Real-time Optimization of Testbeds for Cloudified Radio Access Networks Using Artificial Intelligence

Animesh Singh
Ericsson AB
Stockholm, Sweden
animesh.singh@ericsson.com

Chen Song
Uppsala University
Uppsala, Sweden
chen.song.1637@student.uu.se

Jiecong Yang
Uppsala University
Uppsala, Sweden
jiecong.yang.2357@student.uu.se

Sahar Tahvili
Ericsson AB
Stockholm, Sweden
sahar.tahvili@ericsson.com

***Abstract*—The evolution towards cloudification in Radio Access Networks (RAN) is transforming the telecommunications industry. To validate and assess the performance of Cloudified Radio Access Networks (C-RAN) applications, deploying cloud-native infrastructures in the form of test environments, testbeds, and test infrastructures becomes imperative. However, the intricacies and expenses associated with these testbeds surpass those of traditional testing environments. Effectively utilizing the potential of cloud-native testbeds necessitates real-time decision-making on multiple criteria, including compatibility, capability, cost, capacity, and availability. This paper introduces an Artificial Intelligence-based expert system designed to automatically schedule C-RAN testbeds. The proposed expert system is designed to consider and balance various factors in real time, ensuring optimal utilization of resources and test infrastructures. Employing Artificial Intelligence (AI) based solutions optimizes scheduling decisions, adapts to dynamic environments, maximizes resource utilization, reduces operational costs, and improves turnaround times by dynamically adjusting priorities based on real-time conditions. The feasibility of the AI-based solution proposed in this paper is rigorously assessed through an empirical evaluation conducted on a Telecom use case at Ericsson AB in Sweden. The results demonstrate a remarkable 17% optimization in overall costs with the implementation of the proposed solution.**

Keywords-Software Testing; Artificial Intelligence; C-RAN; Testbed; Optimization; Reinforcement Learning

I. Introduction

The rapidly evolving telecommunication industry places a growing demand on expediting the delivery of telecommunication applications and products. Finding a balance between product quality, cost-effectiveness, and swift deployment remains a persistent challenge in large-scale industries [1] [2]. To address this, various solutions, like Cloudification, have emerged in this domain, offering the potential to maintain scalability, accessibility, and mobility for telecom products. One key concept gaining significant attention is Cloud RAN (C-RAN), which involves the utilization of cloud-based services and infrastructures for radio access networks [3]. While the C-RAN solution provides numerous advantages, it also has its challenges, including system complexity and infrastructure costs, such as those associated with developing, building, constructing, establishing, and utilizing the C-RAN infrastructure. The main differences between a traditional RAN and a C-RAN architecture lie in their respective testing environments, testbeds, and infrastructure. In a traditional RAN setting, the testing infrastructure tends to be well-established and relatively straightforward. The equipment and procedures used for testing are typically well-defined and familiar to telecom professionals. In contrast, the testing environment for C-RAN introduces a range of complexities, flexibilities, and differences. Since C-RAN relies on cloud-based solutions and virtualized network functions, a testing environment (often called a testbed) must embed various virtualized components, such as hardware, radio gateways, simulators, and software components. These components are essential for emulating the C-RAN environment accurately. In essence, a configuration serves as a comprehensive description of a test bed's capabilities and capacities, which can be presented as a unique ID. The utilization of a configuration ID proves invaluable in distinguishing and uniquely identifying various configurations within the testing environment. This systematic approach facilitates efficient management and organization of testing resources. Efficiently optimizing the utilization of C-RAN infrastructure, including testbeds, offers a host of compelling advantages that span well beyond cost reduction, scalability, performance, energy efficiency, and network reliability. Furthermore, considering the crucial aspects of on-time delivery and the diverse dimensions of C-RAN products, traditional optimization models may prove less effective. In contrast, employing the power of AI and Machine Learning (ML) techniques introduces a plethora of advantages for the dynamic optimization of C-RAN testbeds during the testing process. By aligning the availability of testbeds with the timing of requests, the system can strategically power on and off these resources. This on-demand usage minimizes overall energy consumption, promoting energy efficiency and reducing the carbon footprint of the testing infrastructure. Utilizing agile methodologies, such as different IT service management software (e.g., Jira, Azure DevOps) for testbed scheduling can help teams manage software development. However, some manual tasks like request creation, analysis, and information provision still face challenges of ambiguity, uncertainty, and time efficiency. This challenge is more pronounced in large industries, where engineers use distinct terminologies when requesting testbed bookings. For example, the testing team manually initiates a ticket in text format, navigating through predefined options to specify the testing requirements. Subsequently, test managers assess the compatibility and availability of a suitable testbed, relying heavily on domain knowledge. This manual process introduces subjectivity, risking, e.g., double bookings or delaying product delivery. As C-RAN products scale, these inefficiencies highlight the impracticality

and scalability of manual processes in modern software development. Addressing these issues requires automated, streamlined approaches to enhance efficiency, accuracy, and productivity. This paper introduces, implements, and evaluates an AI-based solution for dynamically scheduling testbeds in the context of testing the C-RAN applications. The feasibility of the proposed solution is studied by an empirical evaluation that has been performed on a telecommunication use case at Ericsson AB (EAB) in Sweden. The empirical evaluation demonstrates promising results, indicating the adaptability and potential applicability of the proposed AI-based solution in larger industries. The performance of the proposed AI-based solution in this paper has also been compared with a first-come, first-served (FCFS) queuing approach. The positive outcomes suggest that the scheduling system can be effectively integrated into diverse industrial settings, showcasing its versatility and suitability for broader applications within the telecommunications domain. The organization of this paper is laid out as follows: Section II provides a background of the initial problem and also an overview of research on test environment optimization, Section III describes the proposed AI-based expert system. An industrial case study has been designed in Section IV. Section V clarifies some points of future directions of the present work and finally, Section VI concludes the paper.

