IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 1 July 2024, accepted 22 July 2024, date of publication 29 July 2024, date of current version 6 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3434655

RESEARCH ARTICLE

MFDROO: Matrix Factorization-Based Deep Reinforcement Learning Approach for Stable Online Offloading in Mobile Edge Networks

ENGY A. ABDELAZIM[1,2], (Graduate Student Member, IEEE), SHERIF K. ELDAYASTI[2], (Member, IEEE), HUSSEIN M. ELATTAR[2], (Member, IEEE), AND MOHAMED A. ABOUL-DAHAB[2], (Life Senior Member, IEEE)

[1]Department of Electronics and Communications, Faculty of Engineering, Modern Academy for Engineering and Technology, Cairo 11585, Egypt
[2]Department of Electronics and Communication, Arab Academy for Science, Technology and Maritime Transport, Cairo 11799, Egypt

Corresponding author: Engy A. Abdelazim (engy.azim@student.aast.edu)

ABSTRACT Data-intensive applications coupled with limited mobile resources make opportunistic computation offloading imperative. Therefore, efficient and reliable offloading strategies are crucial for achieving optimal performance in terms of stabilizing data queues at various data rates, efficient task management and reduced waiting times for users. This paper proposes a novel algorithm, Matrix Factorization-Based Deep Reinforcement Learning Approach for Stable Online Offloading in Mobile Edge Networks (MFDROO), that combines Orthogonal Non-negative Matrix Factorization (ONMF) and a Deep Reinforcement Learning approach (DRL) to address the problem of stable online offloading in large networks. The proposed algorithm utilizes ONMF to model network resource heterogeneity by decomposing it into a set of features, capturing the underlying structure of data traffic flows by removing irrelevant information from data. This reduces computational overhead and improves performance. Additionally, MFDROO incorporates a DRL agent to learn optimal offloading decisions over time. By utilizing ONMF in conjunction with DRL, MFDROO overcomes online offloading challenges. It optimizes user computation rates and enhances system performance. Additionally, it maintains data-queue stability for large networks or higher data rates. Large-scale network simulations were extensively conducted to demonstrate MFDROO effectiveness. Maintaining stability while scaling up the users number is a challenge in network computation. Our results demonstrate that MFDROO tackles this challenge and ensures that even with a 3.3% increase in user numbers (up to 100 users), computation networks remain stable and outperform other existing algorithms in terms of utility maximization and stability. This improvement ensures optimal performance and provides scalability and efficiency for various applications.

INDEX TERMS Computation rate, data offloading, deep reinforcement learning (DRL), mobile edge computing, orthogonal non-negative matrix factorization (NMF), queue stability.

I. INTRODUCTION

Edge computing technology has emerged as a promising solution to overcome the challenges associated with processing the vast volumes of data generated by mobile devices. One of the primary goals of edge computing is to minimize latency and ensure real-time responsiveness, which is crucial for applications that require immediate data processing and analysis.

In this paradigm, computing resources are strategically positioned near mobile devices at the edge of the network. This enables local processing of data and facilitates tasks

The associate editor coordinating the review of this manuscript and approving it for publication was Hadi Tabatabaee Malazi.

such as filtering, aggregating and pre-processing before transmitting the data to the cloud for further analysis. Such an approach not only reduces the volume of data that needs to be transmitted over the network but also reduces the processing load on centralized cloud servers [1]. As technology continues to advance, utilizing edge computing can yield too many advantages, such as reducing network traffic, enhancing data security, ensuring uninterrupted services and heightening reliability. This in turn results in optimizing the network performance while meeting the evolving demands of the digital generation. The decision to offload all tasks to mobile edge servers should be carefully considered based on various factors. These encompass evaluating the costs and benefits of the approach, as well as considering the specific requirements of the mobile computing system at hand.

In this respect, the Opportunistic Computation Offloading (OCO) technique for Mobile Edge Computing (MEC) offers a unique advantage. It leverages opportunistic network conditions, such as the presence of an edge server with sufficient processing power and low-latency communication links between the mobile device and the edge server. By employing such a technique, computation tasks can be dynamically allocated either locally or at the edge station [2], [3]. This flexibility enables the optimization of resource utilization, leading to improved system performance and reduced energy consumption [4]. Various approaches were adopted to deal with OCO in MEC using heuristic algorithms [5] such as machine learning-based techniques [6] and game theory-based solutions [7], [8]. The performance of these algorithms were evaluated based on factors such as offloading delay, energy consumption and processing time [9], [10]. However, making decisions based on multiple variable choices often requires solving mixed-integer non-linear programming (MINLP) problems. Solving such problems can be computationally complex, especially when dealing with large datasets. This complexity makes it difficult to interpret the results and make informed decisions based on the optimization solution.

From this perspective, the use of deep reinforcement learning (DRL) techniques is considered a promising solution as it offers several benefits for online computation offloading. Among these are enabling more efficient and effective decision-making and handling complex and dynamic environments. In addition, DRL learns how to balance different factors such as computation time, energy consumption and network bandwidth despite limited computing resources [11]. However, maintaining stable system operation is closely related to guaranteeing DRL stability. Stability is a crucial factor in the success of data offloading as it is an indicator that the system does not undergo an over-fitting scenario in the data offloading system. Over-fitting occurs when a system model is too complex or closely fits the training data but fails to generate well to the unseen data. Therefore, over-fitting could manifest as instability in the data offloading (queue instability) process, resulting in serious consequences of delays in decision-making and missed opportunities due to the loss of some important data and hence reducing the efficiency and driving away potential customers [12]. Therefore, the objective of maintaining a stable system (ex. queue stability) in the context of using the DRL with respect to the OCO technique for MEC is considered challenging.

In [6], the authors introduced an approach that aimed at determining the optimal allocation of tasks between mobile devices and edge servers to minimize costs, these costs related to both computation-rates and energy consumption. However, this approach requires a large amount of training data to learn optimal offloading decisions, posing challenges in practice. Moreover, it did not guarantee queue stability for networks with more than ten users. Queue stability referred to the ability of a data offloading system to maintain consistent and reliable performance under varying loads (specially in networks having higher number of users) and various data rates. In this respect, a stable offloading system was necessary in ensuring smooth data transfer, and efficient processing concerning achieving the user's request tasks requirements regardless of network conditions [7].

The approach proposed in [10] utilized deep learning techniques to optimize offloading decisions and improve computation efficiency in MEC networks through parallel processing. This was accomplished by dynamically allocating tasks either to the local device or the MEC server based on their respective capabilities and workload. While the approach showed promising results in terms of maximizing the weighted sum computation rate, it was crucial to carefully consider that eliminating strict queue stability requirements might reduce its efficiency. The authors in [13] focused on minimizing the overall energy consumption and latency while maximizing the Quality of Service (QoS) for user tasks. They could achieve this goal by dynamically adjusting offloading decisions based on real-time channel conditions and energy availability. Although the proposed algorithm demonstrated promising results, the authors noted that they should focus on long-term performance optimization and maximizing the Spectral Efficiency (SE) of mobile edge networks in future research. In [14], the authors proposed an optimization-based approach for selecting suitable computation offloading candidates in MEC networks, which was a promising approach that combined optimization and machine-learning techniques to improve task allocation efficiency in MEC networks. However, they pointed out that additional research was needed to evaluate the scalability and effectiveness of their approach under different workloads and network conditions. Finally, in [15], the authors proposed a DRL-based binary computation offloading scheme that exploited the energy harvesting capabilities of wireless devices. Simulation results demonstrated the effectiveness of the proposed approach by reducing the computation latency and energy consumption of mobile devices. It was worth notable that the simulations were conducted on a relatively small scale, and the results might not be generalizable to larger and more complex MEC networks.

It is therefore evident that there is still a need for new techniques that would increase the offloading data-queues stability at various data rates, especially for large networks, accelerating request tasks for users and so maximizing their computation rates. For this reason, higher loads need to be considered since they not only strain the network resources but also increase the likelihood of encountering performance bottlenecks. Without proper stability measures (ex. Check Data-Queue stabilization), any proposed system may suffer from over-fitting, that leads to accuracy deterioration due to the excessive complexity and over-fitting to training data (means data arrival rate has surpassed the computation capacity).

In this paper, a novel algorithm is proposed to accelerate data processing and reduce the likelihood of processing delays or queue instability for networks with higher loads (significant user bases). We propose an online computation offloading algorithm, a non-negative Matrix Factorization-Based Deep Reinforcement Learning Approach for Stable Online Offloading named MFDROO. The proposed MFDROO algorithm enables efficient online offloading decision policies in the sense that the decisions are made without the assumption of knowing the future realization of channel conditions and data arrivals. It optimizes computational rates while providing stability for the long-term system. Our simulation results are highly promising, demonstrating the MFDROO's potential to enhance network performance and improve overall user experience. The main contribution of this paper can be summarized as follows:

1- Leveraging orthogonal non–negative matrix factorization (ONMF) machine learning, which is a powerful matrix factorization network technique, to model network resource heterogeneity by decomposing a network into a set of features, capturing the underlying structure of data traffic flows by removing noise and irrelevant information from data. As a result, computational overhead will be reduced, performance will be improved and data offloading will be optimized.
2- MFDROO integrates a DRL agent to provide decision-making capabilities. DRL is a powerful technique that allows agents to learn from experience and optimize their behavior over time. By incorporating a DRL agent, the system achieves optimal offloading decisions.
3- Utilizing ONMF in conjunction with DRL makes MFDROO an innovative algorithm that succeeds in enhancing the computation offloading in terms of stabilizing data-queues at various data rates, efficient task management so accelerating demand tasks for user and therefore optimizing user's computational-rates.
4- Our results demonstrate that MFDROO outperforms other existing algorithms listed in the literature in terms of user's average weighted sum computation-rate optimization and stability control. MFDROO ensures that even with a 3.3% increase in user numbers (up to 100 users), computation networks remain stable and ensure optimal system performance.

II. RELATED WORKS

Numerous studies in the literature have proposed a variety of approaches and architectures for the application of deep reinforcement learning in the context of online data offloading. The authors in [16] proposed a DRL based offloading decision and resource management (DECENT) algorithm that utilizes the advantage actor-critic method that optimizes the offloading decision in real-time, as well as computes the resource allocation for each arriving task. This enabled cumulative weighted response time minimization. However, one limitation was its failure to consider how the task arrival rate influences the effectiveness of the offloading algorithm. In cases where task-arrival rates were high, communication and computing queues could become congested, affecting the accuracy of waiting time estimation and the offloading decision. Moreover, the authors did not consider the impact of network topology on the offloading algorithm. If the network topology was complex, multiple paths or hops between the IoT devices and the MEC servers can affect communication latency and reliability. Therefore, capturing the realistic scenarios of mobile edge computing with multiple objectives and constraints was not realized. The authors in [17] proposed a DRL based Online Offloading (DROO) framework that implemented a deep neural network as a scalable solution, thereby learning the binary offloading decisions from experience. However, this approach failed to account for the influence of task-size on the effectiveness of the offloading algorithm. If the task sizes are large, communication overhead has the potential to dominate computation overhead, which can have a negative impact on energy efficiency and latency of the offloading decision. Moreover, the authors did not take into account how task dependencies would impact their offloading algorithm. For example, If tasks depend on one another, then data dependency and synchronization among them must be factored into any decision-making process for optimal feasibility. The authors in [18] proposed a DRL-based dynamic task offloading (DDTO) adaptable algorithm for the dynamic and complex environment and re-formulate the task offloading problem as a Markov decision process (MDP). As the optimization objective, the proposed algorithm reduces network service delay and energy consumption by weighted sums. However, if lots of mobile terminals aggressively offloaded their computation tasks to be executed on the edge server, this would lead to severe congestion on wireless links and hence result in a significant delay in executing MEC. The authors in [6] proposed a Lyapunov guided DRL (LyDROO) framework that solved the online computation offloading problem, claiming it could achieve high computation performance executed at very low latency while maintaining queue stabilization. However, they did not consider network size's impact on the offloading algorithm. Therefore, realistic scenarios of mobile edge computing with complex tasks and data could not be captured.

From that perspective, focusing on using matrix factorization technique gained researchers attentions last years. Matrix factorization is a technique used in both artificial

intelligence and machine learning to reduce the dimensionality of high-dimensional data by decomposing it into lower-dimensional representations. Such techniques are commonly used in various applications, including image processing, natural language processing, and recommender systems. Furthermore, matrix factorization can be highly advantageous for machine learning tasks where the original feature space is excessively large or noisy [19]. The factorization process aims for feature extraction in data offloading. It helps in capturing and identifying meaningful patterns and the underlying characteristics in the data that can used for machine learning tasks and hence optimize queue operations in real-time. This allows for timely decision-making leading to improved efficiency.

Orthogonal non-negative matrix factorization (ONMF) and non-negative matrix factorization (NMF) are two popular techniques used to analyze and transform data. However, they differ in the way they handle missing data and the computational complexity. NMF is a dimensionality reduction technique that finds a lower-dimensional representation of a non-negative matrix by factorizing it into two non-negative matrices. NMF is known for its ability to capture the underlying patterns or dependencies in the data. However, NMF has several limitations when applied specifically to time-series data. One key limitation is the potential for correlated factors. In NMF, the factors are typically chosen to minimize an objective function, such as the sum of squared residuals or mutual information. However, this can result in correlated factors, which can introduce noise into the decomposition and make it difficult to interpret the results [20].

The ONMF is a type of unsupervised machine learning technique that is widely used in various fields of data analysis. One of the key applications of ONMF is in the field of queue management, where it plays a crucial role in ensuring the stability of queues. ONMF-based queue management systems utilize the concept of matrix factorization. ONMF imposes an additional constraint on the factor matrix to be orthogonal, the idea behind the orthogonality constraint in ONMF is to ensure that the latent factors are aligned in a particular way, resulting in a subset of latent factors that are mutually orthogonal. The motivation for orthogonality constraint in ONMF is that the orthogonal matrices have low rank (reduced dimensionality), meaning they can be decomposed into smaller matrices that are easy to interpret and visualize as the extracted patterns are not confounded by redundant information, leading to more interpretable and meaningful representations [21].

The authors in [22] proposed an improved multi-view clustering algorithm that integrates multiple NMF with co-orthogonal constraints to enforce orthogonality between feature matrices from different views. However, the experimental results were conducted on a limited number of datasets. Therefore, the computational complexity of the proposed algorithm might be high when dealing with many views or high-dimensional data, so further research was needed to improve its scalability. The authors further proposed an advanced initialization procedure for the NMF algorithm, which enabled it to converge more quickly and accurately to the global minimum. Despite the NMF method's effectiveness, it was still computationally expensive compared to other factorization methods. Moreover, the authors did not conduct experiments on synthetic or simulated data to evaluate the robustness or sensitivity of the proposed algorithm and its applications to different scenarios [23]. An NMF model generalized by [24] defined basis vectors as data points instead of separable vectors. The proposed model could be solved efficiently by a convex optimization model or a heuristic algorithm. The authors also claimed that the proposed model could achieve better or comparable results than existing separate NMF or standard NMF algorithms. This was on synthetic, document, and image data sets. However, it was assumed that the data was generated from a specific probabilistic model, which might not always be the practice case, and it might suffer from high computational complexity, particularly when the data size was large. In [25], the authors designed a new formulation of ONMF problem for clustering to improve the performance of clustering algorithms. The work incorporated transferring the orthogonality constraint problem into an ONMF one, which helped in capturing the underlying structure of the data and improving the quality of the clustering results. However, a limited generalizability was considered as the work focused on a specific variant namely the orthogonal NMF problem for clustering. Additionally, a lack of real-world application made it difficult to assess the practical usefulness of the algorithm in real-world scenarios. In [26], the authors introduced a bi-stochastic graph regularization term that encourages the learned factor matrices to be both row-stochastic and column-stochastic, leading to more accurate clustering results. This algorithm was robust to noise and missing data and could manage high-dimensional datasets with varying degrees of sparsity. However, the work only addressed clustering tasks, and the algorithm's generalizability to other problems, such as dimensionality reduction or feature extraction, was not explored. A maximum-entropy-principle approach for solving the ONMF problem was proposed in [27]. This approach incorporated the maximum entropy principle to regularize the factor matrices, leading to more accurate factorization and enforcing orthogonality on the factor matrices. Therefore, the ONMF approach enhances the interoperability of the factors. However, all the proposed DRL or ONMF methods failed to address the performance constraint parameters while optimizing the user's computation-rate under a dynamic larger network. So, from that perspective, the proposed MFDROO algorithm is a promising algorithm in addressing the aforementioned challenges by leveraging the ONMF technique as a preprocessing step before being applied to the subsequent DRL leading to more robust and reliable predictions.

Accordingly, the remainder of the paper will be organized as follows. Section III explains and illustrates the system model and problem formulation in addition to the

Lyapunov optimization technique used to solve stable computation offloading problems [28], [29], [30], [31]. Section IV introduces the MFDROO algorithm to solve the problem in Section III using the combination of ONMF with DRL. Section V evaluates MFDROO through extensive simulations. Lastly, Section VI concludes the paper.

Notation: Throughout this paper, Superscripts $()^{\mathbf{T}}$ and the log function denotes the transpose and the element-wise logarithm operation of a vector respectively.

