Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(12): 21267–21291.
DOI: 10.3934/mbe.2023941
Received: 20 October 2023
Revised: 06 November 2023
Accepted: 15 November 2023
Published: 30 November 2023

Research article

Analysis and prediction of UAV-assisted mobile edge computing systems

Xiong Wang[1], Zhijun Yang[1,2,3,*], Hongwei Ding[1] and Zheng Guan[1]

[1] School of Information Science and Engineering, Yunnan University, Kunming, China

[2] Educational Instruments and Facilities Service Center, Educational Department of Yunnan Province, Kunming, China

[3] Key Laboratory of Education Informatization for Nationalities of Ministry of Education, Yunnan Normal University, Kunming, China

* **Correspondence:** Email: yzj207@aliyun.com.

Abstract: As the demand for the internet of things (IoT) continues to grow, there is an increasing need for low-latency networks. Mobile edge computing (MEC) provides a solution to reduce latency by offloading computational tasks to edge servers. However, this study primarily focuses on the integration of back propagation (BP) neural networks into the realm of MEC, aiming to address intricate network challenges. Our innovation lies in the fusion of BP neural networks with MEC, particularly for optimizing task scheduling and processing. Firstly, we introduce a drone-assisted MEC model that categorizes computation offloading into synchronous and asynchronous modes based on task scheduling. Secondly, we employ Markov chains and probability-generation functions to accurately compute parameters such as average queue length, cycle time, throughput, and average delay in the synchronous mode. We also derive the first and second-order derivatives of the probability-generation function to support these computations. Finally, we establish a BP neural network to solve for the average queue length and latency in the asynchronous mode. Our results from the BP neural network closely align with the theoretical values obtained through the probability-generation function, demonstrating the effectiveness of our approach. Additionally, our proposed UAV-assisted MEC model outperforms the synchronous mode. Overall, our MEC scheduling approach significantly reduces latency, enhances speed, and improves throughput, with our model reducing latency by approximately 11.72% and queue length by around 9.45%.

Keywords: mobile edge computing; partial-differential equations; UAV; polling system; neural network; machine learning; applied numerical methods

1. Introduction

The advent of the fifth generation (5G) has brought forth a wave of novel network applications and services, signifying a transformative societal shift. These changes necessitate the delivery of higher-quality network services to cater to the demands of emerging technologies like virtual reality, mobile social media, the internet of things, the industrial internet, and the internet of vehicles. A majority of these applications impose stringent quality of service (QoS) requirements, particularly in light of the exponential surge in network traffic [1, 2]. This surge in data volumes places a considerable strain on mobile service providers. Without appropriate infrastructure to manage and process this increased workload, cellular networks would inevitably become more congested, resulting in diminished service quality and slower download speeds. Consequently, mobile devices are increasingly reliant on additional computing resources to meet these escalating demands.

Thankfully, certain scholars have introduced the concept of mobile edge computing (MEC) to enhance network service quality. This approach is designed to achieve faster response times and reduce device energy consumption compared to traditional methods. It accomplishes this by deploying computational access points (CAPs) as close as possible to the data source. Moreover, local devices can efficiently offload computing tasks onto these CAPs through judicious decision-making [3–6]. Numerous previous studies have made significant strides in this area.

Computation offloading, on the other hand, entails the transfer of local computational tasks to the cloud or edge server for processing. This approach effectively mitigates the resource limitations of mobile devices [7, 8]. In the realm of edge computing scenarios, particularly within campus networks, compute offloading assumes a significant role. Both students and faculty members frequently employ resource-limited mobile devices to access campus networks. By transferring computational tasks to on-campus edge servers or cloud-based servers, mobile devices can efficiently execute a wide array of tasks, including online learning, video conferencing, and campus navigation. This not only results in improved device performance but also yields reduced energy consumption, consequently extending the lifespan of the device's battery. In simple terms, while computation offloading can save time and resources for the mobile device, it may sometimes introduce latency due to the process of selecting the appropriate cloud server and transmitting workload data from the mobile device to the designated server.

In this context, our primary focus is on the integration of back propagation (BP) neural networks into the realm of mobile edge computing. Our goal is to address complex network issues and optimize the task scheduling and processing within MEC. This integration represents a novel approach to improving the performance and efficiency of MEC systems. Our study aims to achieve quantitative enhancement in comparison to existing state-of-the-art models.

To achieve this, we introduce a drone-assisted MEC model that categorizes computation offloading into synchronous and asynchronous modes based on task scheduling. We employ Markov chains and probability-production functions to accurately calculate critical performance metrics, including average queue length, cycle time, throughput, and average delay in the synchronous mode. Additionally, we derive the first and second-order derivatives of the probability-production function to support these calculations. In the asynchronous mode, we utilize a BP neural network to predict the average queue length and latency, with results that closely align with the theoretical values derived from the probability-production function.

Compared to references [9, 10], our work advances the current state of the art in the MEC field by integrating the strong performance of BP neural networks with the flexibility of drone-assisted computing. We place particular emphasis on the quantitative improvements our model provides and its potential to significantly enhance MEC system performance.

1.1. Our motivation

Accessing dependable computing services remains a significant challenge for user equipment (UE). On one hand, numerous UEs engage in computation-intensive applications in remote regions where communication infrastructure is less robust, and communication conditions are more favorable, rendering the placement of MEC servers more convenient [11, 12]. Conversely, the demand for computationally intensive services surges when the number of users is high, surpassing the capabilities of limited storage and computational resources. In such scenarios, MEC servers, especially in hotspots, are indispensable [13]. Fortunately, unmanned aerial vehicles (UAVs) have flexible deployment and extensive coverage, offering valuable support in executing computationally intensive tasks within MEC systems [14–16].

Nevertheless, UAV-assisted MEC services also grapple with several challenges. As aircraft, UAVs consume substantial energy to sustain their flight, which must also power the onboard communication and computing units to ensure reliable data transmission and processing services [17]. Additionally, the information transmission pathway between UAVs and ground users is exceptionally intricate [18]. Due to the wireless channel's inherent randomness, it is more variable and predictable than wired channels. Consequently, UAV-assisted edge computing becomes a pivotal yet intricate endeavor, demanding the development of suitable models and accurate parameter calculations.

1.2. Related works

A significant portion of academic research focusing on UAV-assisted edge computing relies heavily on deep learning methodologies. For example, in reference [19], an innovative discrete differential evolutionary algorithm, featuring novel mutation and crossover operators, is proposed to optimize the UAV's path over a cluster head. This advancement not only enhances MEC offloading efficiency but also effectively reduces UAV energy consumption. Meanwhile, reference [20] employs inverse induction to scrutinize a proposed game and introduces a dynamic gradient-based iterative search algorithm to boost UAV utilization. Additionally, reference [21] suggests a deep reinforcement learning-based multi-intelligent path planning scheme that prioritizes UE offloading and ensures UAV load balancing. Further contributing to the field, reference [22] offers an online algorithm rooted in perturbed Lyapunov optimization techniques. This approach optimizes UAV energy usage and mission processing rates while maintaining long-term data queue stability. Lastly, reference [23] proposes a Lyapunov-based dynamic resource allocation strategy for UAV-assisted mobile edge computing, effectively curbing energy consumption and computational latency in edge computing.

Simultaneously, some scholars delve into how drones can enhance task scheduling for edge computing to bolster efficiency and curtail energy consumption. Concentrating on scenarios where UAVs navigate intricate terrains replete with obstacles and interdependent tasks, reference [24] introduces a deep reinforcement learning algorithm designed to tackle non-deterministic polynomial (NP) puzzles and trim down MEC processing task latency. In a similar vein, reference [25] introduces the deep

MEC agent, a distributed algorithm rooted in deep reinforcement learning, meticulously designed to optimize computational offloading while minimizing latency. Additionally, reference [26] proposes an energy consumption fairness-aware computational offloading scheme based on genetic algorithms. This approach aims to reduce the disparities in offloading task energy consumption between UAVs and promote uniform energy distribution. Additionally, reference [27] introduces an asymmetric polling model that reduces latency in edge computing. Lastly, reference [28] advocates for a deep reinforcement learning approach that employs multi-intelligent proximal policy optimization, striving to define an optimal computational offloading policy.

Table 1 provides an overview of current research priorities and technology trends in the realm of UAV-assisted edge computing.

Table 1. The literature comparison.