II. Background and Related Work

Test optimization holds a vital role in the software development life cycle. One avenue for achieving this is through the management of the available testing resources, such as testbeds, and test environments, to ensure that the overall quality of the final product is improved while the time to market is reduced [1] [4]. Employing the traditional analytical technique that attempts to find the globally optimal solution through mathematical models can be excessively time-consuming due to the large search space. On the other hand, heuristic or meta-heuristic techniques might discover a sub-optimal solution but within a reasonable timeframe [5]. Moreover, addressing a substantial set of available data in the industry requires a solution capable of handling large-scale data within a tight deadline. The generalizability and extensibility of machine learning-based solutions have been demonstrated and validated across various industrial, real-world problems in the software testing domain [2]. The process of enhancing test activities including test planning and analysis, test design, test execution, and test evaluation, through the application of AI is referred to as test optimization. The test resource scheduling approaches can enable an optimized allocation of resources, ensuring that testbeds are actively used which reduces unnecessary energy consumption and operational costs associated with maintaining idle testbeds. Optimized resource utilization not only benefits the environment but also leads to cost savings. By dynamically managing testbeds based on demand, organizations can minimize operational expenses associated with energy consumption and maintenance. The AI and ML-based solutions prove to be a promising approach for optimization in industrial systems due to their ability to adapt and learn from interactions with dynamic environments [4]. In industrial settings, where system dynamics, constraints, and objectives may evolve, AI and ML-based models, such as reinforcement learning offer a flexible framework. This benefit lets the system learn the best ways to make decisions by trying things out, which helps it improve tasks like scheduling, managing resources, and controlling things. The adaptability of AI-based solutions makes them well-suited for handling complex, uncertain, and changing conditions in industrial systems, ultimately leading to improved efficiency, resource utilization, and overall performance. Testbeds and test resource management can be viewed as dynamic scheduling problems. In this context, AI-based solutions, particularly reinforcement learning, which is capable of handling large-scale data, can be dynamically applied in the industry to make informed decisions regarding the scheduling of testing resources. Q-learning, Genetic Algorithms (GA), and Ant Colony Optimization (ACO) are some examples of the optimization techniques used in scheduling, where each has its strengths and weaknesses. Considering dynamic changes and large data sizes utilizing AI/ML-based approaches for dynamic scheduling has received a great deal of attention. In this regard, AI/ML-based solutions (such as Q-learning) stand out due to their adaptability to dynamic environments and ease of implementation, especially when the state and action spaces are well-defined. On the contrary, while traditional optimization solutions, such as Genetic Algorithms (GA) and Ant Colony Optimization (ACO) demonstrate prowess in managing large search spaces and intricate objective functions, they often encounter challenges in dynamic environments and demand meticulous parameter tuning. These factors can hinder their effectiveness, particularly in scenarios characterized by fluctuating conditions and extensive datasets [6]. In contrast, Q-learning's adaptability positions it as an attractive choice for such dynamic environments and substantial data sizes. Its ability to learn and adjust strategies based on real-time feedback makes it well-suited to navigate unpredictable scenarios efficiently. Additionally, Q-learning's capacity to prioritize cost optimization while considering request priorities enhances its versatility in addressing various scheduling challenges. Furthermore, FCFS scheduling, although simplistic, has its merits [7]. It offers a straightforward and intuitive approach, making it easy to implement and understand. In scenarios where task priorities are relatively homogeneous or where quick task processing is essential, FCFS can provide a pragmatic solution with minimal computational overhead. However, FCFS may struggle in situations with highly variable task priorities or when resource allocation requires more sophisticated decision-making processes. While Q-learning shines in dynamic and data-intensive scheduling environments, FCFS remains a viable option in certain scenarios due to its simplicity and ease of implementation. The choice between Q-learning and FCFS, as well as other optimization algorithms, should be made based on a thorough understanding of the specific characteristics and requirements of the scheduling problem at hand. Using Reinforcement Learning algorithms helps optimize testbeds

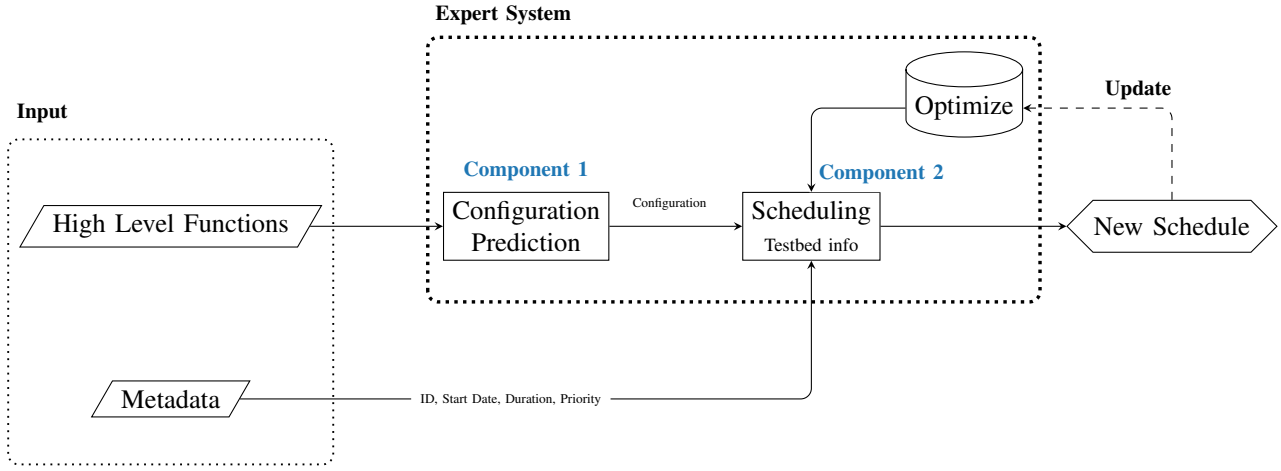

Figure 1. A holistic overview of the proposed expert system in this study.

by learning the best scheduling methods through ongoing interaction with the testing environment. This means making smarter decisions and improving testbed setups over time, using past data and changing needs. Machine Learning also plays a key role in managing test resources effectively, predicting resource needs by studying past usage, test requirements, and outside factors. This predictive analysis helps allocate resources better, making testing more efficient in industries. Reinforcement learning has been explored in previous research for solving scheduling problems. In a study by Reyna et al. [8], they presented a Q-Learning-based approach specifically addressing scheduling problems like the Job Shop Scheduling Problem (JSSP) and Flow Shop Scheduling Problem (FSSP). Another work by Martinez [9] introduced a generic multi-agent reinforcement learning approach adaptable to various scheduling scenarios. However, both studies mainly focused on JSSP, FSSP, and their multi-stage job variants. Kaur et al. [10] explore three distinct approaches within Goal Programming: the Weighted Approach, the Preemptive Approach, and the Chebyshev Approach, providing a comprehensive analysis. Additionally, the study discusses the handling of the three objectives individually as single-objective problems. The proposed optimization models are validated using a dataset from agile-based software development, and sensitivity analysis is conducted to assess the impact of the involved variables. Anand et al. [11] investigate the aspect of multi-upgradation, proposing various optimization problems that address the optimal allocation of testing resources to different versions. The solution presented in [11] comprises a set of models solved using a dynamic programming approach, complemented with numerical illustrations.