III. SYSTEM MODEL, PROBLEM FORMULATION AND LYAPUNOV OPTIMIZATION

This section describes the system model followed by explaining the problem formulation in section B and then utilizing the Lyapunov optimization in section C. A summary of the notations used in this section is summarized in Table 1.

A. SYSTEM MODEL

The MEC multi-user computing network shown in Fig.1 considers an edge station (ES) communicating with M wireless devices (WDs) sequentially within equal time-frames duration (T). On a time division multiple access (TDMA) basis, WDs offload tasks to ES using the same bandwidth (B).

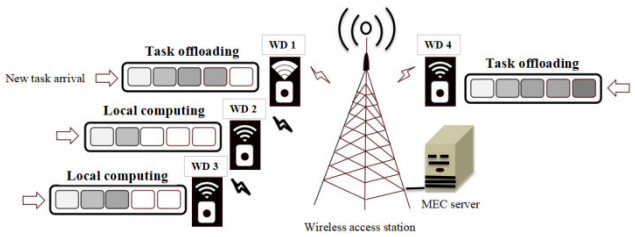


FIGURE 1. MEC multi-user computing network.

For each t time-frame, we denote the raw data-tasks arriving at the i^{th} WD queue as d_i^t (in bits). These tasks follow a general independent and identically distributed (i.i.d) distribution. We also denote c_i^t as the channel gain between the i^{th} WD and ES, assuming a block fading model. This model assumes that the channel gain remains constant within a time-frame but can vary from one frame to another.

Suppose that in t^{th} time-frame, a connecting WD i processes p_i^t bits of data followed by a computation output generated at the termination of the time-frame. In this respect, we follow the binary computation offloading rule [6]. Based on this rule, during each time-frame, it is either possible to process the raw tasks locally at the WD or offload them remotely at the ES for illustration. As shown in Fig.1, the WDs 2&3 compute their tasks locally while WDs 1&4 offload them at the ES. A binary variable v_i^t is used to express the offloading decision, where $v_i^t = 1$ denotes that WD i executes their computing tasks remotely (offload), while $v_i^t = 0$ denotes that the tasks are computed locally.

In the case of local data processing by WD ($v_i^t = 0$), we must consider the local CPU frequency as f_i^t limited by f_i^{max}. The locally processed raw data (in bits) and the amount of energy consumed during the time-frame are represented by (1) below:

$$\begin{aligned} p_{i,L}^t &= f_i^t T/z, \quad \forall v_i^t = 0, \\ e_{i,L}^t &= k \left(f_i^t\right)^3 T, \end{aligned} \tag{1}$$

The parameter k indicates computing energy efficiency ($k > 0$) and the parameter z indicates how many calculation cycles are required to analyze a single bit of raw data ($z > 0$).

Otherwise, for edge processing ($v_i^t = 1$), we represent $\tau_i^t T$ as the time allotted to the i^{th} WD for computation offloading, where $\tau_i^t \in [0, 1]$ and $\sum_{i=1}^{M} \tau_i^t \leq 1$. The transmit power denotes as s_i^t bound by the maximum power $s_i^t \leq s_i^{max}$. We disregard the edge computing delay and hence, the volume of the data that is processed at the edge in the allotted time-frame using the Shannon capacity theorem is:

$$p_{i,O}^t = \frac{B\tau_i^t T}{\omega_u} \log_2 \left(1 + \frac{s_i^t c_i^t}{N_p}\right), \tag{2}$$

where $\omega_u \geq 1$ represents the overhead of communication and the noise-power is represented by N_p. Data offloading consumes energy formulated as $e_{i,O}^t = s_i^t \tau_i^t T$, and then the processed data at the edge can be expressed as:

$$p_{i,O}^t = \frac{B\tau_i^t T}{\omega_u} \log_2 \left(1 + \frac{e_{i,O}^t c_i^t}{\tau_i^t T N_p}\right), \forall v_i^t = 1, \tag{3}$$

The total number of bits computed and the total amount of energy consumed during the time-frame t are represented as:

$$\begin{aligned} p_i^t &\triangleq \left(1 - v_i^t\right) p_{i,L}^t + v_i^t p_{i,O}^t, \\ e_i^t &\triangleq \left(1 - v_i^t\right) e_{i,L}^t + v_i^t e_{i,O}^t, \end{aligned} \tag{4}$$

Therefore, the computation-rate $c_{r_i}^t$ and the power consumption $p_{c_i}^t$ in t^{th} time-frame can be defined as:

$$\begin{aligned} c_{r_i}^t &= \frac{p_i^t}{T} = \frac{\left(1 - v_i^t\right) f_i^t}{z} + v_i^t \frac{B\tau_i^t}{\omega_u} \log_2 \left(1 + \frac{p_{c_{i,O}}^t c_i^t}{\tau_i^t N_p}\right), \\ p_{c_i}^t &= \frac{e_i^t}{T} = \left(1 - v_i^t\right) k \left(f_i^t\right)^3 + v_i^t p_{c_{i,O}}^t, \end{aligned} \tag{5}$$

where $p_{c_{i,O}}^t \triangleq e_{i,O}^t/T$, taking into consideration the assumption of T=1 for exposition simplicity while maintaining generality in the following derivations. As each WD is characterized by its queue length $L_i(t)$ at the beginning of the t^{th} i^{th} time-frame, the queue dynamics can modeled as:

$$L_i(t+1) = \max \left\{L_i(t) - \tilde{P}_i^t + d_i^t, 0\right\}, i = 1, 2, \cdots, \tag{6}$$

where $\tilde{P}_i^t = \min \left(L_i(t), p_i^t\right)$ and $L_i(1) = 0$. For controlled problem-solving tractability, we have considered infinite queue capacity in this paper. Therefore, the queue dynamics can be simplified by enforcing the data causality constraint $p_i^t \leq L_i(t)$, including that $L_i(t) \geq 0$ as follows:

$$L_i(t+1) = L_i(t) - p_i^t + d_i^t, i = 1, 2, \cdots. \tag{7}$$

B. PROBLEM FORMULATION

The paper aims to optimally learn the best decision policies for online offloading while optimizing the resource allocation decisions using a well-designed algorithm framework. The algorithm framework aims to optimize the long-term system performance in terms of maximizing the average weighted sum computation-rate for all WDs while ensuring data-queue stability under the opportunistic computation offloading scheme. A feasible analysis method is adopted, following the multi-stage stochastic Mixed Integer Nonlinear Programming problem (MINLP) formulation [32], [33]. In particular, for each time-frame, the formulation takes into account the absence of knowledge of the future dynamics realizations of random channel conditions and data arrivals. we denote $\mathbf{v}^t = \left[v_1^t, \cdots, v_M^t\right]$, $\boldsymbol{\tau}^t = \left[\tau_1^t, \cdots, \tau_M^t\right]$, $\mathbf{f}^t = \left[f_1^t, \cdots, f_M^t\right]$ and $\mathbf{p}_{\mathbf{c}_O}^t = \left[p_{c_{1,O}}^t, \cdots, p_{c_{M,O}}^t\right]$, and let $\mathbf{v} = \left\{\mathbf{v}^t\right\}_{t=1}^K$, $\boldsymbol{\tau} = \left\{\boldsymbol{\tau}^t\right\}_{t=1}^K$, $\mathbf{f} = \left\{\mathbf{f}^t\right\}_{t=1}^K$ and $\mathbf{p}_{\mathbf{c}_O} = \left\{\mathbf{p}_{\mathbf{c}_O}^t\right\}_{t=1}^K$.

The MINLP problem formulates as follows:

$$\underset{\mathbf{v},\tau,\mathbf{f},\mathbf{p}_{\mathbf{c}O}}{\text{maximize}} \lim_{K\to\infty} \frac{1}{K}\cdot \sum_{t=1}^{K}\sum_{i=1}^{M} \eta_i c_{r_i}^t$$

subject to the following constraints

$$\sum\nolimits_{i=1}^{M} \tau_i^t \le 1, \forall t, \tag{8a}$$

$$\frac{\left(1-v_i^t\right) f_i^t}{z} + v_i^t \frac{B\tau_i^t}{\omega_u} \log_2\left(1+\frac{p_{c_{i,O}}^t c_i^t}{\tau_i^t N_p}\right) \le L_i(t), \forall i, t, \tag{8b}$$

$$\lim_{K\to\infty} \frac{1}{K}\cdot \sum\nolimits_{t=1}^{K} \mathbb{E}\left[\left(1-v_i^t\right) k \left(f_i^t\right)^3 + v_i^t p_{c_{i,O}}^t\right] \le \gamma_i, \forall i, \tag{8c}$$

$$\lim_{K\to\infty} \frac{1}{K}\cdot \sum\nolimits_{t=1}^{K} \mathbb{E}\left[L_i(t)\right] < \infty, \forall i, \tag{8d}$$

$$f_i^t \le f_i^{max}, p_{c_{i,o}}^t \le s_i^{max}\tau_i^t, \forall i, t, \tag{8e}$$

$$v_i^t \in \{0,1\}, \tau_i^t, f_i^t, p_{c_{i,O}}^t \ge 0, \forall i, t. \tag{8f}$$

Here, η_i indicates the fixed weight of the i^{th} WD and E is the expectation taken with the random system events [34]. Constraint 8(a) indicates the offloading time constraint. This constraint indicates that $\tau_i^t = p_{c_{i,O}}^t = 0$, the offloading time must be satisfied at the optimum when $v_i^t = 0$. In other words, if the offloading decision variable v_i^t is equal to 0, then the offloading time must hold at its optimal value. Similarly, $f_i^t = 0$ must hold if $v_i^t = 1$. It corresponds to the constraint (8b) which takes the data causality into account. Moving on, constraint (8c) corresponds to the average power threshold γ_i constraint in the context of the problem. Constraint (8d) represents the data-queue stability constraints. Constraint (8e) corresponds to that f_i^t must be upper bounded by f_i^{max} and (8f) represents the binary variable v_i^t that holds 0 or 1 to denote the offloading decision, and that the variables $\tau_i^t, f_i^t, p_{c_{i,O}}^t$ must hold a values ≥ 0 under the stochastic nature of the channels and data-arrivals. It is worth noting that due to the stochastic nature of the channels and data-arrivals, satisfying the long-term constraints mentioned above can be challenging. These constraints are concerned with maintaining data-queue stability, meaning that it must be ensured that the data-queue will not grow without bounds and will remain stable over time. The variability and unpredictability of the system make it difficult to guarantee the fulfillment of these constraints over extended periods. In the following sub-section, we shed light on the Lyapunov optimization technique, which shall be utilized in the proposed MFDROO algorithm explained in section IV.

C. LYAPUNOV OPTIMIZATION

The multi-stage stochastic problems are characterized by uncertainty and sequential decision-making, making them computationally challenging. To successfully address these problems, it is crucial to consider the inter-dependencies between decisions and uncertainties. To address this complexity, the proposed algorithm used in this work utilizes Lyapunov optimization, which offers a powerful framework for addressing multi-stage stochastic problems. By decoupling the problem into per-frame deterministic sub-problems, our algorithm simplifies the optimization process while still ensuring performance bounds. The applications of Lyapunov optimization are extensive, spanning areas such as wireless networks, energy management, and transportation systems. This technique has proven its effectiveness in enhancing system performance and tackling complex optimization challenges. In the subsequent analysis, we take on the procedure outlined in [6].

Introducing M virtual energy-queues $\{Y_i(t)\}_{i=1}^M$, one for each WD. Particularly, we set $Y_i(1) = 0$ and update the queue as:

$$Y_i(t+1) = \max\left(Y_i(t) + s p_{c_i}^t - s\gamma_i, 0\right), \tag{9}$$

for $i = 1, \cdots, M$ and $t = 1, \cdots, K$, where $p_{c_i}^t$ in (5) is the energy consumption at the t^{th} time-frame and s is a positive scaling factor. $Y_i(t)$ represented as a queue with random energy-arrivals ($s p_{c_i}^t$) and fixed service-rate ($s\gamma_i$).

We define the total queue backlog as $\mathbf{B}(t) = \{\mathbf{L}(t), \mathbf{Y}(t)\}$, where $\mathbf{L}(t) = \{L_i(t)\}_{i=1}^M$ and $\mathbf{Y}(t) = \{Y_i(t)\}_{i=1}^M$ to mutually control the data and energy queues and introducing the Lyapunov function $L(\mathbf{B}(t))$ as:

$$L(\mathbf{B}(t)) = 0.5\left(\sum\nolimits_{i=1}^{M} L_i(t)^2 + \sum\nolimits_{i=1}^{M} Y_i(t)^2\right), \tag{10}$$

and Lyapunov drift $\Delta L(\mathbf{B}(t))$ [34] as:

$$\Delta L(\mathbf{B}(t)) = \mathbb{E}\{L(\mathbf{B}(t+1)) - L(\mathbf{B}(t)) \mid \mathbf{B}(t)\}, \tag{11}$$

The drift-plus-penalty minimization approach is used to simultaneously optimize the time average computation-rate and maintain the stability of the queue $\mathbf{B}(t)$. Each t time-frame has the objective of minimizing the upper-bound of the drift-plus-penalty expression:

$$\Lambda(\mathbf{B}(t)) \triangleq \Delta L(\mathbf{B}(t)) - O\cdot \sum\nolimits_{i=1}^{M} \mathbb{E}\left\{\eta_i c_{r_i}^t \mid \mathbf{B}(t)\right\}, \tag{12}$$

where O > 0 is denotes as the weight to scale the penalty and as proved in [6], the drift-plus-penalty upper bound expression in (12) will be:

$$\hat{B} + \sum_{i=1}^{M} \left\{ L_i(t)\mathbb{E}\left[\left(d_i^t - p_i^t\right)|\mathbf{B}(t)\right] + Y_i(t)\mathbb{E}\left[p_{c_i}^t - \gamma_i|\mathbf{B}(t)\right] - O\mathbb{E}\left[\eta_i c_{r_i}^t|\,\mathbf{B}(t)\right]\right\}. \quad (13)$$

where $\hat{B}$ is a constant derived as proved in [6]. The queue backlog $\mathbf{B}(t)$ plays a crucial role in determining the board of offloading and resource allocation control action. By closely monitoring the queue backlog, we can make informed decisions that aim to minimize the upper bound of (13). The technique of opportunistic expectation minimization [34] is applied to ensure that the resources are allocated efficiently and effectively, mitigating any potential bottlenecks or delays. In the context of a control algorithm, it is important to observe that only the second term in the equation is directly linked to the control action. By eliminating the constant terms from the initial observation at the beginning of the t^{th} time-frame, the algorithm can make decisions based on maximizing the following expression:

$$\sum_{i=1}^{M} \left(L_i(t) + O\eta_i\right) c_{r_i}^t - \sum_{i=1}^{M} Y_i(t) p_{c_i}^t, \quad (14)$$

where $c_{r_i}^t$ and $p_{c_i}^t$ are in (5). When a WD has a long data-queue backlog or a high weight, it intuitively leads to an increase in the computation-rates. This is because a larger backlog implies a greater number of tasks waiting to be processed, which in turn requires the WD to operate at a higher rate to clear the backlog. Similarly, a higher weight assigned to a WD signifies its importance or priority, resulting in a higher computation-rate to meet the expectations associated with that weight. On the other hand, WDs that surpass the average power threshold face penalties. These penalties are imposed to discourage excessive power consumption. By penalizing these WDs, the system encourages a more balanced distribution of power consumption. To facilitate our analysis and decision-making process, auxiliary variable $c_{r_{i,o}}^t$ is introduced for each WD i, the collection of all $c_{r_{i,o}}^t$ values, denoted as $\mathbf{c}_{r_O}^t = \left\{c_{r_{i,o}}^t\right\}_{i=1}^{M}$, provides us with a comprehensive overview of the system's current state. To address the per-frame constraints, it is necessary to solve a deterministic per-frame sub-problem within the t^{th} time-frame. This sub-problem takes into consideration the specific constraints that apply to each frame.

By doing so, we can ensure that each t^{th} time-frame of the algorithm is optimized and meets the requirements.

$$\underset{\mathbf{v}^t,\tau^t,\mathbf{f}^t,\mathbf{p_{cO}}^t\mathbf{c_{cO}}^t}{\text{maximize}} \sum_{i=1}^{M} \left(L_i(t) + O\eta_i\right) c_{r_i}^t - \sum_{i=1}^{M} Y_i(t) p_{c_i}^t$$

$$\text{subject to} \quad \sum_{i=1}^{M} \tau_i^t \le 1, \quad (15a)$$

$$f_i^t/z \le L_i(t), c_{r_{i,o}}^t \le L_i(t), \forall i, \quad (15b)$$

$$c_{r_{io}}^t \le \frac{B\tau_i^t}{\omega_u} \log_2\left(1 + \frac{p_{c_{i,O}}^t c_i^t}{\tau_i^t N_p}\right), \forall i, \quad (15c)$$

$$f_i^t \le f_i^{max}, p_{c_{i,o}}^t \le s_i^{max}\tau_i^t, \forall i, \quad (15d)$$

$$v_i^t \in \{0, 1\}, \tau_i^t, f_i^t, p_{c_{i,O}}^t \ge 0, \forall i. \quad (15e)$$

It is important to note that the constraints (15b) and (15c) can be considered equivalent to constraint (8b). This equivalence arises from the fact that there is only one non zero term on the left hand side of (8b) at the optimum. This equivalence allows us to simplify the problem and focus on addressing the long-term constraints efficiently and effectively.

