GMR
The Original

A MULTI-OBJECTIVE INTELLIGENT LOAD BALANCING ALGORITHM FOR SDN-ENABLED EDGE CLOUD SYSTEMS

[1*]Bakkala Santha Kumar, [2]R Shankar

[1]Research Scholar, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India. santhakumar.bakkala@gmail.com
[2]Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India. shankarrajendran75@kluniversity.in

ABSTRACT
SDN-based systems with edge-clouds are more commonly used to provide support to latency-sensitive and resource-intensive applications with dynamic and heterogeneous operating conditions. In that case, efficient task assignment must consider concurrently the availability of computational resources, network characteristics, reliability of the service, and the distribution of the workload in the long run. The paper suggests a two-tiered smart scheduling model of SDN enabled edge-clouds. During the first stage, a multi-objective scheduler aided by a controller chooses the node of execution based on shared assessment of CPU usage, memory usage, queue However, there are multiple stages to it: In the first stage, a controller-assisted multi-objective scheduler picks the node of execution by combining the assessment of CPU usage, memory utilization, queue occupancy, delay, bandwidth, reliability, packet degradation, and The second stage will focus on the implementation of machine learning proactive load prediction into the scheduler to enhance its ability to make decisions with future workload variation. The experimental findings indicate that the AI-controlled scheduler is the most performance-friendly in latency and the average response time is 1.4796, and the rates of SLA violation are 0.0497. Of all the adaptive variants, the XGBoost-copiloted model has the best balance-based-behavior with an optimal fairness index of 0.2986 and a low load imbalance factor of 0.0685. These results reveal that reactive SDN-aware scheduling can serve effectively delay-sensitive operation whereas lightweight predictive adaptation can enhance fairness and stable workload over the long term.

KEYWORDS: Edge-Cloud Integration, Intelligent Load Balancing, Multi-Objective Optimization, Resource Allocation, Software-Defined Networking.

1. INTRODUCTION

The accelerated deformation of Internet of Things (IoT), smart healthcare, industrial automation, autonomous systems and real-time multimedia services has stimulated the necessity of computing infrastructures that are capable of offering low latency, high scalability, and resource utilization efficiency at the same time. Conventional centralized cloud computing provides a lot of processor capacity, but the distance between the centralized location and the end users usually increases the transmission delay and network congestion of the delay-sensitive applications. Computation closer to data sources on the edges, and the additional centralized control, programmability, and visibility of the network across the entire network on SDN are all very much appreciated in dynamic traffic engineering and resource orchestration. Consequently, edge-cloud integration has become a more promising architecture that can be used to accommodate heterogeneous and latency-sensitive workloads SDN-enabled. According to recent surveys, SDN, orchestration, and AI-aided control are emerging significant enables the management and balancing of resources within a distributed edge-cloud system.

The potential of SDN-enabled edge-cloud environments with efficient load balancing is a challenging issue. This difficulty is not just in the proportionate allocation of incoming tasks, but also in the coordinated decision making to be made under various coexisting constraints, such as heterogenous node capacities, varying conditions of traffic, constrained resources of edges, dynamic queue states, bandwidth change, node service deadlines and reliability. Under these conditions, faulty decision of schedule can overload the adjacent node of the edge, raise response time, breach service level contracts and underutilize the other existing resources. The recent reviews on SDN-based cloud scheduling and QoS-aware SDN-IoT balancing have demonstrated that the latency, throughput, delay, resource use, packet loss, waiting period and reliability are routinely influenced by the load imbalance confirming the acuity of the dilemma in real-world implementations.

The limitations of the current methods additionally enhance the necessity of working on this problem. More traditional methods of load balancing: Round Robin, weighted assignment, least-loaded selection and offloading using a static threshold: simple to code, yet tend to be inherently reactive and unable to work well with highly dynamic loads. Most

of the available strategies base their decisions on the current load of the server or routing state without taking into account the trends in the future workload or by taking a combined view of both the computation and network factors. Alfredo may be based on centralized assumptions or incomplete knowledge of neighbour states (even by recent adaptive methods), and therefore less effective in dynamic edge environments. Models such as reinforcement-learning-based dynamic load balancing have been identified to address some of these flaws, although more recent literature also demonstrates that most these solutions can also make a model more complex, and more difficult to implement in terms of real-time operation.

Despite the previous research on multi-objective scheduling and machine learning based prediction, despite the fact that the previously discussed approaches generally focus on any one of them, using reinforcement-learning models that are energy-intensive and relatively uninterpretable, they are currently mainly implemented separately. Conversely, the current paper unites the SDN-assisted global visibility, score-based multi-objective scheduler and the lightweight predictive refinement layer into a single edge-cloud load balancing structure. It contributes little, however, to the use of prediction by itself, but rather to the production of an interpretable and viable deployable architecture that (jointly) represent the computational state and network state and future load tendency and explicitly characterize the trade-off between long-term fairness increase and latency minimization.

The other research gap in the literature is that a large percentage of the previous literature only considers network-side load balancing in SDN or only considers compute-side task scheduling in edge/cloud systems, but never fully combines the two sides of the coin into a single decision-making process. Nonetheless, in edge-cloud systems with SDN, the quality of execution of a task is no longer determined by the availability of resources on the node but also by the delay in the path, link quality, and controller-aided route choice. Recent survey on SDN-enabled edge cloud networks and SDN-based balancing with a quality-of-service consideration pays attention to the significance of controller intelligence, orchestration, and AI-enriched decision-making mechanisms, but there are limited lightweight and interpretable multi-objective frameworks. This gives a direct necessity of balancing schemes which capitalize on SDN visibility and at the same time are feasible, explainable and computationally efficient.

Out of these gaps, this paper presents an AI-based load balancing scheme in an SDN-enabled edge-cloud integrated environment, and empowers it with adaptive machine learning smartness. The initial phase presents a multi-objective load balancing algorithm with the assistance of the controller that explains how the CPU utilization, memory occupancy, queue occupancy, delay, bandwidth, reliability, packet degradation, and energy cost are jointly considered during the selection of the most suitable execution node. The second stage adds to this decision process a predictive intelligence and involves combining machine-learned future-load forecasting with it such that the system makes proactive, as opposed to a strictly reactive, task allocation decisions. Compared to the heavyweight black-box approaches, the proposed design maintains interpretability by one with a clear score-based decision mechanism and yet, enjoys the adaptive learning. Such a trade off in explainability and adaptability is a significant practical benefit of SDN based edge-cloud systems.

The importance of the proposed work is that it is capable of covering the performance-oriented as well as the balance-oriented objectives in a single context. The AI-driven stage will help minimize response time and violation of SLA by means of controller-aware multi-objective ranking, whereas the adaptive one will increase the levels of fairness and long-term load sharing by predicting overload cases. The importance of such a design is that a real deployment will need not only to be fast in completing its tasks, but it must also be able to maintain the use of the distributed resources during operation. The proposed framework provides a viable solution to intelligent scheduling in a setting where the intensity of workload, network conditions, and resource states are constantly varying based on SDN control-plane visibility and adaptive machine learning.

Primarily, this work has the following contributions. To begin with, there is an AIDS powered multi-objective load balancing algorithm in SDN-enabled edge-cloud networks, and the choices of allocating tasks rely on the compute and network state of information in terms of jointly optimizing the two. Second, adaptive machine learning is used to enrich the proposed scheduler, which means that it will be able to distribute tasks in advance based on the estimation of future loads. Third, the entire experimental scheme is erected to compare the proposed methods in case of a different arrival rate and the infrastructure size in response time, violation rate of the SLA, index of fairness, throughput, and load disproportion factor. Fourth, corresponding comparison is made to compare the relative advantages of reactive AI-based balancing and adaptive ML-enhanced balancing, thus confirming both of the suggested research purposes.