Article	Year	Focused area	Accurate calculation	Deep learning
[19]	2022	Offloading strategy regarding task delays and UAV energy consumption.	No	**Yes**
[20]	2022	Improved MEC offloading efficiency and reduced UAV energy consumption.	No	**Yes**
[21]	2022	Prioritizing offload rates with fairness	No	**Yes**
[22]	2022	Optimizing the energy and mission processing rate of the UAV and meeting long-term data queue stability.	No	**Yes**
[23]	2022	Reducing energy consumption and computational latency in edge computing	No	**Yes**
[24]	2023	Success rate, number of tasks performed, and average task response latency	No	**Yes**
[25]	2023	Calculation latency	Yes	**Yes**
[26]	2023	Fairness of drone energy consumption, computational latency	No	**Yes**
[27]	2023	Minimizing user latency and system power consumption	Yes	**Yes**
[28]	2023	Reduce weighted energy consumption and computational latency	No	**Yes**
Our work	2023	Reduce average user latency, reduce average task queue length, and maximize MEC throughput	Yes	**Yes**

The previously conducted surveys, as mentioned above, shed light on various aspects of computation offloading techniques within edge computing environments. A notable limitation of traditional MEC servers lies in their fixed location, which renders them unable to adapt to the movements of mobile users. In this context, UAVs emerge as a promising and agile solution.

The existing literature has yet to provide a method that accurately encompasses all the essential parameters for UAV-assisted MEC. This underscores the absence of a suitable scheduling approach for this context. Polling, a well-established transmission technique utilized across various sectors

like computing, communications, and industrial control, offers insights into this challenge. Polling service systems can be classified into gated, exhaustive, and limited-k models based on their service policies [29]. In the exhaustive service policy, the server continues operating until the queue is entirely empty. Conversely, the gated service policy dictates that only those clients already in the queue at the start of the access period receive service. In the global gated service policy, only clients present when the server reaches a predefined "parent queue" during an epoch are served. Lastly, the limited-k service policy mandates that the server operates within a queue until a predetermined client is served or until the queue becomes empty [29, 30].

1.3. Our contribution

In this paper, we introduce novel applications of polling and scheduling within the realm of UAV-assisted edge computing. These innovations are as follows:

- We propose a UAV-assisted edge computing model designed to mitigate computational latency and enhance overall system throughput.
- Within the scope of UAV-assisted edge computing, we put forth two distinct scheduling methods: a synchronous-assisted scheduling approach and an asynchronous scheduling approach.
- To accurately assess key performance metrics such as the average queueing length, average period, throughput, and average delay within the synchronous threshold multi-server service (STMS), we leverage probability-generating functions and Markov models.
- Furthermore, we construct a precise BP neural network tailored to forecast the average queueing length and average delay associated with asynchronous threshold multi-server service (ATMS).

1.4. Structure of this paper

This paper follows a structured approach, beginning with an introduction that outlines the research motivation and identifies gaps in UAV-assisted edge computing. We then present the novel aspects of this work. Moving forward, we introduce the STMS model for UAV-assisted mobile edge computation and analyze it using probability distribution functions and Markov models. The subsequent section focuses on the precise computation of critical metrics, including average queue length, cycle time, throughput, and average delay within the STMS model. To validate our findings, we designed a Monte Carlo experiment with 100,000 repetitions, confirming the close alignment between simulated and theoretical values for ATMS. Following this, we propose the ATMS model and detail the construction of a BP neural network. In the penultimate section, we deploy BP neural networks to predict average queue length and latency in ATMS and provide a comparative analysis demonstrating ATMS's superior performance over STMS. Finally, we conclude our study, summarizing key findings, and outline future research priorities.

2. System model

2.1. Network architecture

The UAV-assisted edge computing model, illustrated in Figure 1, consists of multiple shaded regions, each representing an edge zone with a specific number of users denoted as N. In this configuration, S UAVs equipped with mobile edge computing (MEC) servers and communication relay

capabilities collaborate to collectively process user data. From the user's perspective, they interact with a virtual UAV for data exchange.

The data transfer delay is divided into three primary segments: the duration for data transfer from ground users to UAVs, the local computation time, and the delay involved in offloading data and computation among UAVs. It is important to note that UAVs are capable of performing data calculations and offloading simultaneously. In this paper, we focus on the delay associated with transmitting results back to users.

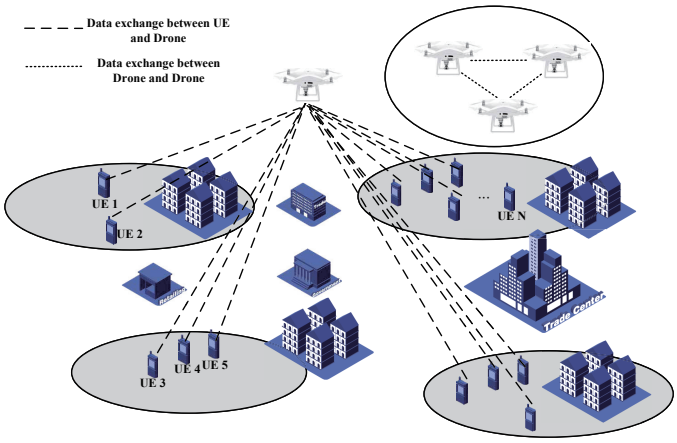

Figure 1. Scenarios of UAV-assisted edge computing.

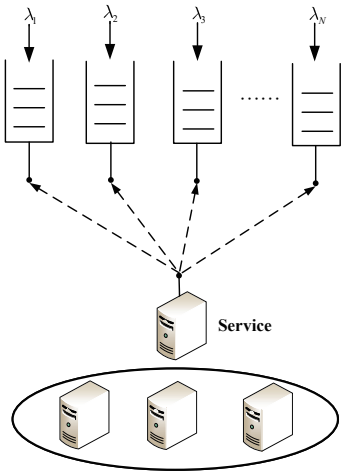

Figure 2. The STMS system model.

2.2. Complexity of the Model

The proposed service solution involves users within a specific region offloading their data to a virtual UAV, necessitating S UAVs to concurrently process the data. This forms an STMS model, as depicted in Figure 2.

The STMS model operates through the following steps: Initially, the virtual server serves UE 1 until the completion of UE 1's data transfer. Subsequently, it proceeds to serve UE 2 in a similar manner, continuing this sequence for each UE. The polling list is updated to determine the next service to be

executed, as illustrated in Figure 3.

The process of the STMS can be summarized as follows:

1. Initialization of the polling list with $i = 1$.
2. The virtual server queries node i.
3. If node i is occupied, it services node i. If not, increment i by 1 and return to step (2).
4. If $i \neq N$, return to step (2). If i reaches N, check whether the next polling will be conducted.
5. If the next polling is scheduled, return to step (1), marking the completion of the polling cycle.

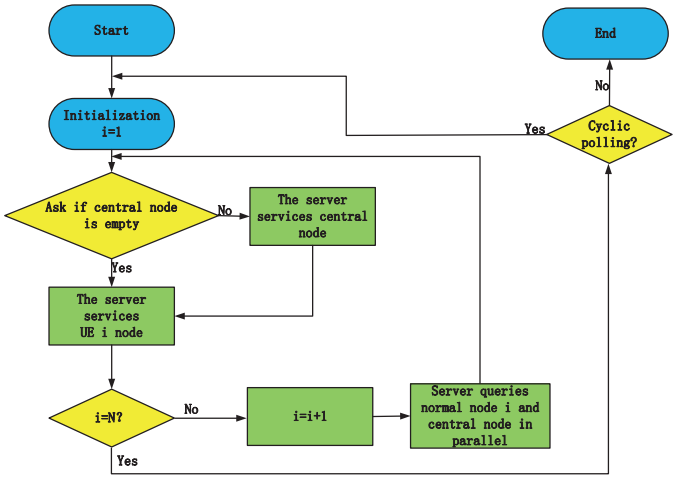

Figure 3. The flowchart of STMS.

3. Model analysis of STMS

3.1. Define random variables

We list the defined variables in Table 2.

Table 2. Definitions of random variables.

Random variables	Definition
$\mu_i(n)$	The virtual server switching time from node i to other nodes (data uploads)
$\nu_i(n)$	The time of the virtual service provided by the server to node i (calculate the data and offload the data)
$\mu_j(\mu_i)$	The amount of data entering the node j within time $\mu_i(n)$
$\eta_j(\nu_i)$	The amount of data entering the node j within time $\nu_i(n)$

3.2. System workflow and conditions of STMS

The STMS system operates through two distinct and precisely defined moments, each playing a vital role in understanding its operational functionality.

In the initial moment, denoted as t_n, the S edge servers collaborate simultaneously with a specific focus on servicing the requirements of node i. This phase is marked by a concentration of resources,

ensuring that node i's needs are met with the highest level of efficiency. Services provided during this interval encompass a range of operations, including data uploads, data computations, and data offloading, all working in concert to comprehensively address the demands of node i.