III. The Proposed Solution

The AI-based expert system proposed in this study contains two main components. Figure 1 shows a detailed representation of the expert system, emphasizing the essential elements, the necessary input, the sequential steps involved, and the expected output. The development path of the expert system introduced in this study embarks on a journey that commences with the analysis of numerous requests which are composed in semi-controlled natural language by the testing team. To analyze this textual data, various natural language processing techniques are deployed. However, even with AI assistance, the process of natural text analysis can be time-consuming and prone to errors. Through our experiences in text analysis, we arrived at a significant insight: a subset of the information provided by the testing team can be sufficient for discerning the anticipated capabilities and characteristics of a Cloud-native testbed. In light of this realization, we turned to different keyword extraction techniques to efficiently identify the essential information required for configuration prediction. This shift in approach not only streamlines the process but also reduces the potential for errors and accelerates the development of the expert system. It underscores the value of leveraging keyword extraction as a powerful tool for discerning the vital details within the textual requests, facilitating the seamless prediction of configurations for a Cloud-native testbed. The following paragraphs provide more details about the data, the embedded components, and the expected output of the proposed AI-based expert system in this study.

A. Input Data

As highlighted in Figure 1, the extracted keywords have been categorized into two primary groups: 1- High-Level C-RAN Functions and 2- Metadata. The metadata category plays a crucial role in facilitating real-time decision-making processes. Metadata refers to additional information or data that provides context or details about the main data. In the context of the proposed AI-based expert system, metadata includes details, such as request ID, timestamps, source locations, priority of each request, or other relevant contextual information. Capturing the metadata enhances the understanding and utility of the primary information, contributing significantly to the system's effectiveness in real-time decision-making for scheduling C-RAN testbeds. Meanwhile, the high-level functions category is instrumental in the automated prediction

of configurations for the testbed. A testbed configuration represents the capacity and capability of a C-RAN testbed. The high-level functions, as illustrated in Figure 1, provide essential elements for configuring the C-RAN testbed. These elements, derived from the extracted keywords, include parameters, such as network topology, bandwidth, number of cells, and other key hardware (HW) and software (SW) settings. The AI-based expert system can efficiently adapt the testbed to various scenarios, optimizing its capacity and capabilities based on the provided information. This ensures a flexible and adaptive C-RAN testbed configuration tailored to specific needs and requirements.

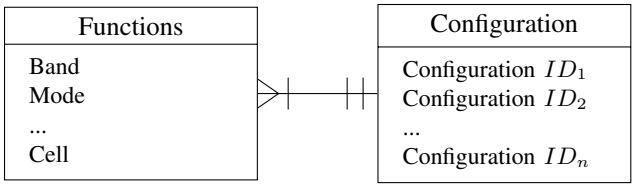

Figure 2. The Many-To-One mapping of the high-level functions using classification models for configuration prediction.

To acquire the requisite input for the proposed expert system, we have created a graphical user interface (GUI). This interface displays a spectrum of metadata choices, including Request ID, Date, Duration, and Request Priority. In essence, the proposed expert system in this study will receive input from all possible combinations of the mentioned data. Figure 2 provides a high-level overview of the prediction process for C-RAN testbed configurations using high-level functions provided by the end users. To generate a testbed configuration, it is recommended to employ multiple classification models, enabling many-to-one mapping. In a general sense, many-to-one mapping signifies a single-valued association, wherein a group of entities can be linked to a similar entity. This methodology improves the adaptability of the system by permitting the creation of diverse configurations grounded in different sets of input parameters. The many-to-one mapping is particularly useful in situations where multiple inputs correspond to a single configuration option. This flexible modeling approach significantly enhances the robustness and versatility of the C-RAN testbed configuration prediction process.

B. Component 1

As illustrated in Figure 1, the proposed expert system in this paper comprises two core AI-based components intricately linked together. As we can see in Figure 1, the output of Component 1 feeds into the second component, establishing a vital connection between these essential elements of the system's architecture. Component 1 focuses on the prediction of configurations based on the high-level function information provided by the end user. As mentioned earlier, a testbed configuration describes crucial details about a testbed's capacity, capability, and properties. In this process, employing various classification models, such as random forest and Support Vector Machine (SVM), enables the expert system to create a unique identifier describing the distinctive features of the testbed. The mentioned classifiers analyze the high-level function information and assign a specific identifier or unique ID to each configuration. This ID serves as a comprehensive representation of the testbed's distinctive characteristics. It is important to note that, even though each configuration is unique, there might be instances where multiple testbeds share the same configuration. This occurrence is due to the inherent complexity of the C-RAN environment, where different testbeds may exhibit similar features or capacities despite being distinct entities. Using classification models enhances the system's ability to efficiently categorize and identify various configurations, contributing to a more nuanced understanding of the C-RAN testbed landscape. Figure 2 offers an insightful depiction of the configuration prediction process, showcasing the innovative solution proposed in this study.

