

Article

Task-Offloading Optimization in Mobile Edge Computing for Smart Library Services

Jingjing Qu ¹, Peiying Zhang ^{2,*} , Ruixin Wang ², Xiangguo Zheng ² and Lijuan Chen ²

¹ Shandong Vocational College of Light Industry, Zibo 255300, China; qujingjing@sdlivc.edu.cn

² Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China; z24070100@s.upc.edu.cn (R.W.); z24070095@s.upc.edu.cn (X.Z.); z24070031@s.upc.edu.cn (L.C.)

* Correspondence: zhangpeiying@upc.edu.cn

Abstract

With the rapid development of artificial intelligence and Internet of Things technologies, smart libraries increasingly require low-latency and energy-efficient computing support for heterogeneous services such as access control, intelligent recommendation, indoor navigation, and book localization. To address the limitations of cloud-only processing, this paper investigates task-offloading optimization in a cloud-assisted mobile edge computing environment for smart library services. A three-tier cloud–edge–device collaborative architecture is first established, and the task-offloading problem is formulated as a multi-objective optimization problem that jointly minimizes task-completion delay and user-side energy consumption under latency, resource-capacity, and coverage constraints. To solve the dynamic decision-making problem, a preference-adaptive dueling double deep Q-network algorithm, termed PA-DDQN, is proposed by integrating preference conditioning, multi-head attention, a dueling architecture, and double Q-learning. Simulation results show that PA-DDQN achieves better performance than fixed offloading strategies and representative reinforcement-learning baselines. Under the heaviest task load, PA-DDQN reduces the average task-completion delay by 23.1% and 31.0% compared with D3QN and DDQN, respectively, while reducing energy consumption by 5.8% and 9.9%. It also improves the task success rate by 14.8% and 21.7%, demonstrating its effectiveness in enhancing service responsiveness, energy efficiency, and reliability in smart library MEC systems.

Keywords: smart library; mobile edge computing; task offloading; multi-objective optimization; deep reinforcement learning; preference adaptation



Academic Editors: Vasco N. G. J. Soares and Marcin Paprzycki

Received: 16 May 2026

Revised: 25 June 2026

Accepted: 4 July 2026

Published: 8 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

With the rapid development of artificial intelligence (AI) and the Internet of Things (IoT), traditional libraries are being transformed from collection-centered information repositories into user-centered smart service ecosystems [1]. Smart libraries are no longer limited to book circulation and digital resource retrieval. Instead, they increasingly provide diversified services, including intelligent navigation, personalized recommendation, indoor positioning, unattended borrowing and returning, security inspection, and real-time learning support. Recent studies on smart libraries emphasize that AI- and IoT-enabled library systems are typically built around smart services, smart sustainability, and smart security, with large numbers of sensors, mobile devices, and intelligent terminals continuously generating service requests and operational data [2]. In this context, the library service environment becomes highly dynamic, heterogeneous, and delay-sensitive.

The continuous expansion of intelligent library services has put forward new requirements for computing and communication infrastructure. Many service tasks in smart libraries, such as visual recognition for unmanned access control, augmented reality navigation, real-time recommendations, and emergency response, require timely data processing and rapid feedback [3]. If all tasks are transmitted to a remote cloud data center, remote communication may result in high transmission latency, unstable service quality, and network congestion, especially during peak periods such as exam weeks, large academic events, and library opening or closing times [4]. Although cloud computing provides abundant computing and storage resources, its centralized architecture is not always suitable for the deployment of latency-critical and context-aware applications near users and physical library spaces [5].

Mobile Edge Computing (MEC) has emerged as an effective computing paradigm for addressing these limitations [6]. By deploying computing, caching, and service capabilities at network edge nodes, MEC can shorten the distance between end users and computing resources, thereby reducing service latency and improving quality of experience. MEC has been widely recognized as a key enabler for delay-sensitive and computation-intensive applications, particularly in environments where terminal devices have limited computing capacity and energy resources [7]. Therefore, integrating MEC into smart library infrastructure can provide a promising solution to support real-time and intelligent library services. However, the smart library scenario is not merely a generic indoor MEC case. Library services combine public-service access control, radio-frequency identification (RFID)-based collection management, shelf-level book localization, indoor navigation, personalized recommendation, and batch inventory in the same physical environment. These workflows generate tasks with distinct service semantics and deadlines: entrance authentication must avoid reader queues, book finding and navigation require zone-aware feedback near shelves, recommendation and retrieval tasks tolerate moderate delay, and inventory tasks can be processed during off-peak periods. Therefore, offloading decisions need to consider library-zone functions, workflow urgency, and deployment constraints, in addition to wireless and computing states.

The main contributions of this paper are summarized as follows:

1. **A smart library-oriented cloud-edge-device collaborative computing framework is developed.** A three-tier mobile edge computing architecture is constructed for smart library services, consisting of heterogeneous user devices, indoor MEC servers, and a remote cloud center. In addition, typical library service tasks are characterized according to their computational workloads and latency requirements, providing practical task settings for the simulation experiments and task-offloading optimization in smart library scenarios.
2. **A multi-objective task-offloading optimization model is formulated for MEC-enabled smart libraries.** By jointly considering task-completion delay, user-side energy consumption, latency constraints, edge-server resource limitations, and indoor coverage conditions, the task-offloading problem is modeled as a mixed-integer nonlinear programming problem. This formulation enables a systematic trade-off between service responsiveness and energy efficiency.
3. **A preference-adaptive reinforcement learning-based offloading algorithm is proposed.** To address the dynamic and multi-objective nature of the problem a preference-adaptive dueling double deep Q-network algorithm, termed PA-DDQN, is designed. The proposed method integrates preference conditioning, multi-head attention, a dueling network architecture, and double Q-learning, allowing a single trained model to adaptively generate offloading decisions under different delay-energy preference settings.

2. Related Work

2.1. Smart Library Services and Edge-Enabled Intelligent Infrastructure

Smart library systems have attracted increasing research attention with the development of the IoT, RFID, wireless sensor networks, cloud computing, and artificial intelligence. In [8], the author proposed a user demand-based hybrid continuous ant colony optimization (BACO) method, which solved the facial image segmentation problem in the construction of a discipline precision service platform for smart libraries. In [9], the author proposed a low-cost smart library architecture, adopted a software-defined network and cluster-based topology, and integrated passive RFID tags for book management. The effectiveness of the proposed architecture was verified through practical deployment and simulation. The authors of [4] proposed an IoT-based low-cost smart library architecture using software-defined networking (SDN). Their architecture integrated RFID tags, RFID readers, SDN controllers, data centers, and network sensors to support authentication, property circulation, and book-loan management. The use of SDN improved network management flexibility and reduced implementation cost. These studies indicate that IoT and RFID technologies are effective for improving the automation level of library services.

In addition to library automation, recent smart library research has also considered intelligent service functions such as recommendation, indoor navigation, environmental monitoring, and user behavior analysis. These services generate heterogeneous tasks with different data sizes, computing demands, and delay requirements [10]. However, most existing smart library studies emphasize service architecture, information management, and system deployment, while the underlying task execution and computing-resource optimization problems are not sufficiently addressed. In particular, few studies have examined how smart library service tasks should be offloaded and processed when local devices, edge servers, and cloud platforms coexist. Therefore, introducing MEC-based task offloading into smart library services is a meaningful research direction.

2.2. Task Offloading in Mobile Edge Computing

Task offloading is a fundamental issue in MEC because it directly affects task-completion delay, energy consumption, and edge-resource utilization. The authors of [11], the proposed a task-offloading problem based on cooperative mobile edge computing networks, formulated it as a convex optimization problem with nonlinear constraints, and further developed a distributed algorithm. The authors of [12] formulated a joint offloading and resource allocation problem for multi-task dependencies in an IoT environment and designed a collaborative algorithm to realize the optimal scheduling of computing and communication resources. The authors of [13] proposed a method combining partial offloading and collaborative mobile edge computing and adopted a two-layer alternating optimization framework. In the upper layer, an improved genetic algorithm was used to generate the population of offloading decisions, while in the lower layer, the deep deterministic policy gradient algorithm was applied to optimize the offloading strategy and weight coefficients. The authors of [14] addressed the task-offloading problem under incomplete edge information, proposed an energy minimization scheme, adopted the exact convex regularization method to estimate resource availability, and realized efficient scheduling. The authors of [15] proposed a task feature extraction method based on dependency-aware graph neural networks and designed a multi-agent deep reinforcement learning algorithm with dependency-aware graphs to address the task-offloading and resource optimization problems in MEC. These studies provide important foundations for MEC task offloading.