TABLE 1. Summary of notations.

Notation	Description
M	No. of wireless devices
T	Time-frames duration
d_i^t	Raw task data arrival
B	Bandwidth
c_i^t	Channel gain
p_i^t	Bits of data computed
v_i^t	Binary variable
f_i^t	CPU frequency
k	Computing energy efficiency
z	Calculation cycles
$\tau_i^t T$	Time allotted to i^{th} WD
s_i^t	Transmit power
e_i^t	Energy consumed
$c_{r_i}^t$	Computation-rate
$p_{c_i}^t$	Power consumption
$L_i(t)$	Queue length
ω_u	Communication overhead
N_p	Noise power
η_i	Fixed weight
γ_i	Power threshold constraint
$Y_i(t)$	Energy-queue
s	Positive scaling factor
$\mathbf{B}(t)$	Total queue backlog
O	Penalty scale weight
$\bar{\mathbf{o}}^t$	Feature extraction layer variable
$\check{\mathbf{o}}^t$	DNN input
$\acute{w}_i$	Weight ∈ (0,1)
$\left\{\acute{w}_{j\|i}\right\}$	Probability mass function (PMF)
E	ONMF cost function
$\boldsymbol{\vartheta}^t$	DNN parameter
d_s	Data samples

IV. ORTHOGONAL NON-NEGATIVE MATRIX FACTORIZATION-BASED DRL FOR STABLE ONLINE OFFLOADING

In this section, we will present an algorithm that solves the per-frame sub-problems in an online manner to satisfy all the long-term constraints. This approach will enable us to handle the time-frame that poses its challenges. Finding an optimal solution within the given time-frame constraints remains the challenge. This requires careful optimization techniques and algorithms suitable specifically for MINLP problems. By addressing the long-term constraints through

the online solving of sub-problems and tackling the MINLP problem in each t time-frame, we can effectively optimize the overall solution and achieve the desired objectives outlined in the problem formulation. The following sections will go into the details of the proposed algorithm employed to solve the long-term and MINLP constraints respectively. By understanding and implementing these methods, we can overcome the challenges presented by the problem and achieve optimal solutions within the given constraints.

To address this problem, we introduce a variable denotes as $\bar{\mathbf{o}}^{\mathbf{t}} \triangleq \left\{\mathbf{c}_{\mathbf{i}}^{\mathbf{t}}, \mathbf{L_i(t)}, \mathbf{Y_i(t)}\right\}_{\mathbf{i=1}}^{\mathbf{M}}$, which consists of the channel gains $\{\mathbf{c_i}\}_{\mathbf{i=1}}^{\mathbf{M}}$ and the system-queue states $\{\mathbf{L_i(t)}, \mathbf{Y_i(t)}\}_{\mathbf{i=1}}^{\mathbf{M}}$. Based on this observation, we can determine the control action $\left\{\mathbf{v}^{\mathbf{t}}, \mathbf{y}^{\mathbf{t}}\right\}$ which includes the binary offloading decision $\mathbf{v}^{\mathbf{t}}$ and the continuous resource allocation $\mathbf{y}^{\mathbf{t}} \left\{\boldsymbol{\tau}_{\mathbf{i}}^{\mathbf{t}}, \mathbf{f}_{\mathbf{i}}^{\mathbf{t}}, \mathbf{p}_{\mathbf{c_{i,o}}}^{\mathbf{t}}, \mathbf{c}_{\mathbf{r_{i,o}}}^{\mathbf{t}}\right\}_{\mathbf{i=1}}^{\mathbf{M}}$. The system queue states represent the queue lengths and the backlog of the data-tasks for each user. These queue states provide information about the current load on the system and the amount of data waiting to be transmitted. By considering these queue states, we can prioritize the users and allocate resources accordingly, to effectively find the optimal solution with much less complexity compared to conventional approaches.

In our pursuit to uncover the inherent structure and patterns within the dynamic queue's datasets, our objective is to obtain a low-rank approximation. To achieve this, we employ the technique known as Orthogonal Non-negative Matrix Factorization (ONMF). ONMF utilizes the concept of decomposing a non-negative matrix into a product of two smaller non-negative matrices, where each matrix is orthogonal to each other. The matrices can be trained to learn different patterns and relationships between different variables. This could be utilized in the proposed algorithm to counteract bottlenecks by handling high-dimensional data efficiently, hence optimizing resource allocation, ensuring that tasks are processed in a fair manner and so maximizing the average sum computation-rates of users [35].

To apply ONMF, certain constraints must be satisfied. Firstly, the matrix being factorized must be non-negative. This ensures that the decomposed matrices also remain non-negative. The non-negative constraint ensures that the factorized matrices capture meaningful patterns rather than arbitrary combinations of features which is necessary for meaningful interpretations and applications. Furthermore, the factorization should be orthogonal. This means that the columns of the first-factor matrix should be uncorrelated with the columns of the second-factor matrix. This constraint ensures that the components extracted by the factorization are uncorrelated, making it more distinct and meaningful. The idea behind the orthogonality constraint in ONMF is to ensure that the latent factors are aligned in a particular way, resulting in a subset of latent factors that are mutually orthogonal [21], [36].

There are other popular feature extraction techniques such as Non-Negative matrix factorization (NMF), Principal Component Analysis (PCA) and Independent Component Analysis (ICA). NMF is known for its ability to capture the underlying patterns or dependencies in the data. However, NMF has several limitations when applied specifically to time-series data. One key limitation is the potential for correlated factors in NMF, the factors are typically chosen to minimize an objective function, such as the sum of squared residuals or mutual information. However, this can result in correlated factors, which can introduce noise into the decomposition and make it difficult to interpret the results. PCA is particularly good at capturing linear relationships and preserving the structure of the data but on the other hand, it does not require the components to be non-negative and focuses on extracting uncorrelated components that may not provide any clear interpretation. Also, PCA does not aim to preserve the original structure but rather seeks to maximize variance. The reconstructed principal components may not fully represent the original data. ICA can handle non-linear relationships and learn representations that capture the underlying structure of the data but on the other hand, ICA assumes that the sources are statistically independent, which may not always be the case in real-world applications. Therefore, it is obvious that ONMF does not impose any independence constraints, allowing it to capture complex relationships between data components [37], [38].

The strict orthogonality formulation constraint leads the authors to use the ONMF instead of other feature extraction as this constraint alleviates addressing the potential problem of over-fitting and linear dependence between basis vectors in deep neural networks (DNN). This has advantages for increasing the classification performance of the subsequent DNN algorithm in the proposed MFDROO algorithm leading to more robust and reliable predictions [39].

ONMF is based on the maximum-entropy principle (MEP-ONMF) [26], which seeks to maximize an entropy-based objective function subject to orthogonal non-negativity constraints where each column(row) ultimately has exactly one non-zero element. The entropy-based objective function represents the degree of randomness or disorder in the data, and the constraints ensure that the learned factor matrices are non-negative and orthogonal. The idea behind MEP-ONMF is to achieve the maximum sparsity for the orthogonal factor in the matrix. Sparsity refers to the presence of a large number of zero elements in the matrix. By ensuring that no row or column in the matrix is entirely zero when R (the features matrix) or U (the mixing matrix) is orthogonal, MEP-ONMF achieves maximum sparsity. The importance of orthogonality in the features matrix lies in its ability to represent the original data in a more compact and meaningful way. Each row or column must have only one non-zero element so that MEP-ONMF can ensure that the features are mutually independent and do not overlap with each other.

In our proposed MFDROO algorithm, the ONMF is interpreted as a facility location problem (FLP). By considering facilities as equivalent to the features matrix ($\mathbf{R}$) and the association probabilities of the MDs to facilities as the

mixing matrix ($\mathbf{U}$), we can gain a deeper understanding of the underlying concepts. FLP is an NP-hard challenging optimization problem that aims to allocate facilities to MDs while minimizing the average distance. One approach to solving FLP is by using the Deterministic Annealing (DA) algorithm [40]. DA is an iterative algorithm that has been successful in addressing FLPs and is efficient in handling large problem instances. By relaxing the strict assignment and introducing probabilistic assignments, the iterative nature of DA allows for refining the probability distribution over multiple iterations. Starting with an initial distribution, the algorithm gradually updates the probability based on the available data and determine the probability distribution by using the maximum-entropy-principle (MEP) [41]. Through this iterative process, The MEP ensures that the probability distribution is maximally unbiased and consistent with the available information. By employing this principle, DA can find a solution that optimally balances the assignment of MDs to facilities. By combining the objective function with the constraints, the proposed algorithm seeks to identify the most informative and interpretable factors that capture the essential features of the data in data-queues and quantify the weights of these features.

For approximating the given data-queue matrix $\mathbf{L(t)}$, it is expressed as the product of two non-negative matrices, namely $\mathbf{R}$ and $\mathbf{U}$. The primary goal of this approach is to capture the crucial information embedded in the queue data set while reducing its dimensionality. The ONMF technique is particularly suitable for our purpose due to its ability to handle non-negative data and its ability to extract non-overlapping, interpretable features representing the original data-queue components. By enforcing non-negativity and orthogonality constraints on the factorized matrices $\mathbf{R}$ and $\mathbf{U}$, we can ensure that the resulting approximation remains non-negative, which is often desirable in real-world scenarios. Then the variable $\bar{o}^{\mathbf{t}}$ expression defined to hold $\tilde{\mathbf{L}}(\mathbf{t})$, donated as $\breve{\mathbf{o}}^{\mathbf{t}} \triangleq \left\{\mathbf{c_i^t}, \tilde{\mathbf{L}}_\mathbf{i}(\mathbf{t}), \mathbf{Y_i(t)}\right\}_{\mathbf{i=1}}^{\mathbf{M}}$, where $\tilde{\mathbf{L}}(\mathbf{t}) = \left\{\tilde{\mathbf{L}}_\mathbf{i}(\mathbf{t})\right\}_{\mathbf{i=1}}^{\mathbf{M}}$. It provides valuable insights and contributes to the overall understanding of the system.

To tackle the resource allocation problem, we will adopt a proven algorithm that has been discussed in [6]. This algorithm will be used to optimize the variable $\mathbf{y^t}$, then we denote $\mathbf{G}\left(\mathbf{v^t}, \breve{\mathbf{o}}^{\mathbf{t}}\right)$ as the optimal value of the above MINLP mentioned problem (15) by optimizing $\mathbf{y^t}$ given the offloading decision $\mathbf{v^t}$ and the parameter $\breve{\mathbf{o}}^{\mathbf{t}}$. By solving this MINLP problem, we are essentially finding the optimal offloading decision, denoted as $\left(\mathbf{v^t}\right)^*$, where it's the argmax of $\mathbf{G}\left(\mathbf{v^t}, \breve{\mathbf{o}}^{\mathbf{t}}\right)$.

In the field of offloading decisions, obtaining the optimal solution for the problem of $\left(\mathbf{v^t}\right)^*$ generally requires enumerating $\mathbf{2^M}$ possibilities. This process, however, results in a significantly high computational complexity, even when the number of users M in the network is small. The time-consuming nature of this approach becomes particularly apparent when M is large. Moreover, the existing search methods encounter significant time constraints, particularly when dealing with a large number of M. Consequently, neither of these methods proves to be practical for online decision-making scenarios. In the light of this, we introduce a groundbreaking algorithm that combines Orthogonal Non-negative Matrix Factorization (ONMF) and Deep Reinforcement Learning (DRL) called MFDROO to address the problem of stable online offloading in large networks. MFDROO aims to construct a policy, denoted as þ, which effectively maps the input to the optimal action $\left(\mathbf{v^t}\right)^*$ with remarkably low complexity. Furthermore, the duration needed for MFDROO to process the observations and produce control actions $\left\{\mathbf{v^t}, \mathbf{y^t}\right\}$ remains within acceptable limits, even with a large user population (e.g., M=200). This implies that the efficiency of MFDROO in this regard is noteworthy as it proves to be an efficient and practical approach for handling a substantial number of users while maintaining stable data-queues and high computation-rates.

A. DESCRIPTION OF THE PROPOSED ALGORITHM

A visual representation of the system architecture, highlighting the flow of data and decision-making within the MFDROO algorithm framework is illustrated in Fig.2. MFDROO is comprised of five main modules. The first module is MEP-ONMF, which takes the input variable expression $\bar{\mathbf{o}}^{\mathbf{t}}$ and generates the new variable expression $\breve{\mathbf{o}}^{\mathbf{t}}$ producing a new matrix of data-queue $\tilde{\mathbf{L}}(\mathbf{t})$, the second module is the actor module which takes the input $\breve{\mathbf{o}}^{\mathbf{t}}$ and outputs a set of candidate offloading actions $\mathbf{v_i^t}$, These candidate offloading actions are potential choices for offloading tasks from the local device either locally or remotely to the edge servers. The third module is the critic module, which assess the candidate offloading-actions $\mathbf{v_i^t}$ generated by the actor-module, based on this evaluation, the critic-module selects the best offloading-action $\mathbf{v^t}$. The fourth module is the policy update module is a crucial component in improving the actor-module over time. This module is responsible for analyzing and updating the policies that guide the decision-making process of the actor module. By continuously evaluating the performance of the actor module and collecting feedback from the system, the policy update module ensures that the actor module evolves and adapts to changing conditions. The last fifth module is the queuing module; it plays a vital role in updating the system queue states $\{\mathbf{L_i(t)}, \mathbf{Y_i(t)}\}_{\mathbf{i=1}}^{\mathbf{M}}$ after executing the offloading actions. This module manages the queue of tasks or requests waiting to be processed by the system. It keeps track of the current queue states. Through repeated interaction with the random environment $\{\mathbf{c_i(t)}, \mathbf{d_i(t)}\}_{\mathbf{i=1}}^{\mathbf{M}}$, the five modules operate sequentially and iteratively. They gather data, analyze, and process it, make decisions, execute actions, and learn from the outcomes. This systematic approach enables the system to adapt and respond effectively to the ever-changing environment. The following subsections give a detailed description of each module.

1) MEP-ONMF MODULE

Consider the data matrix $\mathbf{L(t)} \in \mathbb{R}_{+}^{\mathrm{t}\times \mathrm{M}}$, where t represents t^{th} time-frame and M represents the number of WDs as presented in (1). The product of two non-negative matrices $\mathbf{R} \in \mathbb{R}_{+}^{\mathrm{t}\times a}$ and $\mathbf{U} \in \mathbb{R}_{+}^{a\times \mathrm{M}}$ (where the so called inner dimension $a \ll \min(\mathrm{t}, \mathrm{M})$) is used to achieve the objective of approximating the data matrix. To formalize this objective, we can define an optimization problem.

$$\begin{aligned} &\min \underline{\mathrm{D}}(\mathbf{L}(\mathrm{t}), \mathbf{RU}), \\ &\mathrm{s.t}\ .\ \mathrm{R_{ij}}, \mathrm{U_{ij}} \geq 0 \end{aligned} \tag{16}$$

where $\underline{\mathrm{D}}(\mathbf{L}(\mathrm{t}), \mathbf{RU})$ represents the distance function between the two matrices $\mathbf{L}(\mathrm{t})$ and $\mathbf{RU}$ which measures the reconstruction error and both i^{th}, j^{th} represents node (M) and facility respectively. The goal is to minimize this error to obtain a good approximation of the original data matrix.

One of the key requirements of the ONMF is that one of the matrices, either $\mathbf{R}$ or $\mathbf{U}$ needs to be orthogonal. By imposing this orthogonality constraint, the ONMF formulates the following optimization problem:

$$\begin{aligned} &\min\ \underline{\mathrm{D}}(\mathbf{L}(\mathrm{t}), \mathbf{RU}), \\ &\mathrm{s.t.}\ \mathbf{UU^{T}} = \mathbf{I} \\ &\quad \mathrm{R_{ij}}, \mathrm{U_{ij}} \geq 0 \quad \forall \mathrm{i}, \mathrm{j} \end{aligned} \tag{17}$$

where $\mathbf{I} \in \mathbb{R}_{+}^{a\times a}$ is an identity matrix. In addition to the orthogonality constraint on $\mathbf{U}$, an alternative orthogonality constraint can also be applied, which states that $\mathbf{RR^{T}} = \mathbf{I}$. Having both non-negativity and orthogonality constraints means matrices can take on to the requirement of having only one non-zero element in every row or column when $\mathbf{R}(\mathbf{U})$ is orthogonal.