C. Component 2

Component 2 focuses on scheduling the testbed using both the metadata provided by end users and the configurations generated by Component 1. As previously mentioned, the configuration of a testbed highlights its features, and multiple testbeds may share the same features. However, even if testbeds have identical features, their costs can vary. For instance, the cost of an embedded simulator may differ across testbeds, even if the simulators have the same capacity and capability. The final cost of a testbed is influenced by various factors, including the brand of hardware, software, and suppliers involved. Considering the diversity in costs and features, optimal testbed scheduling involves factoring in the gathered metadata. The metadata, provided by end users, provides some essential details, such as the starting date, duration, and priority of each request. Integrating this metadata information with the generated configurations and the associated costs of each testbed enables the expert system to schedule testbeds more efficiently. Combining the metadata information with the generated configuration and cost of each testbed can help the expert system, to schedule the testbeds more efficiently. Considering a broad spectrum of upcoming requests from end users and a multitude of testbeds with diverse configurations and costs, reinforcement learning models are embedded into Component 2 for the real-time scheduling of the C-RAN testbed. In a reinforcement learning approach, an agent, action, state, and reward are all abstract concepts that can be differently defined to solve various problems. In a Reinforcement Learning (RL) model, agents must learn through interaction with the state by sensing and influencing it. The application of RL to the scheduling problem in this study is fitting, considering an agent capable of continuously assigning newly arrived requests for scheduling a testbed. The state, action, and reward can also be defined as the arrangement of existing requests, the assignment of a new incoming request, and the optimization objective, respectively. In this context, an RL model can be seen as an intelligent search method seeking the most optimal scheduling strategy by interacting dynamically with incoming requests.

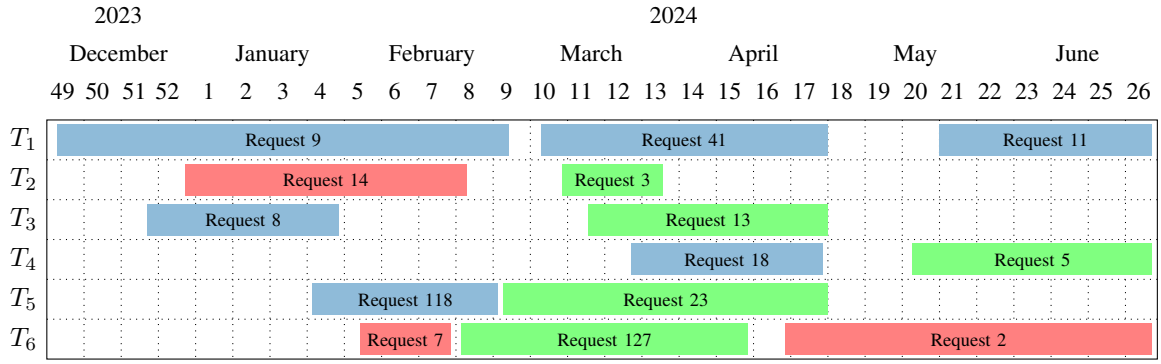

Figure 3. A holistic overview of scheduled C-RAN testbeds utilizing the proposed AI-based expert system. The Y-axis denotes the testbeds, while the X-axis depicts calendar time. Each color represents the priority of each request.

In this paper, Q-learning model has been implemented in the AI-based expert system to determine optimal scheduling for C-RAN testbeds. Q-learning is particularly well-suited for scenarios where an agent interacts with an environment, learning to make decisions that maximize cumulative rewards over time. In Q-Learning, a *Q-function* defines the computation of the expected reward of each state-action pair, and a *Q-table* is used to store the expected reward of all state-action pairs. Their concrete format and structure depend on the problem to be solved. The following list is the overall steps of Q-learning, and it describes how the Q-learning algorithm does scheduling tasks briefly.

1) Initialize the Q-table:
   a) The structure and initialization of the Q-table depend on the concrete problem.
2) Define Q-function and reward:
   a) The format of the Q-function depends on the concrete problem.
3) Learning loop for multiple episodes:
   a) **Select and perform an action to perform**: first, the agent will capture the current state, and then it will choose an action with the highest Q-value.
   b) **Measure the reward**: To learn more optimally, the rewards need to be measured and the Q-table needs to be updated dynamically by the program.
   c) **Update the Q-table**: using the measured reward, the expected reward corresponding to the selected action and current state will be updated based on Equation (1) and Equation (2).

$$TD = r + \gamma \times max(Q) - Q_{old} \quad (1)$$

where, TD is the temporal difference, γ shows discount factor, $max(Q)$ means estimated optimal future (next state) value, Q_{old} is current Q value of the given state and given action.

$$Q_{new} = Q_{old} + \alpha \times TD \quad (2)$$

where, α indicates learning rate, Q_{new} shows the new Q value in the Q-table of the given state and action.

4) Generate the most optimal scheduling decision
   a) The scheduling decision with the highest reward which has been found during the learning loop will be selected as the final decision.

D. Optimization

As depicted in Figure 1, the expert system dynamically receives new requests for scheduling and booking the testbeds. Within the optimization segment of the proposed AI-based expert system in this study, a self-defined optimization objective needs to be established. In this regard, a testbed's total cost (*Overall Cost*) is employed as the objective, calculated as Equation (3):

$$OC = i_1 \cdot makespan + i_2 \cdot \sum_{i=1}^{N} w_i \cdot p_i + i_3 \cdot \sum_{i=1}^{N} c_i \quad (3)$$

where i_1, i_2, and i_3 are weights that require manual configuration by end users, $makespan$ denotes the maximum completion time of all requests, w_i signifies the waiting time (real starting time - release time) for each request, p_i represents the priority of each request, and c_i corresponds to the cost of executing each request on its designated testbed. However, simply summing different objectives with weights can make the setting of weights challenging in practice due to their different scales [12]. Therefore, normalization is necessary to ensure that this single optimization objective yields meaningful information. To bring all three sections to similar scales, the normalized Overall Cost (OC) can be calculated, as shown in Equation (4).

$$OC_{norm} = i_1 \cdot \frac{makespan}{\overline{r} + \overline{d}} + i_2 \cdot \log \sum_{i=1}^{N} w_i \cdot p_i + i_3 \cdot \frac{\sum_{i=1}^{N} c_i}{N \cdot \overline{c}} \quad (4)$$

where $\overline{r}$ and $\overline{d}$ represent the average release time and duration of

all requests, respectively, and $\bar{c}$ denotes the average utilization cost of all testbeds. The use of log in the second section is necessitated by the fact that waiting time cannot be estimated in advance. Although log cannot map the original numbers into a fixed range, it helps align the scale of the second section with the other two. Alternative functions like $sigmoid$ or other normalization methods can also be considered. It is important to note that all Overall Costs mentioned later in this study refer to this normalized form. The objective's intuition is straightforward. By minimizing OC, the generated schedule can, in general, reduce the finish time, decrease waiting times for requests with higher priority, and cut down on the cost of utilizing testbeds. The three weighting factors, set with different ratios, allow different parts to dominate the objective, leading to schedules suitable for different scenarios.