The innovations of the present work is the proposed method that has a four-fold. It develops SDN-enabled edge-cloud load balancing, which is a coherent decision maker whereby computational resource conditions are evaluated together with network delivery conditions and also when placing tasks. Second, it also remains interpretable by having a transparent weighted suitability score, which enables immediate analysis of the impact of each decision factor. Third, it does not require computationally intensive deep reinforcement learning, but rather adds to the scheduler a lightweight proactive prediction layer that predicts node load. Fourth, it considers two different predictive models, XGBoost and LSTM, in the identical line of predictive performance, thus revealing the effect of predictor quality on the tradeoff between response-time performance and fairness over time. This set of characteristics offers a useful and

analytically solvable addition to the current heuristic, traffic-constrained, or black-box adaptive techniques. The present work is novel as it has come up with an open-ended SDN-conscious load balancing framework, which integrates multi-objective task scheduling, network visibility supported by controllers, and lightweight proactive load forecasting under one decision path. In contrast to the previous research, which discusses routing, task offloading, or prediction individually, the proposed approach jointly model system compute-side variables and network-side variables, its explanations are transparent in terms of the explicit weighted score, and various predictors (XGBoost and LSTM) can revise the trade-off between latency-driven and fairness-driven scheduling goals. The result of this is a practically deployable design that is an analytically transparent, and responsive to dynamic workload environments. To be consistent, in the manuscript, the term service-level agreement (SLA), quality of service (QoS), and average response time are used consistently. SLA (service-compliance requirement to deadline or service-level) and QoS (service-quality perspective of end-to-end tax by time and also reliability) and the mean number of seconds each task takes (on average) are end-to-end task completion delays as a major latency-focused metric employed in the experiments. The rest of the paper is structured in the following way. Section 2 examines the literature applicable to SDN-based load balancing, edge-cloud scheduling and adaptive intelligence. The system model is found in section 3. Section 4 develops the problem of optimization. Section 5 explains the suggested AI-based load balancing algorithm. Section 6 presents enhancement of the adaptive machine learning. The experimental set up and results are reported in Section 7. Lastly, the paper ends with Section 8 which provides the research direction in the future.

2. LITERATURE REVIEW

SDN-enabled edge-cloud systems have undergone a shift of the traditional method of load balancing based on rules that are heuristic and based on a threshold to strategies that are predictive, multi-objective, and based on reinforcement learning. Most recent surveys indicate that the current tendency of research is to have algorithms that concurrently optimize latency, throughput, energy, fairness and service reliability instead of considering load balancing to be a one-metric routing problem. Mahdizadeh made a literature review on the task scheduling and load balancing of SDN-based cloud computing and emphasized the increasing value of intelligent algorithms toward enhancing the performance of networks and cloud orchestration [1]. Rostami demonstrated that QoS-conscious load balancing in networks based on SDN IoT is now closely related to the real-time traffic control and service quality enhancement [2]. The same trend is also reflected by broader survey papers: Devi et al. were able to detect that optimization and machine-learning approaches dominated the current research in cloud load-balancing [3], and Kazi et al. were able to consider SDN controllers, orchestration, NFV, and AI-enabled control as the core attributes of edge-cloud management in the future [4]. Sarohe et al. also warned of the fact that load balancing in SDN-IoT has been closely related with energy efficiency, traffic engineering, and routing intelligence [5].

The architectures proposed during 2020-2021 founded the structural premises to the intelligent balancing in the edge and SDN-enabled setup. On one hand, Alonso et al. proposed a network Controlled-IoT architecture based on SDN, NFV, and deep reinforcement learning and demonstrated that one can administer virtualized network resources within QoS constraints with the help of controller-assisted intelligence [6]. Li et al. suggested an edge computing approach to task-allocation, through intermediary nodes, which seeks to routing tasks to relatively light nodes, and enhance the balance within the edge layer [7]. This research work was made out to appear as a load-balancing problem to the IoT by Nezami et al. when approaching the subject of decentralized placement of edge-to-cloud services, where the researchers modeled the trade-off between the local QoS and global spread of workloads [8]. Dong et al. suggested a combined cloud-edge-aware task allocation and load balancing model, and indicated that coordinated deployment of resources in the cloud and in an edge, node could be more useful than unilateral usage [9]. Guo and Hou also investigated software-defined scheduling of tasks in edge computing, and it supported the significance of control-plane visibility in computation offloading and task interaction [10]. These works are significant as it presented edge load balancing as a network and arrange a compute problem, which strongly aligns with the current work.

It changed significantly in 2022, when scientists started to build prediction and learning more into SDN balancing. The use of prediction of the future routing choices to enhance balancing of traffic to the variable traffic was proposed by Chen et al. and they demonstrated that ALBLP is an adaptive SDN architecture with load-balancing that is link-state predictive [11]. Similar, in a related analysis, Chen et al. suggested ALBRL, a reinforcement-based SDN balancing framework that is an automated framework of load distribution contrasted with a manually tuned-based framework [12]. Li et al. proposed a load-balanced system of data-transmission through clouds-edge-end collaboration in the IoT system, that is, it is crucial to consider full-service-chain balancing instead of balancing at a single network-layer [13]. Optimized and deep-learning-assisted SDN balancing policies were also tried out in other studies of this time, as the field switched off traditional fixed-point schedulers. All these works indicated that predictive and learning-based balancing may enhance responsiveness, although a significant number of them were more interested in traffic distribution than joint task-allocation among heterogeneous edge and cloud nodes.

In 2023, studies became more clearer in regard to software-defined edge task definitions and real-time adaptive programming. Zhang et al. suggested RL-SDETA, a reinforcement-based software-defined edge task-allocation

algorithm where the controller utilizes collaborative information of global state to minimize the time to complete and energy use [14]. A Q-learning-based load-balancing solution of real-time tasks in edge computing was proposed by Du et al., and the decision on the distribution of tasks varies with the evolving states of the servers and network conditions [15]. Dreibholz et al. came up with a lightweight cloud and edge platform task-scheduling model, which is motivated by the necessity to operate efficient, low-overhead decision making [16]. During the same time, AI-based load balancing of SDN was surveyed by Alhilali et al. who found throughput, scalability, fault tolerance, and reaction speed among the key evaluation parameters [17]. In schedule delay-sensitive task, Avan et al. reviewed the existing scheduling and observed that centralized and distributed schedulers still have trade-offs between responsiveness and decision overhead [18]. All these studies are indicative that intelligent balancing aids are becoming evaluated not just through reduction of delay but also through run-time viability and practicability of deployment.

The 2024 literature enhanced the argument of balancing using adaptive and multi-objective further. Esmaeili et al. suggested dynamic load balancing in edge computing networks through reinforcement-learning and specifically highlighted three limitations of the existing ones, which included: the use of fixed execution intervals, the use of entire knowledge of the global resources, and the use of centralized dispatch assumptions [19]. The article by Zhou et al. uses SDN and a combination of deep reinforcement learning to optimize energy and load balancing in edge computing, which shows the growing overlap between SDN control and schedules created by DRL [20]. The SDN-enabled MEC studies also analyzed the placement of controllers and choice of edges in both joint delay and energy goals where balancing choices have been found to be profoundly affected by control-plane architecture [21]. Tache et al. found at the survey level that optimization algorithms in SDN were used in routing, traffic engineering, and load balancing [22], whereas Rostami and Mahdizadeh emphasized the significance of QoS intelligent balancing and SDN-IoT networks and SDN-cloud systems [1], [2]. It is apparent in these works that the area is increasingly headed towards adaptive, multi-constraint balancing, although most methods are highly complex in design.

