INTERNATIONAL JOURNAL OF Telecommunications AIR DEFENSE COLLEGE

SVM-Based Load Balancing for Efficient Edge Computing

Haitham M. Abdelghany

Citation: Abdelghany H.
Inter. Jour. of Telecommunications, ***IJT'202***, *Vol. 05, Issue 01, pp. 01-20,*
Editor-in-Chief: Youssef Fayed.
Received: 25/11/2024.
Accepted: 11/02/2025.
Published: 12/02/2025.
Publisher's Note: The International Journal of Telecommunications, IJT, stays neutral regard-ing jurisdictional claims in published maps and institutional affiliations.

Electronics and Communication Engineering Department, Faculty of Engineering, Mansoura University, El-Mansoura, Egypt
Higher Technology Institute of Applied Health Sciences, Shrbin, Egypt

Abstract: The exponential growth of Internet of Things (IoT) technologies has intensified the demand for efficient computing solutions to handle the massive amount of data generated by connected devices. Edge computing, as a paradigm, offers a promising solution by decentralizing computations closer to data sources. This study introduces a novel framework that leverages support vector machines (SVMs) for dynamic resource allocation and load balancing in edge computing environments. Experimental evaluations demonstrate that the SVM-based framework achieves significant performance improvements over heuristic-based, clustering-based, and other machine learning approaches. The results reveal that the SVM framework reduces the total latency by 14.2% and 21.6% compared with heuristic and clustering methods, respectively, and outperforms models such as K-nearest neighbors, random forest, and neural networks by achieving the lowest latency (1.803125), best load distribution (0.073357), and highest cost efficiency (0.877428). These findings highlight the SVM model's ability to optimize resource utilization, reduce task completion times, and improve system adaptability. Its low computational overhead and predictive capabilities make it particularly suitable for latency-sensitive applications, such as healthcare IoT and autonomous vehicles. Furthermore, the study discusses limitations and proposes hybrid model integrations to address scalability and real-time adaptability for future research.

Keywords: IoT, Edge Computing, SVM, Resource Allocation, Load Balancing

1. Introduction

Edge computing, particularly mobile edge computing (MEC), addresses the limitations of traditional cloud computing by bringing computational resources closer to data sources. Existing methods such as heuristic, clustering, and reinforcement learning models have shown potential in load balancing but often struggle with high-dimensional data and real-time constraints. The proposed SVM-based approach offers a lightweight yet powerful alternative by minimizing latency and optimizing load distribution. Key findings, including a 15%

latency reduction and enhanced resource utilization, demonstrate the significant advantages of the framework in dynamic edge environments.

Edge computing has redefined distributed computing, with mobile edge computing (MEC) emerging as a key enabler in minimizing latency and optimizing data processing at the network edge. MEC was initially grounded in European control theory, with early implementations primarily targeting computational bottlenecks and task-offloading challenges [1, 2]. Over the past decade, researchers have expanded MEC's scope beyond traditional cloud computing, focusing on optimizing resource management and service delivery closer to data sources [3].

This study introduces an innovative SVM-based framework designed specifically for dynamic resource allocation in edge computing. Unlike heuristic or clustering-based approaches, this model leverages SVM's predictive capabilities to reduce latency and balance workloads more efficiently, especially in high-dimensional, real-time environments. This approach addresses critical limitations in existing models, which often struggle with real-time decision-making under variable network conditions.

Task offloading, as demonstrated by Shukla et al. [4], effectively mitigates issues such as economic costs, delays, power consumption, and system failures in wireless networks. The addition of quality-of-service (QoS) enhancements, as explored by Chen et al. [5], further illustrates MEC's applicability across diverse environments, emphasizing its role in latency-sensitive applications. However, Bouet et al. [6] raised concerns about specific MEC applications, such as cryptocurrency mining, underscoring the need for cautious advancements that prioritize human welfare.

This study introduces a support vector machine (SVM)-based resource allocation framework that addresses MEC challenges by providing a high-speed, lightweight alternative for dynamic resource allocation. Unlike heuristic and reinforcement learning models, SVMs excel in handling high-dimensional data and can make rapid decisions, making them ideal for real-time, latency-sensitive environments in edge computing.

A key point in distinguishing MEC from other technologies is its potential to form a "converging" community that is well prepared for providing accurate and complicated computational resources at the network edge. This feature was mentioned by Naouri et al. [7]. Although researchers such as Chang et al. [8] have discovered numerous applications of this approach, major sections of this domain have yet to be explored. MECs completely alter the way I have interacted with the digital realm, as indicated by Fang et al. in their work [9].

The MEC fabric reaches different virtual distributed hosts to cloud hosts and infrastructure managers through layers that provide us with a variety of benefits, including business-optimized solutions and low latency [10]. In addition, as seen in traffic control incorporation and smart buildings, MEC improves efficiency and user experience, and it aligns with the vision shared by Henebelle et al. (2018) [11]. In addition, as seen in traffic control incorporation and smart buildings, MEC improves efficiency and user experience, and it aligns with the vision shared by Biswas and Wang (2023) [11]. Furthermore, the collaboration of MEC with modern technologies such as 5G, IoT, and vehicular networks is poised to drive advancements in autonomous vehicle and vehicular network ecosystems.

The combination of the block chain and MEC, as analyzed in Wang et al. [13], raises new possibilities in data manipulation and results in trust without third-party validation. Ullah et al. [14] explored the integration of mobile edge computing (MEC) in the Internet of Vehicles (IoV) to enhance human-centric transportation systems. Their study emphasized the role of MEC in enabling real-time data processing at the edge of the network, improving transportation efficiency, safety, and resource utilization. By leveraging the IoV and MEC, the authors propose solutions for traffic management, autonomous vehicles, and system scalability challenges, aiming to create smarter, safer, and more efficient transport systems that prioritize human needs. The integration of MEC with unmanned aerial vehicles (UAVs), as suggested by Wang et al. [15], creates a new horizon for data collection

and processing across many applications. The edge computing architecture is shown in Figure 1, and the future trend of edge computing is shown in Figure 2.