2.3. Intelligent Optimization Algorithms for MEC Offloading

Because MEC task offloading usually involves binary or partial offloading decisions, wireless-channel variation, computing-resource constraints, and task deadlines, the resulting optimization problem is often nonlinear and combinatorial. Traditional mathematical optimization methods can obtain good solutions in relatively small-scale or static scenarios, but their computational complexity may increase rapidly when the number of users, tasks, and edge servers grows. Therefore, intelligent optimization algorithms have been increasingly applied to MEC offloading problems.

The authors of [16] formulated the task-offloading and resource allocation problem in mobile edge computing under a heterogeneous task environment and proposed a solution based on deep reinforcement learning to optimize channel allocation and reduce task-completion delay. The authors of [17] addressed the problems of slow learning and high task latency in wireless mobile edge computing, proposed a task-offloading framework based on deep reinforcement learning, and optimized the offloading strategy by adopting deep neural networks and the upper bound of the minimum target Q-value. The authors of [18] proposed an energy-efficient task-offloading framework for multi-IoT and multi-server edge computing systems and integrated a load-balancing algorithm, a compression layer, and deep reinforcement learning techniques to minimize system energy consumption.

Although intelligent algorithms can improve the adaptability and decision-making efficiency of MEC systems, the above studies still have several limitations when applied to smart library services. First, existing smart library studies mainly focus on service architecture, RFID-based management, information retrieval, and intelligent service design, but they seldom investigate how heterogeneous service tasks should be executed across local devices, indoor MEC servers, and the remote cloud. Second, many MEC offloading studies optimize task scheduling or resource allocation in general IoT, wireless, satellite, industrial, or mining scenarios, where the task types, service priorities, and indoor deployment assumptions differ from those of smart libraries. Third, most existing learning-based offloading methods are designed under fixed optimization preferences and, thus, cannot flexibly adapt to different delay–energy trade-off requirements caused by real-time access control, interactive recommendation, book localization, and delay-tolerant inventory tasks. To address these limitations, this paper formulates a smart library-oriented cloud–edge–device task-offloading model and proposes a preference-adaptive PA-DDQN algorithm that jointly considers heterogeneous library service tasks, indoor edge-resource constraints, and adjustable delay–energy preferences.

3. System Model and Problem Formulation

3.1. System Architecture

We consider a three-tier cloud-assisted MEC system tailored for smart library environments, consisting of heterogeneous user equipment, edge computing servers deployed at fixed indoor locations, and a remote cloud center. The set of user devices is denoted by $\mathcal{U} = \{1, 2, \dots, U\}$, which includes self-service kiosks, handheld tablets, smartphones, RFID readers, and face recognition terminals. Each device type (u) is characterized by its local computing capacity ($f_u^{\max}$, in CPU cycles per second) and energy efficiency coefficient (κ_u), reflecting the hardware heterogeneity common in public library settings. These devices generate a variety of real-time and delay-sensitive tasks such as identity authentication, real-time book localization, multimedia retrieval, and intelligent recommendation.

The set of MEC servers is denoted by $\mathcal{S} = \{1, 2, \dots, S\}$, each deployed at a specific library zone. In practical smart library deployment, an MEC server can be co-located with a Wi-Fi access point, RFID gateway, or local service gateway in functional areas such as the entrance hall, self-service area, reading rooms, book-stack areas, and multimedia area.

The zone in which a task is generated determines the candidate edge servers because bookshelves, walls, and functional partitions create different coverage and blockage conditions across the library. Each server (s) has a known fixed position ($\mathbf{l}_s$) and a finite computational capacity ($F_s^{\max}$, in CPU cycles per second). The distance ($d_{u,s}(t)$) between a user device (u) and server (s) is bounded by the indoor coverage radius (R_s , typically 20–50 m), and its variations are negligible over short time intervals due to low pedestrian walking speeds. The remote cloud server is indexed as $s = 0$ and is assumed to have abundant computing resources but incurs a fixed round-trip communication delay dominated by wide-area network latency, independent of the user's indoor position.

Each computational task can be executed locally on the user device, offloaded to any nearby MEC server ($s \in \mathcal{S}$) that covers the user's current zone, or sent to the cloud. The offloading decision is captured by a binary variable ($x_{u,s} \in \{0,1\}$, where $x_{u,s} = 1$ means task u is assigned to node s ; otherwise, $x_{u,s} = 0$). Since a task can be processed at exactly one location, the constraint of $\sum_{s \in \{0\} \cup \mathcal{S}} x_{u,s} = 1$ must hold for every $u \in \mathcal{U}$. The set of feasible offloading options for a user depends on its current zone: if the user is within the coverage radius (R_s) of server s , then $x_{u,s}$ may be set to 1; otherwise, that offloading action is invalid. This geographical constraint is formally expressed as $d_{u,s}(t) \leq R_s$ for any allowed offloading to $s \in \mathcal{S}$.

As illustrated in Figure 1, the proposed smart library MEC system follows a cloud–edge–device collaborative architecture. User devices generate heterogeneous service tasks in different library zones, while nearby MEC servers provide low-latency computing support within their coverage areas. When local or edge resources are insufficient, tasks can also be further offloaded to the remote cloud center.

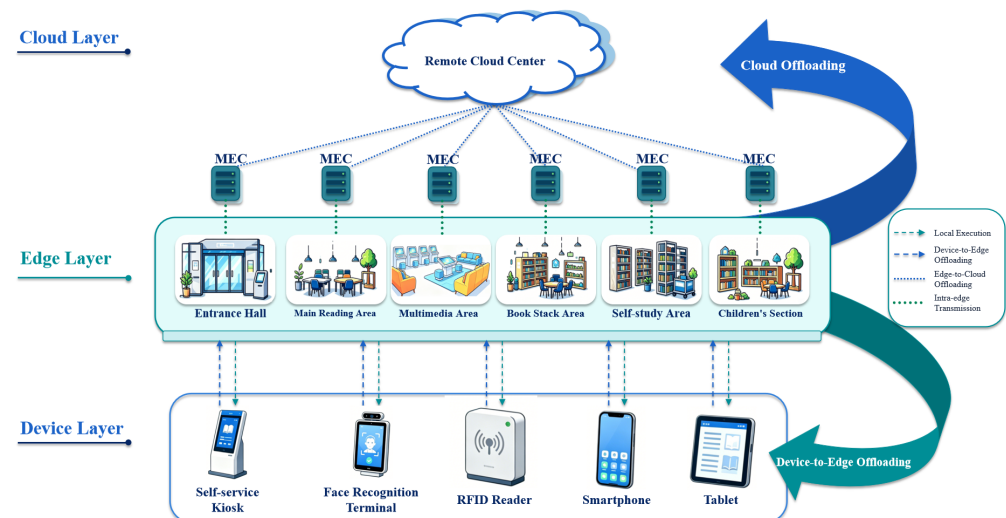


Figure 1. System architecture of the smart library-oriented cloud–edge–device collaborative MEC framework.

3.2. Task Model

In a smart library powered by mobile edge computing, user devices continuously generate computing tasks with diverse characteristics. Based on real-world library operations, typical services are categorized into three classes according to their computational and delay requirements. The first class comprises lightweight real-time tasks such as identity authentication and book-status querying, which involve small data inputs of a few kilobytes; require few CPU cycles, amounting to tens of thousands; and are highly delay-sensitive, with tolerable latency of up to one hundred milliseconds. The second class consists of medium-weight interactive tasks including multimedia retrieval and intelligent recommen-

dation. These tasks have moderate data sizes of hundreds of kilobytes and CPU workloads of millions of cycles, with relaxed deadlines of around five hundred milliseconds. The third class covers heavyweight offline tasks such as book location using three-dimensional indoor mapping or a batch radio-frequency identification inventory. Such tasks are delay-tolerant up to several seconds but consume substantial computing resources and may involve large data inputs on the megabyte scale. This classification enables differentiated offloading policies. For example, in an unattended borrowing workflow, a face-recognition terminal first verifies the reader identity, an RFID reader then checks the book tag and loan status, and the response must be returned before the reader leaves the gate. In an indoor book-finding workflow, a mobile terminal or tablet sends the target-book query, combines it with shelf/RFID information, and receives a zone-level navigation result, which is affected by nearby MEC availability and shelf blockage. In contrast, batch RFID inventories and large-scale data synchronization can be assigned to lightly loaded edge servers or the cloud because they are less delay-critical. Nevertheless, for analytical tractability, each task (u) is modeled as an atomic and non-preemptive unit, meaning it cannot be split into sub-tasks or preempted once started. This simplification is acceptable because the majority of library tasks are processed in short, indivisible sessions. Formally, a task generated by user u is described by a four-tuple:

$$\mathcal{T}_u = (D_u^{\text{in}}, C_u, D_u^{\text{out}}, T_u^{\text{max}}), \quad (1)$$

where D_u^{in} is the size of the input data, including service request parameters; C_u is the total number of CPU cycles required; D_u^{out} is size of the output data; and T_u^{max} is the maximum tolerable end-to-end delay.