This literature momentum is further propelled even onto deep reinforcement learning, multi-objective optimization, and large-scale hybrid environments in the 2025 future. Kazi et al. have conducted a survey of SDN-enabled edge cloud networks and emphasized AI-enhanced controllers as one of the key directions in resource management and orchestration [4]. Sarohe et al. surveyed SDN-IoT load balancing and pointed out the enabling routing-intelligence, energy-awareness and adaptive-control [5]. Multi-objective reinforcement learning multi-objective reinforcement learning has been studied in other 2025 work, such as adaptive load balancing [23], studying a large-scale SDN fairness-aware selection and balancing [24], and provided broader comprehensive views and reviews of SDN load-balancing methods [25]. These papers demonstrate that research is more aimed at meeting several conflicting objectives and dynamically adjusting the policies to the requirements of users and networks.

Although this has been achieved, there are still some gaps. To begin with, it is the case that a lot of SDN research remains focused on the analysis of traffic engineering or the deployment of controllers, whereas a lot of edge-computing research is interested in the task-scheduling problem without any particular SDN-consciousness. Second, deep reinforcement learning systems are more likely to enhance the adaptability, but sacrificing the interpretability and increased levels of runtime complexity. Thirdly, there is a relative dearth of research that compares a reactive multi-objective AI scheduler to a lightweight predictive improvement in one unified framework. Fourth, imbalanced fairness and load imbalance are commonly used as second metrics despite its criticals to the operation of edge-clouds in the long run. The aforementioned gaps drive this current research project, which integrates a decipherable SDN-aware multi-objective balancing algorithm, with an adaptation machine-learning refinement and measures both the latency-based and balance-based performance. This placement is in line with recent proposals of intelligent but sensible balancing solutions in SDN-enabled edge-cloud networks [1]-[5], [19], [22].

In contrast to previous studies that either work on SDN-based traffic balancing, edge-only task-scheduling, or on overweight deep-reinforcement learning arguments, the references have developed a proposed framework that integrates a comprehensible SDN-aware multi-purpose scheduler with a skinny adaptive machine-learning addition. This design combines minimal decision complexity with controller-assisted visibility with the ability to perform proactive balancing (using future-load prediction). Table 1 is an analytical comparison of the studies of the representative works on SDN-enabled edge-cloud load balancing and the framework suggested. The comparison is categorized along lines of optimization scope, adaptive intelligence, justifiable, supportiveness of collaborative compute-network decision making, and significant shortcomings. This overview provides a more analytical literature review demonstrating that most current solutions are either traffic engineering-oriented, or edge-side-task-scheduling-oriented, or based on reinforcement learning-facilitated adaptation, and the suggested approach integrates SDN-aware, multi-objective scheduling with lightweight, predictive, enhancement in a single, interpretable framework.

Table 1. Analytical comparison of representative SDN-enabled edge-cloud load balancing studies and the proposed framework

Study and Year	Main Focus	Adaptive / Predictive Mechanism	Joint Compute	Interpret ability	Limitation

			+ Network Decision		
Alonso et al. [6], 2020	DRL-based SDN resource management in edge-IoT	Deep reinforcement learning	Partial	Low	High model complexity and limited transparency
Li et al. [7], 2020	Edge task allocation through intermediary nodes	No explicit prediction	Mainly compute-side	Moderate	Limited SDN-assisted network visibility
Nezami et al. [8], 2021	Decentralized edge-to-cloud service placement	No explicit prediction	Partial	Moderate	Focuses on placement trade-offs more than controller-assisted scheduling
Dong et al. [9], 2021	Joint cloud-edge task deployment and balancing	No explicit prediction	Partial	Moderate	Limited SDN control-plane integration
Guo and Hou [10], 2021	Software-defined task scheduling for edge computing	No explicit prediction	Partial	Moderate	Emphasizes scheduling, but lacks proactive load forecasting
Chen et al. [11], 2022	Link-state prediction-based SDN load balancing	Predictive architecture	Mainly network-side	Moderate	Focuses more on traffic balancing than heterogeneous task allocation
Chen et al. [12], 2022	Reinforcement learning-based SDN load balancing	Reinforcement learning	Mainly network-side	Low	Black-box decision making and higher runtime complexity
Li et al. [13], 2022	Cloud-edge-end collaborative transmission balancing	Limited adaptation	Partial	Moderate	Service-chain balancing emphasized more than task scheduling
Zhang et al. [14], 2023	RL-based software-defined edge task allocation	Reinforcement learning	Partial	Low	Reduced interpretability and greater implementation overhead
Du et al. [15], 2023	Q-learning-based real-time edge load balancing	Q-learning	Mainly compute-side	Low	Limited joint treatment of network and compute factors
Dreibholz et al. [16], 2023	Lightweight cloud-edge task scheduling	No explicit prediction	Mainly compute-side	High	Lightweight, but lacks SDN-aware global routing visibility
Esmaeili et al. [19], 2024	RL-based dynamic balancing in edge networks	Reinforcement learning	Mainly compute-side	Low	Centralized assumptions and reduced explainability
Zhou et al. [20], 2024	DRL-based SDN-enabled edge resource scheduling	Deep reinforcement learning	Partial	Low	Strong adaptability but computationally expensive
Tache et al. [22], 2024	Optimization algorithms in SDN	Optimization-oriented	Mainly network-side	Moderate	Broad optimization view, not unified

					edge-cloud scheduling
He et al. [23], 2025	Multi-objective reinforcement learning for adaptive balancing	Multi-objective RL	Partial	Low	Heavy learning pipeline and limited practical interpretability
Shahzad et al. [24], 2025	Fairness-aware SDN balancing	Adaptive / fairness-oriented	Mainly network-side	Moderate	Limited compute-resource integration
Proposed framework	**SDN-enabled edge-cloud multi-objective load balancing with proactive prediction**	**XGBoost / LSTM-based proactive load prediction**	**Yes**	**High**	**Not Applicable**

3. RESEARCH METHODOLOGY

This paper will introduce two-step smart load balancing architecture to an SDN-supported edge-cloud integrated system. The first stage will involve the development of a multi-objective load balancing algorithm with the assistant of a controller to allocate incoming tasks to edge or cloud resources depending on the present node and network situation. The SDN controller collects the data on resource usage, queue occupancy, transmission delay, available bandwidth, packet degradation, and reliability and standardizes and combines the data into appropriateness score of each candidate execution node. The node that has the highest suitability measure is then allocated the task and the controller then places the necessary forwarding path. At the second step, the suggested algorithm is supplemented with adaptive machine learning intelligence. A predictive model is learned based on the historical and real-time observations of the systems so as to approximate the near future load of each candidate node. This forecast is combined with up-to-date suitability index to create an adaptative decision measure, which assists in active load balancing. Consequently, the suggested framework is capable of responding to the immediate change of states and adaptively predicting the future overload, thus enhancing responsiveness, fairness and the stability of services in fluctuating edge-cloud settings. The design aligns with current literature trends that focus on the need of SDN-aware and adaptive AI/ML-based load balancing protocols in distributed edge as well as cloud infrastructures. The whole architecture of the suggested SDN-enabled edge-cloud load balancing framework is portrayed in Figure 1. It starts with task generation and task characterization and continues with real-time system monitoring, which is provided by SDN controller. The reactive scheduling stage involves normalizing, and translating, controller-gathered metrics into node suitability scores on candidate edge and cloud resources. Predictive adaptive stage involves a machine learning model predicting the approximate-future load state of the candidate nodes and the predicted values are added to an adaptive score update process. Depending on the final decision score, the preferred execution node is chosen, controller sets the forwarding path and sends the task of the selection to the preferred node. Drawing a clear distinction between reactive and predictive decisions, the updated flow chart enhances the model clarity and better outlines how immediate observations of the system and the expected conditions of future value are combined to make intelligent decisions regarding the load balancing.

