aerospace

MDPI

Article

Edge-Based Intelligent Task Management for Mobile Airfield Lighting Control

Li Jiang [1,2,3,4], Hong Wen [1,3,4,*], Wenjing Hou [1,3,4,*] and Fan Sun [1,3,4]

[1] School of Aeronautics and Astronautics, University of Electronic Science and Technology of China, Chengdu 611731, China; ziyue1985@std.uestc.edu.cn (L.J.); sunfan_uestc@163.com (F.S.)
[2] Research and Development Centre, The Second Research Institute of CAAC, Chengdu 641418, China
[3] Sichuan Intelligent IoT Communication Technology Engineering Research Center, University of Electronic Science and Technology of China, Chengdu 611731, China
[4] Aircraft Swarm Intelligent Sensing and Cooperative Control Key Laboratory of Sichuan Province, Chengdu 610000, China
* Correspondence: sunlike@uestc.edu.cn (H.W.); uestc_hwj@126.com (W.H.)

Abstract

Airfield lighting control (ALC) is critical for ensuring safe, efficient, and compliant airport operations, especially under low-visibility conditions. However, current centralized control architectures cannot adequately meet the real-time responsiveness, scalability, and reliability requirements of Advanced Surface Movement Guidance and Control Systems (A-SMGCS) Level IV. To overcome these limitations, this paper proposes a novel cloud–edge–end collaborative architecture for a mobile ALC scenario, in which we formulate a joint task computing and energy consumption optimization problem to maximize long-term system utility under latency, computation, and communication constraints. In this way, the mobile airfield lighting (MAL) system can also quickly adapt its optimal formation pattern based on the airport environment, lighting conditions, and the type of aircraft taking off or landing via efficient computation, thereby achieving the best navigational assistance effect. For solving such an optimization problem, a framework that combines K-medoids with the Improved Twin Delayed Deep Deterministic Policy Gradient (ITD3) is proposed to integrate the efficiency of clustering for rough allocation and the high-precision dynamic optimization capability of the improved TD3. The training depends on edge nodes and the cloud to achieve online performance. Finally, the extensive simulation proved that our novel algorithm is efficient.

Keywords: airfield lighting system; cloud–fog–edge–end collaboration; hyper-converged infrastructure; task offloading; resource allocation; online incentive mechanism; smart airport

Check for updates

Academic Editor: Miroslav Kelemen

Received: 10 March 2026
Revised: 29 April 2026
Accepted: 29 April 2026
Published: 1 May 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) license.

1. Introduction

Civil aviation is a strategically critical and foundational industry, with its operational efficiency serving as a key indicator of a nation's transportation system modernization and comprehensive support capabilities. In recent years, the continuous growth in air transportation demand, coupled with the increasing complexity of airport operations, including expanded infrastructure, higher flight density, and intricate surface movements, has imposed unprecedented demands on operational efficiency, safety assurance, and service quality [1,2]. Developing novel airport infrastructures capable of collaborative sensing, intelligent decision-making, and safety assurance in complex operational environments

has become a critical technological pathway for enhancing airport operational resilience and overall competitiveness [3].

Within this context, airfield lighting systems provide continuous and reliable guidance for aircraft during all phases of operation, including takeoff, approach, landing, and taxiing under low-visibility conditions. Their safety, reliability, and level of intelligence directly determine an airport's overall operational efficiency and all-weather support capability [4]. The International Civil Aviation Organization (ICAO) recognizes the reliability and availability of lighting systems as key determinants of all-weather operational capability [5]. Traditionally, conventional airfield lighting systems have predominantly employed Programmable Logic Controller (PLC)-based centralized architectures, organizing lighting circuits as basic control units and following typical center-controlled, terminal-executed paradigms. However, as lighting control becomes increasingly integrated with real-time aircraft movement status—exemplified by Advanced Surface Movement Guidance and Control Systems (A-SMGCS) Level IV scenarios requiring dynamic control of thousands of taxiway centerline lights to form continuous guidance paths—the limitations of centralized, circuit-level control paradigms have become evident [6]. Such applications demand high-frequency data acquisition that may reach tens of thousands of data points per second and millisecond-level control response. However, if airfield lighting systems still work on the "full data upload to center, centralized decision-making" model, their fast response cannot be realized due to bandwidth constraints, computational bottlenecks, and single points of failure [7].

Especially as the low-altitude economy continues to evolve, numerous aircraft need to conduct takeoff and landing operations in suboptimal lighting environments. Notably, innovative technologies applied to airfield lighting deployment in mobile scenarios have facilitated on-demand aircraft takeoffs and landings at any time and place. Mobile airfield lighting (MAL), a key smart low-altitude economy technology, offers flexible lighting for temporary airfields. It adapts to environments and aircraft types to ensure safe 24 h takeoffs and landings. The fundamental challenge lies in reconciling heterogeneous task requirements, real-time control response, low energy consuming requirements and analytics tasks that demand intensive computation, which are key research gaps. To bridge these gaps, edge computing has emerged as a promising paradigm to address these challenges by collocating computational and decision-making capabilities near data sources and controlled assets, thereby reducing backhaul dependency and improving response determinism [8,9].

However, isolated edge nodes remain limited by constrained computational capacity and coverage scope, particularly in expansive airport environments with a vast number of terminal devices. A more viable approach involves constructing cloud–edge–end collaborative computing systems: cloud platforms handle long-term analytics, global optimization, and model training, providing global intelligence; edge nodes manage regional data processing, real-time decision-making, and security protection, delivering proximate intelligence; terminal devices perform high-frequency sensing and deterministic actuation, achieving unified global–local coordination [10]. This hierarchical paradigm enables latency-sensitive processing at the edge, event-triggered reporting strategies that conserve bandwidth, and layered autonomous capabilities that maintain essential functionality under network impairments, thereby enhancing overall system resilience [11].

To address these advantages of the cloud–edge–end system, this study presents an innovative cloud–edge–end collaborative framework tailored for MAL applications. The main contributions are as follows. Firstly, a joint optimization problem that coordinates task computation and energy consumption, with the goal of maximizing long-term system utility while satisfying latency, computational, and communication constraints, is established. Secondly, by taking advantage of the cloud–edge–end system, a hybrid architecture

combining the K-medoids algorithm and the improved TD3 method to solve this optimization problem is proposed. Finally, the training process is implemented collaboratively by edge nodes and the cloud platform to support online inference.

2. Related Work

In this section, we reviewed the related research on edge-based MAL control methods and the existing achievements.

Unlike fixed airport lighting systems that are limited to permanent airfields, mobile airfield lighting provides a flexible, portable solution to address the lighting challenges of temporary or mobile landing. The earliest proposal to use Wireless Sensor Networks (WSNs) for airfield lighting system can be traced back to 2015 [7], in which networks energy balance and cost efficiency by taking advantage of the cluster head. Ref. [8] is the first paper to introduce an airfield lighting system that relies on an Internet of Things (IoT) network based on the cloud–fog–edge paradigm to build intelligent platforms and improve operational efficiency, though traditional cloud computing suffers from latency, privacy, and dependency issues that fog and edge computing can address. The rise in edge computing has provided a new deployment paradigm for intelligent control. By offloading computing power to edge nodes close to data sources and controlled objects, intelligent control algorithms can achieve real-time inference and decision-making locally, significantly reducing end-to-end control latency while alleviating cloud computing pressure and bandwidth burden [9–11]. However, these papers do not propose any efficient intelligent algorithm for task management in airfield lighting systems.

Meanwhile, machine learning and deep learning technologies provide powerful data-driven tools for intelligent control. By learning from historical operational data, AI models can automatically extract environmental features, predict system states, and optimize control parameters, thereby achieving adaptive capabilities that traditional mechanism models can hardly match. Ref. [12] is the paper that designs an airfield navaid lighting monitoring system using data mining algorithms to detect abnormal light conditions in real time, enabling timely responses and preventing potential accidents, from which we observed that optimized airfield lighting performance can be achieved through historical data. In particular, in [9], the intelligent algorithms such as Convolutional Neural Networks (CNN), Fast Fourier Transform (FFT) and K-means are applied to analyze the airport environmental data to achieve high detection accuracy, real-time data processing capability, low cost and also high energy efficiency in an edge–cloud collaborative platform. However, until now no research work relating to edge–cloud architecture and low-cost task optimization using deep learning approaches has been carried out on the MAL system.