Task arrivals in a library follow a time-varying, non-homogeneous Poisson process because user activity peaks during rush hours and drops significantly at other times. To capture this, the arrival rate ($\lambda_u(t)$) is modeled as a periodic piecewise function. At the beginning of each time slot, newly arriving tasks are appended to a first in–first out queue per device to ensure fairness. The downlink result is typically much smaller than the uplink input, and the downlink channel rate is generally higher than that of the uplink. Therefore, the delay and energy for result feedback are omitted, which is a standard practice in edge computing research.

3.3. Communication and Computation Model

In the smart library edge computing environment, the performance of task offloading is jointly determined by indoor wireless propagation conditions and the allocation of heterogeneous computing resources. An accurate communication model is required to characterize the uplink transmission rate, delay, and energy consumption, while a refined computation model quantifies execution delay and energy cost under local, edge, and cloud modes. These models serve as the foundation for offloading decisions and resource allocation optimization.

For indoor wireless links between user devices and edge servers, the signal propagation experiences both line-of-sight and non-line-of-sight components due to bookshelves, walls, and other obstacles. A practical indoor path-loss model is adopted, where the channel power gain from user u to node s is expressed as

$$g_{u,s} = \frac{G_0}{d_{u,s}^\alpha} \cdot \psi_{u,s}, \quad (2)$$

where G_0 is the reference gain at one meter, $d_{u,s}$ is the Euclidean distance, α is the path-loss exponent (typically between 2 and 4, depending on the indoor environment), and $\psi_{u,s}$ is a log-normal shadowing term capturing random blockage effects. Since pedestrian walking speeds in a library are low, the distance ($d_{u,s}$) and the shadowing term vary slowly and can

be assumed to remain constant within each time slot. The uplink data transmission rate from user u to node s follows the Shannon–Hartley formula [19]:

$$r_{u,s} = W \log_2 \left(1 + \frac{P_u^{\text{tx}} g_{u,s}}{N_0 + I_{u,s}} \right), \quad (3)$$

where W is the channel bandwidth, P_u^{tx} is the fixed transmission power, N_0 is the noise power spectral density, and $I_{u,s}$ represents the co-channel interference from nearby devices. The transmission delay for uploading of the input data of task u to node s is

$$t_{u,s}^{\text{up}} = \frac{D_u^{\text{in}}}{r_{u,s}}, \quad (4)$$

and the corresponding energy consumption is

$$e_{u,s}^{\text{up}} = P_u^{\text{tx}} \cdot t_{u,s}^{\text{up}}. \quad (5)$$

Consistent with standard edge computing literature, the size of the downlink result is much smaller than the uplink data, and the downlink channel rate is higher, so the delay and energy of result return are omitted.

For the computation model, three execution modes are considered: local, edge-server, and cloud execution. When task u executes locally on the user device, the allocated computing capacity is denoted by f_u^{loc} , which cannot exceed the device's maximum capacity (f_u^{max}) introduced in the system architecture. The local execution delay is

$$T_u^{\text{loc}} = \frac{C_u}{f_u^{\text{loc}}}, \quad (6)$$

and the energy consumption is given by the CMOS dynamic power model [20]:

$$E_u^{\text{loc}} = \kappa_u \left(f_u^{\text{loc}} \right)^2 C_u, \quad (7)$$

where κ_u is the effective capacitance coefficient of the device.

When task u is offloaded to an edge server (s) located in the same library zone, the server allocates a share of its computational capacity ($f_{u,s}^{\text{mec}}$) to this task, with the total assigned to all tasks not exceeding F_s^{max} . The processing delay on the edge server is

$$T_{u,s}^{\text{mec}} = \frac{C_u}{f_{u,s}^{\text{mec}}}, \quad (8)$$

and the total delay for edge offloading comprises both the uplink transmission and the edge processing:

$$T_{u,s}^{\text{all}} = t_{u,s}^{\text{up}} + T_{u,s}^{\text{mec}}. \quad (9)$$

Energy consumed by the edge server is not billed to the user; therefore, the user-side energy for edge offloading is only the transmission part:

$$E_{u,s}^{\text{all}} = e_{u,s}^{\text{up}}. \quad (10)$$

When task u is offloaded to the remote cloud, the cloud is assumed to have abundant computing resources, but the wide-area network introduces a fixed round-trip latency (T^{RTT}), in addition to the uplink transmission and cloud processing times. The cloud processing delay is

$$T_u^{\text{cloud}} = \frac{C_u}{f_{\text{cloud}}}, \quad (11)$$

where f^{cloud} is the cloud CPU frequency allocated to the task. Therefore, the total delay for cloud offloading is

$$T_{u,0}^{\text{all}} = t_{u,0}^{\text{up}} + T^{\text{RTT}} + T_u^{\text{cloud}}. \quad (12)$$

The user-side energy consumption for cloud offloading remains the uplink transmission energy:

$$E_{u,0}^{\text{all}} = e_{u,0}^{\text{up}}. \quad (13)$$

With these expressions, the delay and energy for any offloading decision can be quantitatively evaluated, forming the basis of the optimization problem.

3.4. Problem Formulation

The optimization objective is to minimize a weighted combination of total system delay and total energy consumption, thereby balancing service responsiveness against system efficiency. The total system delay accumulates the completion time of all offloaded tasks and is defined as

$$\mathcal{D} = \sum_{u \in \mathcal{U}} \sum_{s \in \{0\} \cup \mathcal{S}} x_{u,s} T_{u,s}^{\text{all}}. \quad (14)$$

The total system energy consumption includes both local computation energy and transmission energy for offloaded tasks, expressed as

$$\mathcal{E} = \sum_{u \in \mathcal{U}} \left(\left(1 - \sum_{s \in \mathcal{S}} x_{u,s} \right) E_u^{\text{loc}} + \sum_{s \in \{0\} \cup \mathcal{S}} x_{u,s} E_{u,s}^{\text{all}} \right). \quad (15)$$

The weighted-sum objective function is then formulated as

$$\min_{x,f} \omega_T \mathcal{D} + \omega_E \mathcal{E}, \quad (16)$$

where ω_T and ω_E are non-negative weight coefficients satisfying $\omega_T + \omega_E = 1$. These weights reflect the service-level preference of the smart library: a larger ω_T prioritizes low-latency services such as real-time authentication, while a larger ω_E favors energy-conserving operations for battery-powered devices.

The optimization problem is subject to several constraints that capture the practical limitations of the library edge computing environment. First, the end-to-end delay of each task must not exceed its maximum tolerable latency to guarantee quality of service:

$$\sum_{s \in \{0\} \cup \mathcal{S}} x_{u,s} T_{u,s}^{\text{all}} \leq T_u^{\text{max}}, \quad \forall u \in \mathcal{U}. \quad (17)$$

Second, the total computing resources assigned to tasks on each edge server cannot surpass the server's finite capacity:

$$\sum_{u \in \mathcal{U}} x_{u,s} f_{u,s}^{\text{mec}} \leq F_s^{\text{max}}, \quad \forall s \in \mathcal{S}, \quad (18)$$

where F_s^{max} is the maximum computational capacity of edge server s . Third, offloading decisions are binary: $x_{u,s} \in \{0, 1\}$ for every user (u) and every computing node (s), including the cloud. Fourth, each task must be processed exactly once, which is enforced by the per-task uniqueness constraint:

$$\sum_{s \in \{0\} \cup \mathcal{S}} x_{u,s} = 1, \quad \forall u \in \mathcal{U}. \quad (19)$$

Fifth, offloading to an edge server is feasible only if the user is within the server's coverage radius. This geographic constraint is expressed as

$$x_{u,s} \cdot d_{u,s} \leq x_{u,s} \cdot R_s, \quad \forall u \in \mathcal{U}, \forall s \in \mathcal{S}, \quad (20)$$

where R_s is the coverage radius of server s . Finally, the allocated computing resources are non-negative:

$$f_{u,s}^{\text{mec}} \geq 0, \quad \forall u \in \mathcal{U}, \forall s \in \mathcal{S}. \quad (21)$$

The resulting problem is a mixed-integer nonlinear programming formulation, which is known to be NP-hard. In the following section, we design an efficient optimization algorithm to solve it.

4. Multi-Objective Reinforcement Learning-Based Task-Offloading Algorithm

This section elaborates on the design of the proposed multi-objective reinforcement learning-based task-offloading algorithm for MEC-enabled smart library systems. We first introduce the basic concepts of multi-objective optimization and the theoretical framework of multi-objective Markov decision processes [21]. Based on this foundation, the constrained task-offloading problem is reformulated as a sequential decision-making problem. Subsequently, a preference-adaptive multi-objective offloading algorithm is proposed, which employs a dueling double deep Q-network architecture integrated with a multi-head attention mechanism. This design enables an adaptive trade-off between service delay and energy consumption under diverse user preferences that commonly arise in library service scenarios.