E. Expected Output

The expected output of the proposed AI-based expert system in this study is the real-time scheduling of the C-RAN testbeds. Given that each request may vary in time span and priority, the expert system must be executed upon receiving a new request. Furthermore, considering the finite number of testbeds and their distinct costs, the priority assigned to a request (as provided by the end user in the Metadata section, see Figure 1) significantly influences decision-making. Indeed, the decision to schedule a request on a particular testbed may be subject to change if the expert system receives a new request with a higher priority for the same testbed configuration. This dynamic consideration of priorities ensures that the expert system adapts to changing conditions and optimally manages the allocation of resources based on the most pressing needs. Figure 3 provides a comprehensive view of the real-time scheduling of the C-RAN testbeds using the solution presented in Figure 1. As illustrated, multiple requests are received and scheduled concurrently. The Y-axis represents the testbeds, and the X-axis depicts calendar time. Each color represents the priority of each request.

IV. Empirical Evaluation

To analyze the feasibility of the proposed AI-based expert system, we designed an industrial case study at Ericsson AB (EAB) in Sweden, by following the proposed guidelines of Runeson and Höst [13] and also Tahvili and Hatvani in [2]. A subset of the utilized database to simulate the case study can be found at the GitHub repository [14].

A. Unit of analysis and procedure

The units of analysis in this study consist of a set of available C-RAN testbeds and requests submitted by the testing team for scheduling and booking a testbed for testing a C-RAN application. The case study in this paper is conducted in several sequential steps:

- A total of 500 requests have been extracted from the internal database at EAB and are submitted to book a testbed for testing various C-RAN applications.
- Various text analysis techniques are employed to extract critical keywords, which are then presented in the Graphical User Interface (GUI) of the AI-based expert system. This guides end-users in submitting their requests to book a C-RAN testbed.
- A total of 23 unique configurations for several testbeds are analyzed, and the resulting configuration information is incorporated into the internal database of the AI-based expert system. Subsequently, 197 testbeds are identified that match these unique configurations.
- Evaluations from test managers concerning the generated configurations for each testbed and the scheduling of requests are collected and analyzed.

B. Case Study Report

As mentioned earlier, the primary objective of the proposed AI-based expert system is real-time testbed scheduling. Making accurate real-time decisions for scheduling a testbed is directly linked to the upcoming testing requests and the corresponding capacity and capability of the testbeds. Booking a testbed that fails to meet the engineering requirements of a submitted request can have a direct impact on the testing process. In essence, if the capacity, capabilities, and features of a testbed do not align with the testing requirements, the testing team will be unable to successfully execute the test cases. As mentioned before to apply Q-Learning to the problem, both the Q-table and the Q-function need to be defined. The Q-table further depends on how the state and action are defined based on the problem. In the proposed solution in this paper, the Q-table is organized around three different states that signify the state of a request: "Start", "In Process", and "Finish". Simultaneously, an integer set defines the action space: "0" denotes the "Wait" action, while the remaining values represent various test beds. Moreover, the total number of actions is limited by the number of available testbeds, plus one extra action for "Wait".

Table I. An example of a Q-table illustrating the scheduling of 5 different testbeds (TB), with states including "Start", "In Process", and "Finish".

Actions States	**Wait (0)**	**TB1 (1)**	**TB2 (2)**	**TB3 (3)**	**TB4 (4)**	**TB5 (5)**
Start (0)	0	1	0	0	0	0
In Process (1)	0	0	2	0	0	0
Finish (2)	0	0	0	0	3	0

This design results in a Q-table arrangement, where available actions are represented in columns and request states are represented in rows. Each cell in the table has a Q-value representing the expected total reward for carrying out a specific action in a particular state. The learning process is facilitated by the repeated update of these Q-values, allowing the model to intelligently allocate testbeds based on prior experiences and input from the environment. Table I provides an example of a Q-table for scheduling five different testbeds. The definition of the Q-function further depends on the definition of the reward. To minimize the overall cost (OC), the expected reward in our proposed solution is described in Equation (5).

$$r_{total\ reward} = \frac{1}{OC_{norm}} \tag{5}$$

C. Performance Evaluation

Given that the expert system introduced in this paper integrates multiple ML and AI techniques, it becomes imperative to conduct performance evaluations for each step and model independently. Similarly, assessing the performance of an RL model for real-time scheduling presents a multifaceted challenge. This process necessitates the consideration of diverse metrics that are tailored to align with the specific objectives and constraints of the scheduling problem at hand. The selection of evaluation criteria must be thoughtfully tailored to the application's unique requirements, highlighting the importance of striking a balanced approach to optimize real-time scheduling performance.

1) Component 1 Evaluation: In the experiment, the Train/Test split ratio on the dataset is set to $0.8/0.2$. Metrics, such as Precision, Recall, and F1-Score are essential for assessing multi-class classification performance as they offer a thorough understanding of the model's efficacy. Precision highlights the accuracy of positive predictions by calculating the ratio of accurately predicted instances to all instances anticipated as positive. Conversely, Recall evaluates how well the model captures all relevant cases by calculating the ratio of all real positive instances to all correctly predicted positive instances. The experimental results for the mentioned metrics on different classifiers are presented in Table II.

Table II. PERFORMANCE EVALUATION OF COMPONENT 1 USING PRECISION, RECALL, AND F1-SCORE ARE MEASURED ON MULTIPLE CLASSIFIERS.

Classification Model	Precision	Recall	F1-Score
KNN	0.92	0.91	0.91
Random Forest	0.99	0.99	0.99
Logistic Regression	0.98	0.97	0.98
SVM	0.97	0.97	0.97
Multinomial Naive Bayes	0.89	0.89	0.89

Table II, indicating that Random Forest achieved a significantly higher F1-Score compared to the other classifiers. This could be attributed to its ensemble learning approach, which combines multiple decision trees to improve classification accuracy and generalization. The inherent randomness in Random Forest helps reduce overfitting and increases robustness, which can lead to better performance, especially in complex datasets like the one being analyzed.