Expressing the ONMF optimization problem as an FLP one, it can be formulated as:

$$\min_{y_{\mathrm{j}} X_{\mathrm{j|i}}} \sum\nolimits_{\mathrm{ij}=1}^{\mathrm{M}a} \acute{\mathrm{w}}_{\mathrm{i}} \underline{\mathrm{d}}(\mathrm{L_i}, y_{\mathrm{j}}) X_{\mathrm{j|i}}, \tag{18}$$

where $\underline{\mathrm{d}}(\mathrm{L_i}, y_{\mathrm{j}})$ is a function of $\mathrm{L_i} - y_{\mathrm{j}}$ that represents the distance between the i^{th} node and j^{th} facility, $\acute{\mathrm{w}}_{\mathrm{i}} \in (0, 1)$ is a priori known weight that gives the relative importance of the i^{th} WD node ($\sum_{\mathrm{i}} \acute{\mathrm{w}}_{\mathrm{i}} = 1$ and $\acute{\mathrm{w}}_{\mathrm{i}} = 1/\mathrm{M}$ when all the nodes are of equal importance), and the binary variable $X_{\mathrm{j|i}} \in \{0, 1\}$ is 1 only when i^{th} node is assigned the j^{th} facility. Using the notations $\mathbf{L(t)} = [\mathrm{L_1 L_2} \ldots \mathrm{.L_M}] \in \mathbb{R}_{+}^{\mathrm{t}\times \mathrm{M}}$, $\mathbf{R} = [y_1 y_2 \ldots y_j] \in \mathbb{R}_{+}^{\mathrm{t}\times a}$ and $\mathbf{U} = [X_{\mathrm{j|i}}] \in \mathbb{R}_{+}^{a\times \mathrm{M}}$, where then $\underline{\mathrm{D}}(\mathbf{L}(\mathrm{t}), \mathbf{RU}) = \sum_{\mathrm{ij}=1}^{a\mathrm{M}} \acute{\mathrm{w}}_{\mathrm{i}} \underline{\mathrm{d}}(\mathrm{L_i}, y_{\mathrm{j}}) X_{\mathrm{ji}}$. Therefore, for any given non-negative matrix $\mathbf{L(t)} \in \mathbb{R}_{+}^{\mathrm{t}\times \mathrm{M}}$, we can interpret the i^{th} columns of the $\mathbf{L}(\mathrm{t})$ matrix as the node's locations (feature vector). Their corresponding facility set is denoted as $\{y_{\mathrm{j}}(\mathrm{t})\}_{\mathrm{j}=1}^{a}$ and this results that the facility-location matrix $\mathbf{R} \in \mathbb{R}_{+}^{\mathrm{t}\times a}$ and the assignment matrix $\mathbf{U} \in \{0, 1\}^{a\times \mathrm{M}}$ are both non-negative, $\mathbf{U}$ is an orthogonal matrix and $\mathbf{L}(\mathrm{t}) \approx \mathbf{RU}$.

As a result of the complexity that arises from the constraint on $X_{\mathrm{j|i}} \in \{0, 1\}$ to be binary variables, the DA algorithm is employed to alleviate these hard assignments and is replaced by soft assignments $\acute{\mathrm{w}}_{\mathrm{j|i}} \in [0, 1]$, hence the optimization problem can be formulated as:

$$A = \underline{\mathrm{D}}\left(\left\{\acute{\mathrm{w}}_{\mathrm{j|i}}\right\}, \left\{h_{\mathrm{j}}\right\}\right) - \frac{1}{\beta}\mathcal{H}\left(\left\{\acute{\mathrm{w}}_{\mathrm{j|i}}\right\}\right), \tag{19}$$

where $\left\{\acute{\mathrm{w}}_{\mathrm{j|i}}\right\}$ is the probability mass function (PMF) that relates i^{th} data points $\mathrm{L_i}$ to the features h_{j}, while $\underline{\mathrm{D}}\left(\left\{\acute{\mathrm{w}}_{\mathrm{j|i}}\right\}, \left\{h_{\mathrm{j}}\right\}\right) = \sum_{\mathrm{i}=1}^{\mathrm{M}} \acute{\mathrm{w}}_{\mathrm{i}} \sum_{\mathrm{j}=1}^{a} \acute{\mathrm{w}}_{\mathrm{j|i}} \underline{\mathrm{d}}(\mathrm{L_i}, h_{\mathrm{j}})$ is the cost function's expected value in (17) and the Shannon entropy $\mathcal{H}\left(\left\{\acute{\mathrm{w}}_{\mathrm{j|i}}\right\}\right) = \sum_{\mathrm{i}=1}^{\mathrm{M}} \acute{\mathrm{w}}_{\mathrm{i}} \sum_{\mathrm{j}=1}^{a} \acute{\mathrm{w}}_{\mathrm{j|i}} \log \acute{\mathrm{w}}_{\mathrm{j|i}}$ measures the uncertainty of the associated PMF $\left\{\acute{\mathrm{w}}_{\mathrm{j|i}}\right\}$. $\frac{1}{\beta}$ examines how the target cost function affects the formulation, and how much randomness is introduced because of the PMF [27], noting that A is convex in consideration to $\left\{\acute{\mathrm{w}}_{\mathrm{j|i}}\right\}$ for a fixed β.

In the given optimization problem formulation, we view all the characteristics (defined by value $\left\{h_{\mathrm{j}}\right\}$) as having equal importance, a relative weight $w_{\mathrm{r_j}}$ can be assigned to the j^{th} feature by assuming that $w_{\mathrm{r_j}}$ units of j^{th} feature occur at value h_{j} and by considering the MEP and employing the Euclidean distance metric for $\underline{\mathrm{d}}(.,.)$, we can determine the optimal solutions by setting $\frac{\partial A}{\partial \acute{\mathrm{w}}_{\mathrm{j|i}}} = 0$. By doing so, we can then reinterpret the distribution in the PMF as:

$$\acute{\mathrm{w}}_{\mathrm{j|i}} = \frac{w_{\mathrm{r_j}} \mathrm{e}^{-\beta \|\mathrm{L_i} - h_{\mathrm{j}}\|^2}}{\sum_{\mathrm{g}=1}^{a} w_{\mathrm{r_g}} \mathrm{e}^{-\beta \|\mathrm{L_i} - h_{\mathrm{g}}\|^2}}, \tag{20}$$

Setting $\frac{\partial A}{\partial h_{\mathrm{j}}} = 0$ and $\frac{\partial A}{\partial w_{\mathrm{r_j}}} = 0$ yields the locally optimal values of $\left\{h_{\mathrm{j}}\right\}, \left\{w_{\mathrm{r_j}}\right\}$,which yield:

$$h_{\mathrm{j}} = \sum\nolimits_{\mathrm{i}=1}^{\mathrm{M}} \acute{\mathrm{w}}_{\mathrm{i|j}} \mathrm{L_i} \text{ and } w_{\mathrm{r_j}} = \sum\nolimits_{\mathrm{i}=1}^{\mathrm{M}} \acute{\mathrm{w}}_{\mathrm{i}} \acute{\mathrm{w}}_{\mathrm{j|i}}, \tag{21}$$

where $\acute{\mathrm{w}}_{\mathrm{j|i}} = \acute{\mathrm{w}}_{\mathrm{i}} \acute{\mathrm{w}}_{\mathrm{j|i}} / w_{\mathrm{r_j}}$. Thus, at a given we find the optimal (local) solutions for $\left\{\acute{\mathrm{w}}_{\mathrm{j|i}}\right\}$ and $\left\{h_{\mathrm{j}}\right\}$ at a fixed β, these optimal values can then be used to construct matrices $\mathbf{R}$ and $\mathbf{U}$ by simply putting $\mathbf{R}_{:\mathrm{j}} = h_{\mathrm{j}}$ and $\mathbf{U}_{\mathrm{ij}} = \acute{\mathrm{w}}_{\mathrm{j|i}}$, where $\mathbf{R}_{:\mathrm{j}}$ is then will be the updated input to the actor module $\tilde{\mathbf{L}}(\mathrm{t}) \leftarrow \mathbf{R}_{:\mathrm{j}}$.

DA success in optimizing the problem and maximizing the entropy by optimizing each of these non-negative values with respect to θ_{i} to minimize the ONMF cost function:

$$\begin{aligned} \mathrm{E} &= \sum\nolimits_{\mathrm{i}=1}^{\mathrm{M}} \|\mathrm{L_i} - \theta_{\mathrm{i}} h_{\mathrm{i}}\|^2 \\ \theta_{\mathrm{i}} &= \frac{\mathrm{L_i^{T}} h_{\mathrm{i}}}{\|h_{\mathrm{i}}\|^2}, \end{aligned} \tag{22}$$

and replace the ones in the i^{th} columns of U with (θ_{i}) s.

2) THE ACTOR-MODULE

The actor module is a crucial component of the system, comprising two main components: a Deep Neural Network (DNN) with sigmoid activation function at the output layer and an action quantizer. This module enables the system to make informed decisions and take actions based on the current state and policy. At the start of each t^{th} time-frame, the DNN parameter is represented as $\boldsymbol{\vartheta}^{\mathrm{t}}$. It is important to note that $\boldsymbol{\vartheta}^{\mathrm{t}}$ is randomly initialized according to the standard normal distribution when t is equal to 1. In this context, the input

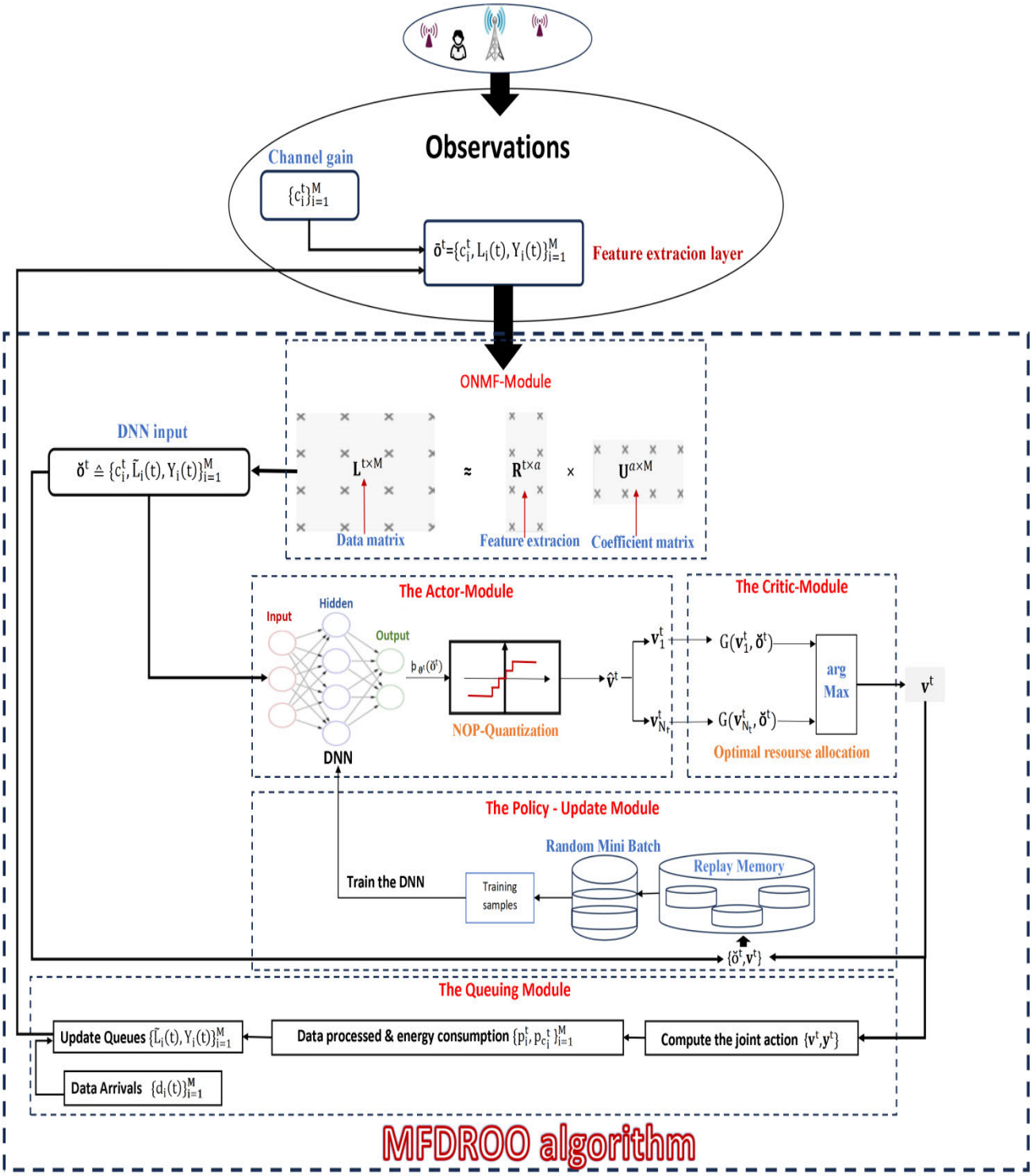


FIGURE 2. MFDROO algorithm framework.

is represented as $\breve{\mathbf{o}}^t$, which corresponds to the observation at a given time t. The DNN processes this input and produces an output $\hat{\mathbf{v}}^t \in [0, 1]^M$ presents a relaxed offloading decision. Then the input-output formalization is expressed as:

$$\mathfrak{p}_{\vartheta^t}: \breve{\mathbf{o}}^t \mapsto \hat{\mathbf{v}}^t = \left\{ \hat{v}_i^t \in [0, 1], i = 1, \cdots, M \right\}, \quad (23)$$

We proceed by discretizing the continuous variable $\hat{\mathbf{v}}^t$ into N_t, where N_t is a feasible time-dependent candidate binary offloading actions, thus can express the quantization function as:

$$qf_{N_t}: \hat{\mathbf{v}}^t \mapsto \mathbf{of}^t = \left\{ v_j^t | v_j^t \in \{0, 1\}^M, j = 1, \cdots, N_t \right\}, \quad (24)$$

where $\mathbf{of}^t$ refers to the offloading-action set in the t^{th} time-frame. To guarantee effective training convergence, the noise order-preserving (NOP) quantization function [32] is employed. This enables the generation of a diverse set of candidate actions $N_t \leq 2M$ and based on the assumption that N_t is an even number, it generates the first $N_t/2$ actions using the order-preserving quantizer (OPQ) described in [17] to process the variable $\hat{\mathbf{v}}^t$. To calculate the first action $\mathbf{v}_1^t = [v_{1,1}^t, \ldots, v_{1,M}^t]$, a specific formula is applied as:

$$v_{1,i}^t = \begin{cases} 1 & \hat{v}_i^t > 0.5, \\ 0 & \hat{v}_i^t \leq 0.5, \end{cases} \quad \text{for } = 1, \ldots, M, \tag{25}$$

where $\hat{v}_{(i)}^t$ represents the i^{th} ordered entry of $\hat{v}^t$ and $\hat{v}_{(i)}^t$'s are used as the decision threshold to quantize $\hat{\mathbf{v}}^t$, where the next $\frac{N_t}{2} - 1$ actions are generated by arranging the entries of $\hat{\mathbf{v}}^t$ on the basis of the distance to 0.5. Then the n^{th} action v_n^t, for n =2,..., $N_t/2$ is obtained as a result of the entry wise comparisons of $\hat{\mathbf{v}}^t$ and $v_{(n-1)}^t$.

That is:

$$v_{n,i}^t = \begin{cases} 1, \hat{v}_i^t > \hat{v}_{(n-1)}^t \text{or} \left\{ \hat{v}_i^t = \hat{v}_{(n-1)}^t \text{ and } \hat{v}_{(n-1)}^t \leq 0.5 \right\} \\ 0, \hat{v}_i^t < \hat{v}_{(n-1)}^t \text{or} \left\{ \hat{v}_i^t = \hat{v}_{(n-1)}^t \text{ and } \hat{v}_{(n-1)}^t > 0.5 \right\} \end{cases},$$
$$\text{for } n = 2, \ldots, \frac{N_t}{2} \& i = 1, \ldots, M \tag{26}$$

Lastly, the remaining $N_t/2$ actions are derived from a noisy version of $\hat{\mathbf{v}}^t$ denoted as $\tilde{\mathbf{v}}^t = \text{Sigmoid}(\hat{\mathbf{v}}^t + \boldsymbol{n})$, where $\boldsymbol{n}$ is the random Gaussian noise and then by replacing $\hat{\mathbf{v}}^t$ with $\tilde{\mathbf{v}}^t$ in (25) and (26) and applying OPQ, the remaining $N_t/2$ actions v_n^t will be derived.

3) THE CRITIC-MODULE

The critic-module plays a crucial role in evaluating the best offloading action in the context of DRL. By analyzing the optimal resource allocation problem, the critic-module can identify and select the best offloading action, denoted as $\mathbf{v}^t$ from the set of available offloading actions $\{v_i^t\}$. MFDROO selects the best action as:

$$\mathbf{v}^t = \arg \max_{v_j^t \in \mathbf{of}_t} G\left(v_j^t, \breve{\mathbf{o}}^t\right), \tag{27}$$

where $G\left(v_j^t, \breve{\mathbf{o}}^t\right)$ is based on resource allocation optimization given v_j^t.

4) THE POLICY-UPDATE MODULE

MFDROO is a reinforcement learning algorithm that utilizes labeled input-output samples $\left(\breve{\mathbf{o}}^t, \mathbf{v}^t\right)$ to update the policy of the DNN. To facilitate this process, MFDROO maintains a replay memory that is updated with the most recent d_s data samples. Before training the DNN, the replay memory is initially empty. However, training commences once more than half of the desired number of data samples, denoted as $d_s/2$ have been collected. This approach ensures that there is a sufficient amount of data in the replay memory to start training the DNN effectively. The DNN is exposed and trained to a diverse range of inputs at Λ_T regular intervals and we check if mod $(t, \Lambda_T) = 0$, then we selecting randomly a batch of data samples $\left\{\left(\breve{\mathbf{o}}^\tau, \mathbf{v}^\tau\right), \tau \in T_s^t\right\}$ which helps to maintain the model's generalization ability and avoid the risk of over fitting and then updating the parameter of the DNN by employing the Adam algorithm [42], which minimizes its average cross-entropy loss function $EF\left(\boldsymbol{\vartheta}^t\right)$ denoted as:

$$EF\left(\boldsymbol{\vartheta}^t\right) = -1/\left|T_s^t\right| . \sum\nolimits_{\tau \in T_s^t} \left[\left(\mathbf{v}^\tau\right)^{\mathbf{T}} \log \mathfrak{b}_{\boldsymbol{\vartheta}^t}\left(\breve{\mathbf{o}}^\tau\right) + \left(1 - \mathbf{v}^\tau\right)^{\mathbf{T}} \log\left(1 - \mathfrak{b}_{\boldsymbol{\vartheta}^t}\left(\breve{\mathbf{o}}^\tau\right)\right) \right], \tag{28}$$

where $\left|T_s^t\right|$ is the sample-batch size and T_s^t is the set of selected samples' time indices. Lastly, after the training completes, the actor module parameter will be updated in the next time-frame to $\boldsymbol{\vartheta}^{t+1}$.