Inspired by this research, this paper aims to address the above research gaps and solve the key problems of poor real-time responsiveness, low resource utilization efficiency, and difficulty in balancing delay and energy consumption in the MAL system.

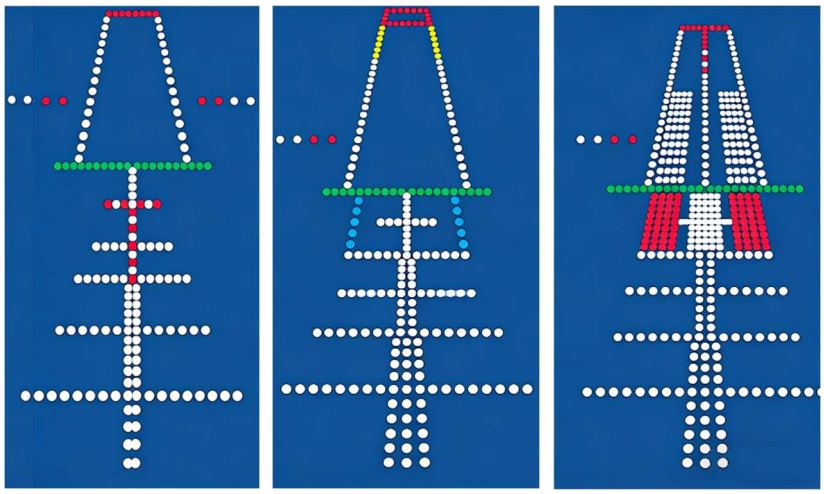
3. Cloud–Edge–End Mobile Airfield Lighting System

3.1. A Mobile Airfield Lighting Scenario

Unlike fixed airport lighting systems that are limited to permanent airfields, mobile airfield lighting provides a flexible, portable solution to address the lighting challenges of temporary or mobile landing sites, which is one of the key technologies of the smart low-altitude economy. This scenario can be designed to support various low-altitude aircraft, including drones, light aircraft, and emergency rescue vehicles, in scenarios such as temporary airfields, disaster relief sites, remote areas, and mobile deployment missions. It solves the problem of insufficient or non-existent lighting infrastructure in suboptimal lighting conditions, ensuring safe takeoff and landing by providing stable, adjustable, and

compliant lighting signals, which makes takeoff and landing possible 24 h a day in any environment to meet the demand for busy air traffic.

Mobile airfield lighting can also quickly adapt its optimal formation pattern based on the airport environment, lighting conditions, and the type of aircraft taking off or landing, thereby achieving the best navigational assistance effect. Scenarios [13] of mobile airfield lighting are shown in Figure 1. The scenario in Figure 1a is suitable for airports with low- to medium-sized flight volumes and basic operational requirements. It provides essential approach and runway lighting guidance, ensuring safe takeoffs and landings under normal visibility conditions. The scenario in Figure 1b is designed for airports that need to operate in relatively poor visibility. The scenario in Figure 1c represents the highest level of airport lighting system, suitable for hub airports with high flight frequencies and extremely high operation requirements. It allows aircraft to land in near-zero visibility conditions [14].



(**a**) (**b**) (**c**)

Figure 1. Scenarios of mobile airfield lighting. (**a**) Low- to medium-sized flight volumes; (**b**) Relatively poor visibility scenarios; (**c**) The highest level of lighting system.

3.2. A Cloud–Edge–End Collaborative Architecture for Mobile Airfield Lighting Systems

To support the scenarios of mobile airfield lighting, real-time control for such scenarios is important. One of the promising solutions is the cloud–edge–end framework, which overcomes the limitations of traditional central frameworks.

3.2.1. Overview of Cloud–Edge–End Computing Paradigm

The evolution of industrial cyber–physical systems has witnessed a gradual shift from centralized, cloud-centric architectures toward distributed computing paradigms that bring processing capabilities closer to data sources. However, as Internet of Things (IoT) deployments expand and applications demand increasingly stringent real-time performance, exclusive reliance on cloud-centric processing reveals inherent limitations in communication latency, bandwidth consumption, and privacy preservation [15]. Figure 2 illustrates a representative cloud–edge–end architecture for smart airfield IoT scenarios, where cloud platforms manage cross-domain connectivity, centralized data management, and global analytics by leveraging sensor networks, edge devices and end nodes to aggregate physical system operational states for process optimization.

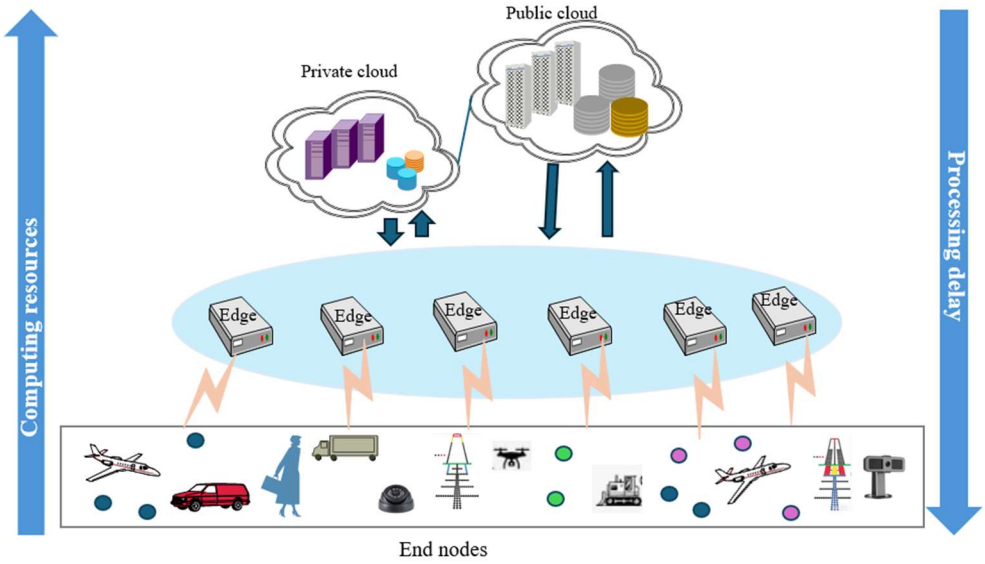


Figure 2. Cloud–edge–end computing architecture for smart airfield IoT scenarios.

To overcome the limitations of cloud-centric architectures, edge computing layers introduce computational capabilities at intermediate and proximal network locations [16]. Edge devices, typically deployed at regional aggregation points such as airport lighting stations, perform local data preprocessing, strategy caching, and regional coordination management. Edge devices, located closest to field assets, handle high-frequency data acquisition, preliminary processing, and deterministic control execution. This hierarchical structure enables latency-sensitive processing at the network edge while maintaining cloud capabilities for non-real-time analytics and global optimization, effectively reducing processing delays and enhancing overall system responsiveness.

3.2.2. Proposed Architecture for Mobile Airfield Lighting Systems

Mobile airfield lighting systems are deployed along runways, taxiways, and other critical operational areas to provide pilots with essential visual guidance through approach lights, runway lights, and taxiway centerline lights—especially under adverse weather conditions such as rain, snow, fog, and haze [17]. Operationally, lighting intensity settings and modes must comply with air traffic control directives and dynamically adjust to visibility conditions, runway usage, and aircraft movement patterns. Typical systems consist of field components (lighting fixtures, connectors) and supporting infrastructure (power distribution, circuit control, monitoring software, sensors). The scale of these systems varies significantly depending on the airport classification and runway–taxiway layout, ranging from hundreds to thousands of lighting fixtures.

The intelligent evolution of airfield lighting systems—driven by an increasing number of control objects, higher data acquisition frequencies, and deeper operational integration—progressively exposes the bandwidth, latency, and scalability limitations inherent in traditional centralized processing models. In particular, high-concurrency scenarios such as real-time guidance and runway incursion prevention reveal vulnerabilities in centralized data processing, including link uncertainties and computational bottlenecks, which hinder the simultaneous achievement of latency-critical responses and continuous availability objectives [18]. To address these challenges, this paper proposes a cloud–edge–end collaborative architecture specifically designed for airfield lighting systems in Figure 2, as illustrated in Figure 3.