As we transition to the second moment, t_{n+1}, a significant operational shift occurs within the system. The previous emphasis on node i gives way to a swift and seamless transition, as the system promptly reorients itself to query and initiate service for node $i + 1$. This agile adaptation ensures that the requirements of node $i + 1$ are promptly and effectively met. This transition highlights the STMS model's dynamic nature and its remarkable ability to respond to the evolving demands of diverse nodes.

The entire system workflow, spanning these two pivotal moments, is vividly depicted in Figure 4. This visual representation serves as an illustrative guide to the system's progression from one moment to the next, showcasing the dynamic nature of the STMS model and its capacity to efficiently serve multiple nodes in a well-coordinated manner.

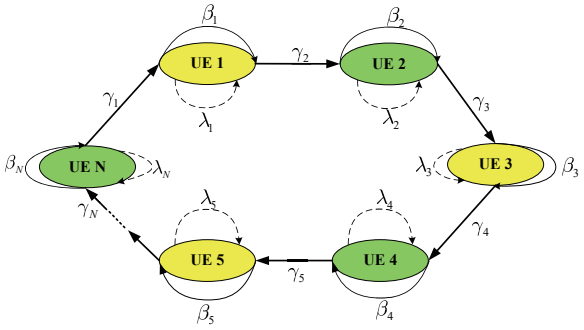

Figure 4. The workflow of STMS.

The chronological sequence follows $t_n < t_{n+1}$. Furthermore, the random variable $\xi_i(n)$ signifies the data volume stored in node i's buffer at time t_n, whereas $\xi_i(n + 1)$ represents the data volume stored in node i's buffer at time t_{n+1}. At both time points, the system's condition variables are expressed as $[\xi_1(n), \xi_2(n), \ldots, \xi_i(n), \ldots, \xi_N(n)]$ and $[\xi_1(n + 1), \xi_2(n + 1), \ldots, \xi_i(n + 1), \ldots, \xi_N(n + 1)]$. The system's state variables are delineated through acyclic and transient Markov chains within each state.

At time $t_n \to t_{n+1}$, we get:

$$\begin{cases} \xi_j(n+1) = \xi_j(n) + \eta_j(\nu_i) + \eta_j(\mu_i) \\ \xi_i(n+1) = \eta_j(\nu_i) + \eta_j(\mu_i) \end{cases}, i \neq j. \tag{3.1}$$

3.3. Mathematical models of STMS

Based on the operational principles of the STMS, the data arrival process, and the data transmission characteristics in the communication process, we define the following operational scenarios:

- Information packets arrive independently at each node following a Poisson distribution. The probability-generating function for a typical node is denoted as $A(z)$. The means and variances for these arrivals are represented as $\lambda_i (i = 1, 2, \cdots, N) = \lambda = A'(1)$ and $\sigma_\lambda^2 = A''(1) + \lambda - \lambda^2$.
- The time it takes for the server to serve any node is independently distributed and follows a Poisson distribution. The corresponding probability-generating function is represented as $B(z)$, with a mean service time denoted as $\beta = B'(1)$, and a variance of $\sigma_\beta^2 = B''(1) + \beta - \beta^2$.

- The switching times of the server from one node, say node i, to another node are independently distributed and follow a Poisson distribution. The corresponding probability-generating function is denoted as $R(z)$, with a mean switch time of $\gamma = R'(1)$ and a variance of $\sigma_\gamma^2 = R''(1) + \gamma - \gamma^2$.
- All data is first in, first out (FIFO)
- Sufficient node capacity, no data overflow

In a state of equilibrium, provided that the condition $\sum_{i=1}^{N} \lambda\beta = N\rho < S$ is satisfied, the probability distribution function for the system's state variables is as follows:

$$\lim_{n\to\infty} P[\xi_1(n), \xi_2(n), \cdots \xi_i(n), \cdots \xi_N(n)] = \pi_i(x_1, x_2, \cdots, x_i, \cdots, x_N). \tag{3.2}$$

According to the Definition of the probability-generating function, the generating function is :

$$\begin{aligned} &G_i(z_1, z_2, \cdots, z_i, \cdots, z_N) \\ &= \sum_{x_1=0}^{\infty}\sum_{x_2=0}^{\infty}\cdots\sum_{x_i=0}^{\infty}\cdots\sum_{x_N=0}^{\infty} \pi_i(x_1, x_2, \cdots, x_i, \cdots, x_N) z_1^{x_1} z_2^{x_2} \cdots z_i^{x_i} \cdots z_N^{x_N}, i = 1, 2, \cdots, N. \end{aligned} \tag{3.3}$$

At the time t_{n+1}, the server starts to transmit the data of $i+1$ node, and the probability-generating function of the system state variable is:

$$\begin{aligned} G_{i+1}(z_1, z_2, \cdots, z_i, \cdots, z_N) &= \lim_{t\to\infty} E[\prod_{j=1}^{N} z_j^{\xi_j(n+1)}] \\ &= \lim_{t\to\infty} E[\prod_{\substack{j=1\\ j\neq i}}^{N} z_j^{\xi_j(n)+\eta_j(\nu_i)+\eta_j(\mu_i)} \cdot z_i^{\eta_i(\nu_i)+\eta_i(\mu_i)}] \\ &= \lim_{t\to\infty} E[\prod_{\substack{j=1\\ j\neq i}}^{N} z_j^{\xi_j(n)+\eta_j(\nu_i)} \cdot z_i^{\eta_i(\nu_i)}] E[\prod_{j=1}^{N} z_j^{\eta_j(\mu_i)}] \\ &= \lim_{t\to\infty} E[\prod_{\substack{j=1\\ j\neq i}}^{N} z_j^{\xi_j(n)+\eta_j(\nu_i)} \cdot z_i^{\eta_i(\nu_i)}] E[\prod_{j=1}^{N} (A_j(\mathrm{z}_j))^{\eta_j(n)}] \\ &= \lim_{t\to\infty} E[\prod_{\substack{j=1\\ j\neq i}}^{N} z_j^{\xi_j(n)+\eta_j(\nu_i)} \cdot z_i^{\eta_i(\nu_i)}] R_i[\prod_{j=1}^{N} A_j(\mathrm{z}_j)] \\ &= R_i[\prod_{j=1}^{N} A_j(\mathrm{z}_j)] G_i[z_1, z_2, \cdots, z_{i-1}, B_i(\prod_{j=1}^{N} A_j(z_j)), z_{i+1} \cdots, z_N]. \end{aligned} \tag{3.4}$$

4. Analysis of system variables

To assess UAV-assisted edge computing systems effectively, it is crucial to compute essential performance metrics such as the average queue length, throughput, cycle time, and delay.

4.1. The average queue length

The average queue length reflects the level of congestion in the STMS system, and a lower average queue length corresponds to quicker data transmission. To define the average queue length $g_i(j)$ for the STMS, we consider it as the average number of information packets stored at node j when node i commences data transmission at time t_n. This can be formulated as:

$$g_i(j) = \lim_{z_1,z_2,\cdots,z_i,\cdots,z_N \to 1} \frac{\partial G_i(z_1, z_2, \cdots, z_i, \cdots, z_N)}{\partial z_j}. \tag{4.1}$$

According to Equations (3.4) and (4.1), we get:

$$g_i(i) = \frac{N\gamma\lambda(S-\rho)}{S - N\rho}. \tag{4.2}$$

4.2. The average cycle

The average cycle time is a fundamental metric for STMS. A shorter average cycle time signifies faster service delivery to UEs. To formally define the average cycle time of STMS, we consider it as the time interval between two successive queries made by the same UE to the edge server. This can be expressed as:

$$E(\theta) = \frac{NS\gamma}{S - N\rho}. \tag{4.3}$$

4.3. The system throughput

Throughput is a crucial parameter that quantifies the system's capacity to handle users and their workload. Higher throughput implies the system's ability to accommodate more users and efficiently manage a larger load. Conversely, lower throughput suggests limited capacity, allowing only a few users with reduced load handling. The throughput of STMS can be formally defined as:

$$T = NS\lambda\beta. \tag{4.4}$$

4.4. The average delay

Average delay serves as a pivotal metric for evaluating data transmission efficiency, directly impacting user data retrieval times and overall system performance. Accurate computation of the system's average delay necessitates the calculation of second-order partial derivatives of the probability-generating function. In this context, we define the second-order partial derivatives of the variables as follows:

$$g_i(j,k) = \lim_{z_1,z_2,\cdots,z_j\cdots,z_k,\cdots,z_N \to 1} \frac{\partial^2 G_i(z_1, z_2, \cdots, z_N)}{\partial z_j \partial z_k}, j = 1, 2, \cdots N; k = 1, 2, \cdots N; j \neq k. \tag{4.5}$$