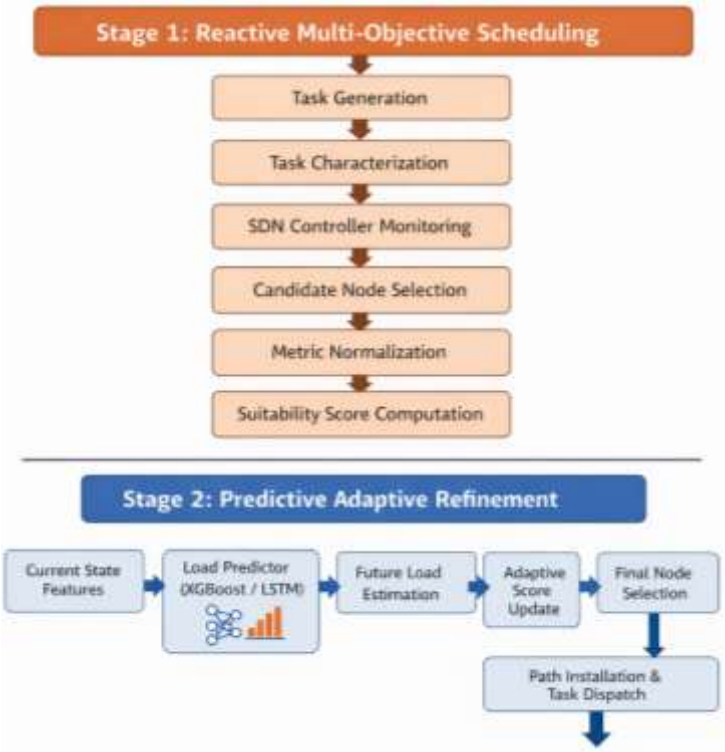

Figure 1. Process flow diagram of the proposed work

The fourfold contributions of this work are its major contributions. To start with, it introduces an AI-based multi-objective load balancing algorithm in an SDN-enabled edge-cloud integrated infrastructure, in which an edge-cloud jointly takes into account compute and network conditions when assigning tasks to controllers by aiding visibility. Second, the proposed load balancer extends with adaptive machine learning that will be used to forecast the subsequent loads of nodes and act proactively to avoid overload of the node to ensure degradation of the service is not experienced. Third, a mathematical optimization model is proposed so as to incorporate delay, fairness, load imbalance, and energy-conscious goals in an integrated decision-making framework. Fourth, a thorough experimental appraisal will be carried out to compare the suggested techniques and traditional and heuristic baselines in dynamic workload, congestion and node variability conditions.

A. System Model

The present study considers an SDN-enabled edge-cloud integrated environment composed of user devices, edge servers, cloud servers, SDN switches, and a logically centralized SDN controller. The system is designed to support heterogeneous application tasks with varying delay sensitivity, computational demand, and bandwidth requirements. Let the set of edge servers be represented as $E = \{e_1, e_2, \ldots, e_n\}$, the set of cloud servers as $C = \{c_1, c_2, \ldots, c_m\}$, and the overall candidate processing set as $N = E \cup C$.

To ensure notation consistency, the manuscript uses T for the task stream, E for the set of edge nodes, C for the set of cloud nodes, $N = E \cup C$ for the full candidate node set, xi for the state vector of node I, Si for the suitability score, L^i(t+1) for predicted future load, and ASi for the adaptive score. The weighting coefficients are denoted by wj with $\sum_{j=1}^{8} w^j = 1$, and the adaptive fusion coefficient is denoted by η. All performance metrics are also represented using consistent notation. Incoming tasks originate from end users or IoT devices and are forwarded through SDN-enabled forwarding elements. Each t_k is characterized by a tuple $t_k = (id, a_k, d_k, s_k, p_k, T_k\}$ where a_k denotes arrival time, d_k denotes computational demand, s_k denotes input data size, p_k denotes task priority, and T_k denotes deadline sensitivity. Tasks with strict latency requirements are preferably served by edge nodes, whereas computation-intensive or overflow tasks may be redirected to neighboring edge nodes or cloud servers depending on resource availability and network conditions.

The SDN controller continuously collects node-level and link-level statistics from the infrastructure. For each candidate node $i \in N$, the controller maintains the state vector $x_i = \{U_i, M_i, Q_i, B_i, P_i, R_i, E_i\}$ where U_i is CPU utilization, M_i is memory utilization, Q_i is queue occupancy, B_i is available bandwidth, P_i is packet loss or path degradation, R_i is node reliability, and E_i is energy cost. This controller-assisted global visibility enables the system to jointly optimize

task allocation and forwarding decisions. The proposed framework has two stages. During the first step, an AI-based multi-objective decision-making process will identify the most optimal processing node considering the state of affairs in the existing system. The second stage is an adaptive machine learning module which predicts the near future node load and augments the decision-making process to allow active balancing.
In solving the above optimization problem, an AI-based multi-objective load balancing algorithm is suggested. The algorithm uses global monitoring capability of SDN controller and calculates a suitability score of each candidate node using normalized resource and network conditions. The intended algorithm works based on the following logic. Upon arrival of a task, the controller initially forms shortlisted candidate nodes. It then normalizes metrics of nodes and paths collected. A suitability score is subsequently calculated in each candidate node. The node that scores the highest is picked and the SDN controller sets up the forwarding rule necessary to redirect the task to the selected point of execution. This process allows the load balancer to collectively take into account both the under-constitution and communications states instead of using fixed threshold policies.

B. Proposed AI-Driven Load Balancing Algorithm

The initial approach that is proposed is a multi-objective controller-aided AI load balancing algorithm. It works in a way to calculate a suitability score of each candidate node based on normalized real-time measurements that are obtained by the SDN controller. The fact that modern load balancing research is carried out more and more in favor of the necessity to not target the selection on a single factor but on a multiplicity of performance dimensions. Let, U_i, M_i, Q_i, B_i, P_i, R_i, E_i be the performance dimensions. After min-max normalization into [0,1], the suitability score is computed with user-defined important weights such that $\sum_{j=1}^{8} w_j = 1$. The node selection rule becomes $i^* = argmax_{i \in N} S_i$. The model is easy to discuss in the paper due to this score-based design; it is also appropriate to ablation experiments. It is also adapted in a natural way to policy preferences like basic edge nodes to give priority to something that requires latency versus basic cloud nodes to give priority to something that requires a lot of computes.

Algorithm 1: Multi-Objective Load Balancing

Input:
  Task stream T
  Candidate nodes N = E U C
  Real-time state from SDN controller
Output:
  Task-to-node allocation A
For each incoming task *tk* in T do
  Retrieve node and network state from SDN controller
  Build candidate set *Nk* based on task type, deadline, and resource constraints
Normalize CPU, memory, queue, delay, bandwidth, packet loss, reliability, and energy values
  For each node I in Nk do
    Compute suitability score:
Si = w1(1-Ui) + w2(1-Mi) + w3(1-Qi) + w4(1-Di)+ w5Bi + w6(1-Pi) + w7Ri + w8(1-Ei)
  End For
  Select best node $i^* = argmax_{i \in N} S_i$
  Install forwarding path to i* through SDN controller
  Dispatch task *tk* to node i*
  Update local and global state tables
End For
Return A

The second contributes to the above algorithm, which has predictive adaptation. Recent researches into adaptive and reinforcement-learning-based load balancing point out that reactive balancing on its own is frequently not enough in a dynamic environment since it does not react to congestion or overload before it makes itself apparent. Predictive and learning-based systems can overcome this deficiency by predicting the future state of nodes. Enhancement, a machine learning model f(.) is learned to forecast the near future load of any node based on the real time and recent system measurements.