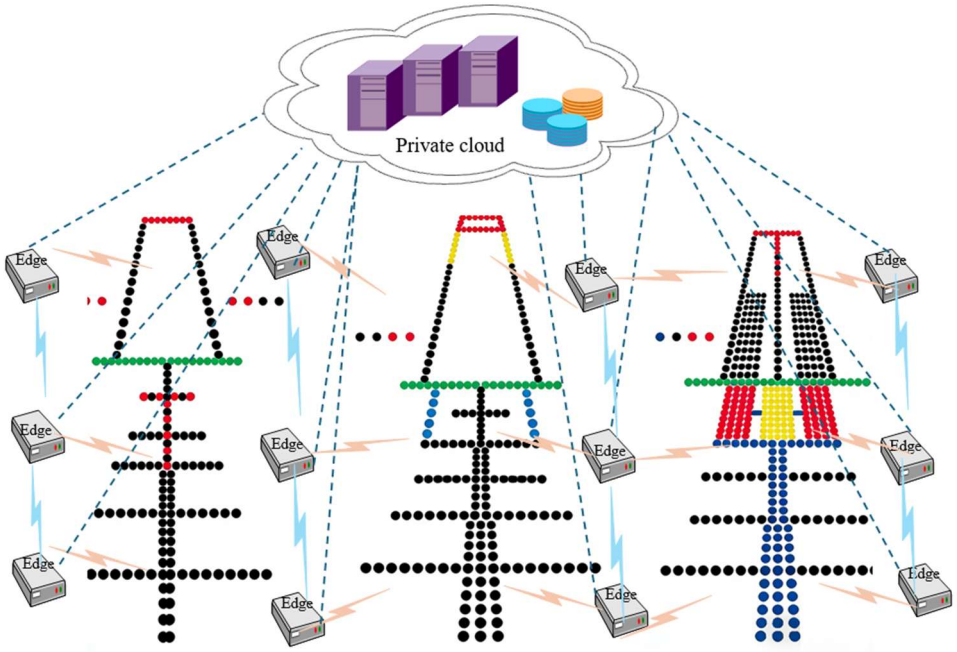


Figure 3. Proposed cloud–edge–end architecture for mobile airfield lighting systems.

The proposed architecture comprises three hierarchical layers:

1. End nodes: The end nodes are field equipment airfield lights, including airfield lights, lightweight computational terminal control units, sensors, and communication radios. These components perform high-frequency status acquisition, essential local preprocessing, limitation computing and auxiliary decision-making. Due to its limited resources, the final decision generally will depend on the upper layer issuance of control commands after comprehensive judgment.
2. Edge Layer: The edge devices have communication radios, sensors and computing capabilities. By communicating with end nodes, they obtain the acquired data or raw preprocessed data and deterministic control instructions for execution. For maintenance-related tasks, edge nodes may run local prediction decisions or diagnostic algorithms to identify anomalies early and trigger alerts. Due to the long total deployment distance of navigation airfield lights, an ALC system supported by multiple edge nodes is usually required. The edge nodes exchange information with the cloud, while transmitting critical status summaries upward to cloud layers to support collaborative and predictive maintenance decision-making. All edge nodes possess a certain store and computing capability.
3. Cloud Layer: Positioned at the architectural apex, the cloud data center manages non-real-time, long-cycle, large-scale data processing tasks, including global data governance and storage, complex model training and iterative updates, cross-regional operational coordination, and strategic planning optimization. To support heterogeneous data source integration and cross-component collaboration, cloud deployments incorporate airfield lighting-specific service components and communication middleware. These components enable reliable, decoupled communication through message brokering mechanisms.

Compared to traditional centralized architectures, the proposed collaborative framework offers three primary advantages. First, latency-sensitive processing and control functions are delegated to the edge and fog layers, reducing reliance on central links and thereby systematically decreasing end-to-end delay and jitter. Second, event-triggered reporting strategies that implement preprocessing and filtering at proximal layers alleviate network bandwidth pressure and improve scalability. Third, the hierarchical autonomy at

the edge enables the system to maintain basic functionality and critical control operations during network impairments or link anomalies, thereby enhancing overall system resilience and availability [19].

4. Edge Intelligent Task Management for MAL Systems

4.1. System Model and Problem Formulation

Airfield lighting control usually requires final adaptive adjustment system parameters of lighting according to environment parameters during aircraft takeoff and landing, which is critical to ensuring flight guidance quality. At this stage, there is a stringent requirement for low latency, and the lighting parameter optimization based on edge-node collaboration must be completed within a short slot. How to achieve efficient task scheduling and rational resource allocation is a critical issue for performance enhancement, which will partly be dealt with in this section.

4.1.1. Communication Model

Our system is the cloud–edge–end architecture of the mobile airfield lighting system shown in Figure 3. Consider a system composed of multiple edge devices and airfield lighting nodes: the system includes M edge nodes, and N airfield lighting nodes (ALN). Each edge node (EN) provides computing and communication services for the ALNs within its coverage area. The ALNs communicate with ENs via wireless channels, and ENs also communicate with the cloud through cable channel or wireless channels. Each EN is at a fixed position when making decisions and performing parameter optimization; their positions are represented by two-dimensional coordinates (x_k, y_k), where $k \in \{1, 2, \ldots, M, L\}$ and L denotes the cloud.

When the distance is the reference distance c the path loss $L(d_0)$ can be calculated using the following formula:

$$L(d_0) = \left(\frac{4\pi d_0 f}{cv}\right)^2 \tag{1}$$

where d_0 is the reference distance, usually set to 1 m; f is the signal carrier frequency; and cv is the speed of light.

Assume the distance between the n-th ALN and the m-th EN is $d_{n,m}$. Then:

$$d_{n,m} = \sqrt{H^2 + (x_m - x_n)^2 + (y_m - y_n)^2} \tag{2}$$

The Rayleigh fading model is adopted to characterize the impact of multipath fading on the channel gain. The channel gain between the n-th ALN and the m-th EN $g_{n,m}$ is given by:

$$g_{n,m} = \frac{1}{L(d_0)}\left(\frac{d_0}{d_{n,m}}\right)^{\beta} h \tag{3}$$

where β is the path loss exponent, typically set to 2 in free space; h follows a Rayleigh distribution, which captures the random variation in signal amplitude caused by multipath fading and its effect on channel gain.

The signal-to-noise ratio (SNR) of the n-th ALN on the c subchannel can be expressed as:

$$SNR_{n,m,c} = \frac{P_{n,c} g_{n,m}}{\sigma^2 + \sum_{n' \neq n} P_{n',c} g_{n',m}} \tag{4}$$

where $P_{n,c}$ denotes the transmit power of the n-th ALN on the c subchannel. SNR is a key indicator of channel quality and the primary basis for calculating delay and energy consumption in the following subsection.

σ^2 is the noise power; $\sum_{n' \neq n} P_{n',c} g_{n',m}$ represents the interference from other ALNs. Based on the Shannon formula, the communication rate between the n-th ALN and the m-th EN can be expressed as:

$$R_{n,m} = B \sum_{c=1}^{C} log_2(1 + SNR_{n,m,c}) \tag{5}$$

4.1.2. Local Computation Delay and Energy Cost Model

Assume that the local computing capability of the n-th ALN is cf_n. Its local pre-process delay can be expressed as:

$$T_n^{local} = \frac{D_n C_n}{cf_n} \tag{6}$$

where D_n denotes the dataset size of parameters of the n-th ALN, and C_n represents the number of calculations per data sample.

After local computing is completed, the n-th ALN uploads the updated pre-process parameters to the m-th EN (the ALN will update the raw parameters if the computing resource cannot support the pre-process), and the time required for the n-th ALN to upload data to the m-th EN $T_{n,m}^{up}$ is:

$$T_{n,m}^{up} = \frac{|w_n|}{R_{n,m}} \tag{7}$$

where $|w_n|$ represents the data size of parameters of the n-th ALN, and $R_{n,m}$ denotes the communication rate between the n-th ALN and the m-th EN. The single-node delay and energy cost of n-th ALN are given by:

$$T_n = T_n^{local} + T_{n,m}^{up}, \quad \forall n \in \{1, 2, \ldots N\} \tag{8a}$$

$$E_n = \kappa \, cf_n^3 T_n^{local} + P_{n,c} T_{n,m}^{up}, \forall n \in \{1, 2, \ldots N\} \tag{8b}$$

where κ is the coefficient of computing energy, cf_n is the CPU frequence of local computing, and $P_{n,c}$ is the transmission power of the n-th ALN under the subchannel c.