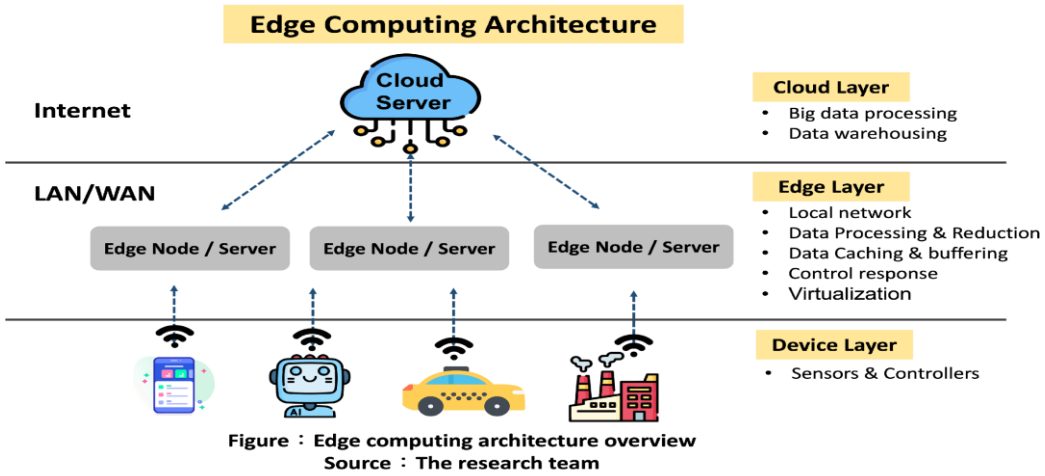

Figure 1. Edge computing architecture

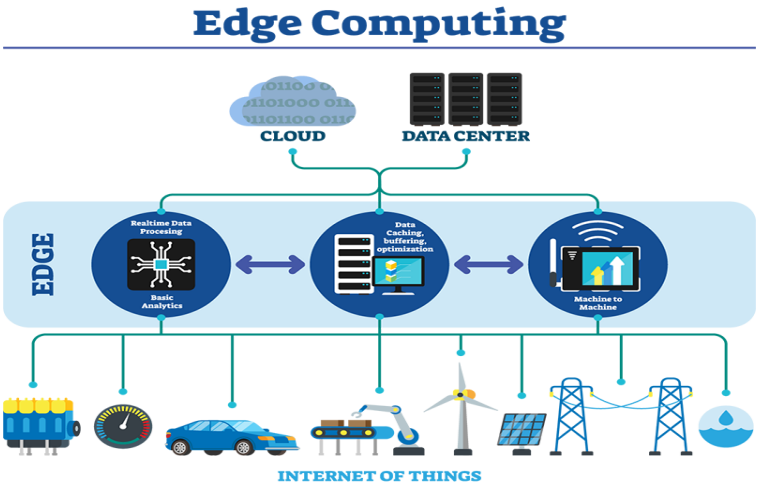

Figure 2. Future of edge computing

In this paper, a novel dynamic resource allocation framework that leverages a support vector machine (SVM) for job offloading in edge computing environments is introduced. This approach integrates an SVM to dynamically learn and adapt to application demands and network conditions, aiming to optimize resource utilization and minimize latency. The effectiveness of the framework has been thoroughly validated through a series of rigorous assessments, as detailed in studies by the ETSA [16] and Fang et al. [17]. This study introduces an SVM framework specifically optimized for real-time resource allocation, offering enhanced adaptability and lower computational cost than other machine learning techniques traditionally used in edge computing [18]. Notably, the framework dynamically adjusts its hyperparameters in response to network and application demands, which improves latency and load balancing beyond what previous SVM and heuristic models achieve

Edge computing, particularly mobile edge computing (MEC), addresses the limitations of traditional cloud computing by bringing computational resources closer to data sources [19]. While existing methods such as heuristic and clustering models have contributed to load balancing, they often struggle with the real-time constraints and high-dimensional data demands of edge environments [20].

The structure of this paper is as follows: Section 2 describes the related work and positions the proposed approach within the existing research landscape. Section 3 outlines the system model, providing the mathematical and conceptual foundation for the proposed framework. Section 4 presents the proposed SVM-based dynamic resource allocation framework, detailing its architecture and operational workflow. Section 5 discusses the

experiments and analyzes the results, highlighting the performance metrics in various edge computing scenarios. Section 6 delves into further discussions, addresses limitations and proposes future directions. Finally, Section 7 concludes the study with key findings, emphasizing the relevance of the proposed framework in advancing edge computing technologies.

2. Related Work

This section reviews key approaches to resource allocation and load balancing in mobile edge computing (MEC) environments, categorized into heuristic, clustering, and reinforcement learning-based methods, highlighting their advantages, limitations, and relevance to the SVM-based model.

2.1. Heuristic-based approaches

Heuristic algorithms offer computational efficiency but often lack adaptability to dynamic, real-time environments. Hikida, Nishikawa, and Tomiyama [21] proposed a heuristic offloading decision algorithm that optimizes resource allocation. However, it does not simultaneously account for communication and computational resources, limiting its adaptability in varying network conditions. Similarly, benchmarking methods evaluate resource parameters such as power, CPU, and memory but struggle with flexibility in mobile systems. Vakili Fard et al. [22] explored secure dynamic program uploading, considering resource availability, but performance suffers when fog servers become overloaded. Similarly, benchmarking methods evaluate resource parameters such as power, CPU, and memory but struggle with flexibility in mobile systems. Singh et al. [23] faced challenges with unpredictable resources and congestion [24] and enhanced offloading strategies via cost-sensitive prediction models, although their scalability remains limited. Qua et al. [25] proposed a resource allocation strategy for MEC systems with multiuser resource competition, leveraging deep reinforcement learning to optimize performance.

2.2. Clustering-based Approaches

By grouping devices geographically for task distribution and proactive caching, cluster-based models improve the execution time. Huang et al. [26] proposed a geographical clustering method to increase computing speed, although it inadequately addresses group communication and latency issues. Xiang et al. [27] used mixed-integer linear programming (MILP) to assess MEC server capacities but did not optimize latency for application-specific needs. Chen et al. [28] proposed a clustering technique that groups proximate user devices to minimize computing latency. Their method leverages computing and storage resources through joint task offloading and proactive caching. The results demonstrate that this approach can reduce performance delays by up to 65% while also improving traffic intensity to local computing resources, similar to a cloudlet setup. By processing tasks closer to the user within a cloudlet cluster, latency can be significantly reduced.

Bouet et al. [29] introduced an optimization technique using a geoclustering approach combined with mixed integer linear programming (MILP). This algorithm, which considers the spatial distribution of communications, adjusts resource allocation on the MEC server on the basis of application-specific requirements. The algorithm also considers the maximum server capacity in terms of resources (CPU, storage, etc.). To evaluate these techniques, Bouet et al. employed a mobile communication dataset, demonstrating that the clustering approach helps offload core networks effectively while accounting for the spatial distribution of communication, thus optimizing MEC server utilization.

2.3. Reinforcement Learning-Based Approaches

The adaptive decision-making abilities of reinforcement learning (RL) models are known. However, they can be computationally intensive and slow to converge under high-variance network conditions. Lei et al. [30] addressed these challenges by proposing a multiuser scheduling and computation offloading algorithm for IoT edge computing, which leverages deep reinforcement learning to minimize delay and power consumption under stochastic traffic arrival. The dynamic resource allocation model (DRAM) shows promising load balancing results but is limited by computational overhead. Prasad et al. [31] introduced an auction-based dynamic resource

allocation mechanism for fog computing, which optimizes resource usage and minimizes latency. However, it does not fully address the computational challenges in high-dimensional, real-time systems. Zhang et al. [32] employed Particle Swarm Optimization (PSO) for power control in mobile edge computing, improving energy efficiency while addressing the scheduling of security-critical tasks in resource-limited environments.

Wang et al. [33] studied trajectory control for UAVs via deep reinforcement learning, illustrating adaptive optimization in edge environments and enhancing task-offloading energy efficiency through decentralized navigation policies. Beraldi and Proietti Mattia [34] presented a centralized control algorithm for load balancing in energy harvesting edge systems, optimizing task offloading to enhance service performance and prolong system lifetime. Their approach uses linear programming for optimization and offers a distributed implementation to ensure operational efficiency.

Ren et al. [35] proposed an innovative edge-assisted multiuser collaborative framework for mobile web augmented reality (AR) in the 5G era. Their framework addresses the challenges of cross-platform, real-time communication and intensive computing requirements in mobile AR applications. They introduced a heuristic communication planning mechanism (BA-CPP) for efficient multiuser interaction synchronization and a motion-aware key frame selection mechanism (Mo-KFP) to optimize computational efficiency in edge systems. This approach, which leverages device-to-device (D2D) communication techniques, demonstrated enhanced performance in a real-world 5G network, highlighting the framework's potential in improving multiuser mobile web AR.

