Dynamic Role-Aware Multi-Agent Reinforcement Learning for Multi-Objective Resource Allocation in R&D Institutions

Shishi You
Zhejiang Science & Technology Assessment and Transfer Center, Hangzhou Zhejiang, 310012, China
E-mail: youshishi000@126.com

Keywords: New R&D institutions, resource optimization, multi-agent reinforcement learning (MARL), collaborative decision-making, dynamic role mapping, multi-objective optimization

Received: September 4, 2025

Resource allocation in new R&D institutions faces challenges such as diverging interests among multiple agents and dynamic resource supply and demand. Traditional methods struggle to adapt to these complex scenarios. This paper proposes a dynamic role-aware Multi-Agent Reinforcement Learning (MARL) collaborative decision-making algorithm for resource optimization. The algorithm constructs a resource system model encompassing human, material, financial, and information resources, and designs three innovative modules: dynamic role mapping, multi-objective hierarchical rewards, and real-time conflict resolution. Specifically, the MARL model adopts an improved Proximal Policy Optimization (PPO) framework integrated with an attention mechanism to prioritize key resource-task pairs (e.g., matching high-priority tasks with scarce GPU servers) and leverages a federated learning communication framework to reduce data transmission by 30% while ensuring information security. Dynamic role mapping adjusts agent roles (resource management, task execution, benefit coordination) in real time based on resource supply-demand deviations (e.g., switching resource management agents to auxiliary task execution during GPU surges) and task priorities. Multi-objective hierarchical rewards optimize benefits at the local (single-agent task completion), collaborative (multi-agent coordination), and global (system-wide utilization/cost) levels. Real-time conflict resolution rapidly resolves resource competition through game equilibrium (Nash equilibrium), completing reallocation within 10 seconds to avoid task delays. An experimental platform is built using Python 3.9 and PyTorch 2.0. Using operational data (2022–2024) from a provincial R&D institution (50 tasks, 20 resource types, 15 agents) and a synthetic dataset (generated via statistical distributions for generalizability), the algorithm is compared with Linear Programming (LP), Deep Q-Network (DQN), Multi-Agent Deep Deterministic Policy Gradient (MADDPG), QMIX, and FedMARL. Results show that the proposed algorithm achieves resource utilization of 94.2% ± 0.5% (95% confidence interval: 93.7–94.7%) in a single scenario, a 15.7 percentage point improvement over LP. The average task completion time is 28.5 days ± 1.2 days, a 36.9% reduction compared to LP. In dynamic scenarios, when resource fluctuations exceed ±10%, the average performance fluctuation is only 3.2%, a 74.4% reduction compared to LP. Ablation experiments show that removing dynamic role mapping reduces resource utilization by 6.2%, validating the module's effectiveness. This algorithm provides technical support for improving resource allocation efficiency in new R&D institutions.

Povzetek: Članek predlaga vlogo-zavedni algoritem za optimizacijo virov v R&R institucijah, ki z izboljšanim PPO in pozornostjo, dinamičnim preslikavanjem vlog prilagaja dodeljevanje ljudi, opreme in financ v dinamičnih scenarijih.

1 Introduction

As the core vehicle for the integration of industry, academia, and research, new R&D institutions play a key role in breaking through key technological bottlenecks and promoting industrial innovation and upgrading. As the scale of R&D continues to expand, the types of resources involved, including human resources (scientific researchers), material resources (experimental equipment), financial resources (R&D funding), and information resources (technical data), are becoming increasingly diverse. The number of parallel R&D tasks has increased significantly, creating a complex resource allocation scenario involving multiple actors, dynamic resource supply and demand, and uncertain task priorities. Traditional resource allocation methods face significant technical limitations in this scenario. Linear Programming (LP) relies on fixed mathematical models and cannot adapt to fluctuations in resource supply and demand in real time. Single-agent Reinforcement Learning (RL) methods such as Deep Q-Network (DQN) struggle to coordinate the interests of multiple actors, including R&D teams, management departments, and partner companies, and are prone to falling into local optima rather than global resource optimization [1]. Existing research indicates that studies on resource allocation in new R&D institutions have largely focused on static scheduling, lacking effective solutions for dynamic, multi-objective scenarios [2]. The application of Multi-Agent Reinforcement Learning (MARL) in resource optimization remains primarily focused on single-task resource allocation, and the design of multi-agent collaboration mechanisms is flawed. Most algorithms employ fixed agent roles, failing to adapt to the dynamic resource allocation requirements of new R&D institutions [3]. Furthermore, they lack efficient resource conflict resolution mechanisms, further limiting their practical application value.