The local computing delay $T_{m,k}^{com}$ and energy cost $E_{m,k}^{com}$ of the m-th EN to process the local parameter task at position k can be expressed as:

$$T_{m,k}^{com} = \sum_{n=1}^{N} \alpha_{n,m} \frac{|w_n| c_m}{cf_m} \tag{9a}$$

$$E_{m,k}^{com} = \kappa \, cf_m^3 T_{m,k}^{com} \tag{9b}$$

where $\alpha_{n,m}$ is a binary variable indicating whether the n-th ALN selects the m-th EN for parameter upload. c_m represents the computing load of the EN per unit bit of data, and cf_m denotes the computing capability of the m-th EN.

4.1.3. Edge and Cloud Computation Delay and Energy Cost Model

Let $|w_m|$ represent the data size of the m-th EN, and let $R_{m,L}$ and P_m denote the communication rate between the m-th EN and the cloud and the transmission power of the m-th EN. Then the EN will upload the updated data to the cloud. The upload delay $T_{m,k}^{up}$ and energy cost $E_{m,k}^{up}$ of the m-th EN are:

$$T_{m,k}^{up} = \frac{|w_m|}{R_{m,L}} \tag{10a}$$

$$E_{m,k}^{up} = P_m T_{m,k}^{up} \tag{10b}$$

https://doi.org/10.3390/aerospace13050424

Let c_L and cf_L denote the computing load of the cloud per unit bit of data and the computing capability of the cloud unit, respectively. The time required for the cloud to process the task is:

$$T_{L,k_L}^{com} = \sum_{m=1}^{M} \frac{|w_m|c_L}{cf_L} \tag{11}$$

If the capacities of ENs can support computing, it will not leverage cloud platform support. In this case, the delay T_{L,k_L}^{com} equals to 0.

Finally, the cloud distributes the updated parameters, which are first sent to each EN and are then forwarded from each EN to the corresponding ALN. The total distribution delay T_{L,k_L}^{down} from cloud to EN is:

$$T_{L,k_L}^{down} = \max_{m=1,2,..,M} \left(\frac{|w_L|}{R_{L,m}} + \sum_{n=1}^{N} \alpha_{m,n} \frac{|w_L|}{R_{m,n}} \right) \tag{12}$$

where $|w_L|$ represents the data size of the L-th EN, and $R_{L,m}$ denotes the communication rate between the L-th ALN and m-th.

And the delay from the EN to the ALN $T_{m,k}^{down}$ is:

$$T_{m,k}^{down} = \sum_{n=1}^{N} \alpha_{m,n} \frac{|w_L|}{R_{m,n}} \tag{13}$$

where $\alpha_{m,n}$ is a binary variable indicating whether the m-th EN distributes the final parameters to the n-th ALN. For general purposes, here we assume that the communication rates of the uplink and downlink are the same.

4.1.4. Total System Delay and Energy Consumption Model

Then, the total delay and energy cost only consider the cost of ALNs and Ens as:

$$T^{all} = \max_{m \in \{1,2,..M\}} \left[\sum_{n \in N_m} \left(T_n^{local} + T_{n,m}^{up} \right) + \left(T_{m,k}^{com} + T_{m,k}^{up} + T_{m,k}^{down} \right) \right] + \left(T_{L,k_L}^{com} + T_{L,k_L}^{down} \right) \tag{14a}$$

$$E^{all} = \sum_{n=1}^{N} E_n + \sum_{i=1}^{M} E_{i,n}^{all} \tag{14b}$$

How to obtain the minimum total delay and energy cost of airfield lighting nodes and edge nodes under transmission condition constraints is our main task and will be addressed in the next subsection.

4.1.5. Optimization Problem Model

By optimizing the total delay and energy consumption of the aforementioned system model, we simultaneously consider the dynamic adjustment of channel allocation and computational resources. The objective function of the optimization problem is:

$$P1: \min_{\alpha_{m,n}, cf_i} \lambda_1 T^{all}(\alpha_{m,n}, f_i) + \lambda_2 E^{all}(\alpha_{m,n}, cf_i) \tag{15}$$

$$\text{s.t.} \begin{cases} C1: T_n(\alpha_{m,n}, cf_i) \le T_{max}, \forall n \in \{1,2,\ldots,N\} \\ C2: E^{all} \le E_{max} \\ C3: \sum_{n=1} \phi_{n,c} \le 1, \forall c \in \{1,2,\ldots,C\} \\ C4: 0 < cf_i \le f_{max}, \forall i \in \{1,2,\ldots,M,L\} \end{cases} \tag{16}$$

where λ_1,λ_2 are the weighting coefficients for delay and energy cost. $\alpha_{m,n}$ and $\phi_{n,c}$ are binary variables indicating whether the n-th ALN occupies the c-th subchannel. Constraint C1 is the computational delay constraint, which requires that the computational delay T_n of

each ALN does not exceed its maximum tolerable delay T_{max}. Constraint C2 is the energy constraint such that the total energy consumption of the ENs must not exceed the total energy storage capacity E_{max}. Constraint C3 is the channel allocation constraint, which requires that each subchannel can only be occupied by one ALN. Constraint C4 is the computational resource constraint, which ensures that the computational resources of each EN cf_i do not exceed its maximum computing capacity f_{max}.

4.2. The Solution of Optimization Problem by Improved TD3 Method

The number of lights in a typical airport's airfield ground lighting system is directly related to the runway length (common runway lengths range from 1800 to 4000 m), with quantities varying from one hundred to three thousand. When the number of lights is less than 100, the optimization problem in Equations (15) and (16) can be solved by traditional alternating iteration and meta-heuristic algorithms [19]. However, the complexity of those algorithms grows exponentially when the number of lights is large. To end this, we proposed an Improved Twin Delayed Deep Deterministic Policy Gradient (TD3) [20] approach to solve this optimization problem.

4.2.1. Overview of Proposed Method

The method combines the clustering method (such as K medoids) with the improved TD3 (Twin Delayed Deep Deterministic Policy Gradient) to integrate the efficiency of clustering for rough allocation and the high-precision dynamic optimization capability of the improved TD3. It is perfectly suited for the practical requirement of online dynamic resource adjustment in the system and outperforms traditional alternating iteration and meta-heuristic algorithms in real-time optimization for large-scale scenarios. The clustering algorithm enables fast rough allocation (addressing the curse of dimensionality in large-scale scenarios), while the improved TD3 achieves high-precision fine optimization (solving the joint optimization problem of mixed-integer and continuous variables).

As an advanced deep reinforcement learning (DRL) algorithm, improved TD3 excels at high-precision optimization in continuous spaces and can be adapted to mixed-integer spaces through improvements. It can perform refined optimization of computational resources f_i and channel allocation $\alpha_{m,n}$ on the basis of clustering-based rough allocation while satisfying multiple constraints. The improved TD3 learns through interaction with the environment without manually designing analytical solutions. It adapts to the dynamics of the system and has far higher robustness than traditional algorithms. It is divided into two highly coupled stages: K-medoids [21] rough allocation and improved TD3-based fine optimization.

K-medoids is selected because it is robust to outliers (suitable for the MAL system with random node state changes) and can quickly realize rough resource allocation to solve the curse of dimensionality in large-scale systems. TD3 is selected because it effectively solves the overestimation bias and training instability of DDPG and is more suitable for the continuous action space optimization of MAL system resource allocation, and the improved TD3 can further adapt to the mixed-integer continuous action space of the problem.

4.2.2. K-Medoids-Based ALN-EN Rough Allocation

The end-to-end delay cost $C_{n,m}$ from ALN to EN is selected as the clustering feature (defined in previous methods, $C_{n,m} = T_n(\alpha_{m,n} = 1; cf_i) + E_n(\alpha_{m,n} = 1; cf_i)$. It is directly calculated using Equation (8) and can directly reflect the optimization objective. For an airfield lighting system with M ENs and N ALNs, the clustering stage is as follows:

Step 1, **Initialization**: Randomly select M ALNs from N ALNs as the initial cluster centroids, with each centroid corresponding to one EN, and determine the initial clusters $C_{n,m} = C_m(m = 1, 2, \ldots, M)$, where C_m is the set of ALNs covered by the m-th EN.

Step 2, **Allocation Phase**: Calculate the similarity $d(n,m) = T_{n,m}^{up}$ from each ALN to M clustering centers, and assign the ALN to the clustering center (corresponding EN) with the **minimum distance,** denoted as $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_M$, where each cluster $\mathcal{G}_m$ corresponds to one EN.