2) Component 2 Evaluation: The performance of the second component is evaluated using the following metrics:

- Cumulative Cost (CC): this metric measures the sum of the costs associated with all tasks or components assigned to the testbeds. The lower the CC, the more cost-effective the solution, aligning with our objective of minimizing expenses.
- Tardiness (T_i), also known as cumulative days difference, represents the difference between the actual start date and the requested start date for the tasks or components assigned to the testbeds. The lower the tardiness, the more efficient the solution, reflecting our aim for expedited project timelines.
- Cumulative Reward (CR): this metric, denoted as CR, is calculated by considering cost, priority, and days which is defined as follows:

$$CR = K \cdot \sum_{i=1}^{N}(L - p_i \cdot IC_i) - M \cdot \sum_{i=1}^{N}(T_i) - Makespan \tag{6}$$

where K, L, and M represent weights that require manual configuration by end users. N denotes the total number of requests, and T_i signifies the tardiness, calculated as the real starting time minus the requested time. Moreover, for each request, p_i denotes the priority of the request and IC_i corresponds to the cost of executing that request on its designated testbed. Makespan in 6 represents the total time required to complete a set of tasks on a given set of resources.

D. Performance comparison between Q-Learning and First Come, First Served (FCFS) scheduling approaches

As reviewed earlier in Section II, various approaches have been proposed for scheduling testbeds and environments in the state of the art. Among these methods, we have chosen to compare the performance of our proposed AI-based solution with the First Come, First Served (FCFS) scheduling approach. FCFS is widely utilized due to its simplicity and straightforward rule: it schedules the first request to arrive and allows it to run to completion. It is important to note that in certain real industrial cases, FCFS scheduling can enhance efficiency compared to more complex and advanced scheduling techniques. This is because FCFS prioritizes simplicity and immediacy, ensuring that tasks are processed in the order they are received. In scenarios where tasks have similar priorities or where quick turnaround times are critical, FCFS can provide a straightforward and effective solution. Additionally, FCFS minimizes the overhead associated with decision-making and prioritization, making it suitable for environments where resource allocation needs to be rapid and uncomplicated [7]. However, it is also crucial to acknowledge that FCFS may not always be the optimal choice, particularly in situations where task priorities vary significantly or where certain tasks require high costs. In such cases, more sophisticated scheduling algorithms, including AI-based approaches, may offer better performance by dynamically adjusting resource allocation based on various factors, such as task urgency, resource availability, and overall system optimization goals. Therefore, while FCFS remains a valuable and widely used scheduling strategy, its effectiveness ultimately depends on the specific requirements and constraints of the given scenario. As mentioned earlier, this case study involves the utilization of multiple testbeds and requests, each operating within distinct timelines. The study adheres to a set of rules extracted from EAB's industrial projects, which govern the allocation and scheduling of resources. These rules are meticulously applied to ensure the accuracy and relevance of the study's findings to real-world industrial scenarios. For the training of the Q-learning model, the following has been initialized:

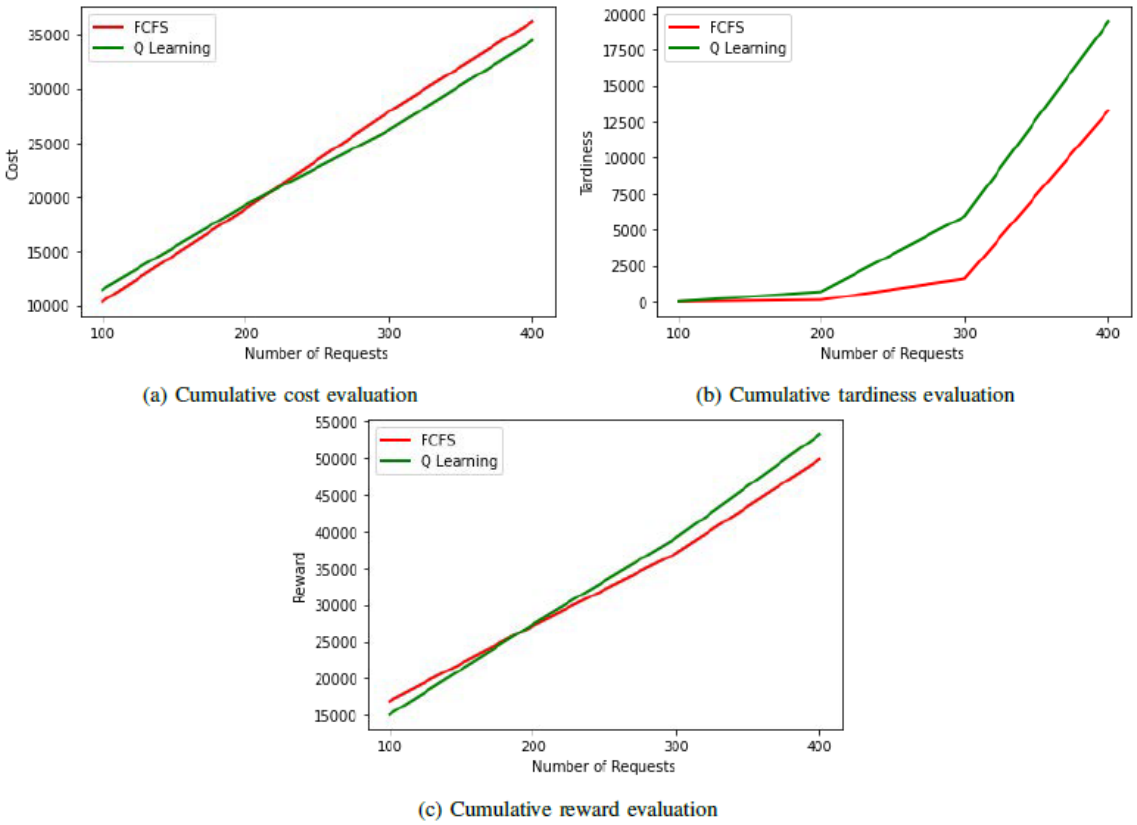

Figure 4. Performance comparison of Component 2 with Q-learning and FCFS approaches, utilizing cost, tardiness, priority, and reward metrics.