According to Equations (3.4) and (4.5), we get:

$$
\begin{aligned}
g(i,i) &= \frac{N}{(1+\phi)(1-N\phi)}\{\lambda^2 R''(1) + \frac{1}{1-N\phi}[(N+2N\phi-1)\gamma^2\lambda^2 \\
&+ (N-1)\gamma\lambda^2\phi + (1+\phi-N\phi)\gamma A''(1) + N\gamma\lambda^3 B''(1)]\} \\
&i = 1, 2, \cdots N; \phi = \frac{\rho}{S}.
\end{aligned} \tag{4.6}
$$

Definition: The average delay within the STMS signifies the temporal span from an information packet's arrival at the node to its subsequent transmission. In particular, the average delay upon entry to the node is denoted as $E(w)$, and we compute it as follows:

$$
\begin{aligned}
E(W) &= \frac{(1+\phi)g(i,i)}{2\lambda g(i)} \\
&= \frac{1}{2}\{\frac{R''(1)}{\gamma} + \frac{1}{1-N\phi}[(N-1)\gamma + (N-1)\phi + 2N\gamma\phi + N\lambda B''(1) + \frac{(1+\phi-N\phi)A''(1)}{\lambda^2}]\}.
\end{aligned} \tag{4.7}
$$

5. Simulation experiment and analysis

We determined performance parameters, including the average queue length and average cycle time, for the multi-server gated service through the mathematical analysis method outlined above. To assess the method's feasibility, we created a system model using MATLAB 2022a and conducted numerical calculations along with experimental simulations. In these simulations, we assumed an ideal data communication process where all data was transmitted successfully without any packet loss or retransmission. The simulation process followed a time-slot-based division of the time axis and adhered to the following conditions:

- The system demonstrates symmetry, with random variables at each site conforming to a Poisson distribution.
- Information packets arrive at nodes during each unit time slot in accordance with a Poisson process, and node storage space is considered infinite.
- System stability is maintained under the condition that $\sum_{i=1}^{N} \lambda\beta = N\rho < S$.
- Each experiment consists of a total of 100,000 trials.

The initial parameters for the experiments are detailed in Table 3.

Table 3. Initial parameters.

i	λ_i	β_i	γ_i
Number of UE	Arrival rate	Service time	Switch time
5	0.001	2	1

Figure 5 depicts how the average queue length changes in response to varying arrival rate for STMS. Notably, the simulated values closely align with the theoretical values, validating the accuracy of our derived formula. As arrival rates increase, the average queue length in STMS similarly rises, which

aligns with real-world observations. Furthermore, an increase in the number of UAVs results in a decrease in the average queue length for STMS, demonstrating the efficacy of collaborative processing. However, excessive UAVs incur exponential costs. In cases where S=2, the average queue length in STMS decreases without significantly escalating expenses.

In Figure 6, we observe how the average cycle period changes concerning the arrival rate for STMS. Notably, the simulated and theoretical values closely align, attesting to the robustness of our simulation. As the arrival rate escalates, the average cycle period for STMS experiences a corresponding increase, a pattern consistent with real-world behavior. Moreover, an increase in the number of UAVs leads to a reduction in the average cycle period for STMS, enhancing system operational speed. When S=2, the average cycle period experiences a pronounced decrease. However, when S exceeds 2, the cycle period of STMS registers only minor reductions. These results highlight the efficient operational dynamics of STMS under varied scenarios.

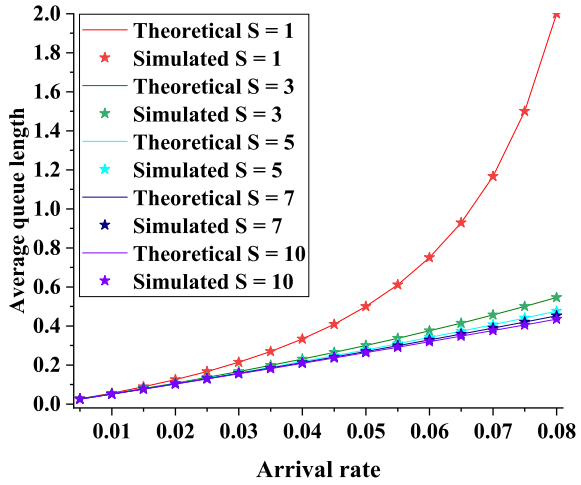

Figure 5. The average queue length of STMS varies with arrival rate.

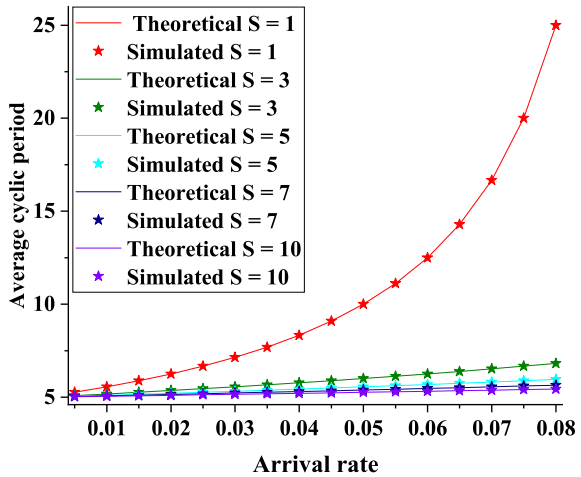

Figure 6. The average cyclic period of STMS varies with arrival rate.

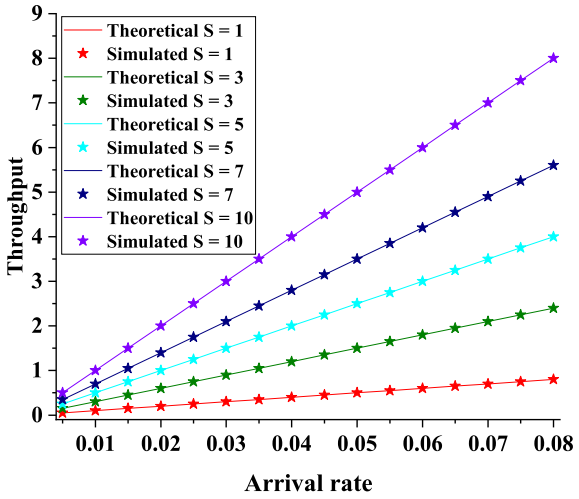

Figure 7. The throughput of STMS varies with arrival rate.

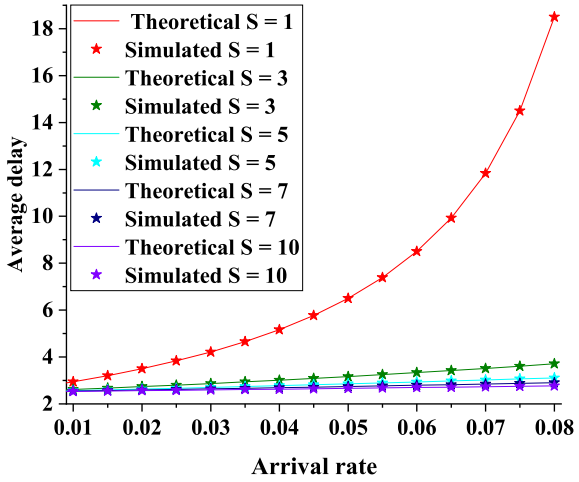

Figure 8. The average delay of STMS varies with arrival rate.

Figure 7 presents the fluctuation in throughput concerning the arrival rate for STMS. The relatively minor variance between the simulated and theoretical values signifies the system's overall coherence. STMS throughput is inherently linked to the arrival rate, the number of users, and the quantity of UAVs. As the arrival rate escalates, the throughput of STMS concurrently experiences an increase, showcasing its adaptability to the influx of data. When the number of UAVs is augmented, the STMS throughput exhibits a gradual ascent, ultimately enhancing the system's capacity to handle increased load. This observation emphasizes the potential for improving STMS operational efficiency by increasing the number of UAVs, particularly when dealing with a substantial number of users, as it bolsters the system's load-bearing capability.