4.1. Multi-Objective Optimization Preliminaries

In a smart library environment, task-offloading decisions must simultaneously optimize two conflicting performance metrics: the total task-execution delay and the overall energy consumption of user devices. A general multi-objective optimization problem seeks to minimize multiple objective functions over a feasible decision set. Without loss of generality, it can be written as

$$\underset{x \in \mathcal{X}}{\text{minimize}} \quad \mathbf{F}(\mathbf{x}) = (F_1(\mathbf{x}), F_2(\mathbf{x}), \dots, F_K(\mathbf{x})), \quad (22)$$

where $\mathbf{x}$ denotes the decision variables, $\mathcal{X}$ is the feasible region, and $F_k(\cdot)$ represents the k -th objective. In this work, we focus on the two conflicting objectives described above, i.e., $K = 2$.

Because these objectives inherently compete with each other, there is no single decision that minimizes both simultaneously. Therefore, the concept of Pareto optimality is adopted [22]. A decision ($\mathbf{x}^* \in \mathcal{X}$) is Pareto-optimal if no other feasible $\mathbf{x} \in \mathcal{X}$ exists such that $F_k(\mathbf{x}) \leq F_k(\mathbf{x}^*)$ for all k and $F_j(\mathbf{x}) < F_j(\mathbf{x}^*)$ for at least one index (j). The set of all such non-dominated solutions constitutes the Pareto set, and its projection onto the objective space is called the Pareto frontier. In the context of library task offloading, the Pareto frontier reveals the possible trade-off levels between fast service response and battery-life preservation.

A widely used technique for solving MOO problems is scalarization, which transforms the vector objective into a single scalar by aggregating the criteria. The linear weighting method is adopted here:

$$\underset{x \in \mathcal{X}}{\text{minimize}} \quad V^{\text{lin}}(x | \lambda) = \sum_{k=1}^2 \lambda_k F_k(x), \quad (23)$$

where $\lambda = (\lambda_1, \lambda_2)$ is a preference vector satisfying $\lambda_k \geq 0$ and $\lambda_1 + \lambda_2 = 1$. By varying λ , one can obtain a series of trade-off solutions that approximate the Pareto frontier. For instance, a library system administrator may assign a higher λ_1 to emphasize low latency for real-time services, whereas a higher λ_2 may be chosen to conserve energy during off-peak hours.

To quantitatively assess the quality of an approximated Pareto frontier, two complementary metrics are employed: the hypervolume indicator and the distribution sparsity.

A reference point ($r = (r_1, r_2)$) is selected such that every point on the Pareto frontier strictly dominates it; typical choices are the worst observed objective values. For a finite set of non-dominated solutions ($\mathcal{P} = \{p_1, \dots, p_N\}$), the hypervolume is defined as the two-dimensional Lebesgue measure of the union of axis-aligned boxes spanning from each solution (p_i) to r [23]:

$$I_H(\mathcal{P}) = \Lambda\left(\bigcup_{i=1}^N [p_i, r]\right), \quad (24)$$

where $[p_i, r] = \{z \in \mathbb{R}^2 \mid p_{i,1} \leq z_1 \leq r_1, p_{i,2} \leq z_2 \leq r_2\}$ and $\Lambda(\cdot)$ denotes the Lebesgue measure. A larger I_H indicates a better approximation in terms of both closeness to the true frontier and solution diversity.

The sparsity metric evaluates how evenly the solutions are distributed along the Pareto frontier. After sorting the members of $\mathcal{P}$ in non-decreasing order according to each objective (k), the sparsity is computed as

$$\Delta(\mathcal{P}) = \frac{1}{2(N-1)} \sum_{k=1}^2 \sum_{i=1}^{N-1} (q_k^{(i+1)} - q_k^{(i)})^2, \quad (25)$$

where $q_k^{(1)} \leq q_k^{(2)} \leq \dots \leq q_k^{(N)}$ are the sorted values of the k -th objective in $\mathcal{P}$. A smaller Δ corresponds to a more uniform distribution, which is highly desirable when a diverse set of trade-off alternatives must be presented to library service managers.

4.2. Formulation of the Multi-Objective Markov Decision Process

The sequential decision-making process for task offloading in an MEC-enabled smart library is formulated as a multi-objective Markov decision process. Formally, such a process is defined by a six-element tuple— $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \rho, \mathbf{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the state-transition probability kernel, $\gamma \in (0, 1)$ is the discount factor, ρ is the initial state distribution, and $\mathbf{R}$ is the vector-valued reward function.

The state ($s_t \in \mathcal{S}$) at discrete time step t captures all time-varying information needed for offloading decisions in a library environment. It is defined as

$$s_t = (\mathcal{T}_t, \mathbf{g}_t, \mathbf{z}_t, \mathbf{c}_t^{\text{rem}}), \quad (26)$$

where $\mathcal{T}_t$ contains the attributes of the currently pending task, including the size of its input data, its required CPU cycles, its maximum tolerable delay, and its service category (lightweight real-time, medium-weight interactive, or heavyweight offline, as defined in the task model); $\mathbf{g}_t$ is the vector of channel power gains from the user to each edge server

and to the cloud; $\mathbf{z}_t$ indicates the current library zone of the user, which determines which edge servers are within coverage; and $\mathbf{c}_t^{\text{rem}}$ denotes the residual computational capacities of all edge servers. This representation allows the agent to perceive both wireless dynamics and zone-dependent resource availability.

The action space corresponds to feasible execution locations for the current task. Given the three-tier architecture and the zone-coverage constraint, the action set is defined as $\mathcal{A} = \{0, 1, 2\}$, where 0 denotes local execution on the user device, 1 denotes offloading to an edge server that covers the user's current zone, and 2 denotes offloading to the remote cloud. Note that action 1 is available only when at least one edge server covers the user's zone; otherwise, only actions 0 and 2 are permissible.

The state-transition probability kernel ($\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$) specifies the probability of moving from s_t to s_{t+1} after taking action a_t . This kernel captures the inherent randomness of the library edge computing environment, including time-varying indoor wireless-channel quality, non-homogeneous task arrivals (with peak and off-peak periods), and fluctuations in server resource availability due to concurrent workloads from multiple users in the same zone.

The discount factor ($\gamma \in (0, 1)$) determines the relative importance of immediate versus future rewards. The initial state distribution (ρ) reflects the random initial conditions at the start of an episode, such as the initial queue lengths at each server and the initial positions (zones) of users.

Unlike classical MDPs that use scalar rewards, the MOMDP framework employs a vector reward to handle the two conflicting objectives. In this work, the immediate reward vector after executing action a_t in state s_t is defined as

$$\mathbf{R}(s_t, a_t) = (R_{\text{delay}}(s_t, a_t), R_{\text{energy}}(s_t, a_t)), \quad (27)$$

with the two components given by

$$R_{\text{delay}}(s_t, a_t) = -\Delta T(s_t, a_t), \quad R_{\text{energy}}(s_t, a_t) = -\Delta E(s_t, a_t), \quad (28)$$

where $\Delta T(s_t, a_t)$ represents the total time incurred by the decision. This includes uplink transmission delay; processing delay on the chosen server, including queue waiting time; and, for cloud offloading, an additional fixed round-trip wide-area network delay. The term $\Delta E(s_t, a_t)$ denotes the corresponding energy consumption, which primarily arises from uplink transmission when offloading or from local CPU execution when the task runs on the device. The negative sign ensures that maximizing cumulative reward is equivalent to minimizing total system delay and energy consumption, aligning with the original optimization objective.

To apply standard single-objective reinforcement learning algorithms, the vector reward must be scalarized [24]. A linear weighted-sum approach is adopted: for a given preference vector ($\boldsymbol{\eta} = (\eta_d, \eta_e)$) with $\eta_d, \eta_e \geq 0$ and $\eta_d + \eta_e = 1$, the scalarized immediate reward is

$$r_{\text{scalar}}(s_t, a_t) = \eta_d R_{\text{delay}}(s_t, a_t) + \eta_e R_{\text{energy}}(s_t, a_t). \quad (29)$$

Different preference vectors correspond to different service policies in the library. For example, during busy hours, the administrator may assign a larger η_d to emphasize low latency for real-time tasks, whereas during off-peak hours, a larger η_e may be used to prolong the battery life of mobile reading devices.

By varying the preference vector, a set of trade-off solutions approximating the Pareto front can be obtained. Then, the cumulative discounted return over an episode of length $T_{\max}$ is

$$G = \sum_{t=0}^{T_{\max}} \gamma^t r_{\text{scalar}}(s_t, a_t). \quad (30)$$

The agent's objective is to learn a policy ($\pi : \mathcal{S} \rightarrow \mathcal{A}$) that maximizes G for a given preference or, ideally, a single policy that can adapt to arbitrary preferences without retraining.