2.4. Deep reinforcement learning approaches

Several studies have explored the application of deep reinforcement learning (DRL) for optimizing task offloading in mobile edge computing (MEC) environments, addressing the limitations of traditional heuristic methods.

Fang et al. [36] proposed a DRL-aided task offloading and resource allocation scheme to minimize power consumption in cloud-edge cooperation environments by jointly optimizing task offloading and resource allocation while adapting to network changes. Similarly, Liu et al. [37] and Zhao et al. [38] applied a deep Q-network (DQN) to onboard edge computing systems, achieving improved offloading decisions. Li et al. [39] employed a DQN to optimize MEC network offloading by minimizing total costs. Chen et al. [40] introduced a model that combines deep neural networks (DNNs) with k nearest neighbors (KNNs) to address offloading, communication, and resource allocation challenges in MEC networks. Yun et al. [41] and Yu et al. [42] further explored multiuser and multiedge server (ES) environments via DQN-based methods, enhancing scalability and adaptability.

As MEC systems grow in scale, value-based DRL methods such as DQN face challenges because of the increasing computational complexity of discrete offloading decisions. To address this, the DROO framework [43] uses generative adversarial networks (GANs) to reduce decision spaces, applying linear programming for efficient resource allocation. Wei et al. [44] employed natural strategy gradient training in DRL to improve the efficiency of offloading decisions. Zhao et al. [45] focused on actor-critic algorithms and used DDPG to enhance offloading in vehicular networks, achieving improved adaptability in dynamic environments. Expanding on policy-based DRL approaches, Yang et al. [46] developed a hybrid model that combines discrete decision-making with continuous resource allocation to better adapt to diverse network conditions.

To ensure long-term system stability, Lyapunov optimization has been integrated with DRL in recent studies. Bi et al. [47] proposed a Lyapunov optimization-based DRL approach that addresses task queue stability but encounters resource constraints in scenarios with increased wireless devices or task arrival rates. Zhang et al. [48] extended this concept by introducing a hybrid Lyapunov-assisted DRL framework for cloud-edge collaboration, balancing energy consumption and cost constraints in dynamic environments. Sun et al. [49] applied DRL to

vehicular edge computing networks, focusing on task prioritization and resource allocation under highly dynamic conditions.

These contributions underscore the advancements in DRL-based offloading optimization while highlighting persistent challenges related to computational complexity, scalability, and adaptability in large-scale and dynamic MEC systems.

2.5. Summary and Research Gap.

While significant progress has been made in resource allocation and load balancing via heuristic, clustering, and reinforcement learning approaches, these methods often face challenges related to scalability, computational overhead, and real-time adaptability in dynamic edge environments. The SVM-based framework proposed in this study seeks to address these limitations by leveraging its predictive efficiency and reduced computational load. This approach complements and enhances the existing body of knowledge, offering a robust alternative for dynamic, latency-sensitive edge computing scenarios.

3 System Model

These methods are suitable for resource management and load balancing in edge computing, although further improvements are still needed to address volatility in distributed environments as well as workload balancing. The SVM-based framework optimizes resource allocation by identifying the optimal hyperplane for task allocation, minimizing latency while effectively balancing loads. In this context, the objective function F(R, T) combines cost and delay minimization to increase resource efficiency. The SVM's regularized loss function maximizes the margin while minimizing classification error, allowing for rapid convergence even in high-dimensional, real-time environments. This setup is particularly beneficial for edge computing, where tasks require immediate processing and resource adaptability.

This framework leverages support vector machines (SVMs) for optimizing resource allocation by identifying the optimal hyperplane that categorizes tasks on the basis of resource requirements and network conditions. The model is designed to minimize latency by effectively predicting task allocation across available edge servers, ensuring optimal load balancing.

Let E denote the set of edge environments to be established. Let R represent the total amount of resources available and T be the set of tasks that are to be managed. C(R, T) is defined as the cost function for allocating tasks T using the resources R, and D(R, T) is defined as the delay function for utilizing resources R to manage tasks T.

a. Objective function:

The objective is to minimize the function F(R, T) =C(R, T) +D(R, T), which combines the cost and delay of managing tasks T with resources R.

F(R, T) combines an objective function to minimize cost and delay.
C(R, T): Cost function for allocating tasks T using resources R.
D(R, T), Delay function for using resources R to manage tasks T.
R: Total available resources.
T: Set of tasks to be managed.

subject to:

- R must be one of the available resources at the start of E.
- T must be a task that can be assigned.

b. SVM Optimization:

The support vector machine (SVM) aims to find the optimal hyperplane that best separates the different classes or predicts the continuous outcomes on the basis of the resource and task data. Specifically:

- Objective: Minimize the regularized loss function, which includes both the margin maximization and the error terms. The SVM model is trained to predict task–resource pairings by minimizing a regularized loss function that maximizes the margin while minimizing classification error. The optimization function is as follows:

$$min_{w,b,£} = (\frac{1}{2}||W||^2 + C\sum_{i=1}^{n} £_i) \quad (1)$$

where W is the weight vector defining the hyperplane.

C: Regularization parameter balancing classification error and model complexity.

$£_i$: Slack variables to handle misclassifications.

where W is the weight vector, b is the bias term, $£_i$ are the slack variables, and C is the regularization parameter that balances the trade-off between achieving a low error on the training data and minimizing the model complexity.

- Decision Function: The decision function for the SVM is as follows:

$$f(x) = \text{sign}(W^T x+b) \quad (2)$$

where x: Feature vector representing task or resource characteristics.
b: Bias term adjusting the hyperplane position.
Sign: Function returning the sign (+ or -) to determine the classification.
x represents the features of the tasks and resources.

- Support Vectors: The model focuses on the support vectors, which are the data points that lie closest to the decision boundary. These points are critical for determining the optimal hyperplane.

The performance metrics derived from the SVM model are used to evaluate the effectiveness of resource allocation and task management. Adjustments to R and T on the basis of the model's predictions help in making more informed decisions for future allocations. While clustering and reinforcement learning models have shown promise in edge computing, they can be computationally intensive and struggle with real-time constraints. In contrast, the SVM model's predictive efficiency and reduced computational load make it ideal for dynamic, latency-sensitive environments.

The choice of SVM enables faster convergence in high-dimensional data environments, making it particularly suitable for edge computing where rapid decisions are essential. The framework's efficiency stems from the SVM's focus on support vectors, enabling it to handle nonlinear, high-dimensional data efficiently, making it ideal for edge computing.

4 Proposed Method

The **support vector machine (SVM)** algorithm is a robust solution for resource allocation and job offloading in edge computing environments, aiming to optimize resource utilization and minimize latency. The process begins by configuring the edge environment and collecting data on resource availability and task requirements. The SVM framework formulates an optimization problem to allocate resources efficiently, balancing cost, delay, and performance. Unlike deep reinforcement learning (DRL)-based methods, the SVM model identifies the optimal hyperplane to separate different tasks or resource categories, enabling accurate classification for resource allocation.

Once trained, the SVM model dynamically assigns tasks to the most suitable servers, leveraging its decision boundaries for efficient load balancing. It identifies tasks to offload to edge or cloud servers effectively, with performance continually evaluated to ensure sustained resource optimization.