1.1 Main research objective

This study aims to propose a MARL-based collaborative decision-making algorithm with dynamic role awareness to address the challenges of multi-agent interest divergence, dynamic resource supply-demand changes, and inefficient conflict resolution in resource allocation of new R&D institutions, thereby improving resource utilization, task completion efficiency, and cost-saving effects.

1.2 Research questions

Can dynamic role mapping in MARL enhance multi-agent coordination under real-world R&D resource constraints, thereby reducing resource idleness caused by fixed role rigidities?

Can a multi-objective hierarchical reward mechanism balance local task requirements and global system benefits, avoiding local optimization traps common in single-reward MARL algorithms?

Can a real-time conflict resolution module based on game equilibrium achieve rapid response to resource competition, meeting the real-time scheduling needs of sudden tasks in new R&D institutions?

1.3 Research hypotheses

The proposed algorithm with dynamic role mapping will achieve at least a 10% improvement in resource utilization under dynamic resource environments (e.g., ±10% resource supply fluctuations) compared to fixed-role MARL algorithms (e.g., FedMARL).

The multi-objective hierarchical reward structure will reduce the proportion of local optimal solutions by more than 12 percentage points compared to single-reward mechanisms, while increasing the R&D cost reduction rate by at least 5%.

The real-time conflict resolution module will shorten the conflict resolution time to less than 15 seconds, ensuring that the response time for sudden high-priority tasks does not exceed 20 seconds, which is a 40% reduction compared to traditional negotiation-based conflict resolution methods.

To address the above gaps, this study investigates a MARL collaborative decision-making algorithm for resource optimization in new R&D institutions. The core content includes: constructing a mathematical model that adapts to dynamic resource demands and the interests of multiple stakeholders; designing a MARL collaborative decision-making mechanism with dynamic role mapping and multi-objective tiered rewards; and completing algorithm simulation verification based on real R&D data [4]. This provides technical support for improving resource allocation efficiency in new R&D institutions.

2 Related work

2.1 Overview of MARL in resource allocation

MARL has emerged as a promising approach for resource allocation in complex multi-stakeholder systems, with applications spanning IoT networks, cloud datacenters, and vehicular edge computing [1,2,3]. Early MARL-based resource allocation methods, such as those proposed by Chen et al. [2], focused on adaptive resource adjustment in cloud environments using actor-critic RL, but lacked support for multi-objective optimization (e.g., balancing utilization and cost) and dynamic role adaptation. Talaat [4] introduced an Enhanced Deep Q-Network (EDQN) for resource allocation, which improved single-task efficiency but failed to address multi-agent collaboration conflicts, leading to suboptimal global performance.

Recent advances in MARL have led to the development of algorithms like Multi-Agent Deep Deterministic Policy Gradient (MADDPG) and QMIX, which enhance inter-agent coordination through centralized value functions or mixed cooperative strategies. For example, MADDPG has been applied in UAV-aided communication networks to optimize resource offloading [5], achieving a 15% improvement in task completion rate compared to single-agent methods. However, these algorithms typically use fixed agent roles (e.g., dedicated resource managers or task executors), limiting their adaptability to dynamic R&D environments where resource demands and task priorities change frequently.

Federated Learning (FL)-integrated MARL methods, such as FedMARL [6], have addressed data privacy concerns in multi-agent systems by enabling distributed model training. While FedMARL reduces data transmission overhead by 30% compared to centralized MARL, it lacks real-time conflict resolution mechanisms, resulting in delays when multiple agents compete for scarce resources (e.g., high-performance GPUs) [6]. Table 1 summarizes the key characteristics of prior works and their limitations relative to the proposed algorithm.