4.3. Preference-Adaptive Dueling Deep Q-Network for Library-Oriented Offloading

To address the multi-objective task-offloading challenge in a smart library environment under dynamic user preferences, we propose a preference-adaptive dueling deep Q-network algorithm called PA-DDQN. The proposed architecture integrates three key components: a preference-conditioning mechanism that encodes the trade-off between service delay and device energy consumption, a multi-head attention module to capture the dynamics of the indoor library environment, and a dueling double Q-network structure for stable value estimation. This design enables a single trained model to produce optimal offloading decisions for arbitrary preference vectors without retraining.

The overall neural network consists of four sequentially connected functional parts: a state-embedding block, a preference integration unit, an attention-based feature enhancement layer, and a multi-objective dueling output head. The raw state (s_t) contains heterogeneous information, including the attributes of the currently pending task, such as the size of its input data, its required CPU cycles, its maximum tolerable delay, its and service category; the channel gains to each edge server and the cloud; the residual computing capacities of all servers; the queue lengths at each server; and the current library zone of the user. A fully connected layer with layer normalization first projects this high-dimensional state vector into a compact hidden representation ($\mathbf{h}_t^{\text{state}}$). Simultaneously, the user-preference vector ($\boldsymbol{\pi} = (\pi_{\text{lat}}, \pi_{\text{eng}})$), which satisfies $\pi_{\text{lat}} \geq 0$, $\pi_{\text{eng}} \geq 0$, and $\pi_{\text{lat}} + \pi_{\text{eng}} = 1$, is passed through a separate linear layer to produce a preference embedding ($\mathbf{h}_t^{\text{pref}}$). The two embeddings are then concatenated to form the joint feature ($\mathbf{f}_t = [\mathbf{h}_t^{\text{state}}, \mathbf{h}_t^{\text{pref}}]$), allowing the network to condition its decisions on the current trade-off emphasis.

To capture the complex correlations among indoor wireless-channel conditions, edge-server loads, and task arrival patterns in a library environment, a multi-head attention mechanism is introduced to refine the joint feature representation. The attention layer maps the input ($\mathbf{f}_t$) into query, key, and value matrices via trainable projections:

$$\mathbf{Q} = \mathbf{W}_Q \mathbf{f}_t, \quad \mathbf{K} = \mathbf{W}_K \mathbf{f}_t, \quad \mathbf{V} = \mathbf{W}_V \mathbf{f}_t, \quad (31)$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are parameter matrices. The scaled dot-product attention is computed as

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V}, \quad (32)$$

with d_k denoting the dimension of the query vectors. To capture different types of correlations in parallel, H independent attention heads are employed, and their outputs are concatenated and linearly transformed:

$$\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1 \parallel \text{head}_2 \parallel \cdots \parallel \text{head}_H] \mathbf{W}_O, \quad (33)$$

where $\text{head}_i = \text{Attn}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i)$ and $\mathbf{W}_O$ is an output projection matrix. The attention output is then passed through a residual connection, followed by layer normalization to stabilize training.

The final layer adopts the dueling architecture, which separately estimates the state-value function and the action advantage function. For a given state (s_t), preference π , and action a_t , the Q-value is computed as

$$Q(s_t, a_t, \pi) = V(s_t, \pi) + A(s_t, a_t, \pi) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} A(s_t, a', \pi), \quad (34)$$

where $V(s_t, \pi)$ denotes the state value independent of the action and $A(s_t, a_t, \pi)$ represents the advantage of taking action a_t over the average action. This decomposition reduces estimation variance and improves learning stability. Double Q-learning is further incorporated by using the target network to evaluate the next-state actions, thereby mitigating the overestimation bias commonly seen in standard DQNs. The overall network is trained end to end by minimizing the temporal-difference error over transitions sampled from a diverse experience replay buffer. To enforce preference-aware generalization, each transition is trained under two different preference weights, forcing the network to recognize that distinct trade-off emphases yield different Q-values for the same state. This design is particularly suitable for library systems, where user priorities may shift between rapid service delivery and energy conservation depending on the time of day or device type.

5. Simulation and Performance Evaluation

5.1. Simulation Setup

To evaluate the effectiveness of the proposed PA-DDQN algorithm, simulation experiments are conducted in a smart library-oriented mobile edge computing scenario based on the system model described in the previous sections. The simulated system consists of user devices, indoor MEC servers, and a remote cloud server. User devices represent typical smart library terminals, such as self-service borrowing devices, RFID readers, handheld query terminals, mobile reading devices, and face recognition terminals. MEC servers are deployed in different functional zones of the library, including the entrance hall, main reading area, multimedia area, book-stack area, self-study area, and children's section. The remote cloud server provides additional computing resources for computation-intensive or delay-tolerant tasks.

The generated tasks are divided into three categories according to their computational workload and delay requirements. Lightweight real-time tasks include identity authentication, access-control verification, and book-status querying. Medium-weight interactive tasks include multimedia retrieval, intelligent recommendation, and reader behavior analysis. Heavyweight offline tasks include indoor book localization, batch RFID inventory, and large-scale data synchronization. Each task is represented by the size of its input data, its required CPU cycles, the size of its output data, and its maximum tolerable delay. The simulation parameters are summarized in Table 1.

At each time slot, a user device generates a task according to the task arrival model. The task can be executed locally, offloaded to a nearby MEC server, or transmitted to the remote cloud server. The offloading decision is made according to the current task attributes, wireless channel condition, server load, and service preference.

5.2. Compared Algorithms

To verify the performance of the proposed PA-DDQN algorithm, several representative baseline algorithms are selected for comparison, including fixed offloading strategies, a

random decision strategy, and reinforcement learning-based strategies. The compared algorithms and their descriptions are listed in Table 2.

Table 1. Simulation parameters.

Parameter	Value
Number of library zones	6
Number of MEC servers	6
Number of user devices	30
Task-load range	100–1000 tasks
Wireless bandwidth	20 MHz
Transmission power	0.5 W
Edge-server coverage radius	30 m
Episode length	100 time slots
Discount factor (γ)	0.99
Learning rate	1×10^{-4}
Training episodes	1000

Table 2. Description of Compared Algorithms.

Algorithm	Description
Random [25]	Randomly selects a feasible execution node for each task.
Local-only [26]	Executes all tasks locally on user devices.
Edge-only [27]	Offloads each task to the nearest available MEC server.
Cloud-only [5]	Offloads all tasks to the remote cloud server.
DDQN [28]	Uses double Q-learning to reduce Q-value overestimation.
D3QN [29]	Uses a dueling double deep Q-network for offloading decisions.
PA-DDQN	The proposed preference-adaptive dueling double DQN algorithm.

The Random algorithm represents a policy without learning ability. Local-only, Edge-only, and Cloud-only are fixed strategies used to evaluate the influence of different execution locations. DDQN and D3QN are learning-based baselines. Compared with these methods, PA-DDQN introduces preference adaptation so that the same model can adjust its offloading decisions according to different delay–energy trade-off requirements.

5.3. Evaluation Metrics

The performance of different algorithms is evaluated from the perspectives of task delay, energy consumption, service reliability, learning performance, and multi-objective trade-off ability.

5.3.1. Average Task-Completion Delay

Average task-completion delay measures the average time required for all tasks to be completed:

$$\bar{T} = \frac{1}{N} \sum_{i=1}^N T_i, \quad (35)$$

where N is the total number of tasks and T_i is the completion delay of task i .

5.3.2. Average Energy Consumption

Average energy consumption measures the average energy cost of local computation or wireless transmission:

$$\bar{E} = \frac{1}{N} \sum_{i=1}^N E_i, \quad (36)$$

where E_i denotes the energy consumption of task i .

5.3.3. Task Success Rate

The task success rate is defined as the proportion of tasks completed before their maximum tolerable delay:

$$R_s = \frac{N_s}{N}, \quad (37)$$

where N_s is the number of successfully completed tasks.

5.3.4. Average Reward

Average reward is used to evaluate the convergence behavior of reinforcement learning algorithms. In this paper, the scalar reward is defined consistently with the multi-objective formulation in Section 4.2, where task-completion delay and energy consumption are jointly considered. The scalar reward at time slot t is given by

$$r_t = \eta_d R_{\text{delay}}(s_t, a_t) + \eta_e R_{\text{energy}}(s_t, a_t), \quad (38)$$

where $R_{\text{delay}}(s_t, a_t)$ denotes the delay-related reward and $R_{\text{energy}}(s_t, a_t)$ denotes the energy-related reward. η_d and η_e are the weighting coefficients of delay and energy consumption, respectively, satisfying $\eta_d + \eta_e = 1$.

The task success rate is not directly included in the reward function but is used as an independent evaluation metric to measure whether tasks can be completed before their maximum tolerable delay. Therefore, the average reward reflects the learned delay–energy trade-off, while the task success rate evaluates the reliability of the obtained offloading policy.