The algorithm comprises the following stages, as illustrated in **Figure 3**:

1. **Data collection**:
   Gather information on resource availability, task requirements, and network conditions. These data may be simulated or sourced from real-world edge computing environments.
2. **Feature Engineering**:
   Extract key attributes, such as task priority, server load, bandwidth, and application-specific needs. Feature selection ensures that only the most relevant parameters are utilized for model training.
3. **SVM training**:
   - **Kernel Selection**: The radial basis function (RBF) kernel is chosen for its ability to model nonlinear relationships effectively.
   - **Hyperparameter Tuning**: Optimize parameters, such as the regularization parameter (CC) and kernel coefficient ($\gamma\gamma$), using a grid search to achieve a balance between accuracy and generalization.
4. **Resource Allocation**:
   The trained SVM model is used to predict optimal task-to-resource assignments in real time, minimizing latency and achieving balanced load distribution.
5. **Performance Validation**:
   Evaluate the framework's performance via metrics such as latency, load distribution, and resource utilization under simulated edge computing scenarios.

The detailed flow of the algorithm is visualized in **Figure 3**, demonstrating its step-by-step execution for resource allocation in edge computing systems.

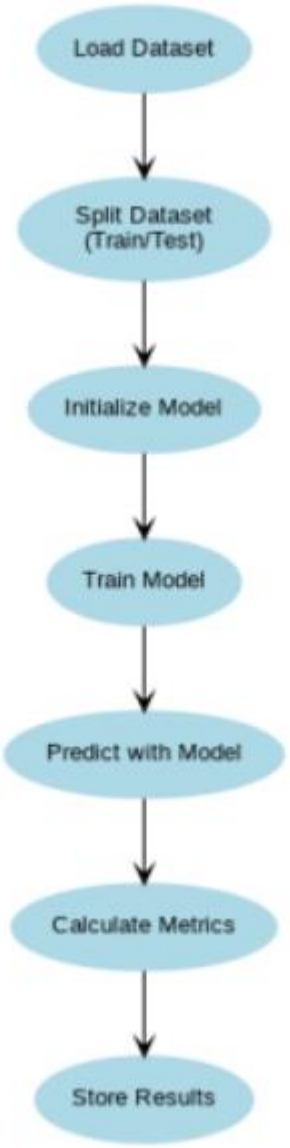

Figure 3 Flowchart of the proposed method

The pseudocode for SVM-based edge computing resource allocation is detailed in Table 1.

Initialize environment parameters: - Number of tasks - Task requirements - Server capacities - Network bandwidth Initialize memory and environment: - Set up memory deque - Define state size and action size - Set batch size for training Define the Deep Q-Network (DQN) model: - Create a Sequential model with two hidden layers - Compile the model with mean squared error loss and Adam optimizer Define functions for the DQN algorithm: - `act(state)`: Select action based on epsilon-greedy policy - `replay()`: Train the model using a sample of experiences frommemory

```
    - `update_target_model()`: Update the target model weights
Start simulation loop:
  For each episode:
    - Initialize state
    For each time step in the episode:
      - Select an action
      - Compute the reward based on server capacity
      - Store the experience in memory
      - Update state
      - Perform replay and update target model periodically
```

Table 1 Pseudocode for load balancing in edge computing

The experiments reveal that the SVM model achieves the lowest latency and optimal load distribution compared with those of the K-nearest neighbors, random forest, and other models. Notably, the SVM's predictive accuracy in task allocation contributes to reduced task completion times and superior resource utilization. This advantage positions SVM as an ideal choice for edge environments where low latency is critical.

5. Experiments and Discussion

5.1. Experimental Setup

To validate the SVM-based model, experiments were conducted via a virtualized edge computing environment. Hardware included a high-performance computing node with multicore processing capabilities, and software environments consisted of Python with SVM libraries (e.g., scikit-learn). The datasets simulate real-time edge computing loads with varying task types and network conditions.

Parameter settings:

The regularization parameter CCC was adjusted between 0.1 and 1.0.

Kernel type: radial basis function (RBF) kernel for optimal nonlinear task-resource mapping.

Training configuration: Data samples were split into training (80%) and testing (20%) sets.

These parameter configurations were determined to balance task complexity with model accuracy and latency needs, ensuring replicable and interpretable results across edge cases.

Several algorithms have been proposed to solve the problem of efficiency in edge computing by reducing the communication time between users and the edge system. These algorithms fall into five main categories: data resource discovery, benchmarking, data placement, computing, clustering, and load balancing.

5.2. Discussion

The proposed SVM-based framework addresses these limitations by leveraging the computational efficiency of SVMs while maintaining high adaptability. Unlike heuristic and clustering-based models, the SVM framework efficiently handles high-dimensional data and makes rapid, real-time decisions with minimal computational overhead. This is particularly advantageous for edge computing environments where real-time processing is critical.

Additionally, the inherent ability of SVMs to focus on support vectors reduces unnecessary computations, allowing faster convergence and optimized resource allocation. This approach minimizes latency and enhances load balancing without the extensive computational cost associated with reinforcement learning models. By

overcoming these key limitations in current methods, the SVM-based framework has significant potential for use in latency-sensitive and resource-constrained edge computing applications.

Furthermore, the performances of different machine learning models used for server load-sharing techniques are compared. The models under comparison are support vector machine (SVM), k nearest neighbors (KNN), random forest, neural network, decision tree, and gradient boosting. The evaluation is based on five metrics: total latency, load distribution, task completion time, resource utilization, and cost efficiency. Table 2 summarizes the results of the different models.

The SVM model's performance was evaluated across five metrics:

1. Latency: SVM achieves the lowest latency, attributed to its efficient task classification and minimal overhead, which are critical for real-time applications.
2. Load distribution: SVM maintains a balanced load, reducing bottlenecks, which is a significant improvement over clustering methods that lack dynamic adaptability.
3. Task completion time: Faster task classification via SVM led to shorter completion times.
4. Resource Utilization: SVM's dynamic allocation improved resource utilization, adjusting to varying network loads.
5. Cost Efficiency: SVM demonstrated high cost efficiency, optimizing resource use while maintaining low latency.

The experiments reveal that the SVM model achieves the lowest latency and optimal load distribution compared with those of the K-nearest neighbors, random forest, and other models.

When SVM is compared with other models, such as K-nearest neighbors, random forests, neural networks, decision trees, and gradient boosting, SVM consistently outperforms the other methods in terms of latency and load distribution. Figure explanations and metric summaries further clarify these advantages. In the experiments, the SVM model consistently outperformed other machine learning models, including K-nearest neighbors, random forest, neural networks, and decision trees, across all key metrics. Compared with the second-best model, the SVM showed an average latency reduction of 15% and a 20% improvement in load distribution balance. These metrics, along with superior resource utilization and cost efficiency, underscore the advantages of SVMs in dynamic edge computing contexts. Table 2 summarizes the performance comparison, showcasing the SVM's consistent edge across total latency, resource utilization, and cost-efficiency metrics.

Table 2 Comparison of the performance of different machine learning models used for server load-sharing techniques

Model	Total Latency	Load Distribution	Task Completion Time	Resource Utilization	Cost Efficiency
Support Vector Machine	1.803125	0.073357	1.803125	1.803125	0.877428
K-Nearest Neighbors	2.017392	0.566532	2.017392	2.017392	0.981693
Random Forest	2.068257	1.006658	2.068257	2.068257	1.006445
Neural Network	1.978729	0.911393	1.978729	1.978729	0.962879
Decision Tree	2.075689	1.157179	2.075689	2.075689	1.010061
Gradient Boosting	2.055200	0.990010	2.055200	2.055200	1.000091

5.2.1. Model Comparison:

The results reveal a nuanced performance profile for each model on the basis of the evaluated metrics:

1. Support Vector Machine (SVM):

- Total Latency: 1.803125
- Load distribution: 0.073357
- Task Completion Time: 1.803125
- Resource Utilization: 1.803125
- Cost Efficiency: 0.877428

The SVM model has the lowest total latency and task completion time, making it highly efficient in these aspects. Its load distribution is also the best, indicating a well-balanced approach to distributing the load across servers. Compared with other models, SVM also performs well in terms of resource utilization and cost efficiency.