5) THE QUEUING-MODULE

The system incorporates the joint computation offloading $\mathbf{v}^t$ and resource allocation action $\mathbf{y}^t$ to effectively handle the processing of data $\left\{p_i^t\right\}_{i=1}^M$ and the consumption of energy $\left\{p_{c_i}^t\right\}_{i=1}^M$ as specified in (5). To adopt the dynamic nature of the system, we considering $\left\{p_i^t, p_{c_i}^t\right\}_{i=1}^M$ and the observed data arrivals $\left\{d_i^t\right\}_{i=1}^M$ during the t^{th} time-frame and then the queuing module continuously update the data and the energy-queues $\{L_i(t+1), Y_i(t+1)\}_{i=1}^M$ utilizing (7) and (9) at the start of the $(t+1)^{th}$ time-frame.With the wireless channel gains observation $\left\{c_i^{t+1}\right\}_{i=1}^M$, the system has the new input parameter $\bar{\mathbf{o}}^{t+1} = \left\{c_i^{t+1}, L_i(t+1), Y_i(t+1)\right\}_{i=1}^M$ and a new iteration is initiated from the MEP-ONMF first module.

MFDROO algorithm described above provides a notable benefit through its ability to decrease the dimensionality of the input data-queues matrix and extract the most crucial features from the beginning. By doing so, the algorithm can concentrate on learning the optimal state-action pairs, which ultimately results in the creation of a more efficient policy. Significantly, this process is carried out efficiently, ensuring that computational resources are utilized effectively. The dimensionality reduction allows the algorithm to identify the key patterns and relationships within the data, enabling it to make informed decisions regarding the best actions to take in a given state. This targeted approach not only improves the algorithm's performance but also enhances its ability to adapt and generalize to new situations. Consequently, the MFDROO algorithm offers a valuable advantage for applications that require efficient and effective policy development based on high-dimensional input data. The pseudo-code for MFDROO is summarized in Algorithm 1.

V. SIMULATION RESULTS

In this section, simulations are conducted to provide valuable insights into the performance and robustness of the proposed MFDROO algorithm. All the computations are executed on

Algorithm 1 MFDROO Algorithm for Solving Offloading Decision Action Problem

input: the wireless channel gain c_i^t of WDs for $i = 1, 2, .., M$ and $t = 1, 2, \ldots$;
output: Control actions $\{\mathbf{v}^t, \mathbf{y}^t\}$;
Initialization set the DNN parameters $\boldsymbol{\vartheta}^1$ randomly.
Empty the initial data-queue $L_i(1) = 0$ and energy-queue $Y_i(1) = 0$, for $i = 1, 2, .., M$
for $t = 1, 2, \ldots, K$ **do**
  Set the input $\bar{\mathbf{o}}^t = \{c_i^t, L_i(t), Y_i(t)\}_{i=1}^M$;
  Apply ONMF on $\{L_i(t)\}_{i=1}^M$ and then update $L_i(t) \rightarrow \tilde{\mathbf{L}}_i(t)$;
  Update $\bar{\mathbf{o}}^t \rightarrow \breve{\mathbf{o}}^t \triangleq \left\{c_i^t, \tilde{\mathbf{L}}_i(t), Y_i(t)\right\}_{i=1}^M$;
  Generate a relaxed-offloading action $\hat{v}^t = \mathfrak{p}_{\boldsymbol{\vartheta}^t}(\breve{\mathbf{o}}^t)$ through the DNN;
  Quantize $\hat{v}^t$ into N_t binary actions $\left\{v_i^t \mid i = 1, \ldots., N_t\right\}$ using NOP method;
  Solve $G\left(v_i^t, \breve{\mathbf{o}}^t\right)$ by executing the optimal resource allocation y_i^t in [6] for each v_i^t;
  Select the best offloading decision action $\mathbf{v}^t = \arg\max_{\mathbf{v}_i^t} G\left(v_i^t, \breve{\mathbf{o}}^t\right)$ and compute the join action $(\mathbf{v}^t, \mathbf{y}^t)$;
  Update the replay memory by adding the action pair $(\breve{\mathbf{o}}^t, \mathbf{v}^t)$;
  If $\mathrm{mod}(t, \Lambda_T) = 0$ then
    Randomly select a batch of data samples $\left\{\left(\breve{\mathbf{o}}^\tau, \mathbf{v}^\tau\right), \tau \in T_s^t\right\}$ from the replay memory;
    Train the DNN using $\left\{\left(\breve{\mathbf{o}}^\tau, \mathbf{v}^\tau\right), \tau \in T_s^t\right\}$ and update $\boldsymbol{\vartheta}^t$ using Adam algorithm;
  end
  $t \leftarrow t + 1$;
  Update $\left\{\tilde{\mathbf{L}}_i(t), Y_i(t)\right\}_{i=1}^M$ based on $(\mathbf{v}^{t-1}, \mathbf{y}^{t-1})$ and data arrival observation $\left\{d_i^{t-1}\right\}_{i=1}^M$ using (7) and (9)
end

a Visual Studio Code platform. The simulation scenario is set up based on a variation in the number of users M=10 WDs to 200 WDs. In this study, we shall assume that the average channel gain $\bar{c}_i$ is governed by a path-loss model $\bar{c}_i = A_g \left(\frac{3*10^8}{4\pi f_c ds_i}\right)^{p_l}$, $i = 1, \ldots, M$, in which $A_g = 3$ represents the antenna gain, $f_c = 915$ MHz represents the carrier frequency, $p_l = 3$ represents the path loss exponent, and $ds_i = 120 + 15(i - 1)$, for $i = 1, \ldots, M$ represents the distance between the WD and ES in meters. A line-of-sight link gain of $0.3\bar{c}_i$ is based on an i.i.d. Rician distribution. With respect to noise power N_p, the noise power spectral density should be considered $N_p = Bn_s$ with $n_s = -174$ dBm/Hz and for the fixed weight η_i, if i is an odd number then $\eta_i = 1.5$ otherwise, $\eta_i = 1$. In all WDs, the arrival of task data follows an exponential distribution with an equal average rate $\mathbb{E}\left[d_i^t\right] = \lambda_i, i = 1, \ldots., M$. Detailed information regarding other parameters is given in Table 2.

TABLE 2. Simulation-parameters.

Parameters	Values
Bandwidth … **B**	2 MHz
Maximum local computing frequency … $\mathbf{f_i^{max}}$	0.3 GHz
Maximum transmit power … $\mathbf{s_i^{max}}$	0.1 watt
Data arrival rate … $\boldsymbol{\lambda_i}$	2.5, ..., 3.8 Mbps
Time-frame … T	1 second
Computation cycles … z	100
Power threshold … $\boldsymbol{\Upsilon_i}$	0.08 watt
Communication overhead … $\boldsymbol{\omega_u}$	1.1
Capacity of memory … **q**	1024
Positive scaling factor … *s*	1000
Computing energy efficiency … $\boldsymbol{k}$	10^{-26}
Time slots … $\boldsymbol{\Lambda_T}$	10
Sample batch size … $\left\|\mathbf{T_s}^t\right\|$	32

MFDROO's actor module is based on a fully connected multi layer perceptron. This multi layer perceptron is comprised of four layers: an input layer, two hidden layers, and an output layer. The input layer serves as the entry point for the system, receiving the relevant data points. The first hidden layer consists of 120 hidden neurons, which are responsible for processing and transforming the input data. The second hidden layer, on the other hand, comprises 80 hidden neurons, further refining the processed information. Finally, the output layer generates the desired output, which is utilized by the system to make informed decision actions.

For performance comparison, the benchmark method LyDROO [6] is considered. The performance of the LyDROO method is verified through computer simulations. The results indicate that this method appears to perform close to optimally. In light of this, we regard LyDROO as the MFDROO algorithm's performance benchmark. However, it is important to note that LyDROO does suffer from limitations considering the system instability under network conditions. The algorithm may perform well in the specific network setup studied, but it remains unclear how it will handle scaling to large networks.

Through simulations, it can be figured out that the benchmark algorithm suffers from significant system instability in complex networks & under certain conditions. Scalability is a critical issue in mobile-edge computing networks. As the number of nodes and applications increases, it becomes

challenging to handle a large number of offloading requests while maintaining low energy consumption. So, regarding the benchmark LyDROO, when the number of users (M) is large or when the data rate exceeds 3 Mbps for the same number of users (M=10), LyDROO experiences notable instability. These findings highlight the limitations of LyDROO in scenarios where there is a high number of users or a higher data rate. It is crucial to consider these factors in real-world applications.

By using Monto-Carlo simulation, the performance of MFDROO is executed and evaluated compared to that of LyDROO. The proposed MFDROO algorithm is first evaluated and compared with the LyDROO benchmark method. Its worth mentioned that the main objective of the proposed MFDROO algorithm is maximizing the long-term average weighted sum computation-rate subject to data-queue stability. However, power consumption should be taken into consideration as it is one of the system performance limitations. The performance of both algorithms is evaluated in terms of average data-queue length, the weighted-sum computation-rate and average power consumption over time.

The proposed MFDROO algorithm through comprehensive simulations proved that it's an innovative algorithm as it elaborates on the advancements regarding stability in complex networks with variable data-arrival rates and focuses on addressing these challenges by developing a robust algorithm to ensure stability of the mobile-edge computing networks. The algorithm incorporates an innovative framework to overcome the stability challenges posed by the benchmark method. One of the key aspects is its ability to adapt dynamically to variations in the data-arrival rate and/or number of users. By leveraging the MFDROO advanced framework, the algorithm can anticipate and mitigate the effects of these variations, ensuring that the network remains stable and delivers reliable performance, making the proposed MFDROO algorithm an indispensable tool for ensuring the efficient operation of mobile-edge computing networks.

This section is organized as follows. Section A examines the novelty of the proposed MFDROO algorithm in comparison with LyDROO under the effect of the different data-arrival rates at a constant number of users (M=10). Section B illustrates the impact of varying the number of users M, ranging from 20 to 200 users examined under different data-arrival rates. Section C examines the Performance of the proposed MFDROO algorithm under different hyper parameters(Batch size & Learning rate). Lastly, Section D examines additional performance evaluation of the proposed MFDROO algorithm.

A. DATA-ARRIVAL RATES EFFECT

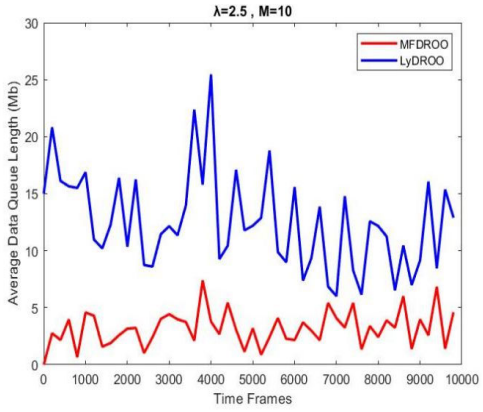
To capture the behavior of the proposed algorithm accurately, we consider i.i.d. realizations of random events in the context of 10,000 time-frames, where each point in the figure represents a moving-window average of 200 time-frames.

This section shows the novelty of the proposed MFDROO algorithm compared to the LyDROO benchmark method under the effect of data-arrival rates variation at a specific number of users (M=10).

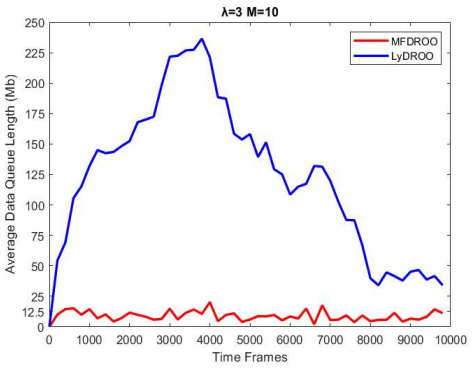
Fig. 3 illustrates the results for two different data arrival rates, $\lambda_i = 2.5$ and 3 Mbps at same number of users M=10. In Fig. 3(a), it is evident that both MFDROO and LyDROO maintain stable data-queues for the given arrival data rates ($\lambda_i = 2.5$ and 3 Mbps). However, in Fig. 3(b), MFDROO outperforms LyDROO in terms of queue length and computation-rate performance. The queue length is significantly shorter in MFDROO over all the time-frames, leading to higher computation-rate performance compared to LyDROO and in Fig. 3(c), it is evident that the proposed algorithm exhibits a higher average power consumption compared to the benchmark method mentioned. However, when TDMA is employed, the power consumption of MFDROO compared to other related papers work in this topic as referred in [13], [43], [44], [45], [46], and [47] is within an acceptable range.

One notable advantage of MFDROO is its fast convergence as it approaches the optimal policy very fast, even in highly dynamic queuing systems. On the other hand, LyDROO requires more time to learn until it reaches the optimal policy for offloading in the early stages, where the amount of data increases rapidly when $t \leq 3000$ (at $\lambda_i = 3$Mbps). This means that MFDROO adapts quickly to changes in the queuing system, while LyDROO may take longer to achieve optimal performance.

Fig. 4 illustrates an evaluation of the impact of the system parameters while fixing the number of users (M) at 10 and varying the data-arrival rate λ_i range from 3.2 to 3.8 Mbps. Upon analysis, we observe from Fig. 4(a) that the average data-queue length associated with the LyDROO method exhibits an almost linear increase over time. This finding suggests that the LyDROO method struggles in stabilizing the data-queues when the data-arrival rate exceeds the computation capacity. These results shed light on the limitations of the LyDROO method in effectively managing data-queues. It becomes evident that as the data arrival rate surpasses the system's computation capacity, the LyDROO method fails to maintain stability in the data-queues. On the other hand, the proposed MFDROO algorithm provides significant advantages when it comes to stabilizing data-queues at various data rates. This flexibility is particularly beneficial in scenarios where data rates fluctuate, such as in wireless communication networks or in applications that require real-time processing of data. Additionally, the MFDROO algorithm offers scalability and ensures low latency making it an effective and reliable solution for a wide range of applications. According to our extensive testing, Fig. 4(b) shows that MFDROO has demonstrated its superiority over LyDROO in terms of computation-rate and data-queue lengths. This comparative analysis highlights MFDROO's ability to deliver significantly higher computation-rates while maintaining shorter data-queue lengths across different time-frames. This combination makes MFDROO a superior choice for data processing and management tasks and Fig. 4(c) shows that MFDROO