Algorithm 2: ML-Enhanced Adaptive Load Balancing

Input:
  Task stream T
  Candidate nodes N
  Real-time system state S

Trained prediction model f(.)
Output:
Adaptive task allocation A*

For each incoming task tk in T do
Retrieve current state from SDN controller
Build candidate set Nk based on task requirements
For each node I in Nk do
Compute current suitability score Si using Algorithm1
Predict future load Lhat_i(t+1) = f(current node/network features)
Compute adaptive score:
ASi = ηSi + (1- η) (1-Lhat_i(t+1))
End For
Select best node i* = argmax(Asi)
If overload risk of i* exceeds threshold ɵ, choose next best node
Install route via SDN controller
Dispatch task tk and log outcome
End For
Periodically retrain or update f(.) using new observations
Return A*.

C. Candidate Machine Learning Models

In the paper, two options were proposed of the adaptive component implementation. The former is an adjusted predictive model, which predicts node loading or protecting time. This is simple to apply and assess in terms of regression measures in the form of RMSE, MAE, and R^2. XGBoost is particularly appropriate in the sense that it can deal with nonlinearity and nonhomogeneous variables effectively.

The second one is a time-series model e.g., LSTM, which forecasts future load based on recent history of node and network states. This can provide better novelty in the event that your experiments model workload bursts and workload timing patterns. The third, more optimistic solution is a reinforcement learning, whereby the agent monitors the system state and trains on the optimal action to assign tasks in order to maximize a reward defined as low delay, fairness and low risk of overload. There is an emerging trend of RL or DRL in adaptive load balancing in recent papers, but complexity of implementation is increased.

Where M represents the number of candidate nodes that are used in each task. In the Algorithms 1 the computation of the scores involves the evaluation of the same number of features per node, which means that the decision complexity per-task is *O(M)*. In Algorithm 2, extra cost in predicting is model dependent. In the case of tree-based models like XGBoost, the inference is also cost-effective yet feasible concerning the real-time scheduling. In this way, the overall complexity of per-task is roughly linear with my candidate node count. This can be used to justify the viability of implementation of the proposed approach in real-time SDN-assisted edge setting.

D. Evaluation Metrics

Several performance metrics are used to assess the performance of the proposed method. The mathematical representations of several performance metrics are computed using equations like:

Average response time: $\overline{RT} = \frac{1}{K}\sum_{k=1}^{K} RT_k$

Throughput: $Throughput = \frac{Total\ completed\ tasks}{Simulation\ Time}$

Task Success ration: $TSR = \frac{Successfully\ completed\ tasks}{Total\ submitted\ tasks}$

The SLA violation rate: $SLA_{viol} = \frac{Deadline{-}missed\ tasks}{Total\ submitted\ tasks}$

The Average utilization across all nodes: RU $= \frac{1}{|N|}\sum_{i\in N}^{K} U_i$

In addition, the load imbalance factor, Jain's fairness index, queue waiting time, and energy consumption are reported. For the ML-enhanced method, prediction accuracy is evaluated using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

4. EXPERIMENTS AND RESULTS

It is suggested to have five test conditions. In case of the former, the rate at which the task enters is low to high and in the latter case, it is high so as to investigate the effects of the congestion and saturation of the system. In the second case the edge nodes number is varied which is used to test the scalability. In the third model, other classes of applications like latency sensitive, compute intensive, and data intensive work loads are factored. The fourth version

involves one of several edge nodes being temporarily overloaded or failed temporarily so that robustness can be tested. The fifth case presents a dynamic bandwidth and delay variability to determine the efficacy of SDN-conscious routing and anticipated balancing.

Three ablation studies be used, the first step would be to compare the existing-state approach where scores only are available to the adaptive score where scores are predicted by the machine learning approach. Second, stress the coefficients of weights in suitability score to study the sensitivity of weights to latency, utilization, and fairness priorities. Third, compare computer-only balancing and the compute-plus-network balancing strategy that is SDN-aware. To assure reliability in the results, the same experiment is to be repeated several times with various random seeds and the average and standard deviation of the respective performance measures are supposed to be presented. One can use a paired t-test or a Wilcoxon signed-rank test to evaluate the statistical significance of the proposed method compared to the strongest baseline.

In order to prove the efficiency of suggested approach, detailed simulative experimental framework is developed. The environment comprises of several edge nodes, one or more cloud nodes, SDN switches, and a centralized SDN controller. Tasks are created variably with arrival rates and of non-homogeneous nature in order to reflect realistic edge-cloud application behavior.

An appropriate implementation stack includes Python as the algorithmic core, SDN emulation layer, e.g., Mininet plus Ryu controller due to its controller logic, and edge/cloud simulator, e.g., iFogSim2, EdgeCloudSim or CloudSim Plus as its workload modeling. To run preliminary experiments, a complete Python-based model is an acceptable simulation of the algorithmic behavior before implementing it in an SDN emulator.

This study uses a set of experiments that are designed around a synthetic workload set created within SDN-enabled edge-cloud simulation environment. The output data set has been taken as temporal node level and network level state observations of a repeated run of simulations and is used to set up and analyze the adaptive predictors. Precisely, the predictive module was trained on 115, 104 samples, which had 18 predictive inputs based on system states observed by controllers, and the performance of scheduling was measured during repeated 350 tasks each-run applications. The simulation environment is 4, 6, 8 and 10 edge nodes, 2 cloud, task arrival rate of 3, 5, 7, 9 and 11, probability of failure is 0.05 and dynamic network jitter factor is 0.15. All experiments were repeated with more than five random seeds in order to minimize stochastic bias.

A. *Setup*

Periodically, the suggested structure was tested on a proposed SDN-enabled edge-cloud integrated environment with the help of a configurable load workload generator and diverse load balancing strategies. It was the aim of the experiments to determine the effectiveness of the proposed AI-driven load balancing algorithm, as well as to evaluate the benefit of the adaptive machine learning improvement under different conditions of traffic, as well as scalability.

The training part of the training involved the adaptive predictors, where 1200 tasks were involved, whereas the evaluation part involved 350 tasks at a run. The training data were created with the arrival rate of the tasks 3, 5, 7, and 9 and with the final experiment count the arrival rate 3, 5, 7, 9 and 11. There were 4, 6, 8 and 10 edge nodes with 2 cloud nodes. All the experiments were repeated five times using various random seeds to minimize stochastic bias. Another aspect that was used to make the environment resemble real deployment uncertainty was a failure probability of 0.05 and a network jitter dynamics factor of 0.15. In the case of the adaptive strategies, the $\eta = 0.70$ model was used (XGBoost-based), and the $\eta = 0.65$ model was used (LSTM-based), and the overload threshold was set at $\eta = 0.9$.

The load balancing techniques that were evaluated include Round Robin, Least Loaded, AI-Driven, Adaptive-XGBoost, and Adaptive-LSTM. The average response time, SLA violation rate, throughput, fairness index, and load imbalance factor helped to evaluate the performance of such methods. The simulation parameters and the experimental configuration used to evaluate the proposed framework are summarized in Table 2.