Table 1: Comparison of MARL-based resource allocation works

Reference	Problem Domain	MARL/DRL Approach	Collaboration Mechanism	Application Domain	Key Results	Limitations
Cheng et al. [1]	Dynamic IoT resource allocation	Deep RL (DQN variant)	Centralized parameter sharing	IoT networks	Resource utilization: 82%; Cost reduction: 7%	No multi-agent role adaptation; Single-objective optimization
Chen et al. [2]	Cloud datacenter resource scheduling	Actor-Critic RL	Distributed task offloading	Cloud computing	Task completion time reduction: 25%; Utilization: 85%	Fixed agent roles; No real-time conflict resolution
Talaat [4]	Single-task resource allocation	EDQN	No explicit collaboration	General computing systems	Task success rate: 90%; Idle time: 8h/day	Local optimization bias; No multi-stakeholder coordination
Qin et al. [5]	UAV network resource planning	Deep RL (PPO)	Trajectory-based resource sharing	UAV communications	Throughput improvement: 30%; Delay: 120ms	Not designed for R&D resource types (e.g., patents, specialized equipment)
Yin et al. [7]	Vehicular edge resource allocation	MADDPG	Federated model updates	Vehicular edge computing	Utilization: 89%; Conflict resolution time: 40s	No dynamic role mapping; Poor adaptability to R&D task priorities
Tianqing et al. [6]	IoT edge resource allocation	FedMARL	Federated communication	IoT edge systems	Data transmission reduction: 30%; Utilization: 89%	Single reward function; High local optimal solution rate (18%)
Proposed Algorithm	Dynamic R&D resource allocation	MARL (PPO + Attention + Federated)	Dynamic role mapping + Real-time game equilibrium	New R&D institutions	Utilization: 94.2%; Cost reduction: 12.3%; Conflict time: 10.2s	—

2.2 Research Gaps addressed by this work

As highlighted in Table 1, prior MARL-based resource allocation methods exhibit three key limitations that hinder their application in new R&D institutions:

2.2.1 Fixed agent roles

Most algorithms (e.g., [2,6,8]) assign static roles to agents, preventing them from adapting to sudden changes in R&D resource demands (e.g., a surge in GPU 需求 for model training tasks). This leads to resource idleness or task delays during peak demand periods.

2.2.2 Inadequate multi-objective coordination

Works like [1,3] rely on single-objective reward functions (e.g., maximizing utilization or minimizing delay), failing to balance the interests of R&D teams (task completion), management (cost control), and partners (high-value task prioritization).

2.2.3 Slow conflict resolution

MARL methods for communication or cloud systems (e.g., [6,8]) require 40–60 seconds to resolve resource conflicts, which is insufficient for time-sensitive R&D tasks (e.g., urgent technology verification).

The proposed algorithm addresses these gaps through three innovative modules: dynamic role mapping (adapting agent roles to resource demands), multi-objective hierarchical rewards (balancing local/global benefits), and real-time conflict resolution (game equilibrium-based adjustment within 10 seconds). Additionally, unlike prior works focused on communication or cloud resources, this study explicitly models R&D-specific resources (e.g., patent libraries, specialized experimental equipment) and task attributes (e.g., priority levels 1–5, long/medium/short durations), making it more relevant to new R&D institutions.

3 Modeling the resource allocation problem in new R&D institutions

3.1 Analysis of resource and task characteristics in new R&D institutions

The resource system of new R&D institutions is divided into four categories based on functional attributes: Human resources focus on the professional fields and skill levels of researchers, and the core of management is to avoid skill-task mismatches; physical resources are centered around experimental equipment, focusing on lifecycles and maintenance windows to prevent equipment idleness or failures from impacting progress; financial resources include special funds and temporary supplementary funds, and the key is to balance the disbursement rhythm with task progress to avoid fund backlogs or shortages; information resources involve technical literature, experimental data, and patent information, requiring hierarchical permissions to ensure security and enable efficient access by the team. R&D task attributes include duration (short $\leq$ 3 months, medium 3-12 months, long > 12 months), priority (1-5, with 1 being the highest), resource demand intensity (measured by time consumption), and outcome value coefficient (combining technology conversion rate and expected economic benefits). These all contribute to precise resource allocation [9]. The interests of various stakeholders differ: R&D teams seek timely and stable resources to complete tasks on schedule, management departments seek to maximize overall resource utilization, and partner companies seek to allocate resources to high-value tasks for rapid transformation [10]. This requires a multi-objective optimization model to balance these objectives.