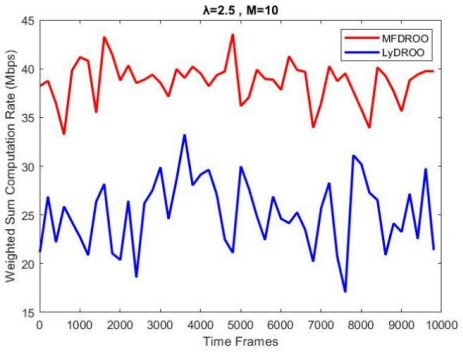


(i) $\lambda_i = 2.5$ Mbps

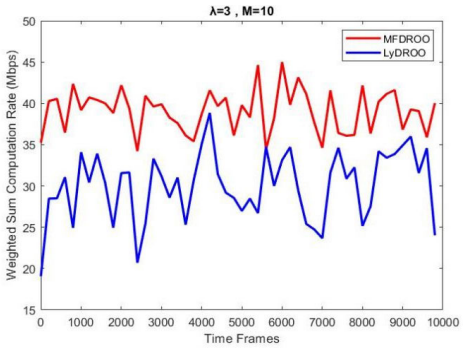


(ii) $\lambda_i = 3$Mbps

(a) Average data-queue length

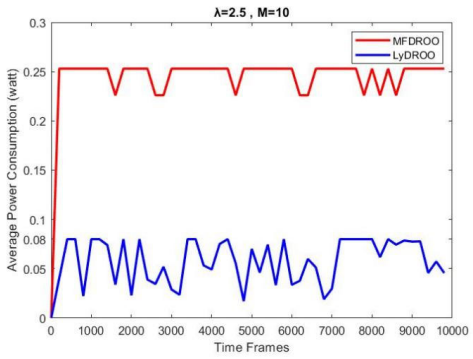


(i) $\lambda_i = 2.5$ Mbps

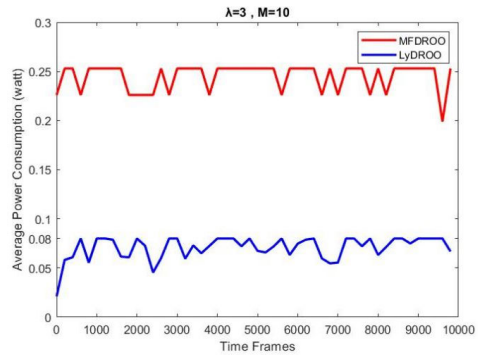


(ii) $\lambda_i = 3$ Mbps

(b) Weighted-Sum Computation-Rate



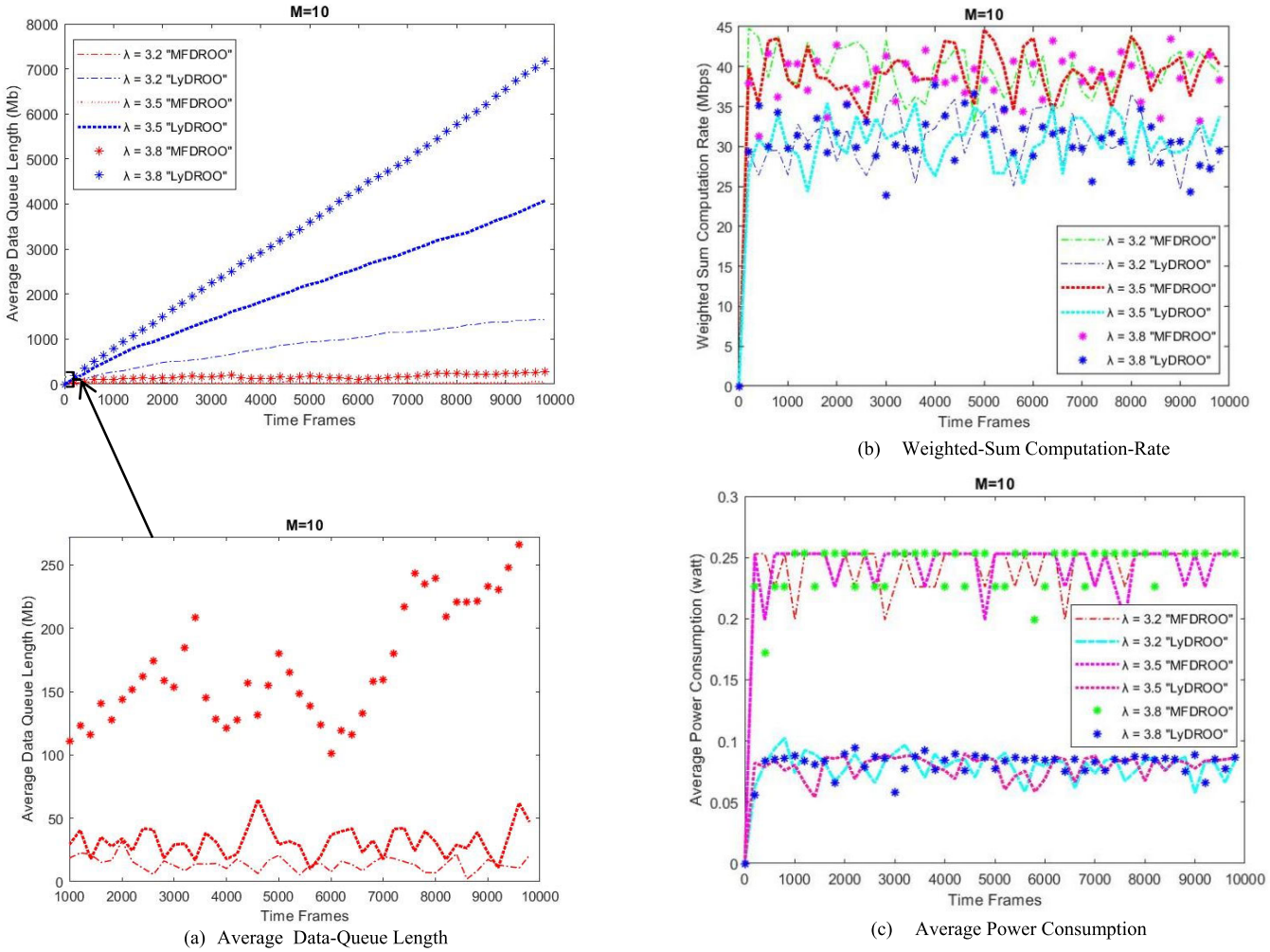
(i) $\lambda_i = 2.5$ Mbps



(ii) $\lambda_i = 3$ Mbps

(c) Average Power Consumption

FIGURE 3. Different schemes' convergence performance under $\lambda_i = 2.5$ and 3Mbps. (a) Average data-queue length (b) Weighted-sum computation-rate (c) Average power consumption. Each of (a),(b) and (c) is plotted at (i) $\lambda_i = 2.5$Mbps and (ii) $\lambda_i = 3$Mbps.



(a) Average Data-Queue Length

(b) Weighted-Sum Computation-Rate

(c) Average Power Consumption

FIGURE 4. Performance comparisons under different data-arrival rates λ_i (a) Average data-queue length (b) Weighted-sum computation-rate (c) Average power consumption.

algorithm may consume more power on average but it's obvious that power consumption is within the normal range compared to other related papers work and it is still operating efficiently and effectively.

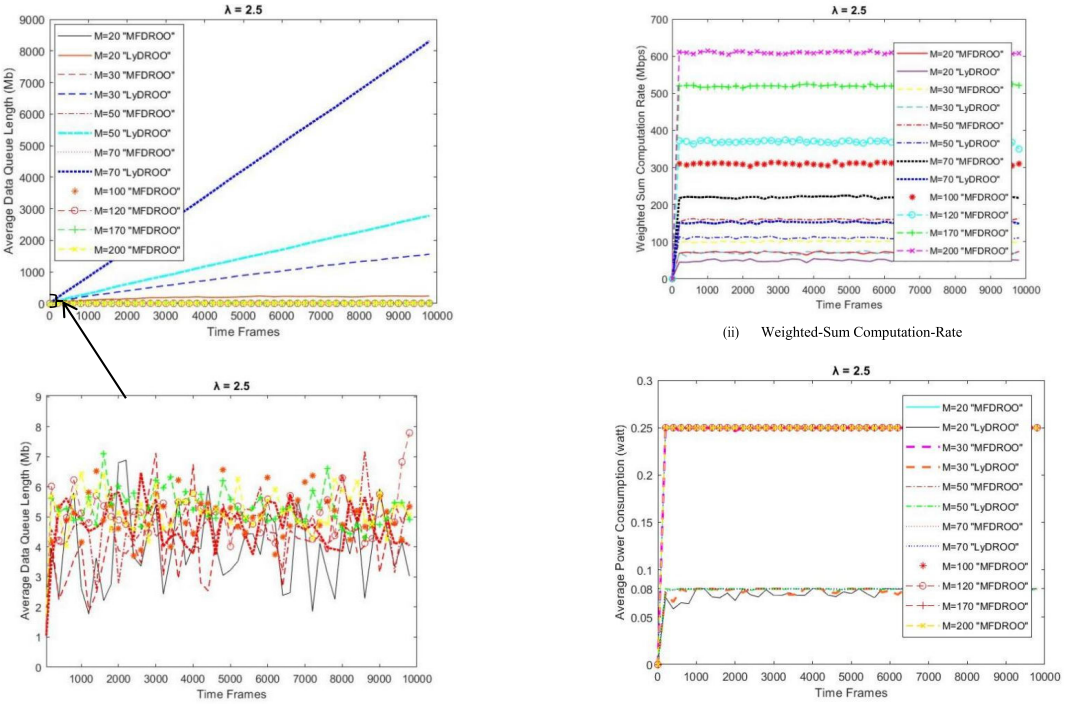
B. IMPACT OF CHANGING USERS NUMBER

This section examines the outperforming of the proposed MFDROO algorithm under the impact of varying the number of users M, ranging from 20 to 200 users at different data-arrival rates.

Fig.5 illustrates the impact of varying the number of users M, ranging from 20 to 200 users. Throughout this analysis, we maintain a fixed data rate ($\lambda_i$at 2.5, 3 and 3.2 Mbps). In Fig.5(a), it is remarkable that the proposed algorithm successfully maintains data-queue stability even when the number of users reaches up to 200. In contrast, when we examine the benchmark method, LyDROO, it is observed that it exhibits unstable behavior in terms of the data-queue. Particularly, the data-queue becomes unstable when the number of users is only 20.

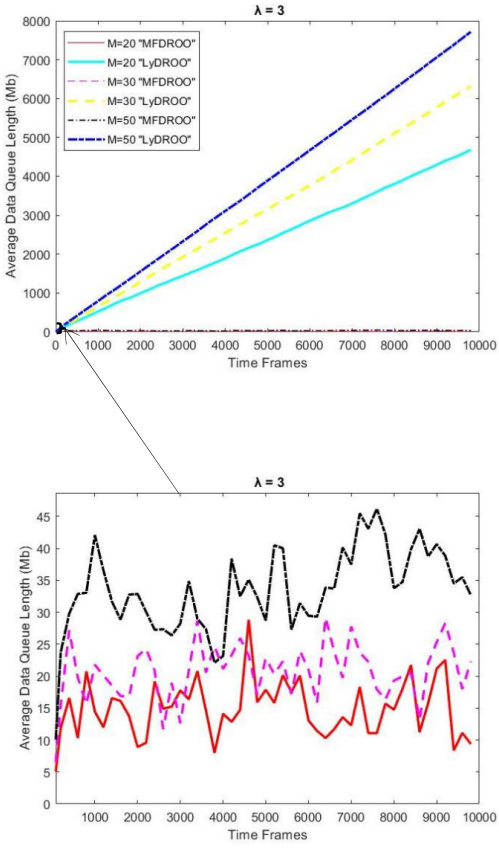
These findings highlight the effectiveness and superiority of our algorithm, MFDROO, in ensuring data-queue stability within communication networks. It demonstrates its capability to handle larger numbers of users compared to the benchmark method, LyDROO, which struggles to maintain data-queue stability even with a relatively small number of users. Additionally, MFDROO exhibits a shorter queue length, indicating efficient task management and reduced waiting times for users. In Fig. 5(b), the proposed algorithm exhibits its capability to handle up to 50 users at a higher data rate of $\lambda_i = 3$Mbps achieving a high computation-rate with lower queue length, while the benchmark LyDROO exhibits an almost linear increase over time demonstrates struggles to stabilize the data-queues with lower computation-rate in comparison with MFDROO. The results depicted in Fig. 5(c) illustrate the performance of MFDROO at a data rate of $\lambda_i = 3.2$ Mbps. The proposed algorithm maintains queue stability even with up to 30 users, resulting in a shorter queue length.

In contrast, the benchmark LyDROO fails to maintain queue stability when the number of users exceeds 20 at



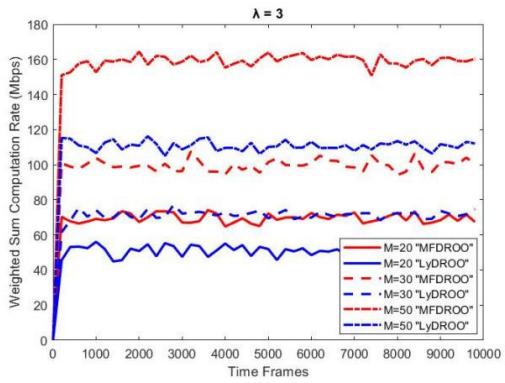
(i) Average Data-Queue Length (ii) Weighted-Sum Computation-Rate (iii) Average Power Consumption

(a) Data-Arrival Rate $\lambda_i = 2.5$ Mbps

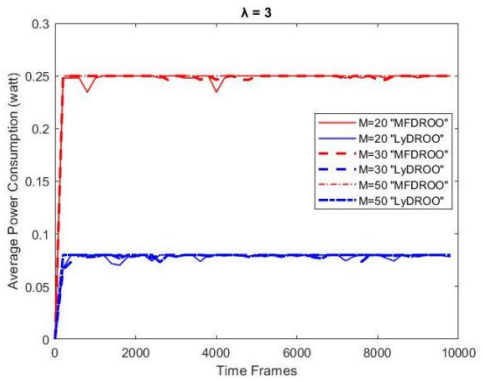


(i) Average Data-Queue Length

FIGURE 5. Performance comparisons under variation in number of users M for different data-arrival rates λ_i: (a) $\lambda_i = 2.5$Mbps, (b) $\lambda_i = 3$Mbps and (c) $\lambda_i = 3.2$Mbps. Each of (a), (b) and (c) is plotted as function of (i) Average data-queue length (ii) Weighted-sum computation-rate (iii) Average power consumption.

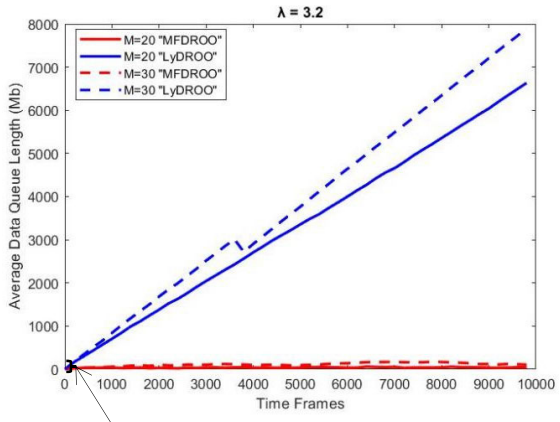


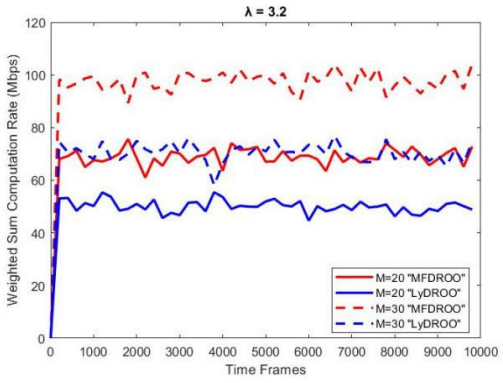
(ii) Weighted-Sum Computation-Rate



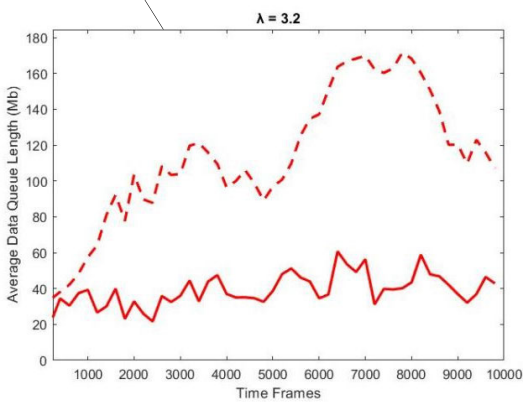
(iii) Average Power Consumption

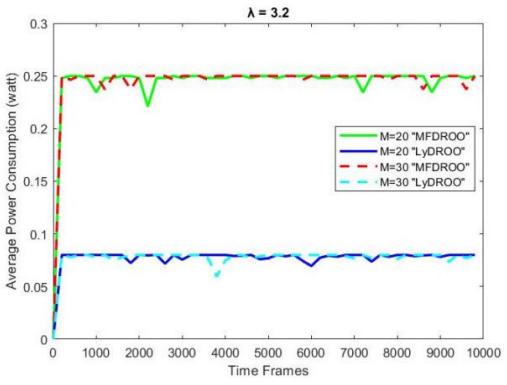
(b) Data-Arrival Rate $\lambda_i = 3$ Mbps





(ii) Weighted-Sum Computation-Rate





(i) Average Data-Queue Length

(iii) Average Power Consumption

(c) Data-Arrival Rate $\lambda_i = 3.2$ Mbps

FIGURE 5. ***(Continued.)*** **Performance comparisons under variation in number of users M for different data-arrival rates λ_i: (a) $\lambda_i = 2.5$Mbps, (b) $\lambda_i = 3$Mbps and (c) $\lambda_i = 3.2$Mbps. Each of (a), (b) and (c) is plotted as function of (i) Average data-queue length (ii) Weighted-sum computation-rate (iii) Average power consumption.**

a data rate of $\lambda_i \geq 2.5$Mbps. This highlights the limitations of LyDROO in handling larger user populations and higher data rates. This means that the proposed algorithm provides a robust solution that ensures queue stability even under demanding conditions. These findings demonstrate the effectiveness of the proposed algorithm in managing network resources and ensuring a smooth user experience.

Based on the obtained results, it is evident that the proposed algorithm exhibits a higher average power consumption compared to the benchmark method mentioned. This is due to the

fact of using ONMF in the proposed MFDROO algorithm as the ONMF involves solving optimization problems required for the decomposition process which may require significant computational power. However, when TDMA is employed, the power consumption of MFDROO compared to other related papers work on this topic as referred to in [13], [43], [44], [45], [46], and [47] is within an acceptable range. Moreover, it outperforms the aforementioned references for scenarios involving 10 up to 200 users and arrival data rates. This suggests that while the proposed algorithm may consume more power on average, it is still operating efficiently and effectively.