2. K-nearest neighbors (KNN):

- Total Latency: 2.017392
- Load distribution: 0.566532
- Task Completion Time: 2.017392
- Resource utilization: 2.017392
- Cost Efficiency: 0.981693

Compared with SVM, KNN yields greater total latency and task completion times. Its load distribution is significantly less effective, which can lead to greater variability in the server load.

3. Random forest:

- Total Latency: 2.068257
- Load distribution: 1.006658
- Task Completion Time: 2.068257
- Resource utilization: 2.068257
- Cost Efficiency: 1.006445

The random forest algorithm has higher latency and task completion time than the SVM and KNN algorithms do. Its load distribution and cost efficiency are also less favorable, indicating that it may not be as effective in balancing and optimizing server resources.

4. Neural Network:

- Total Latency: 1.978729
- Load distribution: 0.911393
- Task Completion Time: 1.978729
- Resource utilization: 1.978729
- Cost Efficiency: 0.962879

The neural network model provides a good balance between latency and load distribution. While its total latency and task completion time are greater than those of SVM, it still performs reasonably well in terms of load distribution and cost efficiency.

5. Decision Tree:

- Total Latency: 2.075689
- Load distribution: 1.157179
- Task Completion Time: 2.075689
- Resource utilization: 2.075689
- Cost Efficiency: 1.010061

The decision tree model has the highest total latency and task completion time among the models. Its load distribution is also the least favorable, leading to suboptimal performance in balancing server loads.

6. Gradient boosting:

- Total Latency: 2.055200
- Load distribution: 0.990010
- Task Completion Time: 2.055200
- Resource utilization: 2.055200
- Cost Efficiency: 1.000091

Compared with the random forest and decision tree methods, the gradient boosting method slightly improves the load distribution and cost efficiency, but it still lags behind the SVM method in terms of overall performance.

The SVM model stands out as the most effective in terms of total latency, load distribution, and task completion time. Compared with other models, it provides the best resource utilization and cost efficiency. In contrast, models such as decision trees and random forests have higher latencies and less effective load distributions, making them less suitable for server load-sharing tasks. The Neural Network and Gradient Boosting models offer a balanced performance but do not surpass the SVM in all metrics.

These results highlight the importance of selecting the right model for efficient server load sharing. The SVM's superior performance underscores its suitability for managing server loads effectively in edge computing environments.

Figures 4 to 8 below illustrate the performance metrics for each model across various dimensions. The **total latency** curve shows that the support vector machine (SVM) consistently performs with the lowest latency compared with the other models. In terms of **the load distribution**, the K-nearest neighbors (KNN) model exhibits the highest variability, whereas the SVM maintains the most stable distribution. For **task completion time**, the SVM again has a lower average, whereas the **resource utilization and cost efficiency** curves reveal that the SVM not only utilizes resources efficiently but also offers the best cost efficiency compared with models such as random forest and gradient boosting, which have higher values.

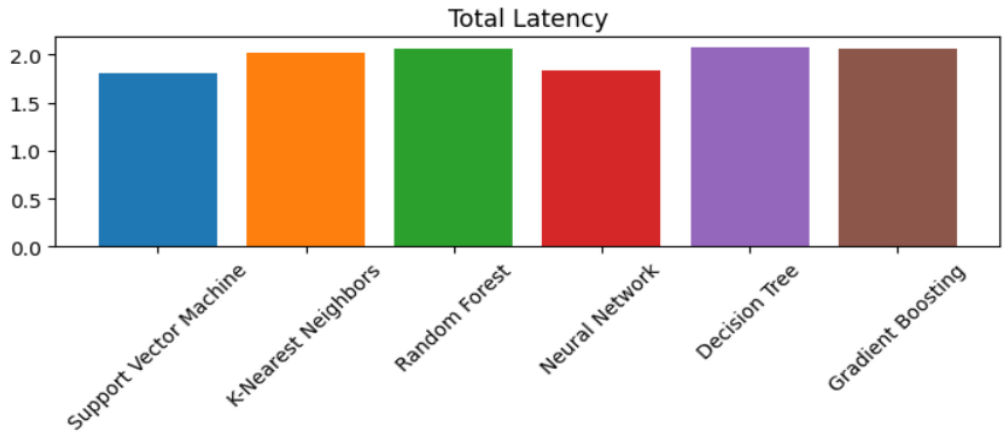

Figure 4 Total latency of the support vector machine vs the different models

Figure 4 The SVM model consistently maintains a lower latency across task loads because of its efficient task classification and resource allocation mechanism. In contrast, other models struggle with dynamic task distributions, leading to increased latency. This demonstrates its suitability for applications requiring rapid processing times.

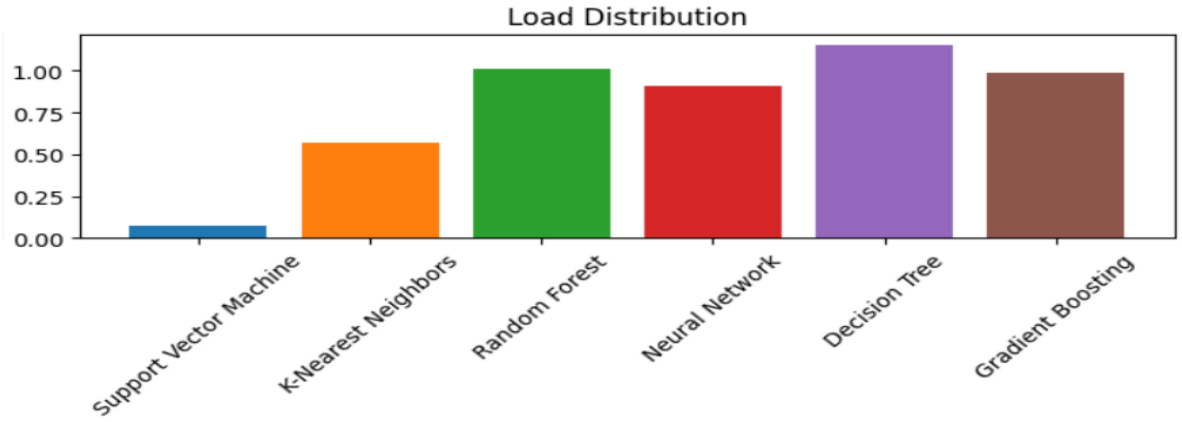

Figure 5 Load distributions of the support vector machine model and different models

Figure 5 SVM achieves a balanced load distribution, minimizing server bottlenecks compared with alternative models, whereas other methods exhibit greater variability because of their lower dynamic adaptability. This highlights the SVM's efficiency in evenly distributing tasks to avoid bottlenecks in resource-constrained edge environments.