Step 3, **Cluster center update**: For each cluster, traverse all ALNs in the cluster and select the ALN that minimizes the total distance of all samples in the cluster as the new clustering center. For each cluster $\mathcal{G}_m$, screen the feasible ALN set $\mathcal{G}_m^* = \{n \in \mathcal{G}_m \mid C_{n,m} \leq T_{max}\}$ that satisfies the C1 delay constraint to ensure the rough allocation result meets the basic constraints.

Step 4, **Convergence judgment**: Stop clustering if the clustering centers no longer change or the variation in the total intra-cluster distance is less than the threshold; otherwise, return to Step 2.

Step 5, **Rough allocation result**: Obtain the initial channel allocation matrix $\left\{\alpha_{m,n}^{(0)}\right\}$ for each cluster $\mathcal{G}_m^*$ by selecting the ALN n^* with the minimum delay cost $C_{n,m}$ and set $\alpha_{m,n^*}^{(0)} = 1$, $\alpha_{m,n}^{(0)} = 0$.

The workflow of K-medoids-based ALN-EN rough allocation is shown in Figure 4.

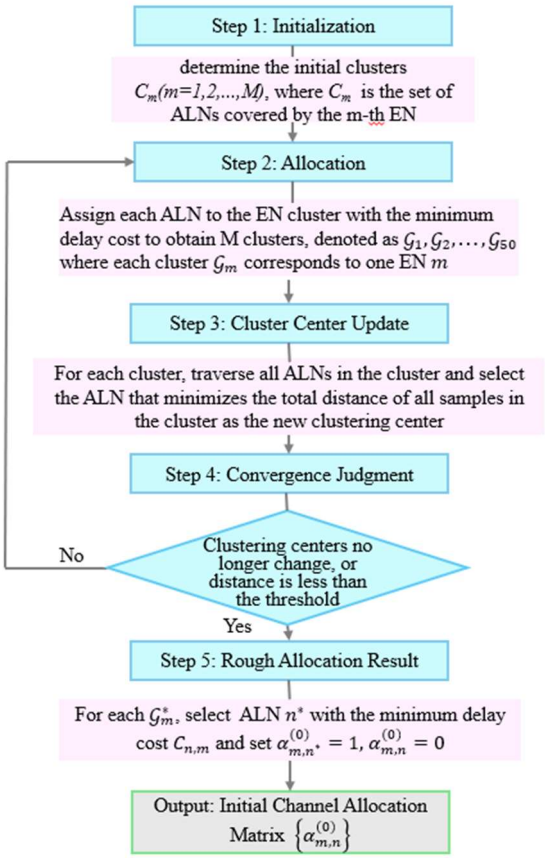


Figure 4. K-medoids-based ALN-EN rough allocation.

4.2.3. Improved TD3-Based Refined Resource Optimization

TD3 is a classic deep reinforcement learning algorithm for continuous action spaces, which solves the overestimation and training instability problems of Deep Deterministic Policy Gradient (DDPG) through dual critic networks, delayed policy update, and action noise regularization. The improved TD3 is adapted to mixed-integer action space (binary $\alpha_{m,n}$+ continuous cf_i) through action space discretization and hard constraint embedding, with the optimization objective of minimizing the total delay T^{all} and total energy cost E^{all}

in Equation (16) while satisfying all C1/C2/C3/C4 constraints in Equation (16). The state, reward, and termination conditions of the environment are defined as follows.

1. State space S

State space S contains the core dynamic parameters of the system, including delay cost, communication rate, computational resource, single-node delay, and total delay. State space s is defined as:

$$S = \{cf_m,\ cf_n, P_{n,c}, R_{n,m}, T_n, E_n, E_m^{rem}\} \tag{17}$$

where $\forall n \in \{1, 2, \ldots, N\}$, $\forall m \in \{1, 2, \ldots, M\}$ and $\forall i \in \{1, 2, \ldots, M, L\}$. cf_m denotes the computing frequency of the m-th EN and cf_n denotes the computing frequency of the n-th ALN in the cluster. $P_{n,c}$ is the subchannel transmission power of the n-th ALNs in the cluster, and represents the associated end-to-end delay cost. $R_{n,m}$ is the communication rate in Equation (5) and T_n is the delay of a single EN. E_n in Equation (8b) is the energy consumption of the n-th ALNs and E_m^{rem} is the remaining energy of the m-th EN that should satisfy the total energy constraint C2. All state variables are normalized to the range [0, 1] to eliminate the influence of dimensionality and improve the training efficiency of neural networks.

2. Action space A

The action space includes the selection of parameter upload nodes for ALNs, channel allocation and computing resource allocation. The action space is expressed as:

$$A = \{\alpha_{m,n}, \phi_{m,c}, P_m\}, \forall n \in \{1, 2, \ldots, N\}, \forall m \in \{1, 2, \ldots, M, L\} \tag{18}$$

where $\alpha_{m,\,n}$ is a binary switch variable that intuitively represents whether the nth airfield lighting node uploads parameters to the mth edge node. $\phi_{m,c}$ denotes the channel allocation decision and P_m denotes the proportional allocation of computing resources according to tasks.

3. Reward function $r(S, A)$

The reward function is the core of this improved TD3, which is designed as the negative value of the weighted objective function plus constraint penalty terms, so that the agent minimizes the objective function by maximizing the reward and strictly satisfies all constraints (C1–C4).

$$r(S, A) = -\lambda_1 \frac{T^{\text{all}}(S, A)}{T_{\max}} - \lambda_2 \frac{E^{\text{all}}(S, A)}{E_{\max}} - \gamma \sum_{k=1}^{4} P_{en_k}(S, A) \tag{19}$$

where penalty terms $Pen_k(S, A)$ are imposed on behaviors that violate constraints C1–C4, and γ is a large penalty coefficient (e.g., 10) to ensure constraint satisfaction:

$$P_{en_1} = 1 \text{ if } T_n > T_{max}, \text{ otherwise } Pen_1 = 0 \tag{20a}$$

$$P_{en_2} = 1\ E^{all} > E_{max}, \text{ otherwise } Pen_2 = 0 \tag{20b}$$

$$P_{en_3} = 1 \sum_n \varphi_{n,c} > 1, \text{ otherwise } Pen_3 = 0 \tag{20c}$$

$$P_{en_4} = 1 \text{ if } f_i > f_{max}, \text{ otherwise } Pen_4 = 0 \tag{20d}$$

Such a penalty term γ restricts excessive model complexity, such as large weights and a larger penalty coefficient strengthens regularization, simplifying the model.

4.2.4. Design of the Improved TD3 Algorithm

The standard TD3 is designed for Markov Decision Processes (MDPs) with continuous action spaces and has problems such as difficulty in handling hybrid discrete–continuous

actions, overestimation bias, and low exploration efficiency. To address the problem considered in this paper, three core improvements are made to adapt to the optimization requirements of the airfield lighting system, including K-medoids for rough resource allocation and ITD3 for refined optimization.

1. Actor Network

A dual-branch action output structure is introduced in the actor network to decouple the training of continuous and discrete actions, avoiding the interference of discrete variables on the optimization of continuous variables:

Continuous branch: A fully connected layer is used to output the normalized computing frequency f_i and transmission power $P_{n,c}$, which are mapped to the actual value range through denormalization ($0 < f_i \leq f_{\max}$, $P_{\min} \leq P_{n,c} \leq P_{\max}$).

Discrete branch: A softmax layer is used to output the probability distribution $P(\phi_{n,c}) = 1$ of channel allocation. Gumbel-Softmax sampling is adopted to realize the differentiable training of discrete variables; then, the binary allocation variable $\varphi_{n,c}$ is obtained through a hard threshold (0.5), ensuring that the channel exclusive constraint C3 is satisfied.

2. Critic Networks

The standard TD3 alleviates the Q-value overestimation bias through dual critic networks + delayed policy update + action noise regularization. On this basis, this paper adds constraint-aware Q-value correction. The two critic networks $Q_1(S, A)$ and $Q_2(S, A)$ calculate the Q-values respectively, and the minimum value is taken as the target Q-value to avoid overestimation of a single network. For actions that violate constraints, their Q-values are corrected to a minimum value, so that the critic networks learn that "actions violating constraints have no value", forming a dual constraint with the penalty term of the reward function. The update of the actor network is delayed relative to the critic networks (the delay step is set to 2) to avoid the transfer of unconverted errors of the critic networks to the actor network.