Figure 8 illustrates the relationship between the average delay of STMS and the arrival rate. The proximity of the simulated values to the theoretical values affirms the effectiveness of the simulation. With an escalating arrival rate in STMS, the average delay proportionally increases, indicating that the system's capacity is constrained by a predefined number of UAVs. As the number of UAVs rises, the average delay within STMS consistently decreases, aligning with the trends observed in Figures 5

and 6. Our primary objective is to optimize system efficiency while minimizing costs. Ideally, a dual-system setup is desirable to ensure reliable and cost-effective operation. However, in scenarios with a substantial user base, deploying additional UAVs becomes a necessity to uphold system efficiency and service delivery.

6. Asynchronous mode performance prediction based on BP neural networks

6.1. Threshold multi-server asynchronous mode

The ATMS approach employs a strategy where multiple servers do not concurrently serve the same node. For the purposes of illustration, let's assume $S = 2$, and the system's configuration is depicted in Figure 9. When Server 1 initiates service at node i, it checks whether Server 2 is concurrently serving that very node. If Server 2 is found to be servicing node i, then Server 1 immediately reassigns itself to node $i + 1$. Conversely, if no concurrent service by Server 2 is detected, Server 1 proceeds to serve node i before transitioning to the subsequent node following its service completion. This operational sequence is visualized in the flowchart presented in Figure 10.

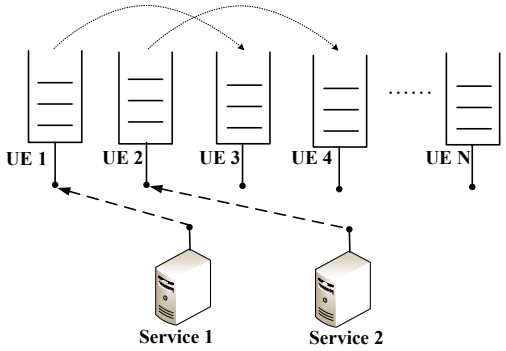

Figure 9. The system model of ATMS.

The process of ATMS is as follows:

1. Initialization of the polling list with i set to 1. The idle server queries node i.
2. If node i is unoccupied, increment i by 1, and return to step 2. Otherwise, proceed to service node i.
3. If node i is currently being serviced, increment i by 1, and return to step 2. Otherwise, the idle server commences service at node i.
4. If i is not equal to N, return to step 2. Otherwise, inquire about the need for the next polling cycle.
5. If the next polling cycle is required, return to step 1. Otherwise, conclude the polling process.

The average queue length and average delay provide valuable insights into the system's data processing efficiency. Figures 11 and 12 present simulated values for average queue length and average delay in ATMS. When comparing Figure 11 with Figure 5, it's clear that the arrival rate has a consistent impact on the average queue length. Similarly, comparing Figure 12 with Figure 8 reveals that the arrival rate consistently impacts the system's delay. Remarkably, under identical initial parameters, ATMS outperforms STMS, making it the preferred choice for UAV-assisted edge computing, providing enhanced data processing capabilities.

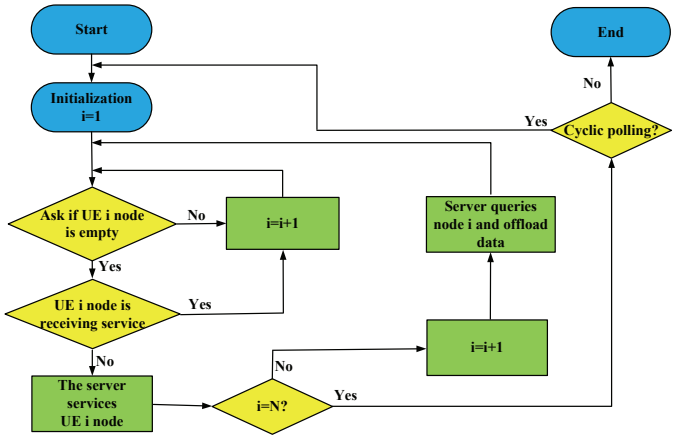

Figure 10. The flowchart of ATMS.

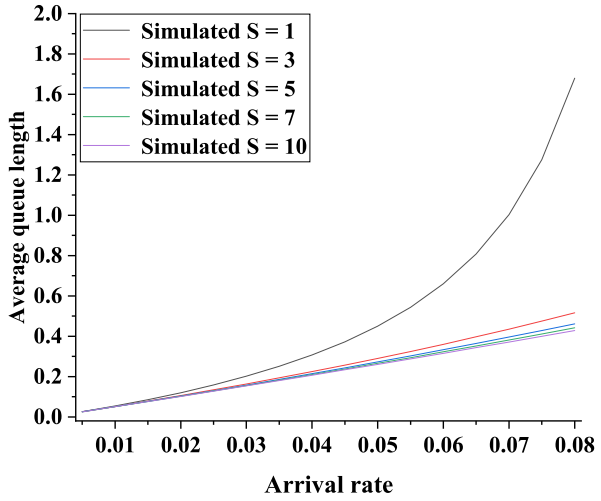

Figure 11. The average queue length of ATMS varies with arrival rate.

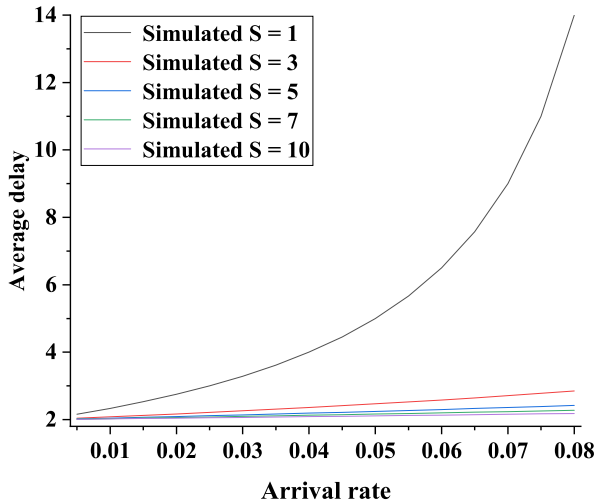

Figure 12. The average delay of ATMS varies with arrival rate.

6.2. The principle of BP neural network

The rapid expansion of mobile edge computing and the increasing demands for efficient data processing have significantly complicated network deployment and management. Accurate performance forecasting is essential for evaluating system functionality and mitigating deployment challenges. In our experiments, we introduced an ATMS model, assessing its performance through mathematical methodologies and simulation trials. We also developed a BP neural network for performance prediction, rigorously validating its reliability. Moreover, our comprehension of system performance under specific arrival rates is invaluable for anticipating system behavior in unpredictable scenarios. This understanding facilitates the fine-tuning of system parameters and the selection of an optimal system model, contributing to more efficient network management.

A BP network is a multi-layer neural network with at least three layers, each consisting of multiple neurons. The architecture of a BP neural network is depicted in Figure 13. Neurons in the left and right layers are fully interconnected, signifying that every neuron in the left layer is linked to each neuron in the corresponding right layer, with no connections bridging neurons across different layers. BP neural networks are trained through supervised learning. When the network encounters a pair of learning patterns, the activation values of its neurons propagate from the input layer, traverse through the hidden layer, and arrive at the output layer, where neurons generate a network response corresponding to the input pattern. Subsequently, output errors are iteratively rectified, starting from the output layer, and proceeding backward through each hidden layer, ultimately reaching the input layer. During this process, connection strengths are adjusted following the principle of minimizing expected errors. This sequential correction procedure, known as the error backpropagation algorithm, results in enhanced network accuracy in responding to input patterns through consistent training over time.

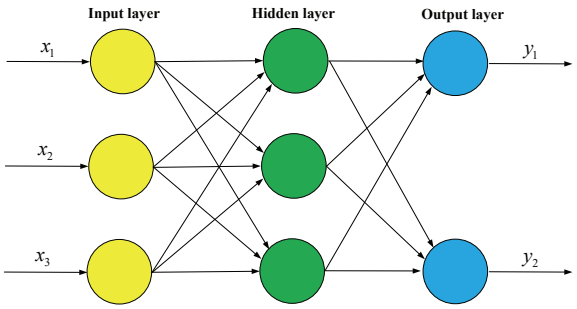

Figure 13. The BP neural network structure diagram.

6.3. Parameters of BP neural network

The architecture of a BP neural network must consider the following elements: the network's layer count, the quantity of neurons within each layer, initial settings, and the learning rate.