5.4. Analysis of Experimental Results

5.4.1. Training Performance Analysis

Figure 2 shows the training curves of the proposed PA-DDQN algorithm in terms of average energy consumption, average task-completion delay, task success rate, and average reward. It can be observed that the proposed method exhibits a clear convergence trend during training.

At the early stage of training, the average energy consumption is around 450, the average task-completion delay is about 8.3, the task success rate is only about 0.58, and the average reward is approximately 0.15. This indicates that the agent has not yet learned an effective offloading policy and still makes a large number of suboptimal decisions.

As the number of training episodes increases, the agent gradually learns the relationships among task characteristics, wireless conditions, and computing resource states. Therefore, the average energy consumption and delay continuously decrease, while the task success rate and reward steadily increase. After about 300–400 training episodes, the performance curves begin to stabilize, indicating that the proposed algorithm has learned a relatively stable policy. At the end of training, the average energy consumption decreases to about 260, the average task-completion delay falls to about 5.6, the task success rate rises to about 0.91, and the average reward increases to approximately 0.54.

These results demonstrate that PA-DDQN has a good convergence ability and can effectively optimize task-offloading decisions in the smart library MEC environment.

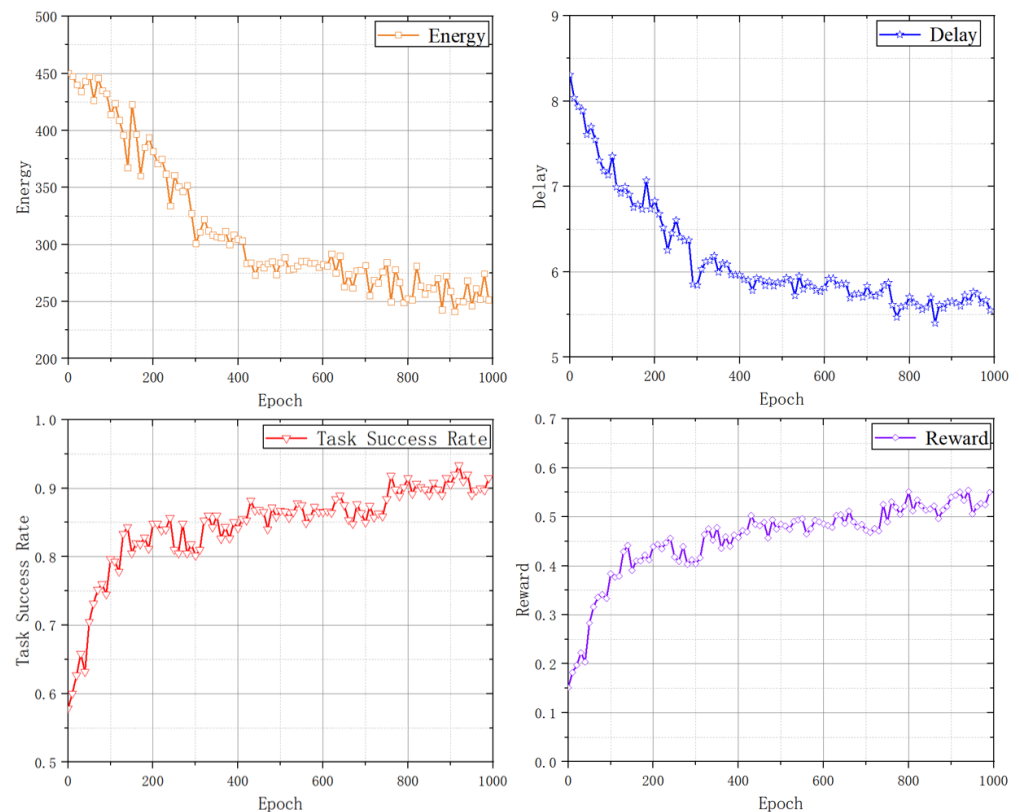


Figure 2. Training performance of PA-DDQN in terms of energy consumption, task-completion delay, task success rate, and average reward.

5.4.2. Delay Performance Analysis

Figure 3 compares the average task-completion delay of different algorithms under increasing task loads. As the number of tasks increases from 100 to 1000, the average delay of all algorithms increases because more computation and communication resources are occupied under heavier system load.

Among all compared methods, PA-DDQN consistently achieves the lowest delay. When the task number reaches 1000, the average delays of Random, Local-only, Edge-only, Cloud-only, DDQN, D3QN, and PA-DDQN are approximately 13.3, 14.2, 12.9, 12.3, 11.6, 10.4, and 8.0, respectively. Compared with D3QN and DDQN, the proposed PA-DDQN reduces the average delay by about 23.1% and 31.0%, respectively.

The Local-only strategy shows the worst delay performance under high task load because the computing capability of user devices is limited and cannot efficiently handle a large number of tasks. Cloud-only also suffers from relatively high delay due to the additional wide-area transmission and round-trip latency. Although Edge-only performs better than Local-only and Cloud-only, it still lacks adaptive decision-making ability and cannot flexibly balance server load and communication cost. In contrast, PA-DDQN can jointly consider task attributes, server states, and network conditions, thereby selecting more suitable execution nodes and achieving the best delay performance.

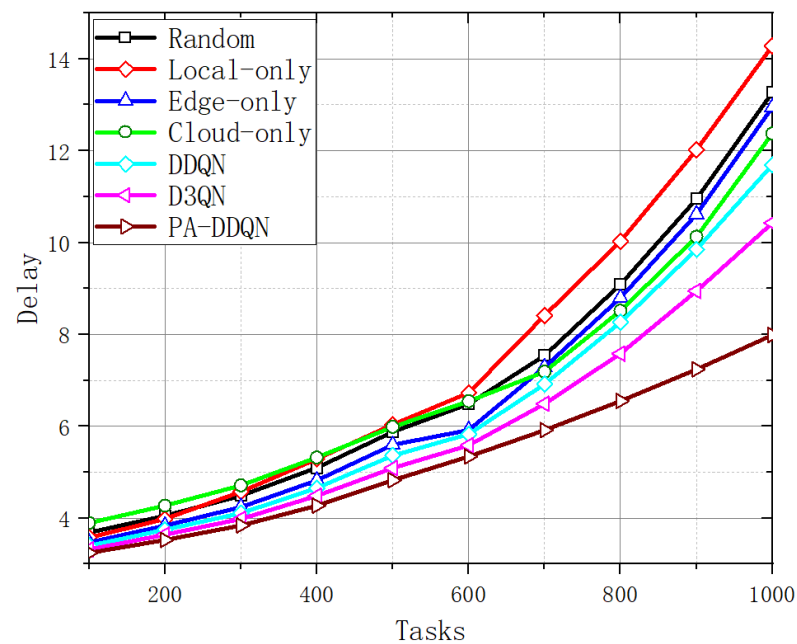


Figure 3. Average task-completion delay under different numbers of tasks.

5.4.3. Energy Consumption Analysis

Figure 4 illustrates the average energy consumption of different algorithms under different task loads. Similar to the delay results, the energy consumption of all algorithms increases as the number of tasks grows. This is because more tasks lead to more local computation or wireless transmission operations.

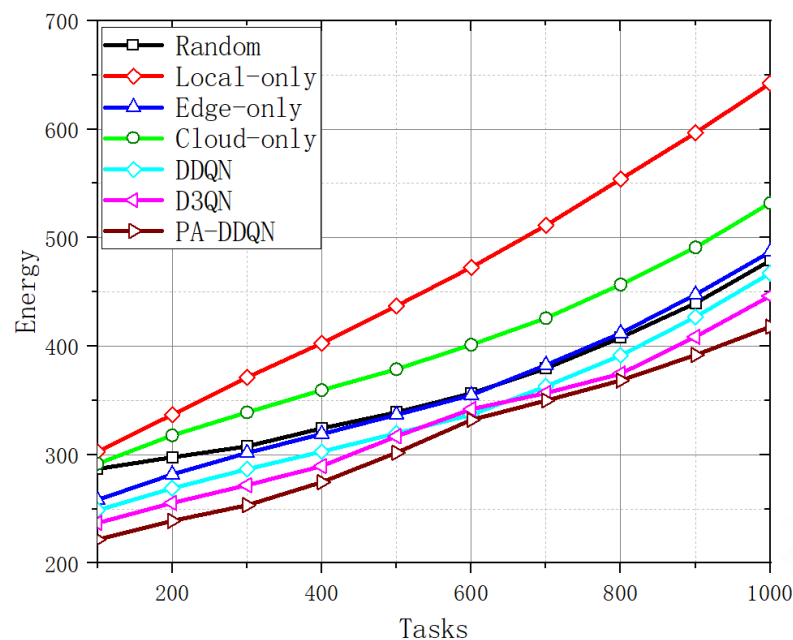


Figure 4. Average energy consumption under different numbers of tasks.