3. Network Structure of the Improved TD3

The standard TD3 adopts uniform experience replay, which has the problem of low utilization efficiency of effective experience. This paper introduces prioritized experience replay (PER). Priorities are assigned to experience samples (S, A, r, S') according to the Temporal Difference (TD) error. The larger the TD error, the higher the sample priority (containing more unlearned information). Weighted sampling is performed according to priorities during sampling to improve the utilization rate of high-value samples and accelerate network convergence. Importance Sampling Weights (ISWs) are introduced to correct the bias caused by priority sampling and ensure the unbiasedness of training.

Table 1 presents the network structure of the improved TD3. ReLU is an activation function, which introduces non-linearity by outputting the input directly if it is positive, otherwise outputting zero.

$$\mathrm{ReLU}(x) = \max(0, x) \tag{21}$$

Tanh is used for the continuous action branch and is formulated as:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{22}$$

Table 1. Network structure of the improved TD3.

Network Type	Structure (Input → Hidden Layers → Output)	Activation Function	Optimizer
Actor Network	State S → 2-layer FC(256,128,64) → Dual-branch (Continuous + Discrete)	ReLU+Tanh/Softmax	Adam
Critic Network Q1	[S, A] → 2-layer FC(256,128,64) → 1D Q-value	ReLU+Linear	Adam
Critic Network Q2	[S,A] → 2-layer FC(256,128) → 1D Q-value	ReLU+Linear	Adam

Softmax is used for the discrete action branch and is formulated as:

$$\text{Softmax}(x) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{23}$$

Adam (Adaptive Moment Estimation) [22] is used to update network weights.

The action space includes discrete variables (channel allocation, node association) and continuous variables (computing frequency, transmit power). However, standard TD3 cannot directly train discrete actions because they are non-differentiable. Softmax enables differentiable sampling of discrete actions, allowing end-to-end training of the actor network while satisfying exclusive channel constraints, which is essential for handling the hybrid discrete–continuous action space in MAL systems.

The Improved Twin Delayed Deep Deterministic Policy Gradient (ITD3) proposed in this paper introduces three distinct novel components that differentiate it from the standard TD3 algorithm: (1) a dual-branch actor network structure, consisting of a continuous branch for computing frequency and transmit power optimization, and a discrete branch with Gumbel-Softmax for differentiable channel allocation, enabling end-to-end training in hybrid discrete–continuous action spaces. (2) Constraint-aware Q-value correction, which explicitly penalizes constraint-violating actions in the critic networks, forming a dual constraint mechanism with the reward function to strictly satisfy delay, energy, channel, and computing resource limits. (3) Prioritized experience replay (PER) integration which replaces uniform experience replay with TD-error-based weighted sampling to improve sample efficiency and accelerate network convergence.

4.2.5. Training Process of the Improved TD3

The training process of the improved TD3 will perform the following steps:

Step 1 (Initialization): Initialize the actor network π_i, the target actor network π_{target}, the dual critic networks Q_1/Q_2, and the target critic networks $Q_{1target}/Q_{2target}$, and copy the parameters $(\pi_{target}\pi, Q_{target}Q)$; initialize the prioritized experience replay buffer D (capacity 10^6).

Step 2 (Exploration and sampling): For the current state S_e, the actor network outputs the action A, and Gaussian noise is added (the exploration rate decays from 0.1 to 0.01 with the number of training steps) to obtain the exploratory action A'; execute A' to obtain the reward r and the next state S', and store the experience sample (S, A', r, S') in D.

Step 3 (Experience sampling): Sample a batch of samples (batch size 64/128) from D according to priorities and calculating the Importance Sampling Weights (ISWs) [23].

Step 4 (Target Q-value calculation): Obtain the next action $A'_{target} = \pi_{target}(S')$ through the target actor network, add target policy noise (to avoid overfiting), and calculate the target Q-value: $y = r + \gamma \cdot \min\left(Q_{1target}\left(S', A'_{target}\right), Q_{2target}\left(S', A'_{target}\right)\right) \cdot ISW$.

Step 5 (Critic network update): Minimize the loss functions $L_{Q1} = E\left[(y - Q_1(S, A))^2\right]$ and $L_{Q2} = E\left[(y - Q_2(S, A))^2\right]$, and update the parameters of Q_1/Q_2.

Step 6 (Actor network update): Minimize the policy loss $L_\pi = -E[Q_1(S, \pi(S))]$, and update the parameters of the actor network π.

Step 7 (Soft update of target networks): Update the parameters of the target networks in a soft update manner with an update rate $\tau = 0.005$.

$$\theta_{\pi_{target}} \leftarrow \tau\theta_\pi + (1-\tau)\theta_{\pi_{target}} \tag{24}$$

$$\theta_{Q_{traget}} \leftarrow \tau\theta_Q + (1-\tau)\theta_{Q_{traget}} \tag{25}$$

Step 8 (Convergence judgment): Stop training if the sliding average of the reward function has no improvement for 500 consecutive steps or the maximum number of training steps is reached; otherwise, return to Step 2.

To summarize, we first use K-medoids clustering to quickly group lights and edge nodes to reduce the problem size and avoid overly complex calculations. Then we use an improved TD3 reinforcement learning method to let the system learn by interacting with the environment: it observes the current state (delay, energy, computing resources), makes actions (node assignment, channel allocation, resource adjustment), and receives rewards. Good behavior earns positive rewards; violations of delay or energy limits incur penalties.

We improved TD3 to fit our task: use a dual-branch network to handle both discrete and continuous actions, add constraint checks to avoid invalid decisions, and use priority sampling to speed up training. In short, clustering performed rough grouping, and improved deep reinforcement learning performs precise optimization. Together they minimize delay and energy while keeping the system stable and ensuring real-time performance for mobile airfield lighting control.

5. Simulation Results and Discussion

5.1. Simulation Environment for Algorithm Evaluation

To evaluate the performance of the proposed improved TD3 algorithm for the MAL system, we conducted extensive simulations using a discrete-event simulator implemented in Python 3.12. This subsection details the simulation setup, including the network topology, task generation model, resource configuration, benchmark algorithms, and performance metrics.

The simulated system consists of M ENs and N ALNs. The experimental parameter settings for task management for the MAL system are shown in Table 2. The improved TD3 network training parameters setting are shown in Table 3 according to the actual physical constraints of MAL systems (e.g., computing capacity of edge nodes and transmission power of terminal nodes) and the empirical settings from mainstream related studies in the field. The simulation parameters are calibrated based on the actual technical specifications of commercial MAL system equipment.

Table 2. Experimental parameter settings for task management for MAL system.

Parameters	Value
The number of ALNs N	50/100/500/1000/2000
The number of ENs M	$N/25$
The computing capacity of EN cf_m	[3, 4] GHz
The computing capacity of ALN cf_n	[0.5, 2.5] GHz
Transmission power of ALN $P_{n,c}$	[0.5, 5] W
Band of wireless channel B	2 MHz
Noise power σ^2	-50 dBm
Size of training data D_n	[0.1, 2] MB
Size of upload data $\|w\|$	[0.1, 2] MB

Table 3. The improved TD3 network training parameters.

Module	Parameter Name	Recommended Value
K-medoids	Number of clusters K	Number of ENs (M)
	Convergence threshold	10^{-3}
Improved TD3	Batch size	64/128
	Hidden layer dimension	256/128
	Learning rate (Actor) μ	10^{-3}~10^{-5}
	Learning rate (Critic) ρ	10^{-2}~10^{-4}
	Discount factor	0.95~0.99
	Soft update rate	0.005
	Exploration noise std	0.1 $\rightarrow$ 0.01 (decay)
	Experience replay buffer capacity	10^6
Reward Function	Penalty coefficient γ	10
	Weighting coefficients λ_1/λ_2	0.7/0.3 (low-latency priority)

The system operates in discrete time slots of duration $\Delta t = 100$ ms. Each simulation run covers T = 3600 time slots (equivalent to 6 min of operation), and the results are averaged over 10 independent runs with different random seeds. $\lambda_1 = 0.7$ and $\lambda_2 = 0.3$ are chosen because **low latency is critical** for MAL real-time control; we choose such values based on prior experience, in which 0.7/0.3 achieves the best balance between delay and energy for airport lighting scenarios, compared with λ_1/λ_2 = 0.5/0.5, 0.6/0.4, 0.7/0.3, and 0.8/0.2.