- The number of network layers: In the realm of neural network design, the selection of architecture is a pivotal consideration. In the case of a BP neural network, it is commonly advised to incorporate a minimum of one hidden layer utilizing sigmoid activation, along with a linear output layer, resulting in a three-tier structure. This adheres to the foundational design principle of BP neural networks, which excel at approximating rational functions. While expanding the number of network layers can lead to diminished errors and heightened accuracy, it must be noted

that this approach can simultaneously introduce complexities and prolong the training process for weight adjustments. Alternatively, an effective strategy for enhancing error accuracy involves increasing the number of neurons within the hidden layer. This approach offers the benefit of more straightforward observation and adjustment of training outcomes, making it a preferable choice over the mere addition of extra layers. Consequently, the recommendation often leans towards augmenting the number of neurons within the hidden layer, effectively balancing error reduction and network simplicity, aligning with the goal of achieving both accuracy and efficiency in practical applications.

- Hidden layer: Increasing the number of neurons within the hidden layer can be beneficial for improving network accuracy, but an excessive number of neurons may lead to overfitting. Therefore, a practical approach involves training the network with various neuron quantities and comparing their performance to determine the ideal number. Additionally, the selection of initial item weights is critical, and it's advisable to opt for random initial weights within the range of (-1,1) to promote efficient learning without the hindrance caused by excessively large or small initial weights. This combination of careful neuron selection and appropriate weight initialization contributes to a well-balanced and efficient neural network design.
- Learning efficiency: The learning rate is a critical factor that governs the extent of weight adjustments made during each training cycle. A high learning rate can lead to system instability, while a low learning rate demands a lengthier training process, resulting in slower convergence but guaranteeing that the network's error value eventually approaches the minimum threshold. Therefore, to maintain system stability while achieving efficient learning, we select a learning rate within the range of 0.01 to 0.8, ensuring a well-balanced and reliable system performance.

6.4. Optimization attempts

- Learning Rate Scheduling: Learning rate scheduling is a crucial optimization strategy in neural network training, playing a pivotal role. The core idea of this strategy is to gradually reduce the numerical value of the learning rate, which can be seen as introducing a self-adjusting mechanism in network learning to better adapt to data and enhance overall performance. The learning rate is a key parameter that controls the magnitude of weight updates in a neural network. A higher learning rate can result in excessively large weight updates, potentially causing system instability and even oscillations that hinder network convergence. Conversely, a lower learning rate leads to smaller weight updates in each step, requiring longer training time to reach the optimal state. Therefore, selecting an appropriate learning rate range is of paramount importance. Typically, we choose a moderate learning rate range, often between 0.01 and 0.8. This range maintains system reliability and stability while improving learning efficiency. The strategy of gradually reducing the learning rate allows the network to adapt to data more rapidly during the early stages of training and become more stable in the later stages, contributing to higher accuracy and generalization capability. This strategy is a key step in enhancing neural network performance, ensuring that the network balances learning speed and system stability more effectively, ultimately improving overall performance.
- Activation Function Selection: In our pursuit of enhancing our neural network's performance, we embarked on an extensive exploration to identify the most suitable activation function for our specific task. Throughout this journey, we conducted thorough experiments with various acti-

vation functions, including ReLU (rectified linear unit), Leaky ReLU (rectified linear unit with leakage), and Swish, among others. Our goal was to meticulously select an activation function that would maximize our neural network's performance. During this exploration, we delved into both the mathematical properties of these activation functions and their real-world performance. We conducted a comprehensive analysis, carefully comparing and evaluating the strengths and weaknesses of each activation function. Our assessment covered various aspects, including their non-linearity, their ability to address gradient vanishing issues, computational efficiency, and their impact on model convergence speed and generalization capabilities. After extensive experimentation and careful consideration, we ultimately opted for the Swish activation function. Swish demonstrated exceptional performance in our specific task, combining swift convergence with remarkable generalization capabilities. This choice signifies not only a technical triumph but also a testament to our profound understanding of the task requirements, ensuring the best possible performance for our neural network.

6.5. Data preparation and problem description

The evaluation of mobile edge computing systems revolves around two principal performance metrics: the average queueing length and the average data transmission delay. The former constitutes a first-order performance indicator, while the latter is a second-order characteristic. Smaller values for both metrics signify superior system performance under equivalent workloads.

Building upon the insights from Section 6.1, we maintained consistent parameters across all five ATMS sites. Parameters encompassing the number of servers, arrival rates, and average time intervals were provided as inputs for forecasting the average queue length and average delay. To broaden the scope of our predictions, we increased the number of trials to simulate average queue lengths across varying arrival rates. In these experiments, a dataset comprising 100 data sets was employed for predictive analysis.

During the data processing phase, each sample value underwent a normalization process, scaling it to a range of (-1,1). This normalization step was instrumental in standardizing the statistical distribution of the data samples, thereby enhancing the accuracy of our predictive models. The formula for this normalization process is detailed in Equation (6.1).

$$y = \frac{(y_{\max} - y_{\min}) \cdot (x - x_{\min})}{x_{\max} - x_{\min}} + y_{\min}. \tag{6.1}$$

The dataset used for training the neural network was divided into three sets: 70% for training, 15% for testing, and 15% for validation.

The forecasting task within the framework of ATMS can be formulated as a mathematical problem. We seek to predict $y1(l+1)$ given an arrival rate of $l+1$. Utilizing N historical data points as input, we derive $y1(l+1)$, which is represented in the form of the matrix y_1.

$$y_1 = \begin{bmatrix} y_{11} \\ y_{12} \\ \vdots \\ y_{1N} \end{bmatrix} = \begin{bmatrix} y_{11}(l-N+1) & y_{11}(l-N+2) & \cdots & y_{11}(l) \\ y_{12}(l-N+1) & y_{12}(l-N+2) & \cdots & y_{12}(l) \\ \vdots & \vdots & & \vdots \\ y_{1N}(l-N+1) & y_{1N}(l-N+2) & \cdots & y_{1N}(l) \end{bmatrix} \tag{6.2}$$

We can obtain y_2 by following the same method.

6.6. Model establishment

Following data preprocessing, we employed MATLAB 2022a to construct the neural network. After extensive iterative testing, we determined the network configuration to consist of an input layer with 3 nodes, a hidden layer with 10 nodes, and an output layer with 2 nodes. Equation (6.3) outlines the formula for calculating the mean square error (MSE). We set the number of training iterations at 1000, the error threshold at 4.4479×10^{-10}, and the learning rate at 0.01. Figure 14 illustrates the flowchart depicting the training process of the neural network.

$$MSE(y, \widehat{y}) = \frac{1}{n} \sum_{i=1}^{n} (\widehat{y} - y)^2. \tag{6.3}$$

The neural network model, as depicted in Figure 15, was successfully established. To overcome the issues of sluggish convergence and irregular computed values often associated with conventional BP network models in the context of ATMS modeling, we implemented iterative calculations with weights denoted as ω and thresholds as θ to serve as the initial parameters. Through this iterative approach, we derived updated initialization parameters ω' and θ' at specific points in time. Subsequently, we employed these new parameters to create the ATMS model, effectively addressing the previously mentioned challenges.

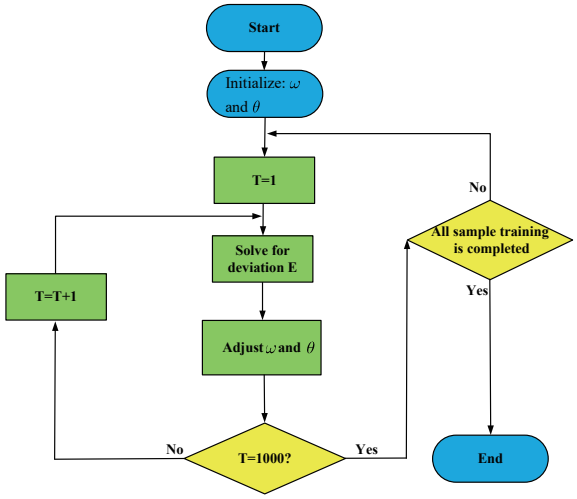

Figure 14. The flowchart of BP neural network.

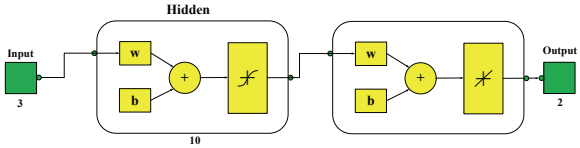

Figure 15. The BP neural network models.

7. Prediction results and analysis

The loss function is a critical metric that quantifies the disparity between predicted values and ground truth, where a lower loss value signifies a more accurate prediction. To gain a deeper insight

into the model's performance, we have visualized the loss functions for the three distinct datasets in Figure 16. The graph reveals an interesting pattern during the training process. Initially, there is a rapid and substantial reduction in the loss values, indicative of the network's rapid learning and adaptation to the data. However, as the training progresses, the rate of reduction gradually diminishes, leading to a phase where the loss values plateau. This plateauing phase highlights the model's ability to reach a certain level of predictive accuracy. Upon closer examination of the training process, the optimal validation performance is achieved at approximately the 1000th iteration. At this point, the model attains an impressively low error rate of 4.7749×10^{-10}, which underscores the network's remarkable precision in capturing the underlying patterns within the data. This observation provides valuable insights into the model's convergence and the point at which it achieves its best predictive performance.