Table 2. Experimental Configuration of the SDN-Enabled Edge-Cloud Simulation Environment

Parameter	Value
Number of tasks for training	1200
Number of tasks for evaluation	350
Training arrival rates	3, 5, 7, 9
Evaluation arrival rates	3, 5, 7, 9, 11
Edge node configurations	4, 6, 8, 10
Number of cloud nodes	2
Number of repeated runs	5
Failure probability	0.05
Dynamic network jitter	0.15
Adaptive weight for XGBoost	0.70

Adaptive weight for LSTM	0.65
Overload threshold	0.90
LSTM sequence length	6
LSTM training epochs	8
LSTM batch size	64
LSTM hidden dimension	32

B. Load Prediction Performance of the Adaptive Models

The predictive quality of the machine learning models was tested before being incorporated into the scheduling loop. The learned training set had 115,104 samples, 18 features, which are temporal node-level and network-level system states. XGBoost model had an MAE of 0.0258, RMSE of 0.0489 and R^2 of 0.8576 as opposed to LSTM model which had MAE of 0.0569, RMSE of 0.0900 and R^2 of 0.5245. These findings suggest that future load patterns were well predicted using the XGBoost model compared to the LSTM model in the existing experiment environment. The predictive performance of the adaptive machine learning models is summarized in Table 3.

Table 3. Comparison of predictive performance of the adaptive load forecasting models

Model	MAE	RMSE	R^2
XGBoost	0.0258	0.0489	0.8576
LSTM	0.0569	0.0900	0.5245

The difference is significant, as the adaptive scheduling step requires futuristic load estimation accuracy. The issue of XGBoost being the stronger predictor indicates that tree-based models can be more appropriate to the current simulation data, in which system dynamics are modeled by structured controller statistics instead of long and high-order temporal relationships.

C. Performance under Varying Task Arrival Rates

To test the robustness with changing traffic intensity, 3, 5, 7, 9, and 11 arrival rates were used and experimented. The findings indicated that the suggested methods were unstable at all the workload levels, and average response time and the SLA violation rate changed mediumly. The comparative performance of the evaluated load balancing methods under varying task arrival rates presented in Table 4.

Table 4. Performance comparison of the evaluated load balancing methods under varying task arrival tasks

Method	Arrival Rate	Avg. Response Time	SLA Violation Rate	Throughput	Fairness Index	Load Imbalance Factor
AI-Driven	3	1.4617	0.0474	2.9390	0.2828	0.1120
	5	1.4988	0.0520	4.8441	0.2828	0.0821
	7	1.5433	0.0583	7.1747	0.2608	0.1513
	9	1.4809	0.0446	9.1500	0.3099	0.1528
	11	1.4134	0.0463	10.8781	0.2384	0.0484
Adaptive-LSTM	3	1.6217	0.0726	2.9390	0.2444	0.0787
	5	1.5248	0.0606	4.8441	0.2893	0.1284
	7	1.6210	0.0657	7.1747	0.2177	0.1550
	9	1.5090	0.0560	9.1500	0.2726	0.1421
	11	1.4508	0.0549	10.8781	0.4021	0.0693
Adaptive-XGBoost	3	1.5314	0.0554	2.9390	0.2787	0.0649
	5	1.5573	0.0617	4.8441	0.2347	0.0726
	7	1.6038	0.0680	7.1747	0.3143	0.0723
	9	1.5333	0.0451	9.1500	0.2657	0.0841
	11	1.4687	0.0543	10.8781	0.3996	0.0484

In the AI-Driven strategy, the average response time was always in the close range between 1.4134 and 1.5433, whereas the violation of the SLA was between 0.0446 and 0.0583. Namely, the AI-based approach generated response times of 1.4617, 1.4988, 1.5433, 1.4809 and 1.4134 in cases of arrival rates of 3, 5, 7, 9 and 11 accordingly. The rates of SLA violation were respectively 0.0474, 0.0520, 0.0583, 0.0446 and 0.0463.

In the case of the Adaptive-LSTM strategy, the means of the mean response time were 1.6217, 1.5248, 1.6210, 1.5090, and 1.4508 and SLA violation rates were 0.0726, 0.0606, 0.0657, 0.0560, and 0.0549 given the fixed arrival rates.

These findings point to the fact that the LSTM-enhanced strategy was practical and stable, but could not be better than the AI-based baseline in terms of delay and deadline compliance.

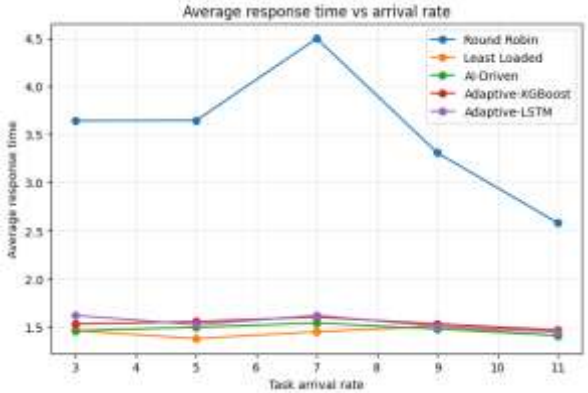

Figure 2. Average Response Time of the evaluated load balancing methods

Figure 2 shows that the AI-driven scheduler consistently maintained the lowest response time across different workload intensities, while the adaptive variants remained competitive under increasing arrival rates. The results also demonstrated that the Adaptive-XGBoost strategy was also competitive at all arrival rates and mostly acted more similar to the AI-oriented one as compared to the LSTM-inspired one. This coincides with the superior individual predictive exactness of XGBoost.

In general, the experiments of arrival rates reveal three significant findings. First, the recommended scheduling paradigm was maintained even with rising level of workloads. Second, AI-driven approach recorded the highest response time and minimum SLA violation rate on average. Third, the XGBoost-based enhancement was more efficient among the adaptive variants than the LSTM-based enhancement, which substantially proved the idea that the prediction quality is strong in terms of downstream scheduling performance. Figure 3 illustrates the confirmation of satisfaction behavior that the AI-driven method. The figure confirms the AI-driven method achieved the lowest SLA violation rate overall (under varying task arrival rates). Whereas the adaptive strategies exhibited slightly higher violation levels under heavier loads.

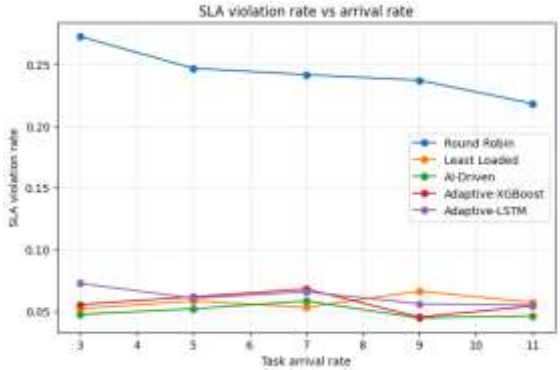

Figure 3. SLA isolation rate of the evaluated load balancing methods

D. Comparison of the Proposed Models

To isolate the effect of adaptive intelligence, three variants, AI-Driven, Adaptive-XGBoost and Adaptive-LSTM, were compared on averages of the results of the arrival-rate experiments. The overall averaged performance of the AI-driven and adaptive variants of the proposed framework is summarized in Table 5.