5.2. Performance Evaluation

All simulations and training were performed on a computing platform equipped with an Intel Core i9-10700K CPU with an NVIDIA RTX 4060 GPU under Python 3.12.

Under scenarios with the number of ALNs ranging from 50 to 2000 and the number of ENs ranging from two to 80, the traditional alternating iteration (TAI) [19], Deep Deterministic Policy Gradient (DDPG) [24], Proximal Policy Optimization (PPO) [25], TD3 [20] and improved TD3 (ITD3) in this paper are employed to evaluate the solution of the MALC optimization problem shown in Equations (15) and (16).

The variations in total delay and total energy consumption with the number of training iterations were illustrated in Figure 5, in which the number of Airport Lighting Nodes (ALNs) N was set to 50, 100, 500, 1000, and 2000 respectively, and the delay and energy consumption of various algorithms were calculated. Then the overall total delay and total energy consumption were obtained by averaging the results. As can be seen from Figure 5, the ITD3 method proposed in this paper demonstrates certain advantages in terms of both total delay and total energy consumption. Moreover, when the number of training epochs exceeds 2000, the improvement of all algorithms becomes negligible.

The ITD3 algorithm outperforms ATI, DDPG, PPO and traditional TD3 due to the fact that the K-medoids rough allocation reduces the algorithm search space, the improved TD3 solves the overestimation bias and adapts to the mixed action space, and the prioritized experience replay accelerates network convergence.

We conducted simulation experiments on the total delay and energy efficiency with different numbers of ALNs, as shown in Figure 6. The number of ALNs was set to 50, 100, 500, 1000, and 2000, respectively. Figure 6 shows that both the total delay and total energy consumption of the system increase with the rising number of ALNs. This is because more

nodes participating in training will inevitably lead to longer local training time and higher training energy consumption, thus resulting in increased delay and energy consumption.

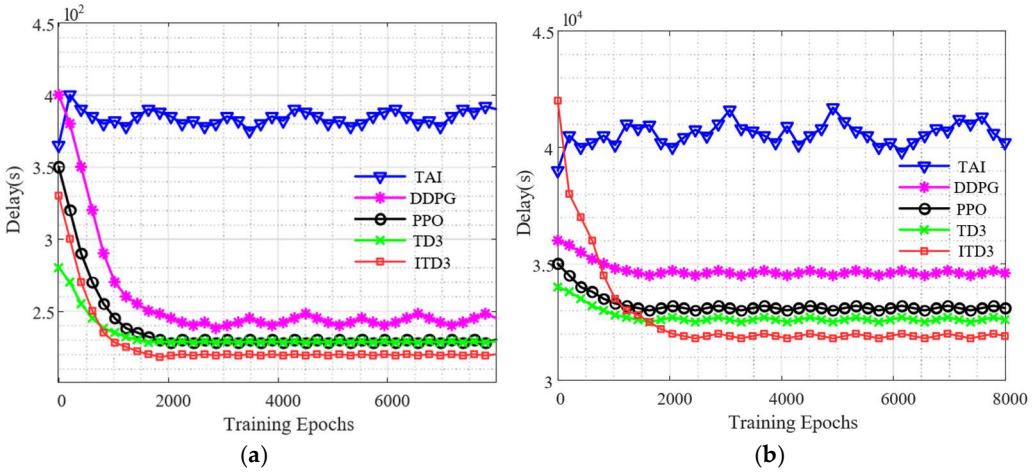


Figure 5. Total delay and total energy consumption via train epochs. (**a**) Comparison of delay under different Algorithms; (**b**) comparison of energy under different algorithms.

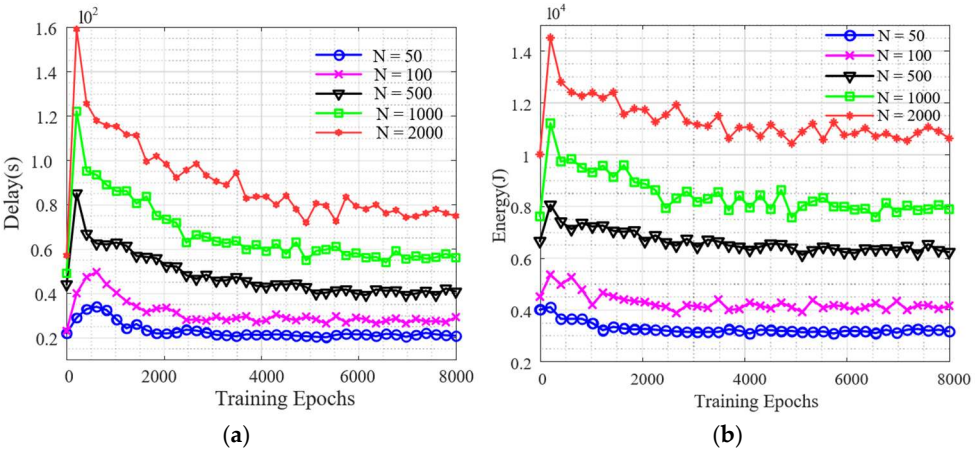
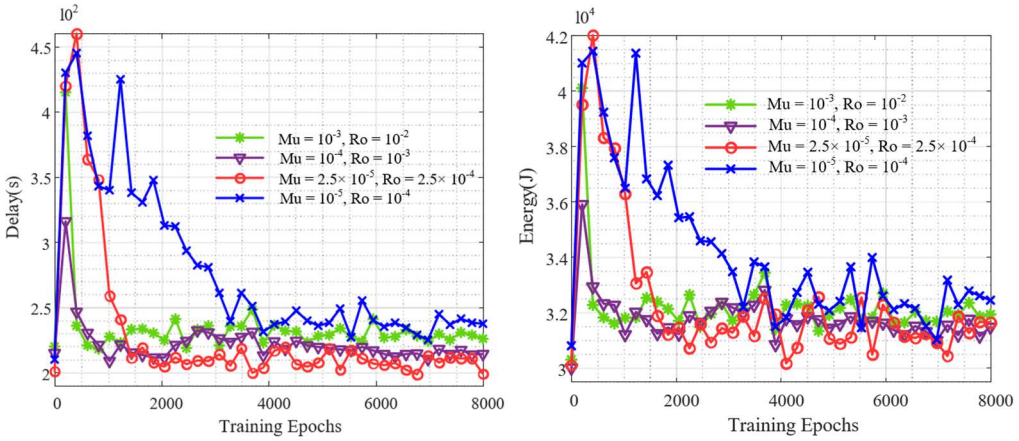


Figure 6. The comparison of total delay and total energy consumption of ITD3 under different numbers of ALNs. (**a**) Comparison of total delay of ITD3; (**b**) comparison of total energy of ITD3.

Figure 7 investigates the performance of the proposed ITD3 algorithm under different learning rate settings. Figure 7a,b show the performance of the algorithm in terms of total delay and total energy consumption under different learning rates, where "Mu" denotes the learning rate of the actor network μ and "Ro" denotes the learning rate of the Critic network ρ.

As μ and ρ decrease from 0.001 and 0.01, the algorithm performance improves in both total delay and total energy consumption. When μ = 0.000025 and ρ = 0.00025, the algorithm converges slowly in the early stage but achieves the optimal stable values in total delay and total energy consumption after the number of train epochs exceeds 2000. In contrast, when the learning rates are further reduced to μ = 0.00001 and ρ = 0.0001, both total delay and total energy consumption performance become worse. When the number of training epochs is less than 2000, the algorithm maintains a good convergence speed with favorable total delay and total energy consumption values at μ = 0.0001 and ρ = 0.001. The algorithm performance changes with different learning rates (e.g., a too high learning rate leads to training oscillation, while a too low learning rate leads to slow convergence, which means that the optimal learning rate balances the convergence speed and stability).



(**a**) (**b**)

Figure 7. Total delay and total energy consumption via train epochs. (**a**) Total delay under different learning rates; (**b**) total energy under different learning rates.

Results show the proposed method maintains a stable and superior performance within reasonable ranges of key hyperparameters, including the number of ALNs nodes and the learning rate, which verified that the advantages do not rely on extreme or carefully tuned parameters.