1) The release timetable: each element is generated as a discrete random integer variable ranging from 0 to 20 (representing different starting weeks) with a uniform distribution.
2) The compatibility table: each element is generated as a boolean variable with a uniform distribution. Additionally, at least one testbed is compatible with each request.
3) The testbeds cost table: each element is randomly chosen from a set of real cost numbers of the testbeds provided by the industrial partner.
4) The priority table: each element is generated as a discrete random integer variable ranging from 1 to 5. The distribution decreases from 1 to 5, simulating that most requests have low priority while only a few requests have high priority.
5) The duration table: each element is generated as a discrete random integer variable ranging from 1 to 6 (representing different numbers of weeks). The distribution of 2 is higher while others are the same, simulating that the number of requests requiring two weeks is relatively higher based on our observation of real data.

Figure 4 presents a comparative performance study of the Q-learning and First-Come-First-Served (FCFS) scheduling approaches. This study delves into the metrics of cumulative cost, tardiness, priority, and reward as they relate to the number of processed requests. The results are encapsulated within three graphs, each depicting a unique aspect of performance and providing valuable insights into the effectiveness of the scheduling methods. Figure 4a demonstrates that the Q-learning method consistently surpasses FCFS in terms of cost optimization across different request volumes. This trend persists, highlighting the robustness of the Q-learning approach in reducing costs by up to 17% compared to the FCFS approach. Contrarily, Figure 4b demonstrates FCFS's strength in minimizing tardiness. It consistently exhibits high efficiency in this aspect compared to Q-learning. The Q-learning model excels in prioritizing both the cost-optimized allocation of testbeds for each request and their associated priorities. As depicted in Figure 4c, with the increase in requests, both strategies demonstrate a similar and consistent rise in rewards. Furthermore, the Q-learning approach outperforms the FCFS model, showcasing its superior performance as the workload intensifies. This convergence of reward trajectories highlights the comparable overall performance between Q-learning and FCFS, with Q-learning demonstrating its effectiveness in managing increasing request volumes. When the number of requests escalates from 100 to 400, employing the Q-learning approach leads to a significant 19% growth in the overall cumulative reward compared to the FCFS approach. In conclusion, the Q-learning algorithm offers a more economical solution

despite its tendency to incur more tardiness compared to FCFS. The cumulative reward evaluation illustrates a noticeable trend of reward growth favoring Q-learning over FCFS. This trend suggests that Q-learning gradually outperforms FCFS in optimizing task allocation, showcasing its capability to achieve higher rewards as the number of requests increases. The Q-learning model used for comparison was trained over 1000 episodes with a learning rate of 0.1 and a discount factor of 0.9. This rigorous training regime underscores the reliability of the findings and the robustness of the Q-learning approach in scheduling testbeds effectively.

V. Discussion and Future Direction

The main objective of this study is to design, implement, and evaluate an AI-based expert system dedicated to scheduling C-RAN testbeds for diverse applications. In pursuit of this goal, we present the following contributions:

- Multiple Natural Language Processing (NLP) based approaches are used to extract critical keywords, enabling end users to configure and schedule a C-RAN testbed effectively. The extracted keywords are later embedded and presented in the GUI of the proposed solution in this study.
- Multiple machine learning models are employed to predict configurations for C-RAN testbeds based on end-user input.
- A Reinforcement Learning approach has been proposed for the optimal scheduling of testbeds, utilizing Q-learning to identify feasible time slots.
- These outlined phases are seamlessly integrated into a Python-based tool.

However, during conducting this study, several challenges have been faced including the lack of sufficient and balanced datasets makes the usage of oversampling necessary which may affect the model's performance in production due to overfitting [2]. Moreover, some potential threats to the validity of the obtained results in this study can be summarized as follows. Addressing these threats through sensitivity analysis, robust experimental design, and validation against real-world data can strengthen the reliability and generalizability of the results.

- Simulation assumptions: the results may be influenced by the assumptions made in the simulation environment. Assumptions regarding testbed compatibility, request priorities, and cost distributions could impact the generalizability of the findings to real-world scenarios.
- Parameter sensitivity: the performance of the algorithms could be sensitive to the choice of hyperparameters, such as learning rate, discount factor, and exploration-exploitation trade-off in the case of Q-learning. Small variations in these parameters may lead to different results and interpretations.
- Algorithm initialization: the initial state of the Q-learning algorithm, including the initialization of Q-values and exploration strategy, could impact the learning process and subsequent performance. Variations in initialization methods may yield different results.
- Environmental dynamics: the simulation environment may not fully capture the complexity and dynamics of real-world industrial scheduling scenarios. Factors, such as changing priorities, unexpected events, and resource constraints could influence the performance of the algorithms differently in practice.
- Evaluation metrics: the choice of evaluation metrics, such as cumulative cost, tardiness, and reward, may not fully capture the overall performance of the scheduling algorithms. Other important factors, such as resource utilization, scalability, and robustness, should also be considered for a comprehensive evaluation.

The future scope of this paper may involve transitioning to a multi-label categorization paradigm. In contrast to the existing method, multi-label categorization acknowledges the potential overlap or presence of distinct qualities by allowing the simultaneous examination of multiple test beds for a given set of functionalities. This transition would empower the model to identify more intricate linkages and nuances in the data, fostering a more comprehensive understanding of the underlying patterns. Expanding to multi-label categorization has the potential to unveil previously undiscovered aspects of predictive power. This is especially relevant in situations where a group of selections may concurrently belong to several testbeds. Such an extension could result in a classification model that is more resilient and adaptable, effectively handling scenarios in which examples exhibit different attributes and simultaneously belong to different categories. The enhanced model's ability to capture complex relationships within the data may lead to more accurate predictions and a deeper insight into the intricacies of the underlying system. Moreover, in this paper, our focus was on scheduling the currently existing testbed. However, the same approach can be extended to the creation of new testbeds in response to upcoming requests. In practice, testbeds can be commissioned for a limited period, leveraging the provided metadata specified in the submitted build requests. The decision on whether to retain, decommission, or update a testbed can be informed by the nature of upcoming requests. This expansion of the approach to include the creation of new testbeds allows for a more dynamic and adaptive system. The system can respond proactively to emerging requirements, optimizing resource allocation not only for the current testbeds but also for the potential introduction of new ones. This adaptability enhances the overall efficiency and responsiveness of the scheduling system, ensuring that the available resources are aligned with the evolving needs of the testing environment. Utilizing the above-mentioned approach has several advantages, particularly in terms of energy efficiency and environmental impact. The approach's adaptability to create new testbeds based on upcoming requests ensures that the testing environment can scale efficiently. This scalability is essential for accommodating growth in testing demands without compromising energy efficiency.