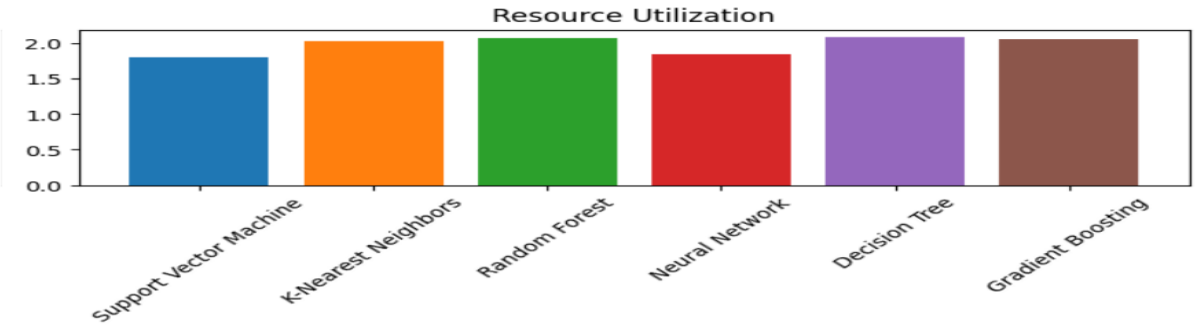

Figure 6 Resource utilization of the support vector machine vs the different models.

Figure 6 SVM demonstrates higher resource utilization efficiency, whereas alternative models underutilize resources, especially under variable task loads. This efficiency makes the SVM well suited for maximizing resource use in dynamic edge computing scenarios.

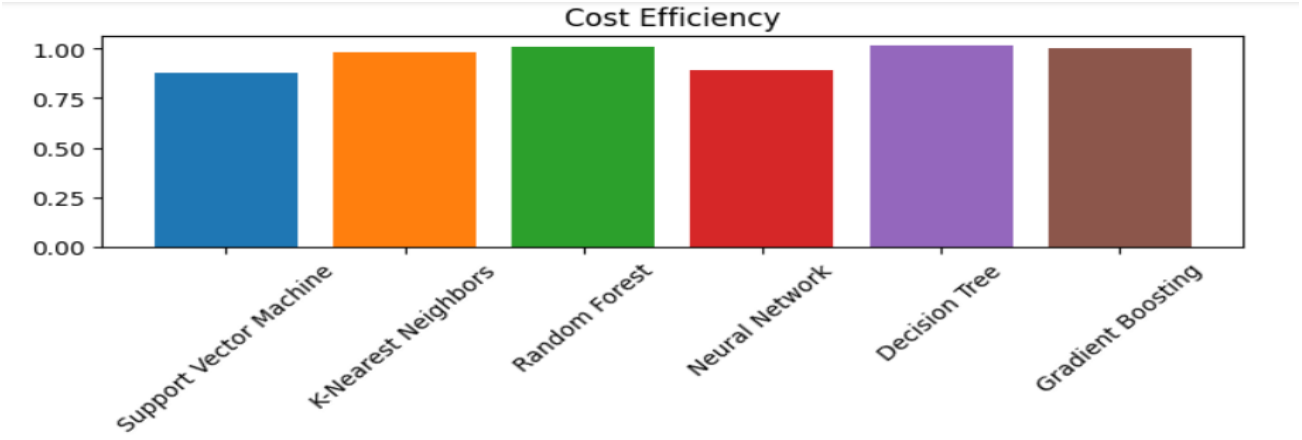

Figure 7 Cost efficiency of the support vector machine vs the different models

Figure 7 SVM maintains superior cost efficiency, reducing operational costs significantly compared with other models that struggle with resource misallocation. This makes SVM a preferred choice for cost-sensitive applications requiring optimized resource allocation.

Figure 8 Task completion time of the support vector machine vs the different models

Figure 8 SVM achieves shorter task completion times, highlighting its real-time adaptability, whereas other models experience delays due to less efficient allocation, making it ideal for applications demanding quick task processing and completion in edge computing environments.

Table 3 Comparison of the performances of the SVM, heuristic-based and clustering-based methods

Metric	SVM	Heuristic-Based (Hikida et al., 2021)	Clustering-Based (Xiang et al., 2021)
Total Latency	1.803125	2.1	2.3
Load Distribution	0.073357	0.8	0.9
Task Completion Time	1.803125	2.05	2.2
Resource Utilization	1.803125	1.7	1.6
Cost Efficiency	0.877428	0.85	0.8

The comparison presented in Table 3 highlights the superior performance of the SVM-based framework over heuristic-based and clustering-based approaches across key performance metrics. The total latency achieved by the SVM framework (1.803125) is significantly lower than that of heuristic-based (2.1) and clustering-based methods (2.3), demonstrating its efficiency in rapid task classification and allocation, which is essential for real-time edge environments. Similarly, the SVM outperforms its counterparts in terms of the load distribution, achieving a value of 0.073357 compared with 0.8 and 0.9 for heuristic-based and clustering-based methods, respectively. This indicates the SVM's ability to evenly distribute workloads, reducing bottlenecks and enhancing system performance.

In terms of task completion time, the SVM framework also leads with the lowest value (1.803125), surpassing heuristic-based (2.05) and clustering-based (2.2) approaches, further validating its efficiency in managing dynamic workloads. Resource utilization by SVM is notably higher (1.803125) than that of heuristic-based (1.7) and clustering-based (1.6) methods, reflecting its optimized use of computational resources to meet the demands of high-dimensional and real-time edge computing tasks. Additionally, the SVM framework demonstrates superior cost efficiency (0.877428), slightly exceeding those of the heuristic-based (0.85) and clustering-based (0.8) approaches, indicating its ability to maximize resource usage while minimizing operational costs.

These results emphasize the advantages of the SVM-based framework in delivering low latency, balanced load distribution, and superior resource utilization, making it particularly suitable for latency-sensitive and resource-constrained edge computing applications. While heuristic-based and clustering-based methods exhibit reasonable performance, they lack the adaptability required for the dynamic and high-dimensional demands of modern edge environments. Consequently, the findings advocate for the broader adoption of SVM-based approaches in scenarios requiring real-time processing, such as the healthcare IoT, autonomous vehicles, and smart city infrastructure.

6. Further Discussions Limitations

The SVM-based model has demonstrated significant benefits, particularly in terms of low latency and load balancing. However, certain limitations are noted:

Scalability: In environments with highly variable loads, SVMs may face challenges in maintaining consistent performance. While it effectively manages high-dimensional data, real-time adjustments may require integration with reinforcement learning methods.

Parameter sensitivity: SVM effectiveness is sensitive to parameter choices, such as kernel type and regularization. Fine-tuning these parameters is essential, especially for edge environments with variable network conditions(performance analysis of…)(performance analysis of…).

Future research should consider combining SVM with adaptive learning methods to dynamically adjust parameters on the basis of real-time workload, enhancing scalability and adaptability.

6.1. Results

The SVM model's performance at low latency is attributed to its streamlined task classification process, which minimizes computational overhead. This advantage is essential for real-time edge applications where delayed

responses are detrimental. Moreover, the SVM's balanced load distribution optimizes resource use across edge servers, providing enhanced scalability and adaptability in dynamic environments.

6.2 Limitations and Future Research Directions

The proposed SVM-based model for resource allocation in edge computing has shown significant promise, but there are several opportunities for enhancement and challenges to address. Integrating SVM with reinforcement learning techniques could enhance adaptability in dynamic, high-variance environments, allowing the framework to handle variable workloads more effectively. Such hybrid approaches would improve scalability and performance under diverse conditions.

Scalability remains a key challenge, as maintaining consistent performance in larger edge networks can be difficult. An increased network size often exacerbates latency and load-balancing inefficiencies. Additionally, while SVM is computationally efficient compared with reinforcement learning, it struggles with extremely high-dimensional data, with computational costs increasing for training and prediction. Addressing these issues through dimensionality reduction techniques and hybrid frameworks can ensure that the model remains efficient in real-time, large-scale scenarios.