From this performance, we can summarize that the ITD3 algorithm adopts dual critic networks, delayed policy updates, and constraint-aware reward design, all of which are classic mechanisms to improve training stability and generalization ability. The performance gain of our method mainly comes from the cloud–edge–end collaborative architecture, the hybrid optimization mechanism, and the improved TD3 structure, rather than delicate hyperparameter tuning. Even if batch size, penalty coefficient, or exploration noise varies within a reasonable range, the advantages in latency reduction, energy efficiency, and convergence speed can still be maintained.

6. Conclusions

This paper has addressed the fundamental challenges of real-time responsiveness, resource heterogeneity, and operational reliability in intelligent MAL systems for smart airports. The main contributions and findings are summarized as follows.

First, we have proposed a cloud–edge–end collaborative architecture that enables unified resource pooling, elastic scalability, and hierarchical autonomy across all computing layers. Second, this paper proposes a dedicated cloud–edge–end collaborative framework that formulates a joint optimization problem to minimize the total delay and energy consumption of the MAL system, which can maximize long-term system utility under latency, computing, and communication constraints. Third, we have designed a hybrid scheme combining K-medoids clustering and an improved TD3 algorithm which is designed to solve the problem. Training is conducted jointly by the edge and cloud to enable efficient online inference.

Experimental evaluation on a MAL system prototype and extensive simulations have demonstrated the effectiveness of the proposed framework. This work provides a systematic and practically viable solution for the intelligent evolution of airfield lighting systems, offering a clear migration path from legacy centralized systems to distributed intelligent control.

Future research directions include conducting field trials on real airport MAL systems, incorporating machine learning techniques for enhanced adaptability, extending the proposed approach to other smart airport applications, investigating more sophisticated incentive mechanisms, and integrating 5G URLLC capabilities with lightweight security

mechanisms suitable for resource-constrained edge devices. It is also valuable to check the performance and improvement of other deep learning algorithms (such as DDPG, PPO) when combined with the clustering stage.

Author Contributions: Conceptualization, L.J., H.W. and W.H.; methodology, L.J., H.W. and W.H.; soft-ware, L.J. and W.H.; validation, H.W.; formal analysis, F.S.; investigation, F.S.; writing—original draft preparation, L.J., F.S. and W.H.; writing—review and editing, H.W. and W.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Sichuan Science and Technology Program under Grant 2024ZDZX0007 and NSFC under Grant U23B2021 and 62302082.

Data Availability Statement: The data underlying this article cannot be shared publicly due to its proprietary nature data as airport operation data.

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

References

1. Waris, M.; Rauf, S.; Adisasmita, S.A.; Ramli, M.I. The development of airport with the concept of smart airport in supporting strategic areas of national tourism in South Sulawesi Province. *Int. J. Eng. Trends Technol.* **2024**, *72*, 229–242. [CrossRef]
2. Yang, L.; Yin, S.; Han, K.; Haddad, J.; Hu, M. Fundamental diagrams of airport surface traffic: Models and applications. *Transp. Res. Part B Methodol.* **2017**, *106*, 29–51. [CrossRef]
3. Derraa, A.; Ouaaline, N.; Nassiri, B. A new airfield lighting system network architecture. *Int. J. Electr. Comput. Eng.* **2024**, *14*, 3607–3615. [CrossRef]
4. Naung, Z.W.; Purushothaman, M.B.; Ghaffarianhoseini, A. LED airfield lighting: Exploring the synergy of sustainable solutions, safety, and smart technology. *Smart Sustain. Built Environ.* **2026**, *23*, 1–35. [CrossRef]
5. International Civil Aviation Organization (ICAO). *Annex 14—Aerodromes, Volume I—Aerodrome Design and Operations*, 9th ed.; ICAO: Montreal, QC, Canada, 2022.
6. Meier, C.; Jakobi, J.; Adamson, P.; Lozito, S.; Martin, L. Benefits of advanced surface movement guidance and control systems (A-SMGCS). *J. Maz. Univ. Med. Sci.* **2009**, *16*, 169–180. [CrossRef]
7. Wang, B.Y.; Zhang, X.-Q.; Tian, K.; Gao, M. An efficient WSN routing protocol on airfield lighting monitoring system. In *2015 IEEE 5th International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*; IEEE: Piscataway, NJ, USA, 2015.
8. Mijuskovic, A.; Bemthuis, R.; Aldea, A.; Havinga, P. An enterprise architecture based on cloud, fog and edge computing for an airfield lighting management system. In *2020 IEEE 24th International Enterprise Distributed Object Computing Workshop (EDOCW)*; IEEE: Piscataway, NJ, USA, 2020.
9. Liu, X.; Wang, Q.; Zou, C.; Yu, M.; Liao, D. Edge intelligence for smart airport runway: Architectures and enabling technologies. *Comput. Commun.* **2022**, *195*, 323–333. [CrossRef]
10. Ahmed, Z.R.; Askar, S.; Hussein, D.H.; Ibrahim, M.A. Fog computing challenges and opportunities in IoT networks: A review. *Procedia Comput. Sci.* **2025**, *259*, 1749–1764. [CrossRef]
11. Mao, Y.; Zhang, C.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [CrossRef]
12. Zhao, X.S. Design and research of airfield navaid lighting monitoring system based on data mining algorithm. In *International Conference on Innovative Computing*; Springer: Singapore, 2023.
13. Liu, Z.B.; Wang, Q.; Li, H.; Wang, C.Y.; Fei, J.Y. Insulation resistance measurement of airport navigational lighting system based on deep learning and transfer learning. *Math. Probl. Eng.* **2022**, *2022*, 7754356. [CrossRef]
14. Karen Rafferty, K.; Niblock, J.; Peng, J.X. Monitoring the performance of aerodrome ground lighting. *Int. Airpt. Rev.* **2008**, *12*, 60–64.
15. Dumitru, M.C.; Moisescu, M.A.; Pietraru, R.N.; Pop, E. Extending Internet of Things systems with fog and edge computing. In *2024 IEEE International Conference on Engineering, Technology, and Innovation (ICE/ITMC)*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–6.
16. Jiang, L.; Wang, P.F.; Zhang, Y.; Liu, Y.; Chen, H. A Navigation Lighting Control Strategy Selection Method Based on Edge Computing. Chinese Invention Patent CN115599125A, 7 April 2023.
17. Amalia, D.; Soleh, A.M.; Febriansyah, A.; Rizko, R.; Salbiah, S.; Suryan, V.; Septiani, V. Development of airfield lighting system digital learning media: An application usability testing. *JMKSP (J. Manaj. Kepemimp. Dan Supervisi Pendidik.)* **2024**, *9*, 240–255. [CrossRef]

18. Yang, X.; Yang, H.; Li, T.; Ma, X.; Chen, F. Design of single-lamp monitoring system for airfield lighting based on broadband power line carrier communication. *J. Phys. Conf. Ser.* **2021**, *2078*, 012061. [CrossRef]
19. Liang, M.; Zheng, B.; Zhao, R. Alternating iterative methods for solving tensor equations with applications. *Numer. Algorithms* **2019**, *80*, 1437–1465. [CrossRef]
20. Woo, J.H.; Wu, L.; Park, J.B.; Roh, J.H. Real-time optimal power flow using twin delayed deep deterministic policy gradient algorithm. *IEEE Access* **2020**, *8*, 213611–213618. [CrossRef]
21. Wang, T.; Li, Q.; Bucci, D.J.; Liang, Y.; Chen, B.; Varshney, P.K. K-medoids clustering of data sequences with composite distributions. *IEEE Trans. Signal Process.* **2019**, *67*, 2093–2106. [CrossRef]
22. Diederik, P.K.; Jimmy, B. Adam: A Method for Stochastic Optimization. 3rd International Conference for Learning Representations. *arXiv* **2014**, arXiv:1412.6980. [CrossRef]
23. Kong, A. A note on importance sampling using standardized weights. *Univ. Chic. Dept. Stat. Tech. Rep.* **1992**, *348*, 14.
24. Sumiea, E.H.; Abdulkadir, S.J.; Alhussian, H.S.; Al-Selwi, S.M.; Alqushaibi, A.; Ragab, M.G.; Fati, S.M. Deep deterministic policy gradient algorithm: A systematic review. *Heliyon* **2024**, *10*, e30697. [CrossRef] [PubMed]
25. Wang, Y.; He, H.; Tan, X. Truly proximal policy optimization. In *Uncertainty in Artificial Intelligence*; PMLR: Cambridge, MA, USA, 2020; pp. 113–122.

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.