Error statistics for the three data types are presented in Figure 17, revealing a relatively focused error distribution. The mean squared error (MSE) is computed at 4.4479×10^{-10}, with the maximum error amounting to 0.000144. These values indicate a strong model fit and high prediction accuracy.

The error statistics, as illustrated in Figure 17, provide valuable insights into the model's performance across the three different data types. The error distribution, as observed, exhibits a notably concentrated pattern, which is indicative of the model's consistent and precise predictions. This focused error distribution implies that the majority of predictions are remarkably close to the ground truth values, contributing to a high level of prediction accuracy. In particular, we calculated the MSE to quantify the average magnitude of errors. The computed MSE, which stands at 4.4479×10^{-10}, signifies the overall closeness of predictions to the actual data points. This remarkably low MSE suggests that, on average, the model's predictions deviate insignificantly from the true values, further confirming the model's strong fit to the data. Furthermore, it's worth noting that while the MSE provides a comprehensive view of prediction accuracy, we also examined the maximum error, which, in this case, amounts to 0.000144. This maximum error represents the most significant deviation between predictions and actual values within the dataset. Even at its maximum, the error remains exceptionally low, underscoring the model's ability to maintain high precision and consistency in its predictions. In summary, the error statistics depicted in Figure 17, including the low MSE and maximum error, collectively validate the robustness and high prediction accuracy of the model across a diverse range of data types. These findings reaffirm the model's ability to provide reliable predictions with remarkable consistency.

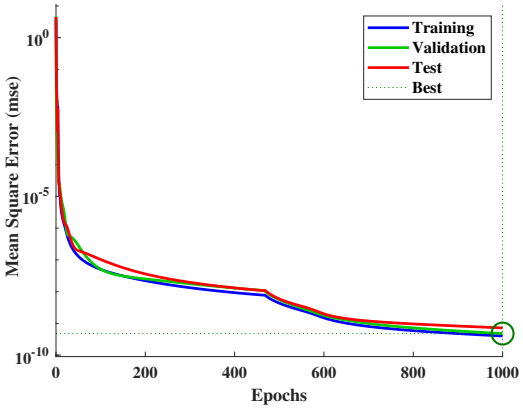

Figure 16. The loss function curves for BP neural networks.

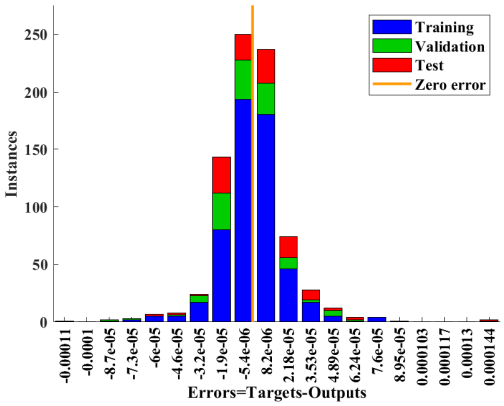

Figure 17. The error statistics for BP neural networks.

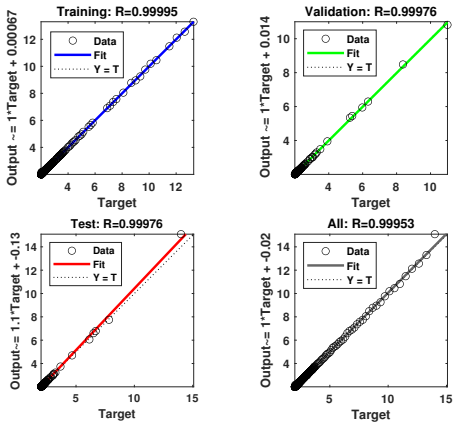

Figure 18. The regression analysis.

Figure 19 illustrates the relationship between ATMS's average queue length and the arrival rate, emphasizing the BP neural network's accurate predictions. As the arrival rate increases, the average queue length in ATMS rises, reflecting its ability to manage growing data loads efficiently. Moreover, a higher number of UAVs under constant conditions reduces the average queue length, enhancing system responsiveness. This is due to UAVs' efficiency in mobile edge computing, facilitating faster data processing and delivery. Comparing ATMS with STMS in Figure 5 under identical parameters, ATMS consistently outperforms STMS. This highlights ATMS as a superior choice for managing data traffic in mobile edge computing environments, reaffirming its efficiency and predictive strength.

Figure 20 presents the prediction graph for ATMS delays, vividly showcasing the minimal disparities between predicted and theoretical values, further emphasizing the outstanding performance of the established BP neural network model in delivering highly accurate predictions. In line with the earlier conclusion, delays in ATMS increase with the arrival rate. And, an encouraging finding is the direct comparison between Figure 20 and Figure 8, which highlights ATMS's significant advantage over STMS under the same parameters. It's worth noting that in both ATMS and STMS, increasing the number of auxiliary UAVs results in a remarkable reduction in latency and a substantial improvement in throughput. This observation further underscores the potential of auxiliary UAVs in enhancing system

responsiveness and performance. In summary, the results from Figure 20, along with the comparison with Figure 8, provide compelling evidence of the predictive capabilities of the BP neural network model and the superior performance of ATMS relative to STMS. This will offer valuable guidance for future research and system optimization.

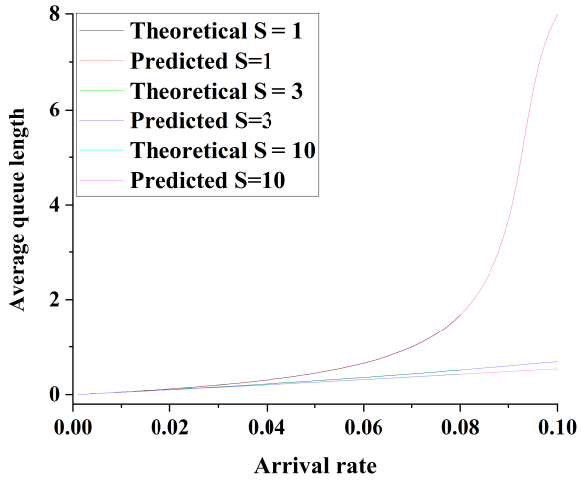

Figure 19. The average queue length of ATMS varies with arrival rate.

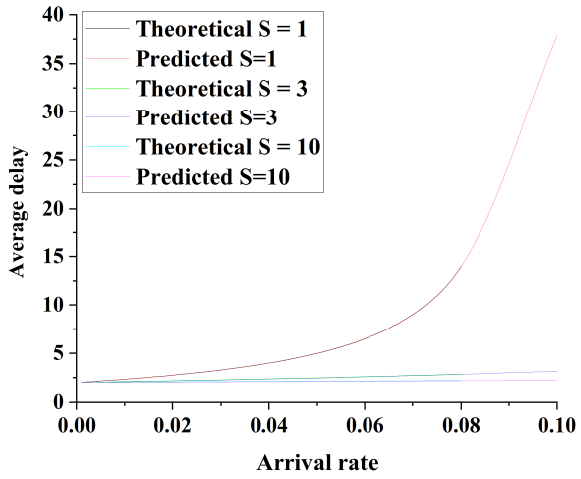

Figure 20. The average delay of ATMS varies with arrival rate.