The Local-only strategy has the highest energy consumption in almost all cases. When the number of tasks reaches 1000, its average energy consumption is about 640, which is significantly larger than that of other algorithms. This is mainly because local devices need to complete all computation tasks by themselves, resulting in a heavy energy burden. Cloud-only also exhibits relatively high energy consumption because remote transmission

introduces additional communication overhead. Edge-only performs better than Local-only and Cloud-only, but it still cannot fully exploit the system resources.

The learning-based methods—DDQN, D3QN, and PA-DDQN—achieve lower energy consumption than fixed strategies. In particular, PA-DDQN obtains the best result. At 1000 tasks, the average energy consumptions of DDQN, D3QN, and PA-DDQN are approximately 466, 446, and 420, respectively. Therefore, PA-DDQN reduces energy consumption by about 5.8% compared with D3QN and by about 9.9% compared with DDQN. This shows that the proposed method can effectively reduce unnecessary transmission and computation overhead through more reasonable offloading decisions.

5.4.4. Task Success-Rate Analysis

Figure 5 presents the task success rates of different algorithms as the task number increases. It can be observed that the success rate of all algorithms decreases gradually with increasing task load. This is because more tasks intensify resource contention and make it more difficult to complete all tasks before their delay deadlines.

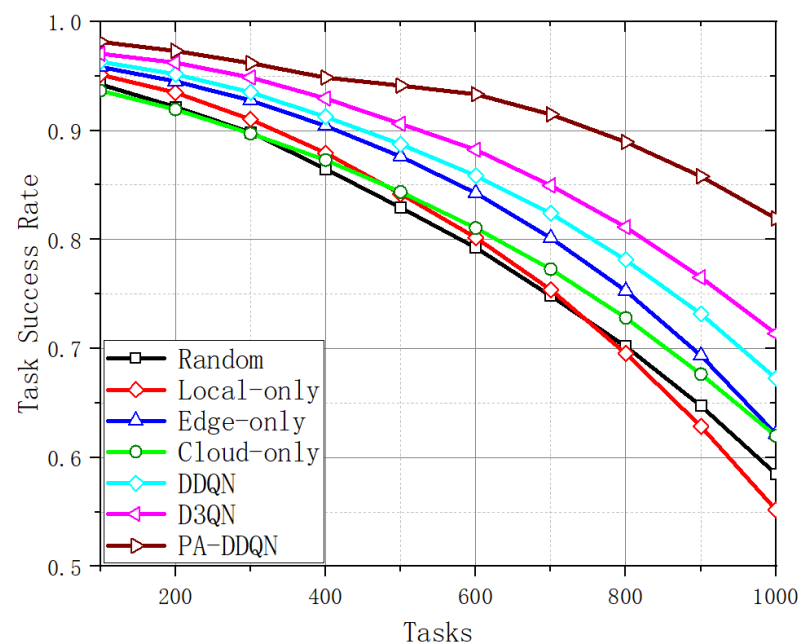


Figure 5. Task success rate under different numbers of tasks.

However, PA-DDQN maintains the highest task success rate across all task loads and shows the slowest degradation trend. When the number of tasks increases from 100 to 1000, the task success rate of PA-DDQN decreases from about 0.98 to about 0.82, which is still significantly higher than the other methods. At 1000 tasks, the task success rates of Random, Local-only, Edge-only, Cloud-only, DDQN, D3QN, and PA-DDQN are approximately 0.58, 0.56, 0.63, 0.62, 0.67, 0.71, and 0.82, respectively.

Compared with D3QN and DDQN, PA-DDQN improves the task success rate by about 14.8% and 21.7%, respectively, under the heaviest task load. This result indicates that the proposed method can better satisfy task delay constraints and improve the service reliability of the smart library system.

5.5. Ablation Study

This experiment is designed to evaluate the contribution of each key component in PA-DDQN. Since the proposed algorithm introduces preference adaptation, a dueling network structure, double Q-learning, and attention-based feature enhancement, it is necessary to analyze whether these components effectively improve task-offloading performance. The

ablation study settings are summarized in Table 3. The following algorithm variants are constructed for comparison.

Table 3. Ablation study settings.

Variant	Description
PA-DDQN	Complete proposed model.
PA-DDQN w/o Preference	Removes preference conditioning.
PA-DDQN w/o Dueling	Replaces the dueling head with a standard Q-value head.
PA-DDQN w/o Attention	Removes the attention module.
DDQN	Removes preference conditioning, the dueling architecture, and the attention module.

Table 4 lists the ablation results. All variants are trained under the same simulation environment, task arrival process, training settings, and evaluation metrics. Their performance is compared in terms of average task-completion delay, average energy consumption, task success rate, and average reward. Through this design, the effectiveness of each module in PA-DDQN can be independently analyzed.

Table 4. Results of the ablation study.

Variant	Delay	Energy	Task Success Rate	Reward
PA-DDQN	5.28	315.5	0.924	0.538
PA-DDQN w/o Preference	5.66	324.9	0.906	0.509
PA-DDQN w/o Dueling	5.84	331.8	0.894	0.494
PA-DDQN w/o Attention	5.97	336.4	0.886	0.483
DDQN	6.39	340.7	0.852	0.468

It can be observed that the complete PA-DDQN achieves the best performance among all variants. Specifically, it obtains the lowest average delay and energy consumption while simultaneously achieving the highest task success rate and average reward. Compared with DDQN, PA-DDQN reduces the average task-completion delay by about 17.4%, reduces average energy consumption by about 7.4%, improves the task success rate by about 8.5%, and increases the average reward by about 15.0%.

After removing the preference input vector, the performance drops for all metrics. This indicates that preference adaptation plays an important role in enabling the agent to adjust its decision strategy according to different delay–energy trade-off requirements. Without explicit preference guidance, the learned policy becomes less flexible and less effective in handling multi-objective optimization.

When the dueling network structure is removed, the performance also degrades noticeably. This is because the dueling architecture can separately estimate the state value and the action advantage, which helps the agent better distinguish the contributions of different offloading actions under the same environmental state. Therefore, the dueling mechanism improves the accuracy and stability of Q-value estimation.

Similarly, removing the attention module leads to an obvious decline in performance. This result suggests that the attention-based feature enhancement mechanism is effective in extracting important state information, such as task urgency, channel quality, and server resource state. By focusing on more critical features, the proposed model can make more accurate offloading decisions.

Overall, the ablation study verifies that preference adaptation, the dueling architecture, and the attention mechanism all contribute positively to the performance improvement of

PA-DDQN. The best results are achieved only when all components are jointly incorporated into the final model.

5.6. Summary

This section evaluates the proposed PA-DDQN algorithm in a smart library-oriented MEC environment. The simulation considers heterogeneous smart library devices, multiple indoor service zones, MEC servers, and a remote cloud server. Several fixed strategies and reinforcement learning-based methods are selected as baselines for comparison.

The experimental results show that PA-DDQN achieves stable convergence during training. Compared with the benchmark methods, the proposed method consistently obtains lower task-completion delay and lower energy consumption under different task loads. At the same time, PA-DDQN maintains a higher task success rate, especially in high-load scenarios where communication and computing resources become more limited.

The delay analysis shows that PA-DDQN can effectively reduce the task completion time by selecting appropriate execution nodes according to task characteristics, wireless channel conditions, and server resource states. The energy consumption analysis further demonstrates that the proposed method can reduce unnecessary local computation and wireless transmission overhead. In addition, the task success-rate results indicate that PA-DDQN can better satisfy the delay constraints of smart library tasks and improve service reliability.

The ablation study verifies the effectiveness of the main components in the proposed algorithm. The results show that the dueling network structure, double Q-learning mechanism, and attention-based feature enhancement all contribute to the improvement of the overall offloading performance.

Overall, PA-DDQN can dynamically optimize task-offloading decisions according to system states and task requirements. Therefore, it effectively improves the service efficiency and reliability of smart library MEC systems.

6. Conclusions and Future Work

This paper investigated the task-offloading optimization problem in smart library service scenarios. A cloud–edge–device collaborative computing architecture was constructed, and heterogeneous service tasks in smart libraries were modeled and optimized by jointly considering task-completion delay, user-side energy consumption, edge resource capacity, and service coverage. Simulation results show that the proposed task-offloading scheme can effectively reduce task-completion delay and energy consumption while improving the task success rate under different task loads. Compared with several benchmark algorithms, the proposed scheme can still maintain good service stability and reliability in high-load scenarios, indicating that it can effectively improve service responsiveness, energy efficiency, and resource utilization in MEC-enabled smart library systems.

Future work will focus on extending the proposed model to more practical smart library scenarios. In particular, future studies may consider task dependency and task partitioning for complex services and further validate the proposed framework in real MEC-enabled library environments. In addition, privacy and security issues should be further investigated to support safe and reliable intelligent library services.