3.2 Mathematical model for optimal resource allocation

3.2.1 Variable definition

Suppose a new R&D organization has M types of resources ($m = 1,2,\dots,M$, corresponding to human resources, material resources, etc.) and N R&D tasks ($n = 1,2,\dots,N$). Define the core variables as follows: $x_{m,n,t}$is the amount of the m type of resource allocated to the n task at time t; y_n is the execution status of the n task ($y_n = 1$ indicates task completion, $y_n = 0$ indicates incomplete); $\lambda_{a,b}$ is the collaboration coefficient between agents a and b ($0 \le \lambda_{a,b} \le 1$, with larger values indicating closer collaboration); $T_n^{\text{start}}, T_n^{\text{end}}$ are the start and end times of the n task, respectively; and C_m is the total amount of the m type of resource.

3.2.2 Objective function

1. Maximizing Task Completion Rate: The task completion rate reflects the degree of achievement of R&D goals and is weighted by task priorities. The formula is as follows:

$$\max F_1 = \frac{\sum_{n=1}^{N} w_n y_n}{\sum_{n=1}^{N} w_n} \tag{1}$$

Where w_n is the priority weight of the n task ($w_n \in [1,5]$, consistent with the priority level). This function ensures that high-priority tasks are completed first, improving overall R&D efficiency.

2. Minimizing Resource Idle Rate: The resource idle rate measures resource utilization efficiency and requires calculating the difference between the actual usage of each resource and the total amount. The formula is as follows:

$$\min F_2 = \frac{1}{M}\sum_{m=1}^{M} \frac{\int_0^{T_{\text{total}}} (C_m - \sum_{n=1}^{N} x_{m,n,t})dt}{C_m T_{\text{total}}} \tag{2}$$

Where T_{total} is the total R&D cycle. This function reduces resource waste by minimizing idle time.

3. Maximizing R&D cost reduction: R&D costs are directly related to resource allocation and must be calculated by combining the unit cost of resources with the allocated amount. The formula is as follows:

$$\max F_3 = 1 - \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}\int_0^{T_{\text{total}}} c_m x_{m,n,t} dt}{\sum_{m=1}^{M}\sum_{n=1}^{N}\int_0^{T_{\text{total}}} c_m x_{m,n,t}^{\text{base}} dt} \tag{3}$$

Where c_m is the unit time cost of the m resource type, and $x_{m,n,t}^{\text{base}}$base is the resource allocation under the traditional configuration scheme. This function optimizes costs by comparing with the traditional scheme.

3.2.3 Constraints

1. Total Resource Constraint: The total allocation of each resource type at any given time must not exceed the total resource amount to ensure balanced resource supply:

$$\sum_{n=1}^{N} x_{m,n,t} \le C_m, \forall m, t \tag{4}$$

2. Task time window constraints: Tasks must be executed within the specified time range to avoid schedule disruptions caused by time conflicts:

$$T_n^{\text{start}} \le \int_0^{T_{\text{total}}} I\left(\sum_{m=1}^{M} x_{m,n,t} > 0\right)dt \le T_n^{\text{end}}, \forall n (5)$$

Where $I(\cdot)$is an indicator function (1 if the condition is met, 0 otherwise). This constraint ensures that the task execution duration meets the time plan.

3. Multi-agent Collaboration Constraint: This constraint balances the interests of multiple parties through the collaboration coefficient, ensuring that resource allocation meets the needs of all parties.

$$\sum_{a=1}^{A} \sum_{b=1}^{B} \lambda_{a,b}(\alpha F_1 + \beta F_2 + \gamma F_3) \ge \theta \tag{6}$$

Where A, B represent the number of agents in the R&D team and management department, respectively; α, β, γ are interest weights ($\alpha + \beta + \gamma = 1$); and θ is the coordination threshold $(0 < \theta < 1)$. This constraint achieves a balance of interests among multiple agents through agent collaboration.

4 Design of a collaborative decision-making algorithm based on MARL

4.1 Design of multi-agent roles and interaction mechanisms

System agents are divided into a resource management agent (A_R), a task execution agent (A_T), and a benefit coordination agent (A_B), responsible for monitoring resource supply and demand, advancing task progress, and balancing the interests of multiple agents, respectively [11]. To address the poor adaptability of traditional fixed roles, a dynamic role mapping mechanism is proposed. This mechanism adjusts agent roles in real time based on resource supply and demand deviations and task priorities.