Table 5. Overall comparison of different variants of the proposed work

Method	Avg. Response Time	SLA Violation Rate	Throughput	Fairness Index	Load Imbalance Factor
AI-Driven	1.4796	0.0497	6.9972	0.2749	0.1093
Adaptive-LSTM	1.5455	0.0619	6.9972	0.2852	0.1147
Adaptive-XGBoost	1.5389	0.0569	6.9972	0.2986	0.0685

The AI-Driven approach had the lowest average time of responding, which is 1.4796, and the lowest SLA violation at 0.0497. In the Adaptive-XGBoost, the average response time was 1.5389 with an SLA violation rate of 0.0569 and an Adaptive-LSTM method generated 1.5455 and 0.0619, respectively. These findings suggest that adaptive intelligence added in the present simulation did not enhance latency or deadline achievement compared to the AI-based control condition of the simulation.

Nevertheless, load distribution quality was increased through the adaptive enhancement. Adaptive-XGBoost strategy showed the best value of the fairness index, 0.2986, and the lowest load imbalance factor, 0.0685, and used in balance-oriented metrics compared with the AI-driven approach and Adaptive-LSTM one. Comparing the two, the strategy based on AI came up with a fairness index of 0.2749 and a load imbalance factor of 0.1093 whereas the Adaptive-LSTM returned 0.2852 and 0.1147.

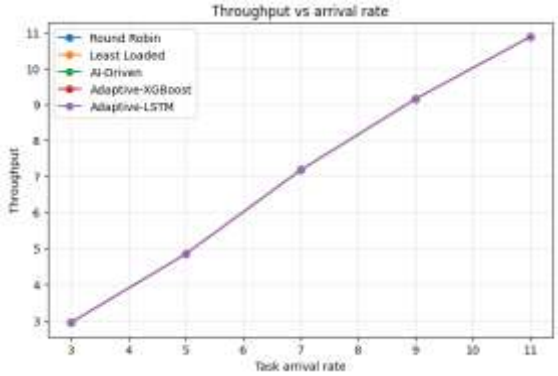

Figure 4. Throughput vs arrival time of the evaluated load balancing methods

Figure 4 illustrates that the task arrival rate grew steadily through all the methods assessed. Nevertheless, the curves almost overlap meaning that the choice of the load balancing strategy did not have a significant impact on throughput in the current simulation setup. Thus, the primary advantage of the suggested framework would be the enhancement of delay, fairness, and load balance as opposed to the growth of raw throughput. These findings indicate that the adaptative XGBoost model was especially successful in evenly redistributing work to the available nodes, although the redistribution failed to translate into the optimal response time. In that way, the experiments demonstrate there was a significant trade-off in that the AI-driven approach was more focused on satisfying the delays and deadlines, and the Adaptive-XGBoost one was more focused on fairness and balance. Figure 5 indicates that the adaptive-XGBoost variant achieved the highest fairness index, demonstrating its ability to distribute workloads more evenly across the infrastructure.

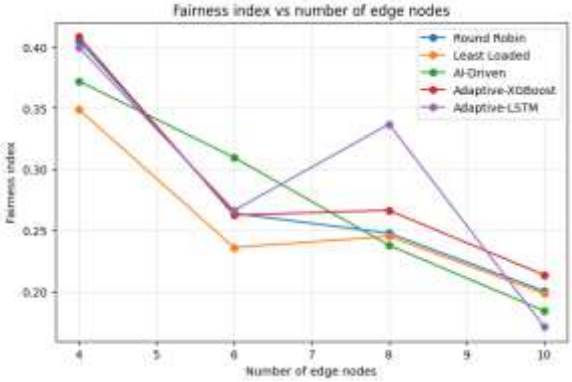

Figure 5. Fairness comparison of the AI-driven and adaptive variants of the proposed load balancing framework

The AI-based strategy provided the highest performance in terms of latency when averaged over all the considered scenarios of arrival rates, with the lowest average response time and SLA violation rate. Conversely, the Adaptive-XGBoost model had the biggest fairness index and the lowest load imbalance factor which means that the infrastructure has managed to distribute the workload in a better fashion. This finding is suggestive of a key trade-off capstone of smart edge-cloud load balancing: predictive adaptation has the potential to increase fairness and balance, but such enhancements may not always result in minimal latency. Consequently, the choice of the AI-based and adaptive strategies can be based on the target deployment focusing on efficiency of the responses or balancing stability over a long period of time.

E. Scalability Analysis

The number of edges nodes was varied in the experiment of scalability in 4, 6, 8, and 10 nodes and the task arrival process was active. The question was to proceed to establish whether the strategies that were proposed had any effect with an increasing size of the edge layer. The findings revealed that all the suggested methods scaled in a non-unstable manner. The larger the edge node count also the more flexibilities to the framework that could place tasks enhanced workload distribution and minimized the possibility of local congestion. This was significant in that SDN-assisted global state view allowed the scheduler to take advantage of the increased candidate pool. The adaptive mechanisms particularly the XGBoost based version showed better balancing behavior as the edge layer grew and this is also expected to be in line with its fairness focused finding on the arrival-rate experiments. Figure 6 illustrates that the proposed framework remained stable and effective as the number of edges increased, demonstrating its suitability for larger edge-cloud deployments.

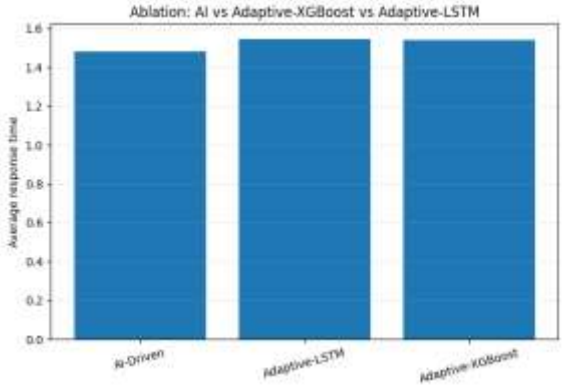

Figure 6. Comparison of AI-driven scheduler and its adaptive enhancements

F. Discussions and Key Findings

The results acquired make a number of valuable observations. To start with, the multi-objective AI-driven scheduler proved quite useful in reducing the response time and deadline missed. It means that the online awareness of the resource utilization, queue conditions, delay, bandwidth and reliability with the help of the controller is adequate to generate powerful reactive balancing choices. Second, adaptive enhancement was advantageous in fairness and quality of load balancing especially in the XGBoost as the predictors. Third, the LSTM model was not as successful as the XGBoost in prediction accuracy or even timeliness, suggesting that the synthetic controller-state information in these experiments is better to structured tabular learning than deep sequential modeling. To isolate the contribution of adaptive intelligence, an ablation comparison of the proposed variants is presented in Table 6.

Table 6. Ablation comparison of different variants of the proposed load balancing framework

Variant	Avg. Response Time	SLA Violation Rate	Throughput	Fairness Index	Load Imbalance Factor
AI-Driven (without adaptive prediction)	1.4796	0.0497	6.9972	0.2749	0.1093
AI-Driven + Adaptive-LSTM	1.5455	0.0619	6.9972	0.2852	0.1147
AI-Driven + Adaptive-XGBoost	1.5389	0.0569	6.9972	0.2986	0.0685

Regarding the system design, these findings are useful as they demonstrate that adaptive intelligence is not to be included in the system just to make it more complex. Rather, its advantage is based on the deployment purpose. The reactive approach based on the AI can be already enough in the case the objective is to reduce the delay and SLA breaches. When it is desired to enhance the fairness and minimize unbalance in the long term on a broader infrastructure, then an adaptive predictor, especially the XGBoost can be beneficial.