VI. Conclusion

Test optimization plays a crucial role in the software development life cycle, and effective scheduling of testbeds and environments stands out as a key strategy for achieving this optimization. In this paper, we have introduced, implemented, and evaluated our proposed approach and tool for scheduling C-RAN testbeds to facilitate testing across various applications. The AI-based expert system presented in this study provides a user-friendly GUI, allowing end-users to input requests in the form of high-level functions and metadata. The system comprises two main components. The first component automatically predicts and presents configurations that include the capability, capacity, and required features for testing a C-RAN application. The second component prioritizes testing requests for execution based on their metadata. Empirical evaluations conducted at Ericsson AB, combined with an analysis of results from an industrial project, confirm that the proposed AI-based system is a practical tool for scheduling testbeds effectively. Furthermore, the proposed system exhibits versatility in handling a diverse set of testing requirements and testbeds with distinct configurations. Its adaptability is evident in dynamically receiving and analyzing upcoming testing requests, providing different decisions based on the inserted requests. This adaptability ensures that the system remains responsive to changing testing needs and effectively manages the scheduling of testbeds accordingly. By optimizing the utilization of testbed resources, our approach not only improves operational efficiency but also aligns with sustainable practices. The dynamic management of testbeds, powering them on and off based on demand, contributes to energy efficiency and reduced environmental impact. This holistic approach positions our AI-based expert system as a comprehensive solution for testbed scheduling, combining user-friendliness, adaptability, and sustainability for an enhanced testing environment. In conclusion, while First Come, First Served (FCFS) offers a straightforward and easily implementable approach to scheduling, the utilization of reinforcement learning, such as Q-learning, presents significant advantages in handling dynamic environments, large datasets, and improving accuracy over time. The continuous learning capability of reinforcement learning algorithms allows for adaptability to changing conditions and optimization of resource allocation strategies. We need to consider that AI/ML-based approaches to industrial processes require an initial investment in terms of cost and effort. However, the return on investment typically outweighs these initial expenses [15] [16]. The enhanced efficiency, improved resource allocation, and ability to adapt to evolving conditions offered by reinforcement learning justify the adoption of such advanced techniques in industrial settings. Therefore, while FCFS and other traditional scheduling and optimization approaches remain viable options in certain scenarios, the recommendation is to leverage reinforcement learning algorithms for scheduling tasks in dynamic and data-intensive environments. This strategic shift towards AI/ML-based approaches promises to unlock new levels of productivity and efficiency in industrial operations, ultimately leading to substantial gains in performance and competitiveness.

Acknowledgments

This work was supported by the VINNOVA grant 2023-00244 through the D-RODS project.

References

[1] S. Tahvili, "Multi-criteria optimization of system integration testing", Ph.D. dissertation, Mälardalen University, Dec. 2018, ISBN: 978-91-7485-414-5.

[2] S. Tahvili and L. Hatvani, *Artificial Intelligence Methods for Optimization of the Software Testing Process With Practical Examples and Exercises*, Elsevier, Ed. Elsevier, Jun. 2022, ISBN: 978-0323919135.

[3] A. Younis, T. X. Tran, and D. Pompili, "Bandwidth and energy-aware resource allocation for cloud radio access networks", *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, pp. 6487–6500, 2018.

[4] M. Felderer, E. P. Enoiu, and S. Tahvili, "Artificial intelligence techniques in system testing", in *Optimising the Software Development Process with Artificial Intelligence*, F. C. José Raúl Romero Inmaculada Medina-Bulo, Ed., Springer, Jun. 2023, ISBN: 978-981-19-9947-5.

[5] A. Arisha, P. Young, and M. El Baradie, "Job shop scheduling problem: An overview", in *International Conference for Flexible Automation and Intelligent Manufacturing (FAIM 01)*, 2001, pp. 682–693.

[6] N. Sariff and N. Buniyamin, "Comparative study of genetic algorithm and ant colony optimization algorithm performances for robot path planning in global static environments of different complexities", Jan. 2010, pp. 132–137.

[7] T. Aladwani, "Types of task scheduling algorithms in cloud computing environment", *Scheduling Problems-New Applications and Trends*, pp. 1–12, 2020.

[8] Y. Fonseca-Reyna, Y. Martinez, J. Cabrera, and B. Méndez-Hernández, "A reinforcement learning approach for scheduling problems", *Investigacion Operacional*, vol. 36, pp. 225–231, Jan. 2015.

[9] Y. M. Jiménez, "A generic multi-agent reinforcement learning approach for scheduling problems", *PhD, Vrije Universiteit Brussel*, vol. 128, 2012.

[10] J. Kaur, O. Singh, A. Anand, and M. Agarwal, "A goal programming approach for agile-based software development resource allocation", *Decision Analytics Journal*, vol. 6, p. 100 146, 2023, ISSN: 2772-6622.

[11] A. Anand, S. Das, O. Singh, and V. Kumar, "Resource allocation problem for multi versions of software system", in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 571–576.

[12] Z. Wang, K. Tang, and X. Yao, "Multi-Objective Approaches to Optimal Testing Resource Allocation in Modular Software Systems", *IEEE Transactions on Reliability*, vol. 59, no. 3, pp. 563–575, 2010.

[13] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering", *Empirical Softw. Engg.*, vol. 14, no. 2, pp. 131–164, Apr. 2009, ISSN: 1382-3256.

[14] A. Singh, *Real time optimization of testbeds*, https://github.com/Animesh963/Real-time-Optimization-of-Testbeds, 2024.

[15] H. Eljak *et al.*, "E-learning-based cloud computing environment: A systematic review, challenges, and opportunities", *IEEE Access*, vol. 12, pp. 7329–7355, 2024.

[16] S. Tahvili *et al.*, "Cost-benefit analysis of using dependency knowledge at integration testing", in *The 17th International Conference On Product-Focused Software Process Improvement*, Nov. 2016.