Author Contributions: Conceptualization, J.Q. and P.Z.; methodology, J.Q., P.Z. and R.W.; software, J.Q.; validation, R.W., X.Z. and L.C.; formal analysis, R.W. and L.C.; investigation, J.Q. and L.C.; resources, J.Q.; data curation, R.W.; writing—original draft preparation, J.Q., R.W., X.Z. and L.C.; writing—review and editing, J.Q. and P.Z.; visualization, R.W.; supervision, P.Z.; project administration, J.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Natural Science Foundation of Shandong Province under Grants ZR2023LZH017 and ZR2024MF066.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Stepanov, V.K. A User-Centered Model of a Science Library. *Sci. Tech. Inf. Process.* **2023**, *50*, 69–73. [\[CrossRef\]](#)
- Bi, S.; Wang, C.; Zhang, J.; Huang, W.; Wu, B.; Gong, Y.; Ni, W. A Survey on Artificial Intelligence Aided Internet-of-Things Technologies in Emerging Smart Libraries. *Sensors* **2022**, *22*, 2991. [\[CrossRef\]](#) [\[PubMed\]](#)
- Su, X.; Chen, N. Intelligent Information Service System of Smart Library Based on Virtual Reality and Eye Movement Technology. *Sci. Program.* **2022**, *2022*, 917476. [\[CrossRef\]](#)
- Xu, H.; Liu, W.D.; Li, L. An IoT-based low-cost architecture for smart libraries using SDN. *Sci. Rep.* **2024**, *14*, 7022. [\[CrossRef\]](#) [\[PubMed\]](#)
- Mammadov, S.; Kucukkulahli, E. A User-Centric Smart Library System: IoT-Driven Environmental Monitoring and ML-Based Optimization with Future Fog–Cloud Architecture. *Appl. Sci.* **2025**, *15*, 3792. [\[CrossRef\]](#)
- Younis, A.; Maheshwari, S.; Pompili, D. Energy-Latency Computation Offloading and Approximate Computing in Mobile-Edge Computing Networks. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 3401–3415. [\[CrossRef\]](#)
- Zhou, J.; Hou, X.; Zeng, Y.; Cong, P.; Jiang, W.; Guo, S. Quality of Experience and Reliability-Aware Task Offloading and Scheduling for Multi-User Mobile-Edge Computing Systems. *IEEE Trans. Serv. Comput.* **2025**, *18*, 1683–1696. [\[CrossRef\]](#)
- Bai, Y. Construction of a Smart Library Subject Precise Service Platform Based on User Needs. *Math. Probl. Eng.* **2022**, *2022*, 5675291. [\[CrossRef\]](#)
- Zhou, Q. Smart library architecture based on internet of things (IoT) and software defined networking (SDN). *Heliyon* **2024**, *10*, e25375. [\[CrossRef\]](#) [\[PubMed\]](#)
- Tang, C.; Ding, Y.; Xiao, S.; Huang, Z.; Wu, H. Collaborative Service Caching, Task Offloading, and Resource Allocation in Caching-Assisted Mobile Edge Computing. *IEEE Trans. Serv. Comput.* **2025**, *18*, 1966–1981. [\[CrossRef\]](#)
- Wang, D.; Zhu, H.; Qiu, C.; Zhou, Y.; Lu, J. Distributed Task Offloading in Cooperative Mobile Edge Computing Networks. *IEEE Trans. Veh. Technol.* **2024**, *73*, 10487–10501. [\[CrossRef\]](#)
- Chai, F.; Zhang, Q.; Yao, H.; Xin, X.; Gao, R.; Guizani, M. Joint Multi-Task Offloading and Resource Allocation for Mobile Edge Computing Systems in Satellite IoT. *IEEE Trans. Veh. Technol.* **2023**, *72*, 7783–7795. [\[CrossRef\]](#)
- Li, S.; Li, W.; Zheng, W. Multi-user joint task offloading and resource allocation based on mobile edge computing in mining scenarios. *Sci. Rep.* **2025**, *15*, 16170. [\[CrossRef\]](#) [\[PubMed\]](#)
- Zhang, Y.; Zhu, T.; Fang, X.; Xu, T.; Liu, Y.; Luo, J. Task Offloading Scheduling for Mobile Edge Computing Networks with Incomplete Edge Information. *IEEE Internet Things J.* **2026**, *13*, 25354–25366. [\[CrossRef\]](#)
- Chen, X.; Bi, Y.; Zhang, J.; Yuan, X.; Niyato, D.; Zhao, L.; Wang, X. Task Offloading and Resource Optimization Based on Dependency-Aware Graph and Collaborative Deep Reinforcement Learning in Mobile Edge Computing. *IEEE Trans. Mob. Comput.* **2026**, *25*, 10998–11015. [\[CrossRef\]](#)
- Jiang, T.; Chen, Z.; Zhao, Z.; Feng, M.; Zhou, J. Deep-Reinforcement-Learning-Based Task Offloading and Resource Allocation in Mobile Edge Computing Network with Heterogeneous Tasks. *IEEE Internet Things J.* **2024**, *12*, 10899–10906. [\[CrossRef\]](#)
- Yang, W.; Liu, Z.; Liu, X.; Ma, Y. Deep Reinforcement Learning-Based Low-Latency Task Offloading for Mobile-Edge Computing Networks. *Appl. Soft Comput.* **2024**, *166*, 112164. [\[CrossRef\]](#)
- Zhao, L.H.; Zhang, J.; Bah, M.J. Energy-Efficient Task Offloading in Multi-server Mobile Edge Computing Networks: A Deep Reinforcement Learning Approach. *Arab. J. Sci. Eng.* **2025**, *50*, 16221–16242. [\[CrossRef\]](#)
- Sun, H.; Wang, Z.; Yu, Y.; Sha, K.; Wu, Y. EdgeBrain: A Game Based Collaborative Computational Task Offloading Framework for Edge Video Analytics. *IEEE Trans. Serv. Comput.* **2024**, *17*, 2287–2303. [\[CrossRef\]](#)
- Panda, S.K.; Lin, M.; Zhou, T. Energy-Efficient Computation Offloading with DVFS Using Deep Reinforcement Learning for Time-Critical IoT Applications in Edge Computing. *IEEE Internet Things J.* **2023**, *10*, 6611–6621. [\[CrossRef\]](#)
- Hayes, C.F.; Rădulescu, R.; Bargiacchi, E.; Källström, J.; Macfarlane, M.; Reymond, M.; Verstraeten, T.; Zintgraf, L.M.; Dazeley, R.; Heintz, F.; et al. A practical guide to multi-objective reinforcement learning and planning: CF Hayes et al. *Auton. Agents Multi Agent Syst.* **2022**, *36*, 26. [\[CrossRef\]](#)
- Liu, E.; Wu, Y.-C.; Huang, X.; Gao, C.; Wang, R.-J.; Xue, K.; Qian, C. Pareto set learning for multi-objective reinforcement learning. *Proc. AAAI Conf. Artif. Intell.* **2025**, *39*, 18789–18797. [\[CrossRef\]](#)

23. Guerreiro, A.P.; Fonseca, C.M.; Paquete, L. The hypervolume indicator: Computational problems and algorithms. *ACM Comput. Surv.* **2021**, *54*, 119. [[CrossRef](#)]
24. Peng, N.; Tian, M.; Fain, B. Multi-objective reinforcement learning with nonlinear preferences: Provable approximation for maximizing expected scalarized return. *arXiv* **2023**, arXiv:2311.02544. [[CrossRef](#)]
25. Qin, Y.; Chen, J.; Jin, L.; Yao, R.; Gong, Z. Task offloading optimization in mobile edge computing based on a deep reinforcement learning algorithm using density clustering and ensemble learning. *Sci. Rep.* **2025**, *15*, 211. [[CrossRef](#)] [[PubMed](#)]
26. Anand, J.; Karthikeyan, B. Adaptive and intelligent customized deep Q-network for energy-efficient task offloading in mobile edge computing environments. *Sci. Rep.* **2026**, *16*, 5456. [[CrossRef](#)] [[PubMed](#)]
27. Tang, X.; Tang, T.; Shen, Z.; Zheng, H.; Ding, W. Double deep Q-network-based dynamic offloading decision-making for mobile edge computing with regular hexagonal deployment structure of servers. *Appl. Soft Comput.* **2025**, *169*, 112594. [[CrossRef](#)]
28. Kumari, S.; Gupta, D. Task offloading based on deep reinforcement learning in edge computing network. *J. Supercomput.* **2025**, *81*, 1479. [[CrossRef](#)]
29. Mustafa, E.; Shuja, J.; Rehman, F.; Namoun, A.; Bilal, M.; Bilal, K. Deep Reinforcement Learning and SQP-driven task offloading decisions in vehicular edge computing networks. *Comput. Netw.* **2025**, *262*, 111180. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.