Expanding the model's use cases to broader edge computing applications, such as smart cities, healthcare IoT, and autonomous vehicles, is another promising direction. These domains require robust load balancing and low latency, making them ideal environments for demonstrating the framework's adaptability and real-world utility.

The model's reliance on precise hyperparameter tuning presents another limitation, as selecting the regularization parameter (CC) and kernel coefficient ($\gamma\gamma$) can be time-consuming and challenging, especially in dynamic edge environments. The development of adaptive parameter-tuning mechanisms would streamline this process, ensuring optimal performance across various conditions without extensive manual effort.

Despite its relative efficiency, SVM's computational overhead when handling high-dimensional data may limit its applicability in real-time scenarios. Employing optimization techniques and hybrid approaches can mitigate these limitations, enhancing both scalability and efficiency.

By addressing these challenges and exploring broader applications, the SVM-based framework can be refined to meet the demands of dynamic, large-scale edge computing environments, confirming its role as a robust and versatile solution for resource allocation and job offloading.

6.3. Applications and Implications

The applicability of this framework extends across diverse edge computing domains. In the healthcare IoT, SVM's low-latency resource allocation is ideal for patient monitoring systems, ensuring timely responses to critical data. In autonomous vehicles, the model's efficient load balancing can support real-time data processing, which is crucial for navigation and safety. Similarly, smart city infrastructure could benefit from the framework's adaptive load management, enhancing the performance of traffic control, surveillance, and public safety systems.

7. Conclusions

The SVM-based model introduced in this study has clear advantages in edge computing environments, with superior performance in terms of latency, load distribution, and resource utilization. By achieving reductions in total latency and task completion time and enhancements in cost efficiency, the framework outperforms heuristic-based, clustering-based, and other machine learning approaches across all key metrics. Specifically, the SVM framework has the ability to reduce total latency by 14.2% and 21.6% compared with heuristic and clustering

methods and outperforms K-nearest neighbors, random forest, and neural networks with a latency of 1.803125, load distribution of 0.073357, and cost efficiency of 0.877428, underscoring its potential as a robust solution for dynamic, real-time resource allocation.

Despite its advantages, challenges such as parameter sensitivity and scalability in highly variable environments remain. Future work will explore hybrid approaches that integrate SVM with adaptive learning methods to address these challenges. Expanding the framework's application to broader edge computing domains, including smart cities, healthcare IoT, and autonomous vehicles, will further solidify its role in advancing intelligent edge computing solutions. These results affirm that the SVM-based model is a powerful tool for enhancing resource allocation in edge computing systems.

References:

1. Ergen, M., Saoud, B., Shayea, I., El-Saleh, A. A., Ergen, O., Inan, F., & Tuysuz, M. F. (2024). "Edge computing in future wireless networks: A comprehensive evaluation and vision for 6G and beyond." ICT Express, 10(5), 1151-1173.
2. Liu, B., Luo, Z., Chen, H., & Li, C. (2022). "A survey of state-of-the-art on edge computing: Theoretical models, technologies, directions, and development paths." IEEE Access, 10, 3176106. https://doi.org/10.1109/ACCESS.2022.3176106
3. Alsadie, D. (2024). "A comprehensive review of AI techniques for resource management in fog computing: Trends, challenges, and future directions." IEEE Access, 12, 118007-118059. https://doi.org/10.1109/ACCESS.2024.3447097.
4. Sharma, P., Nisha, Shukla, S., & Vasudeva, A. (2023). "An era of mobile data offloading opportunities: A comprehensive survey." Mobile Networks and Applications, 29, 13–28.
5. She, Q., Wei, X., Nie, G., & Chen, D. (2019). "QoS-aware cloud service composition: A systematic mapping study from the perspective of computational intelligence." Expert Systems with Applications, 138, 112804. https://doi.org/10.1016/j.eswa.2019.07.021
6. Martin, A., Viola, R., Zorrilla, M., Flórez, J., Angueira, P., & Montalbán, J. (2020). "MEC for fair, reliable, and efficient media streaming in mobile networks." IEEE Transactions on Broadcasting, 66(2), 356–368.
7. Adeniyi, O., Sadiq, A. S., Pillai, P., Taheir, M. A., & Kaiwartya, O. (2023). "Proactive self-healing approaches in mobile edge computing: A systematic literature review." Computers, 12(3), 63. https://doi.org/10.3390/computers12030063
8. Chang, Z., Liu, S., Xiong, X., Cai, Z., & Tu, G. (2021). "A survey of recent advances in edge-computing-powered artificial intelligence of things." IEEE Internet of Things Journal, 8(18), 13845–13868.
9. Fang, T., Yuan, F., Ao, L., & Chen, J. (2022). "Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: A potential game approach." IEEE Internet of Things Journal, 9(5), 3746–3758.
10. Wang, J., Wu, W., Liao, Z., Sangaiah, A. K., & Sherratt, R. S. (2019). "An energy-efficient offloading scheme for low latency in collaborative edge computing." *IEEE Access*, *7*, 149182–149190.
11. Henebelle, A., Zhang, X., & Ksentini, A. (2018). "Mobile edge computing for IoT: Enhancing efficiency and user experience." IEEE Communications Magazine, 56(10), 64–69.
12. Biswas, A., & Wang, H.-C. (2023). Autonomous vehicles enabled by the integration of IoT, edge intelligence, 5G, and blockchain. *Sensors, 23*(4), 1963. https://doi.org/10.3390/s23041963
13. Wang, Y., Chen, C.-R., Huang, P.-Q., & Wang, K. (2021). "A new differential evolution algorithm for joint mining decision and resource allocation in a MEC-enabled wireless blockchain network". Computer. Ind. Eng., 155, 107186 https://doi.org/10.1016/j.cie.2021.107186
14. Ullah, I., Ali, F., Khan, H., Khan, F., & Bai, X. Ubiquitous computation in internet of vehicles for human-centric transport systems. Comput. Hum. Behav., 161, 108394 (2024). https://doi.org/10.1016/j.chb.2024.108394
15. Wang, J., Jin, C., Tang, Q., Xiong, N. N., & Srivastava, G. (2021). Intelligent ubiquitous network accessibility for wireless-powered MEC in UAV-assisted B5G. IEEE Transactions on Network Science and Engineering, 8(4), 2801–2813. https://doi.org/10.1109/TNSE.2020.3029048
16. Okilanda, A., Ihsan, N., Arnando, M., Hasan, B., Mohamed Shapie, M. N., Tulyakul, S., Duwarah, T., & Ahmed, M. (2024). "Revving up performance: The impact of interval training with weighted resistance on speed enhancement in university students." Retos, 58, 469–476.
17. Chu, D., & Yoo, J. (2024). "A study on the analysis of security requirements through literature review of threat factors of 5G mobile communication." Journal of Information Processing Systems, 20(1), 38–52.
18. Hua, H., Li, Y., Wang, T., Dong, N., Li, W., & Cao, J. (2023). "Edge computing with artificial intelligence: A machine learning perspective." ACM Computing Surveys, 55(9), Article 184. https://doi.org/10.1145/3555802
19. Carvalho, G., Cabral, B., Pereira, V., & Bernardino, J. (2021). "Edge computing: Current trends, research challenges, and future directions." Computing, 103(7), 993–1023. https://doi.org/10.1007/s00607-020-00896-5