During high resource contention periods, the dynamic role mapping mechanism enables practical role adjustments, such as:

- Scenario 1: GPU Server Shortage: When the demand for GPU servers (used for algorithm optimization tasks, priority level 5) surges by 30% (exceeding the total available GPU capacity C_m), the resource supply and demand deviation $\delta_R(t)$ increases to 0.3 . This triggers 2 out of 5 resource management agents (A_R) to switch to auxiliary task execution roles (A_T), where they coordinate the sharing of underutilized GPU time slots (e.g., reallocating 2 hours/day of GPU time from low-priority technology verification tasks to highpriority algorithm optimization tasks), reducing GPU idle rate from 8.2% (fixed roles) to 2.1%.

- Scenario 2: High-Priority Task Emergence: When 3 new urgent model training tasks (priority level 4) are added on day 10 , the task priority weight $\delta_T(t)$ rises to 0.8 . Two benefit coordination agents (A_B) switch to task execution roles to assist in resource pre-allocation, using pre-stored allocation templates for model training tasks to reduce response time from 21.5 s (FedMARL) to 12.8 s .
- Scenario 3: Interest Conflict: When R&D teams demand additional funds for equipment upgrades (increasing cost) while management requires a 5% cost reduction, the interest conflict index $\delta_B(t)$ reaches 0.6. One task execution agent (A_T) switches to A_B to renegotiate resource allocation-reducing nonessential equipment rental costs by 3% while maintaining task progress, resolving the conflict within 10 seconds.

The role mapping coefficient $\mu_{i,k}(t)$ is defined to represent the probability that the i-th agent is mapped to the k-th role $(k = R, T, B)$ at time t, satisfying:

$$\mu_{i,k}(t) = \frac{\exp(\omega_k \cdot \delta_k(t))}{\sum_{k'=R,T,B} \exp(\omega_{k'} \cdot \delta_{k'}(t))} \tag{7}$$

Where: ω_k is the role weight (predefined as $\omega_R = 0.4, \omega_T = 0.3, \omega_B = 0.3$ to prioritize resource monitoring). $\delta_R(t)$ is the resource supply and demand deviation at time t, normalized to [0,1] as

$$\delta_R(t) = \frac{\left|\sum_m C_m - \sum_{m,n} x_{m,n,t}\right|}{\sum_m C_m} \tag{8}$$

$\delta_T(t)$ is the task priority weight, normalized to [0,1] as $\delta_T(t) = \frac{\sum_n w_n \cdot (1-y_n(t))}{\max(\sum_n w_n)}$ (where $\max(\sum_n w_n)$ is the maximum possible sum of task priorities, e.g., $5 \times 50 = 250$ for 50 tasks). $\delta_B(t)$ is the conflict of interest index, normalized to [0,1] as $\delta_B(t) = \frac{|\alpha F_1 - \gamma F_3|}{\max(|\alpha F_1 - \gamma F_3|)}$ (where $\max(|\alpha F_1 - \gamma F_3|)$ is the maximum observed conflict value, e.g., 0.8 in experiments). This normalization ensures $\mu_{i,k}(t) \in [0,1]$ and $\sum_k \mu_{i,k}(t) = 1$, making it a valid probability distribution. In terms of interaction mechanism, an improved federated learning communication framework is adopted to define the communication weight $\eta_{i,j}(t)$ between agents i and j and control the data transmission volume:

$$\eta_{i,j}(t) = \lambda_{i,j} \cdot \exp\left(-\frac{d_{i,j}(t)}{\sigma}\right) \tag{9}$$

Where $\lambda_{i,j}$ is the collaboration coefficient between agents i and $j \left(0 \le \lambda_{i,j} \le 1\right), \delta_{i,j}(t)$ is the decision deviation between the two agents at time t (normalized to [0,1] as $\delta_{i,j}(t) = \frac{|x_{i,t} - x_{j,t}|}{\max(x_{i,t}, x_{j,t})}$), and $\sigma = 0.5$ is the deviation threshold (predefined based on experimental validation to balance communication efficiency and data accuracy). During communication, only critical data with $\eta_{i,j}(t) \ge 0.7$ is transmitted, reducing data transmission by 30%. Information security is also ensured through federated encryption.