8. Conclusion

In this research, we have explored UAV-assisted MEC networks and developed a multi-server polling scheduling methodology for both STMS and ATMS models. For STMS, we have harnessed the power of probability generating functions and harnessed Markov chains to compute fundamental metrics encompassing average queue length, average cycle duration, throughput, and mean delay. Numerous experiments have confirmed the accuracy of our equations by showing the convergence of theoretical and simulated values. In contrast, for ATMS, we used a neural network to predict performance. Our research findings have invariably underscored the supremacy of ATMS over STMS under

identical parameters within the domain of MEC networks. The increasing demand for internet services and the need for improved service quality guide our future research. We will focus on developing multi-server service strategies to meet these growing demands. We will particularly focus on carefully determining parameters to enhance precision. Furthermore, in our future research, we will actively explore optimization methods and algorithms to reduce energy consumption. Our main goal is to improve energy efficiency and sustainability by implementing strategies and algorithms, thus reducing energy costs.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This research received support from multiple funding sources, including Yang Zhijun's Industry Innovation Talents Project of Yunnan Xingdian Talents Support Plan (No. YNWR-CYJS-2020-017), the Yunnan Province Wu Zhonghai Expert Workstation (No. 202305AF150045), the National Natural Science Foundation of China (NSFC) under grant numbers 61461054 and 61461053, as well as funding from the Yunnan University Graduate Research Innovation Project (No. TM-23237070).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. S. M. AHuda, S. Moh, Survey on computation offloading in UAV-Enabled mobile edge computing, *Network Computer Appl.*, **5** (2022), 103341. https://doi.org/10.1016/j.jnca.2022.103341
2. C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, L. Guo, Computation offloading in mobile edge computing networks: A survey, *Network Comput. Appl.*, (2022), 103366. https://doi.org/10.1016/j.jnca.2022.103366
3. L. Chen, S. Tang, V. Balasubramanian, J. Xia, F. Zhou, L. Fan, Physical-layer security based mobile edge computing for emerging cyber-physical systems, *Comput. Commun.*, **194** (2022), 180–188. https://doi.org/10.1016/j.comcom.2022.07.037
4. X. Lai, J. Xia, L. Fan, T. Q. Duong, A. Nallanathan, Outdated access point selection for mobile edge computing with cochannel interference, *IEEE Transact. Vehicular Technol.*, **71** (2022), 7445–7455. https://doi.org/10.1109/TVT.2022.3167405
5. Y. Guo, R. Zhao, S. Lai, L. Fan, X. Lei, G. K. Karagiannidis, Distributed machine learning for multiuser mobile edge computing systems, *IEEE J. Select. Topics Signal Process.*, **16** (2022), 460–473. https://doi.org/10.1109/JSTSP.2022.3140660
6. H. Jiang, X. Dai, Z. Xiao, A. K. Iyengar, Joint task offloading and resource allocation for energy-constrained mobile edge computing, *IEEE Transact. Mobile Comput.*, (2022). https://doi.org/10.1109/TMC.2022.3150432

7. Y. Y. Cui, D. G. Zhang, T. Zhang, J. Zhang, M. Piao, A novel offloading scheduling method for mobile application in mobile edge computing., *Wireless Networks*, **28** (2022), 2345–2363. https://doi.org/10.1007/s11276-022-02966-2

8. S. K. U. Zaman, A. I. Jehangiri, T. Maqsood, N. U. Haq, A. I. Umar, J. Shuja, et al., LiMPO: Lightweight mobility prediction and offloading framework using machine learning for mobile edge computing, *J. Cluster Comput.*, **26**(2023), 99–117. https://doi.org/10.1007/s10586-021-03518-7

9. B. Jiang, S. Chen, B. Wang, B. Luo, MGLNN: Semi-supervised learning via multiple graph cooperative learning neural networks, *Neural Networks*, **153** (2022), 204–214. https://doi.org/10.1016/j.neunet.2022.05.024

10. A. Singh, K. Raj, T. Kumar, S. Verma, A. M. Roy, Deep learning-based cost-effective and responsive robot for autism treatment, *Drones*, **7** (2023), 81. https://doi.org/10.3390/drones7020081

11. N. Zhao, Z. Ye, Y. Pei, Y. C. Liang, D. Niyato, Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing, *IEEE Transact. Wireless Commun.*, **21** (2022), 6949–6960. https://doi.org/10.1109/TWC.2022.3153316

12. Q. Chen, H. Zhu, L. Yang, X. Chen, S. Pollin, E. Vinogradov, Edge computing assisted autonomous flight for UAV: Synergies between vision and communications, *IEEE Commun. Magaz.*, **59** (2022), 28–33. https://doi.org/10.1109/MCOM.001.2000501

13. P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, S. Papavassiliou, Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty, *IEEE Transact. Mobile Comput.*, **22** (2023), 175–190. https://doi.org/10.1109/TMC.2021.3069911

14. S. R. Sabuj, D. K. P. Asiedu, K. J. Lee, H. S. Jo, Delay optimization in mobile edge computing: Cognitive UAV-assisted eMBB and mMTC services, *IEEE Transact. Cognit. Commun. Network.*, **8** (2023), 1019–1033. https://doi.org/10.1109/TCCN.2022.3149089

15. W. Lu, Y. Mo, Y. Feng, Y. Gao, N. Zhao, Y. Wu, et al., Secure transmission for multi-UAV-assisted mobile edge computing based on reinforcement learning, *IEEE Transact. Network Sci. Eng.*, **10** (2023), 1270–1282. https://doi.org/10.1109/TNSE.2022.3185130

16. Y. Liu, J. Yan, X. Zhao, Deep reinforcement learning based latency minimization for mobile edge computing with virtualization in maritime UAV communication network, *IEEE Transact. Vehicular Technol.*, **71** (2022), 4225–4236. https://doi.org/10.1109/TVT.2022.3141799

17. Z. Liu, Y. Cao, P. Gao, X. Hua, D. Zhang, T. Jiang, Multi-UAV network assisted intelligent edge computing: Challenges and opportunities, *China Commun.*, **19** (2023), 258–278. https://doi.org/10.23919/JCC.2022.03.019

18. T. Tan, M. Zhao, Z. Zeng, Joint offloading and resource allocation based on UAV-assisted mobile edge computing, *ACM Transact. Sensor Networks (TOSN)*, **18** (2022), 1–21. https://dl.acm.org/doi/abs/10.1145/3476512

19. M. H. Mousa, M. K. Hussein, Efficient UAV-based mobile edge computing using differential evolution and ant colony optimization, *PeerJ Comput. Sci.*, **8** (2022). https://doi.org/10.7717/peerj-cs.870

20. H. Zhou, Z. Wang, G. Min, H. Zhang, UAV-Aided Computation offloading in mobile-edge computing networks: A stackelberg game approach, *IEEE Int. Things J.*, **10** (2023), 6622–6633. https://doi.org/10.1109/JIOT.2022.3197155

21. Z. Wang, H. Rong, H. Jiang, Z. Xiao, F. Zeng, A load-balanced and energy-efficient navigation scheme for UAV-mounted mobile edge computing, *IEEE Transact. Network Sci. Eng.*, **9** (2022), 3659–3674. https://doi.org/10.1109/TNSE.2022.3188670

22. Z. Yang, S. Bi, Y. J. A. Zhang, Dynamic offloading and trajectory control for UAV-enabled mobile edge computing system with energy harvesting devices, *IEEE Transact. Wireless Commun.*, **21** (2022), 10515–10528. 10.1109/TWC.2022.3184953

23. J. Lin, L. Huang, H. Zhang, X. Yang, P. Zhao, A novel Lyapunov based dynamic resource allocation for UAVs-assisted edge computing, *Comput. Networks.*, **205** (2022), 108710. https://doi.org/10.1016/j.comnet.2021.108710

24. X. Wei, L. Cai, N. Wei, P. Zou, J. Zhang, S. Subramaniam, Joint UAV trajectory planning, DAG task scheduling, and service function deployment based on DRL in UAV-empowered edge computing, *IEEE Int. Things J.*, **10** (2023), 12826–12838. https://doi.org/10.1109/JIOT.2023.3257291

25. X. Zhang, Y. Wang, DeepMECagent: Multi-agent computing resource allocation for UAV-assisted mobile edge computing in distributed IoT system, *Appl. Intell.*, **53** (2023), 1180–1191. https://doi.org/10.1007/s10489-022-03482-8

26. B. Kim, J. Jang, J. Jung, J. Han, J. Heo, H. Min, A computation offloading scheme for UAV-edge cloud computing environments considering energy consumption fairness, *Drones*, **7** (2023), 139. https://doi.org/10.3390/drones7020139

27. X. Wang, Z. Yang, H. Ding, Application of polling scheduling in mobile edge computing, *Axioms*, **12** (2023), 709. https://doi.org/10.3390/axioms12070709

28. W. Liu, B. Li, W. Xie, Y. Dai, Z. Fei, Energy Efficient Computation offloading in aerial edge networks with multi-agent cooperation, *IEEE Transact. Wireless Commun.*, **22** (2023), 5725–5739. https://doi.org/10.1109/TWC.2023.3235997

29. M. A. A. Boon, R. D. van der Mei, E. M. M. Winands, Applications of polling systems, *Surveys Operat. Res. Manag. Sci.*, **16** (2011), 67–82. https://doi.org/10.1016/j.sorms.2011.01.001

30. R. Suman, A. Krishnamurthy, Analysis of tandem polling queues with finite buffers, *Ann. Operat. Res.*, **293** (2020), 343–369. https://doi.org/10.1007/s10479-019-03358-0

AIMS Press