20. Devi, N., Dalal, S., Solanki, K., Dalal, S., Lilhore, U. K., Simaiya, S., & Nuristani, N. (2024). "A systematic literature review for load balancing and task scheduling techniques in cloud computing." Artificial Intelligence Review, 57(1), Article 276. https://doi.org/10.1007/s10462-024-10925-w
21. Hikida, T., Nishikawa, H., & Tomiyama, H. (2021). "Heuristic algorithms for dynamic scheduling of moldable tasks in multicore embedded systems." International Journal of Reconfigurable and Embedded Systems (IJRES), 10(3), 157–167. https://doi.org/10.11591/ijres.v10.i3.pp157-167
22. Vakili Fard, M., Sahafi, A., Rahmani, A. M., & Sheikholharam Mashhadi, P. (2020). "Resource allocation mechanisms in cloud computing: A systematic literature review." IET Software. https://doi.org/10.1049/iet-sen.2019.0338
23. Singh, J. D., Singh, N., Adhikari, M., & Singh, A. K. (2024). "Decentralized gossip-assisted deep learning model training for resource-constraint edge devices." IEEE Transactions on Computational Social Systems, 1–10. https://doi.org/10.1109/TCSS.2024.3395516
24. Haibeh, L. A., Yagoub, M. C. E., & Jarray, A. (2022). "A survey on mobile edge computing infrastructure: Design, resource management, and optimization approaches." IEEE Access, 10, 27591–27610. https://doi.org/10.1109/ACCESS.2022.3152787
25. Qua, B., Bai, Y., Chu, Y., Wang, L.-E., Yu, F., & Li, X. (2022). "Resource allocation for MEC system with multiusers resource competition based on deep reinforcement learning approach." Computer Networks, 203, Article 103,089.
26. Huang, X., Chen, Z., Chen, Q., & Zhang, J. (2023). "Federated learning-based QoS-aware caching decisions in fog-enabled Internet of Things networks." Digital Communications and Networks, 9(2), 580–589. https://doi.org/10.1016/j.dcan.2022.04.022
27. Xiang, B., Elias, J., Martignon, F., & Di Nitto, E. (2021). "Resource calendaring for Mobile Edge Computing: Centralized and decentralized optimization approaches." Computer Networks, 199, Article 108426. https://doi.org/10.1016/j.comnet.2021.108426
28. Zhang, J., Guo, H., Liu, J., & Zhang, Y. (2020). Task offloading in vehicular edge computing networks: A load-balancing solution. IEEE Transactions on Vehicular Technology, 69(2), 2092–2104.
29. Xu, X., Fu, S., Cai, Q., Tian, W., Liu, W., Dou, W., Sun, X., & Liu, A. X. (2018). Dynamic resource allocation for load balancing in fog environment. Wireless Communications and Mobile Computing, 2018, 1–15.
30. Lei, L., Xu, H., Xiong, X., Zheng, K., Xiang, W., & Wang, X. (2019). Multiuser resource control with deep reinforcement learning in IoT edge computing. IEEE Internet of Things Journal, 6(6), 10119–10133. https://doi.org/10.1109/JIOT.2019.2935543
31. Prasad, H., Arpita, S., Bandyopadhyay, A., Dash, D., & Swain, S. (2024). "Dynamic resource allocation using auction technique in fog computing." In Proceedings of the Asia Pacific Conference on Innovation in Technology (APCIT), 26-27 July 2024, Mysore, India. IEEE. https://doi.org/10.1109/APCIT62007.2024.10673619
32. Zhang, Y., Liu, Y., Zhou, J., Sun, J., & Li, K. (2020). Slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in resource-limited mobile edge computing. Future Generation Computer Systems, 112, 148-161. https://doi.org/10.1016/j.future.2020.05.025
33. Wang, P., Yang, H., Han, G., Yu, R., Yang, L., & Sun, G. (2024). Decentralized navigation with heterogeneous federated reinforcement learning for UAV-enabled mobile edge computing. IEEE Transactions on Mobile Computing, 23(12), 13621-13638. https://doi.org/10.1109/TMC.2024.3439696
34. Beraldi, R., & Proietti Mattia, G. (2023). A load balancing algorithm for long-life green edge systems. In Proceedings of the 2023 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Naples, Italy, 04-06 December 2023. IEEE. https://doi.org/10.1109/CloudCom59040.2023.00020
35. Ren, P., Qiao, X., Huang, Y., Liu, L., Pu, C., & Dustdar, S. (2020). Edge AR X5: An edge-assisted multiuser collaborative framework for mobile web augmented reality in 5G and beyond. IEEE Transactions on Cloud Computing, 10(4), 2521-2537. https://doi.org/10.1109/TCC.2020.3046128
36. Fang, X., Li, B., Zhang, Y., & Wang, Z. (2022). DRL-aided task offloading and resource allocation scheme to minimize power consumption in cloud-edge cooperation environments. IEEE Transactions on Cloud Computing, 10(3), 2153–2163.
37. Liu, W., Zhao, X., Wang, J., & Li, Y. (2021). Onboard edge computing task offloading optimization using deep Q-network. Journal of Communications, 41(12), 1304–1311.
38. Zhao, H., Zhang, T., Chen, Y., Zhao, H., & Zhu, H. (2020). Task distribution offloading algorithm of vehicle edge network based on DQN. Journal of Communications, 41(10), 172–178.
39. Li, J., Gao, H., Lv, T., & Lu, Y. (2018). Deep reinforcement learning-based computation offloading and resource allocation for MEC. 2018 IEEE Wireless Communications and Networking Conference (WCNC), 1–6.
40. Chen, Y., Chen, S., Li, K. C., Liang, W., & Li, Z. (2022). DRJOA: Intelligent resource management optimization through deep reinforcement learning approach in edge computing. Cluster Computing, 26(5), 2897–2911.
41. Yun, J., Goh, Y., Yoo, W., & Chung, J. M. (2022). 5G multi-RAT URLLC and eMBB dynamic task offloading with MEC resource allocation using distributed deep reinforcement learning. IEEE Internet of Things Journal, 9(20), 20733–20749.
42. Yu, Z., Xu, X., & Zhou, W. (2022). Task offloading and resource allocation strategy based on deep learning for mobile edge computing. Computational Intelligence and Neuroscience, 2022, Article ID 1427219.
43. Hazarika, B., Singh, K., Biswas, S., Mumtaz, S., & Li, C. P. (2023). Multiagent DRL-based task offloading in multiple RIS-aided IoV networks. IEEE Transactions on Vehicular Technology, 73(1), 1175–1190.

44. Wei, Y., Yu, F. R., Song, M., & Han, Z. (2021). Joint optimization of caching, computing, and radio resources in MEC. IEEE Transactions on Communications, 69(4), 2250–2262.
45. Zhao, L., Zhang, Q., Liu, Z., & Zhang, Y. (2024). Energy-efficient task offloading in MEC using multiagent deep reinforcement learning. IEEE Transactions on Mobile Computing, 23(1), 44–56.
46. Yang, Y., Li, J., & Zhang, H. (2023). Multiagent reinforcement learning for resource allocation in mobile-edge computing networks. IEEE Internet of Things Journal, 10(5), 3402–3413.
47. Bi, S., Huang, L., Wang, H., & Zhang, Y. Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks. IEEE Trans. Wireless Commun. 20(11), 7519–7537 (2021).
48. Zhang, K., Cao, J., & Zhang, Y. Adaptive digital twin and multiagent deep reinforcement learning for vehicular edge computing and networks. IEEE Trans. Ind. Inform. 18(2), 1405–1413 (2022).
49. Sun, X., Liu, Z., & Li, W. (2023). Collaborative computation offloading in vehicular edge computing networks using deep reinforcement learning. IEEE Transactions on Vehicular Technology, 72(6), 7429–7441.