The predictor with the best load forecasting was the XGBoost with the estimates of MAE = 0.0258, RMSE = 0.0489 and R^2=0.8576. The AI-based scheduler showed the optimal performance in terms of latency, and the total average response time is 1.4796, SLA violation rate is 0.0497. Adaptive-XGBoost scheduler registered the best balance-oriented results, having the best fairness index of 0.2986 and the lowest load imbalance factor of 0.0685. The Adaptive-LSTM strategy was not any better than the other variants, which was not surprising given that it has lower standalone

prediction performance. Figure 6 depicts that the Adaptive-XGBoost variant achieved the lowest load imbalance factor, highlighting the effectiveness of predictive adaption in improving workload distribution.

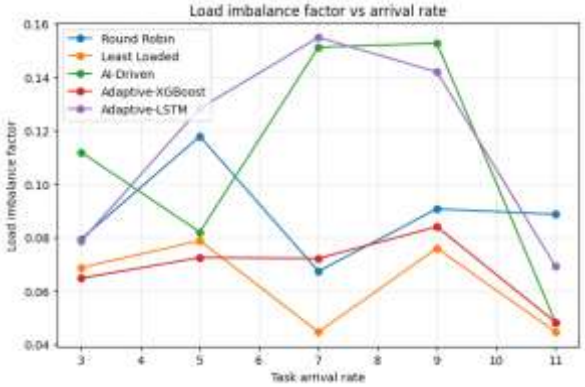

Figure 6. Load balancing factor of the AI-driven and adaptive variants of the proposed framework

To conclude, the experiments prove that the suggested structure is efficient in the intelligent load balancing in SDN-based edge-cloud settings. The AI-based approach demonstrated the highest results in regard to the response time and compliance with the SLA, whereas the Adaptive-XGBoost strategy was more fair and less imbalanced. These outcomes thus confirm both research objectives: the former objective is justified by the good performance of the AI-driven scheduler, and the latter one is justified by the exhibited balancing profit of the adaptive XGBoost enhancement.

CONCLUSION

In the SDN-enabled edge-cloud integrated environments setting, this paper answered two research questions. To begin with, an AI-assisted multi-objective load balancing algorithm was to be suggested, which aims to utilize the SDN controller visibility to manage allocated tasks with edge resources and cloud resources. Second, adaptive intelligence based on machine learning load prediction was added to the proposed algorithm to ensure that balancing in response to dynamic workload conditions can be proactive. Both goals were confirmed by the findings of the experiment. The AI assisted scheduler scored the best latency-optimized performance, which means that real-time multi-objective decision-making through the assistance of controllers is very efficient in minimizing response time, as well as SLA violations. Simultaneously, the measure of adaptive XGBoost-assisted enhancement enhanced equity and lowered load unequalization, which shows the usefulness of predictive knowledge in a more steady workload allocating to non-homogeneous sources. The comparative study also revealed that the quality of prediction is rather important: XGBoost was more appropriate than LSTM in relation to the structured controller-state data provided in this study. In general, the findings suggest that clever load balancing of SDN-enabled edge-cloud systems need to be based on deployment objectives. A reactive AI-based multi-objective scheduler can work decently in case the priority is on minimum delay and compliance with deadlines. Predictive-based scheduling made by adaptive predicting is more beneficial in case long-term stability of the load and equitableness is the priority. This framework can be extended in future work in the form of deep reinforcement learning, SDN coordination among multiple controllers, real Mininet/Ryu implementation, energy-conscious coordination, and resilient scheduling of edge-cloud infrastructures of large scale.

REFERENCES

[1] M. Mahdizadeh, “Task scheduling and load balancing in SDN-based cloud computing: A review of relevant research,” *Array*, vol. 24, 2024.

[2] M. Rostami, “An overview of QoS-aware load balancing techniques in SDN-based IoT networks,” *Journal of Cloud Computing*, 2024.

[3] N. Devi, S. Dalal, K. Solanki, S. Dalal, U. K. Lilhore, S. Simaiya, and N. Nuristani, “A systematic literature review for load balancing and task scheduling techniques in cloud computing,” *Artificial Intelligence Review*, vol. 57, art. 276, 2024.

[4] B. U. Kazi *et al.*, “A survey on software defined network-enabled edge cloud networks: Challenges and future research directions,” *Network*, vol. 5, no. 2, 2025.

[5] S. Sarohe *et al.*, “A systematic and comprehensive survey of load balancing in SDN-IoT,” *Computer Networks*, 2025.

[6] R. S. Alonso, D. Prieto, and J. M. Corchado, “Deep reinforcement learning for the management of software-defined networks and network function virtualization in an edge-IoT architecture,” *Sustainability*, vol. 12, no. 14, art. 5706, 2020.

[7] G. Li *et al.*, “A new load balancing strategy by task allocation in edge computing based on intermediary nodes,” 2020.
[8] Z. Nezami, K. Zamanifar, K. Djemame, and E. Pournaras, “Decentralized edge-to-cloud load balancing: Service placement for the Internet of Things,” *IEEE Access*, 2021.
[9] Y. Dong, G. Xu, M. Zhang, and X. Meng, “A high-efficient joint cloud-edge aware strategy for task deployment and load balancing,” 2021.
[10] Y. Guo and J. Hou, “Software-defined task scheduling strategy towards edge computing,” 2021.
[11] J. Chen *et al.*, “ALBLP: Adaptive load-balancing architecture based on link-state prediction in software-defined networking,” *Wireless Communications and Mobile Computing*, 2022.
[12] J. Chen *et al.*, “ALBRL: Automatic load-balancing architecture based on reinforcement learning in software-defined networking,” *Wireless Communications and Mobile Computing*, 2022.
[13] J. Li *et al.*, “Load balanced data transmission strategy based on cloud-edge-end collaboration in the Internet of Things,” *Sustainability*, vol. 14, no. 15, art. 9602, 2022.
[14] T. Zhang, X. Zhu, and C. Wu, “Reinforcement-learning-based software-defined edge task allocation algorithm,” *Electronics*, vol. 12, no. 3, art. 773, 2023.
[15] Z. Du *et al.*, “A Q-learning-based load balancing method for real-time tasks in edge computing,” *Electronics*, vol. 12, no. 15, art. 3254, 2023.
[16] T. Dreibholz *et al.*, “Towards a lightweight task scheduling framework for cloud and edge platform,” *Internet of Things*, 2023.
[17] A. H. Alhilali *et al.*, “Artificial intelligence-based load balancing in SDN: A comprehensive survey,” 2023.
[18] A. Avan *et al.*, “A state-of-the-art review of task scheduling for edge computing: A delay-sensitive application perspective,” *Electronics*, vol. 12, no. 12, art. 2599, 2023.
[19] M. E. Esmaeili *et al.*, “Reinforcement learning-based dynamic load balancing in edge computing networks,” *Computer Networks*, 2024.
[20] X. Zhou *et al.*, “Deep reinforcement learning-based resource scheduling in SDN-enabled edge computing environments,” *Computer Networks*, 2024.
[21] C. Li *et al.*, “Deep reinforcement learning based controller placement and edge selection in SDN-based multi-access edge computing,” 2024.
[22] M. D. Tache *et al.*, “Optimization algorithms in SDN: Routing, load balancing, and traffic engineering,” *Applied Sciences*, vol. 14, no. 14, art. 5967, 2024.
[23] X. He *et al.*, “Multi-objective reinforcement learning for adaptive load balancing,” *Computer Networks*, 2025.
[24] M. Shahzad *et al.*, “Fair switch selection for large scale software defined networks,” *Telecommunication Systems*, 2025.
[25] R. Farahi *et al.*, “A comprehensive overview of load balancing methods in software-defined networking,” 2025.