The computational overhead of the proposed MFDROO algorithm is examined by studying the time computational complexity big O notation which is found to be $O(\mathrm{M}\log_2\left(\frac{\Delta}{\sigma_o}\right)+\mathrm{M}^3\overline{Len})\mathrm{N_t}$, where the first term describes the bi-section search in the adopted resource allocation algorithm [6] and $\overline{Len}$ is the input length in a binary form used in the second term that corresponds to solving the linear programming problem in the resource allocation algorithm and this is exactly the same as that of the benchmark LyDROO [6].

However, as the DNN offloading action generation is the most consuming time process. Therefore, the proposed MFDROO algorithm CPU computation time is computed and presented in Table 3 under different number of M users compared to the benchmark LyDROO. This is done by running both algorithms on a Visual Studio Code V8 platform with an Intel Core i7-7500U CPU @ 2.70 GHz and of 8GB memory.

TABLE 3. Computation rate, queue stability and CPU computation time under variation in number of users M.

M	Computation rate (Mbps) / Queue Stability		CPU computation time (second)	
	MFDROO	LyDROO	MFDROO	LyDROO
10	39.08 / stable	20.02 / stable	0.10	0.093
20	68.15 / stable	49.74 / unstable	0.11	0.1025
30	97.71 / stable	70.07 / unstable	0.117	0.1186

As shown from the above table, the proposed MFDROO algorithm exhibits higher CPU computation time than the benchmark LyDROO and that translates in the computational complexity overhead of the proposed algorithm. So, it is worth noting that due to this time computational overhead, there is a consequent increase in the energy consumption of the algorithm, as more CPU cycles incorporated a more energy is consumed.

Lastly, the paper would like to highlight that the strategic use of the Edge Computing network and Task offloading approach helps mitigate the power consumption issue from the mobile device to the edge server, allowing the mobile devices M to remain light and resource efficient.

C. HYPER PARAMETERS EFFECT

This section examines the performing of the proposed MFDROO algorithm under the effect of different hyper parameters.

Fig.6 illustrates the performance of MFDROO under different hyper parameters.

In Fig.6(a), we observe the convergence effect of MFDROO while varying the batch size. It is evident that when the batch size is set to 128, MFDROO exhibits the most favorable convergence behavior. Moving on to Fig.6(b), we focus on the impact of different learning rates on the algorithm's convergence. The abscissa in this figure represents the range of learning rates, spanning from too-high to too-low values. It is evident that the best convergence effect is observed when the learning rate is set to 0.01. This finding highlights the importance of selecting an appropriate learning rate and batch size to ensure optimal algorithm performance.

We present a comparison between MFDROO and LyDROO in terms of computation-rate performance and queue stability.

The comparison is conducted under different numbers of wireless devices (WDs) denoted by M and varying data arrival rates λ_i. The results are summarized in Table 4 for users up to 100.

From the analysis, it is observed that MFDROO achieves a higher computation-rate compared to LyDROO across the range of WDs (M=10 up to 200) while maintaining queue stability. This indicates that MFDROO is more efficient in terms of computational processing, allowing for increased data throughput.

Furthermore, the queue stability of MFDROO remains unaffected by the variation in the number of users. This implies that even with an increase in the number of WDs, MFDROO can effectively manage the incoming data and maintain a stable queue. This is crucial for ensuring smooth and uninterrupted data transmission in wireless networks. By optimizing the utilization of available resources, MFDROO maximizes the computational capacity of the system, resulting in improved processing speed.

D. ADDITIONAL PERFORMANCE EVALUATION

In the following sub-section, to further evaluate and shed light on the novelty of the proposed MFDROO algorithm, we present an additional benchmark algorithm (QDRL) [48]. We compare the proposed MFDROO algorithm with the QDRL benchmark algorithm concerning three commonly used metrics: the normalized real computation-rate (RCR), the RCR, and the average task queue length. The RCR is commonly used to measure the real-time performance and efficiency of algorithm whereas a larger RCR indicates how great will be the system benefits and performance. we also consider the average task queue length as a measure of the system stability where the average task queue length represents the average number of tasks waiting to be executed during a given period, so, a small average task queue length

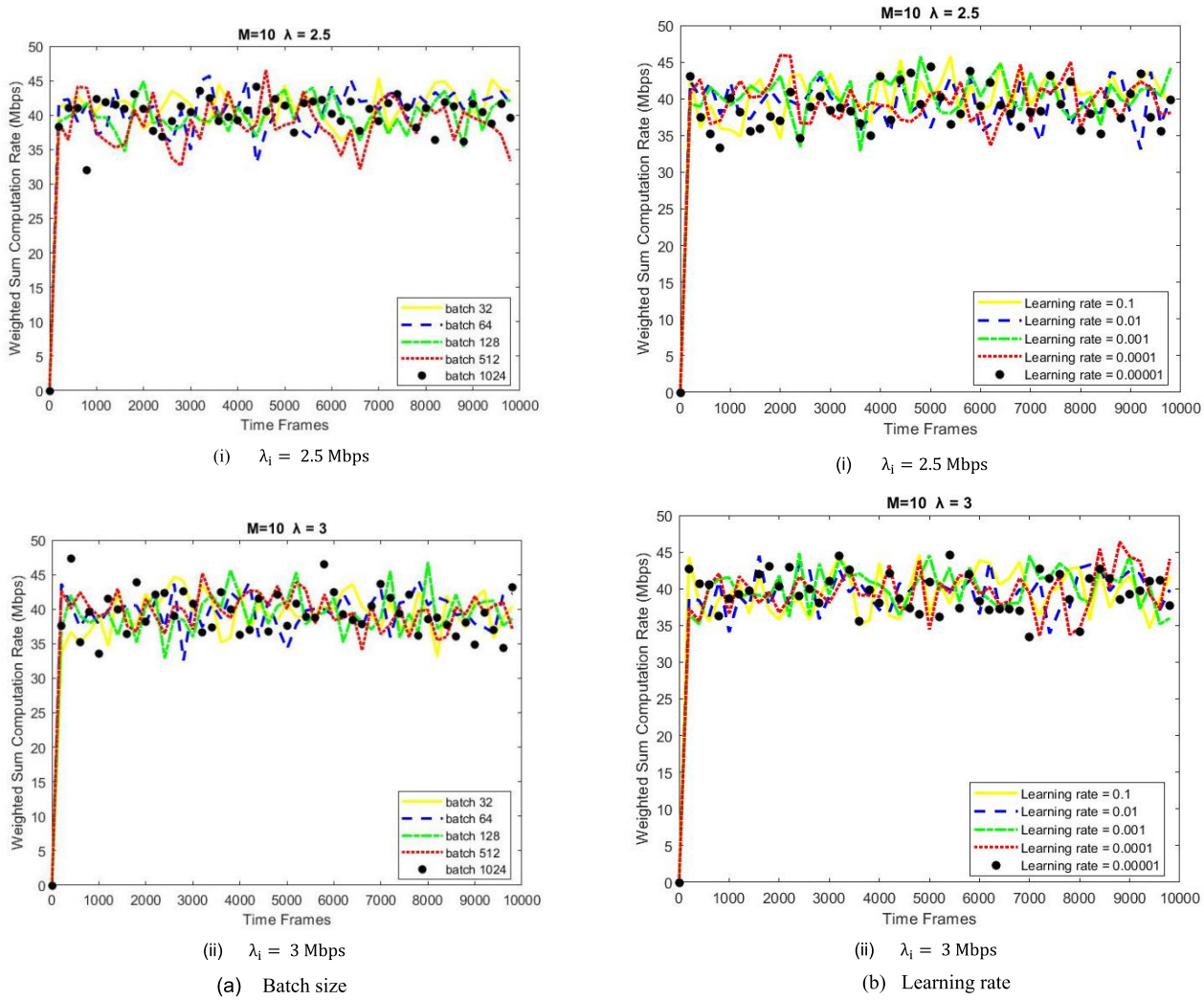


FIGURE 6. Performance of MFDROO under different hyper-parameters.(a) Batch size, (b) Learning rate. Each of (a) and (b) is plotted at (i) $\lambda_i = 2.5$ Mbps, (ii) $\lambda_i = 3$ Mbps.

means the system is more stable. The normalized RCR should be close to 1 to agree for better performance. As we adopt in the above subsections of the simulation results section, LyDROO as the benchmark method, it will be used as a normalized reference to define the normalized RCR. Then we define the normalized RCR $= \mathrm{RCR}_{\mathrm{algorithm}}/\mathrm{RCR}_{\mathrm{LyDROO}}$.

To capture the behavior of the proposed MFDROO algorithm, we conduct the comparison under the realization of random events in the context of 1000 time-frames as used in [48]. Fig.7 illustrates the results of RCR under data rate $\lambda_i = 3$Mbps and fixed number of users (M=10). The figure shows that MFDROO outperforms QDRL in terms of real computation-rate performance and that's a pivotal effect on the system's efficiency and performance.

Fig.8 shows the performance comparison under the normalized long-term CR constraint. The figure shows that both the proposed MFDROO algorithm and the benchmark QDRL algorithm even exceed the near-optimal solution (i.e., LyDROO)

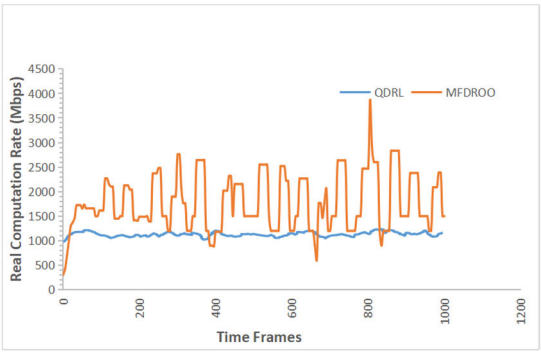


FIGURE 7. Performance of MFDROO under fixed parameters ($\lambda_i = 3$ Mbps & M=10).

Fig.9 illustrates the average task-queue length of both the proposed MFDROO algorithm and the benchmark QDRL.

It evident that MFDROO is slightly higher than QDRL with a very small value as the average value of the task-queue length over all the time-frames with respect to the QDRL

TABLE 4. Computation-rate evaluation and queue stability check at different number of users M.

M	Computation-Rate (Mbps) / Queue Stability			
	$\lambda = 2.5$ Mbps		$\lambda = 3$ Mbps	
	MFDROO	LyDROO	MFDROO	LyDROO
10	38.73 / Stable	25.17 / Stable	39.08 / Stable	20.02 / Stable
20	69.38 / Stable	48.84 / Un-Stable	68.15 / Stable	49.74 / Un-Stable
30	98.77 / Stable	69.13 / Un-Stable	97.71 / Stable	70.07 / Un-Stable
50	157.4 / Stable	108.7 / Un-Stable	156 / Stable	108.7 / Un-Stable
70	215.8 / Stable	148.2 / Un-Stable	208.5 / Stable	148.2 / Un-Stable
100	303.8 / Stable	205.7 / Un-Stable	305.1 / Stable	208 / Un-Stable

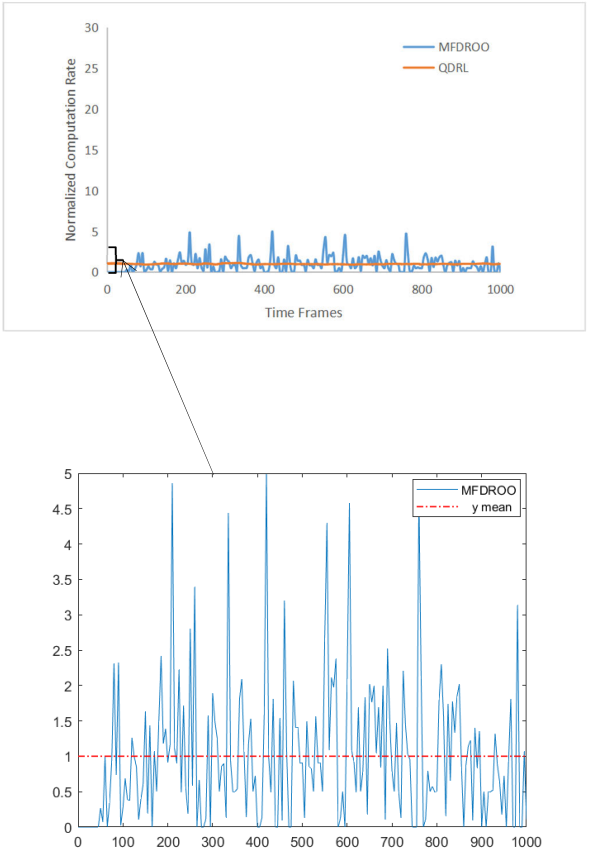


FIGURE 8. Performance of MFDROO under normalized long-term constraint CR.

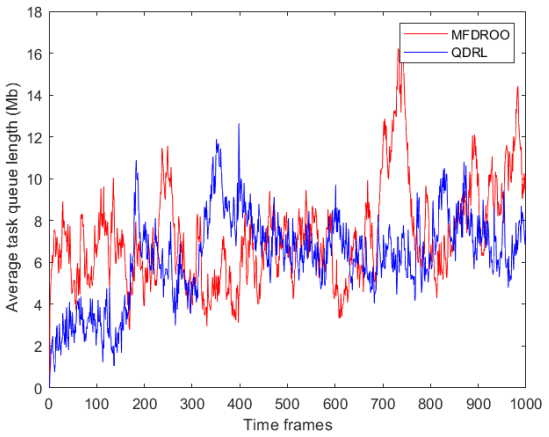


FIGURE 9. Average task-queue length plotted at ($\lambda_i = 3$ Mbps & M=10).

benchmark is 6.25 Mb and for the proposed MFDROO algorithm is 7.2 Mb and that difference sounds nothing considering the higher real computation-rate that MFDROO achieved it.

Fig.10 study the evaluation of the proposed MFDROO algorithm under the variation of arrival data rates. The figure shows that from $\lambda_i = 2.5$ up to 2.7 Mbps, MFDROO is lower than the benchmark QDRL means that the system remains more stable in that cases and from $\lambda_i = 2.8$ up to 3 Mbps, the proposed MFDROO is slightly higher than QDRL but still guaranteeing the system stability.

TABLE 5. Comparison with various approaches.

References	Year	Offloading Scheme Type	No. of users	λ(Mbps)	Computation Rate (Mbps)	Queue length (Mb)	Queue Stability	Power Consumption (watt)
[14]	2022	Binary Offloading	10	2.5	N/A	10	Stable	N/A
				3	3.2	20	Stable	
				3.2	N/A	150	Stable	
				3.5		N/A		
				3.8				
			20	2.5				
				3	4.4			
				3.2	N/A			
			30	2.5				
				3	6			
				3.2	N/A			
			70	2.5				
				3				
			100	2.5				
			170	2.5				
			200	2.5				
[13]	2022	Partial Offloading	10	2.5	0.8 at M=9 0.99 at M=12	N/A		2.3
				3				
				3.2				
				3.5				
				3.8				
			20	2.5	N/A			N/A
				3				
				3.2				
			30	2.5				
				3				
				3.2				
			70	2.5				
				3				
			100	2.5				
			150	2.5				
			200	2.5				
[10]	2023	Binary Offloading	10	2.5	3.14	N/A		
				3				
				3.2				
				3.5				
				3.8				
			20	2.5	4.29			
				3				
				3.2				
			30	2.5	5.73			
				3				
				3.2				
			70	2.5	N/A			
				3				
			100	2.5				
			170	2.5				
			200	2.5				
[15]	2023	Binary Offloading	10	2.5	0.85 at M=9 1.29 at M=12	N/A		
				3				
				3.2				
				3.5				
				3.8				
			20	2.5	N/A			
				3				
				3.2				
			30	2.5				
				3				
				3.2				
			70	2.5				
				3				
			100	2.5				
			170	2.5				
			200	2.5				
Proposed Algorithm	2023	Binary Offloading	10	2.5	38.73	3.025	Stable Queue	0.244
				3	39.08	8.71		0.244
				3.2	39.17	13,13		0.2382
				3.5	39.17	31.74		0.2404
				3.8	38.37	162.9		0.2344
			20	2.5	69.38	3.866		0.25
				3	68.15	14.75		0.25
				3.2	68.55	38.28		0.25
			30	2.5	98.77	4.18		0.2449
				3	97.71	20.51		0.25
				3.2	98.01	111.7		0.25
			70	2.5	215.8	4.661		0.245
				3	208.5	46.6		0.26
			100	2.5	303.8	4.951		0.245
			170	2.5	517.4	5.214		0.245
			200	2.5	607.1	5.27		0.245

However, Fig.11 shows that MFDROO achieves higher computation rate under the different arrival data rates in comparison with QDRL achieving both stable system and maximizing the users satisfaction leads to efficient and reliable system performance.

The comparisons with both benchmarks LyDROO and QDRL proved the novelty of the proposed MFDROO algorithm and how it can maximize the computation-rates guaranteeing system stability and ensuring its ability to outperform in high-scale networks under variation of

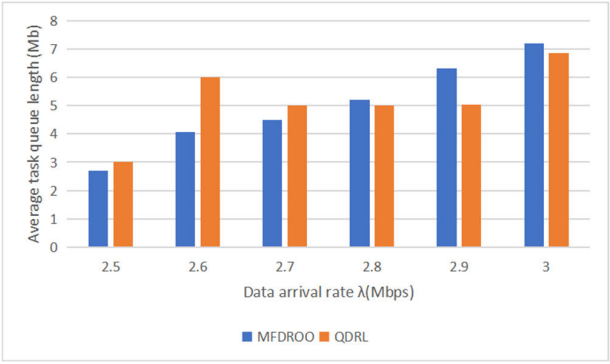


FIGURE 10. Average task-queue length plotted under variation of data arrival rates (λ_i = 2.5 up to 3 Mbps) at M=10.

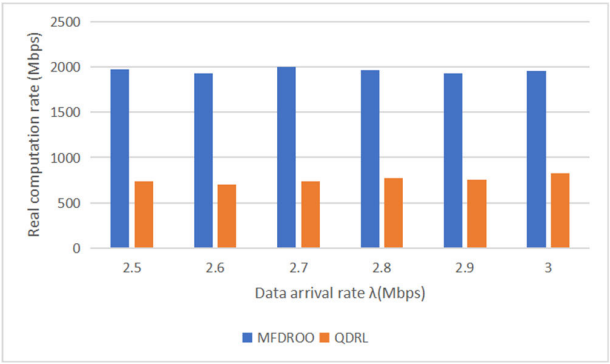


FIGURE 11. Real computation rate plotted under variation of data arrival rates (λ_i = 2.5 up to 3 Mbps) at M=10.

arrival data rates, achieving better system performance and efficiency.

In addition, Table 5 shows a comparison of the proposed algorithm with other existing approaches available in the literature. It provides an overview of the key features and performance metrics considered for the comparison.

Also, it's worth noting that the authors in [45] demonstrated the results of their existing method that tackled the same problem. However, it is not included in the comparison since the data arrival rate parameter used in this work is in the range of Kbps, this is because the objective of the study as stated by the authors in this article was increasing the battery lifetime and enhancing the capability of IIoT networks to achieve higher business requirements by achieving low energy consumption systems. The authors did that by offloading all Delay-sensitive and compute-intensive (DSCI) tasks type workloads to mobile edge computing (MEC) servers for processing because of the limited battery capacity of the devices and the authors discarding offloading massive amounts of tasks as it will lead to higher energy consumption in the system. Therefore, the used data rate in the mentioned article is much less than the proposed MFDROO algorithm, as the used data arrival rate in MFDROO is in the range of Mbps.

VI. CONCLUSION

This paper focuses on the problem of stable offloading computations in a multi-user MEC network, considering the uncertainties introduced by the stochastic nature of wireless channel and arrivals of data-tasks. Our objective is to formulate a multi-stage stochastic MINLP problem that assures long-term data-queue stability while optimizing the average weighted sum computation-rate of all WDs.

To address this problem, we proposed a novel algorithm based on utilizing ONMF in combination with the DRL approach, called (MFDROO). The MFDROO algorithm combined the advantages of three key techniques: Lyapunov optimization, non-negative matrix factorization (NMF), and deep reinforcement learning (DRL).

The simulation results demonstrated the ability of MFDROO to achieve optimal computation-rate and simultaneously satisfying the long-term constraint.

The proposed MFDROO was a promising algorithm that offers a significant improvement in both the efficiency and robustness of computation and represents a powerful tool for optimizing computation-rate performance while maintaining a high level of reliability and stability. The results demonstrate the effectiveness of MFDROO in achieving better overall system performance, which helped in guaranteeing data-queue stability, even in scenarios with a large number of users.

The proposed algorithm in this paper considers the absence of knowledge of the future dynamics realizations of random channel conditions neglecting the user's dynamic behavior during computing offloading and resource allocation. This oversight affects the performance of the solution.

In future research, we will investigate distributed machine learning techniques in MEC to study end-user mobility during task processing, enhance user's quality of service (QOS) and also extend the algorithm to incorporate other long-term performance relevant optimization factors such as energy efficiency to include the IoT devices by overcoming the limitations observed in the MFDROO algorithm.

REFERENCES

[1] Y. Wu, "Cloud-edge orchestration for the Internet of Things: Architecture and AI-powered data processing," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12792–12805, Aug. 2021, doi: 10.1109/JIOT.2020.3014845.

[2] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, "A survey on computation offloading modeling for edge computing," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102781, doi: 10.1016/j.jnca.2020.102781.

[3] A. H. A. Salam, H. M. Elattar, and M. A. Aboul-Dahab, "Smart technique for cache-assisted device to device communications," *IEEE Access*, vol. 8, pp. 181485–181499, 2020, doi: 10.1109/ACCESS.2020.3028565.

[4] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A truthful online mechanism for collaborative computation offloading in mobile edge computing," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4832–4841, Jul. 2020, doi: 10.1109/TII.2019.2960127.

[5] S. Li, X. Hu, and Y. Du, "Deep reinforcement learning and game theory for computation offloading in dynamic edge computing markets," *IEEE Access*, vol. 9, pp. 121456–121466, 2021, doi: 10.1109/ACCESS.2021.3109132.

[6] S. Bi, L. Huang, H. Wang, and Y. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7519–7537, Nov. 2021, doi: 10.1109/TWC.2021.3085319.

[7] S. Chen, S. Sun, H. Chen, J. Ruan, and Z. Wang, "A game theoretic approach to task offloading for multi-data-source tasks in mobile edge computing," in *Proc. IEEE Int. Conf Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Sep. 2021, pp. 776–784, doi: 10.1109/ispa-bdcloud-socialcom-sustaincom52081.2021.00111.

[8] S. Wang, Z. Hu, Y. Deng, and L. Hu, "Game-theory-based task offloading and resource scheduling in cloud-edge collaborative systems," *Appl. Sci.*, vol. 12, no. 12, p. 6154, Jun. 2022, doi: 10.3390/app12126154.

[9] F. Zhang, J. Ge, C. Wong, C. Li, X. Chen, S. Zhang, B. Luo, H. Zhang, and V. Chang, "Online learning offloading framework for heterogeneous mobile edge computing system," *J. Parallel Distrib. Comput.*, vol. 128, pp. 167–183, Jun. 2019, doi: 10.1016/j.jpdc.2019.02.003.

[10] A. Acheampong, Y. Zhang, and X. Xu, "A parallel computing based model for online binary computation offloading in mobile edge computing," *Comput. Commun.*, vol. 203, pp. 248–261, Apr. 2023, doi: 10.1016/j.comcom.2023.03.004.

[11] M. Chen, T. Wang, S. Zhang, and A. Liu, "Deep reinforcement learning for computation offloading in mobile edge computing environment," *Comput. Commun.*, vol. 175, pp. 1–12, Jul. 2021, doi: 10.1016/j.comcom.2021.04.028.

[12] A. P. Program and S. Barbara, "The problem of stability," *Queueing Syst.*, vol. 4, pp. 287–317, Dec. 1989.

[13] W. Chen, G. Shen, K. Chi, S. Zhang, and X. Chen, "DRL based partial offloading for maximizing sum computation rate of FDMA-based wireless powered mobile edge computing," *Comput. Netw.*, vol. 214, Sep. 2022, Art. no. 109158, doi: 10.1016/j.comnet.2022.109158.

[14] X. Li, L. Huang, H. Wang, S. Bi, and Y. A. Zhang, "An integrated optimization-learning framework for online combinatorial computation offloading in MEC networks," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 170–177, Feb. 2022, doi: 10.1109/MWC.201.2100155.

[15] G. Shen, W. Chen, B. Zhu, K. Chi, and X. Chen, "DRL based binary computation offloading in wireless powered mobile edge computing," *IET Commun.*, vol. 17, no. 15, pp. 1837–1849, Sep. 2023, doi: 10.1049/cmu2.12658.

[16] Z. Akhavan, M. Esmaeili, B. Badnava, M. Yousefi, X. Sun, M. Devetsikiotis, and P. Zarkesh-Ha, "Deep reinforcement learning for online latency aware workload offloading in mobile edge computing," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2022, pp. 2218–2223, doi: 10.1109/GLOBECOM48099.2022.10001678.

[17] L. Huang, S. Bi, and Y. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020, doi: 10.1109/TMC.2019.2928811.

[18] Y. Chen, W. Gu, and K. Li, "Dynamic task offloading for Internet of Things in mobile edge computing via deep reinforcement learning," *Int. J. Commun. Syst.*, pp. 1–16, Mar. 2022, doi: 10.1002/dac.5154.

[19] S. Peng, W. Ser, B. Chen, and Z. Lin, "Robust orthogonal nonnegative matrix tri-factorization for data representation," *Knowl.-Based Syst.*, vols. 201–202, Aug. 2020, Art. no. 106054, doi: 10.1016/j.knosys.2020.106054.

[20] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1336–1353, Jun. 2013, doi: 10.1109/TKDE.2012.51.

[21] M. Charikar and L. Hu, "Approximation algorithms for orthogonal non-negative matrix factorization," *Proc. Mach. Learn. Res.*, vol. 130, pp. 2728–2736, 2021.

[22] N. Liang, Z. Yang, Z. Li, W. Sun, and S. Xie, "Multi-view clustering by non-negative matrix factorization with co-orthogonal constraints," *Knowl.-Based Syst.*, vol. 194, Apr. 2020, Art. no. 105582, doi: 10.1016/j.knosys.2020.105582.

[23] X. Lin and P. C. Boutros, "Optimization and expansion of non-negative matrix factorization," *BMC Bioinf.*, vol. 21, no. 1, pp. 1–10, Dec. 2020, doi: 10.1186/s12859-019-3312-5.

[24] J. Pan and N. Gillis, "Generalized separable nonnegative matrix factorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 5, pp. 1546–1561, May 2021, doi: 10.1109/TPAMI.2019.2956046.

[25] J. Dehghanpour and N. Mahdavi-Amiri, "Orthogonal nonnegative matrix factorization problems for clustering: A new formulation and a competitive algorithm," *Ann. Oper. Res.*, pp. 1–17, Mar. 2022, doi: 10.1007/s10479-022-04642-2.

[26] Q. Wang, X. He, X. Jiang, and X. Li, "Robust bi-stochastic graph regularized matrix factorization for data clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 390–403, Jan. 2022, doi: 10.1109/TPAMI.2020.3007673.

[27] S. Basiri and S. Salapaka, "A novel maximum-entropy-driven technique for low-rank orthogonal nonnegative matrix factorization with ℓ_0-norm sparsity constraint," 2022, *arXiv:2210.02672*.

[28] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019, doi: 10.1109/TCOMM.2019.2898573.

[29] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[30] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017, doi: 10.1109/JSAC.2017.2760160.

[31] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1079–1092, Feb. 2019, doi: 10.1109/TVT.2018.2883156.

[32] M. E. Bruni, "Solving nonlinear mixed integer stochastic problems: A global perspective," in *Global Optimization: From Theory to Implementation*. Boston, MA, USA: Springer, 2006, pp. 75–109, doi: 10.1007/0-387-30528-9_4.

[33] S. Zhang and X. A. Sun, "Stochastic dual dynamic programming for multistage stochastic mixed-integer nonlinear optimization," *Math. Program.*, vol. 196, nos. 1–2, pp. 935–985, Nov. 2022, doi: 10.1007/s10107-022-01875-8.

[34] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, Jan. 2010, doi: 10.2200/s00271ed1v01y201006cnt007.

[35] D. Tolić, N. Antulov-Fantulin, and I. Kopriva, "A nonlinear orthogonal non-negative matrix factorization approach to subspace clustering," *Pattern Recognit.*, vol. 82, pp. 40–55, Oct. 2018, doi: 10.1016/j.patcog.2018.04.029.

[36] J.-H. Yoo and S.-J. Choi, "Nonnegative matrix factorization with orthogonality constraints," *J. Comput. Sci. Eng.*, vol. 4, no. 2, pp. 97–109, Jun. 2010, doi: 10.5626/JCSE.2010.4.2.097.

[37] D. Kitamura and N. Ono, "Efficient initialization for nonnegative matrix factorization based on nonnegative independent component analysis," in *Proc. IEEE Int. Workshop Acoustic Signal Enhancement*. Tokyo, Japan: National Institute of Informatics, 2016, pp. 1–5.

[38] J. Xie, P. K. Douglas, Y. N. Wu, A. L. Brody, and A. E. Anderson, "Decoding the encoding of functional brain networks: An fMRI classification comparison of non-negative matrix factorization (NMF), independent component analysis (ICA), and sparse coding algorithms," *J. Neurosci. Methods*, vol. 282, pp. 81–94, Apr. 2017, doi: 10.1016/j.jneumeth.2017.03.008.

[39] Y. Qiu, G. Zhou, and K. Xie, "Deep approximately orthogonal nonnegative matrix factorization for clustering," 2017, *arXiv:1711.07437*.

[40] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proc. IEEE*, vol. 86, no. 11, pp. 2210–2239, 1998, doi: 10.1109/5.726788.

[41] Karmeshu, *Entropy Measures, Maximum Entropy Principle and Emerging Applications*, vol. 119. Berlin, Germany: Springer, 2003, doi: 10.1007/978-3-540-36212-8.

[42] M. Stephen, *Machine Learning an Algorithmic Perspective*, 2nd ed., Boca Raton, FL, USA: CRC Press, 2014. [Online]. Available: https://b-ok.cc/book/2543746/ef80cb

[43] Y. Chen, N. Zhang, Y. Wu, and S. Shen, "Dynamic computation offloading for energy efficiency in mobile edge computing," in *Energy Efficient Computation Offloading in Mobile Edge Computing*. Cham, Switzerland: Springer, 2022, doi: 10.1007/978-3-031-16822-2_2.

[44] K. Li, J. Zhao, J. Hu, and Y. Chen, "Dynamic energy efficient task offloading and resource allocation for NOMA-enabled IoT in smart buildings and environment," *Building Environ.*, vol. 226, Dec. 2022, Art. no. 109513, doi: 10.1016/j.buildenv.2022.109513.

[45] H. Wu, J. Chen, T. N. Nguyen, and H. Tang, "Lyapunov-guided delay-aware energy efficient offloading in IIoT-MEC systems," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 2117–2128, Feb. 2023, doi: 10.1109/TII.2022.3206787.
[46] B. Rong, *Energy Efficient Computation Offloading in Mobile Edge Computing*, vol. 30. New York, NY, USA: IEEE Wireless Communications, 2023, doi: 10.1109/mwc.2023.10105148.
[47] J. Liu, "Task offloading and resource allocation algorithm based on mobile edge computing in Internet of Things environment," *J. Eng.*, vol. 2021, no. 9, pp. 500–509, Sep. 2021, doi: 10.1049/tje2.12056.
[48] A. Xu, Z. Hu, X. Zhang, H. Xiao, H. Zheng, B. Chen, M. Zheng, P. Zhong, Y. Kang, and K. Li, "QDRL: Queue-aware online DRL for computation offloading in industrial Internet of Things," *IEEE Internet Things J.*, vol. 11, no. 5, pp. 7772–7786, Mar. 2024, doi: 10.1109/JIOT.2023.3316139.

ENGY A. ABDELAZIM (Graduate Student Member, IEEE) received the B.Sc. degree in electronics and communication engineering from Modern Academy for Engineering and Technology, Egypt, in 2011. She is currently pursuing the M.Sc. degree with the Department of Electronics and Communications Engineering, Arab Academy for Science, Technology and Maritime Transport (AASTMT), Cairo, Egypt. Her current research interests include mobile communication systems, data offloading, queueing and scheduling optimization, and learning in wireless communication systems and networks.

SHERIF K. ELDAYASTI (Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering (electronics and communications) from the Arab Academy for Science, Technology and Maritime Transport (AASTMT), Cairo, Egypt, in 1998 and 2006, respectively, and the Ph.D. degree in electrical engineering (electronics and communications) from Ain Shams University, Cairo, in 2018. He is currently an Assistant Professor with the Department of Electronics, AASTMT.

HUSSEIN M. ELATTAR (Member, IEEE) received the B.Sc. degree in electrical engineering (electronics and communications) from the Arab Academy for Science, Technology and Maritime Transport (AASTMT), Cairo, Egypt, and the M.Sc. and Ph.D. degrees in electrical engineering (electronics and communications) from Ain Shams University, Egypt. He is currently a Professor with the Department of Electronics and Communications Engineering, AASTMT. His current research interests include radio resource management, cognitive networks, optimization techniques, the Internet of Things (IoT), and 5G applications.

MOHAMED A. ABOUL-DAHAB (Life Senior Member, IEEE) was born in Zagazig, Al-Sharkia, Egypt, in October 1950. He received the B.Sc. degree in electrical engineering (electronics and communications) from the Faculty of Engineering, Cairo University, Egypt, in 1973, and the M.Sc. and Ph.D. degrees in electrical engineering (communications) from the Faculty of Engineering, Alexandria University, Egypt, in 1980 and1986, respectively. He joined as a Teaching Assistant with the Arab Academy for Science, Technology and Maritime Transport (AASTMT), in 1975, where he became a Professor of communications engineering, in 1999. He has been an Advisor for the AASTMT President for Scientific Affairs, since 2013. He has several publications in the area of education. His research interests include adaptive antenna arrays, microstrip antennas, signal processing techniques in wireless communication systems, channel modeling, and satellite communications. He was a member of the Advisory Board for Electrical Engineering, from 2015 to 2018, and the National Radio Science Committee, from 1995 to 2016. He is a Registered Consultant Engineer with Egyptian Engineering Syndicate. He was the Chairperson of the National Committee for Information and Communication Technology, from 2018 to 2022. He is currently one of its board members, both are under the umbrella of the Academy of Scientific Research and Technology, Egypt.