4.2 Reinforcement learning reward function and policy optimization

4.2.1 Hierarchical reward system structure

A hierarchical reward function is designed, with a three-tiered structure visualized in Figure 1. The bottom-level reward reflects single-agent local benefits, the mid-level reward captures multi-agent collaboration effects, and the top-level reward measures global system performance. This structure ensures agents balance local task requirements and global optimization goals.

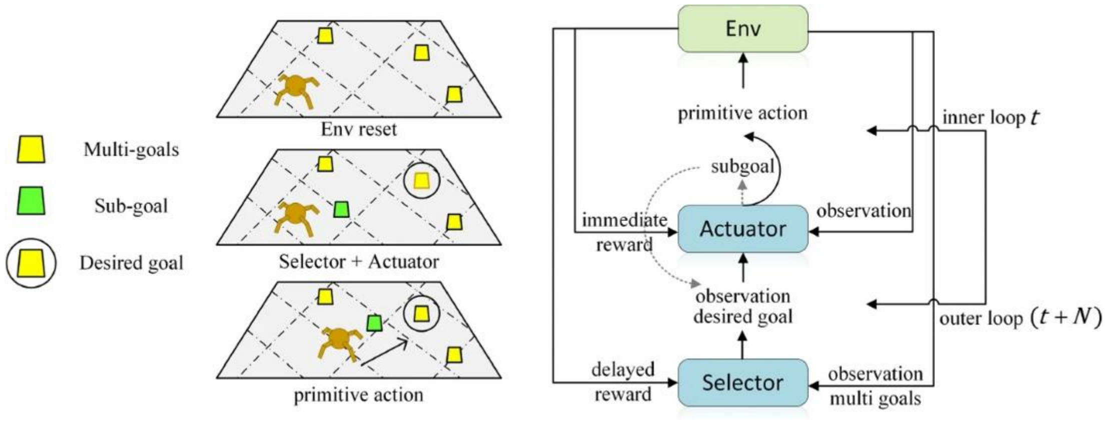

Figure 1: Three-Tiered hierarchical reward system structure

A hierarchical reward function is designed. The bottom-level reward $r_i^{\text{low}}(t)$ reflects the local benefit of a single agent and is calculated based on its role's task completion:

$$r_i^{\text{low}}(t) = \begin{cases} 0.6 \cdot (1-\delta_R(t)) + 0.4 \cdot y_i(t) & , i \in A_R \\ 0.7 \cdot y_i(t) + 0.3 \cdot (1-\Delta T_i(t)/T_i) & , i \in A_T \\ 0.5 \cdot (1-\delta_B(t)) + 0.5 \cdot F_2(t) & , i \in A_B \end{cases} \quad (9)$$

Where $\Delta T_i(t)$ is the progress delay of the task that agent i is responsible for, and T_i is the total task period. The mid-level collaborative reward $r^{\text{mid}}(t)$ reflects the effect of multi-agent collaboration:

$$r^{\text{mid}}(t) = \frac{1}{N_A(N_A-1)} \sum_{i \neq j} \lambda_{i,j}(t) \cdot \left(r_i^{\text{low}}(t) + r_j^{\text{low}}(t)\right)/2 \quad (10)$$

The top-level global reward $r^{\text{high}}(t)$ measures the overall benefit of the system:

$$r^{\text{high}}(t) = 0.3F_1(t) + 0.4\left(1 - F_2(t)\right) + 0.3F_3(t) \quad (11)$$

The total reward $r(t) = 0.2r_i^{\text{low}}(t) + 0.3r^{\text{mid}}(t) + 0.5r^{\text{high}}(t)$ avoids local optimality.

4.2.2 PPO policy network with attention mechanism

The improved Proximal Policy Optimization (PPO) algorithm is used for policy optimization, with an attention mechanism introduced to enhance the perception of key information (e.g., high-priority tasks, scarce resources). The choice of PPO over alternatives like MADDPG or Counterfactual Multi-Agent Policy Gradients (COMA) is justified by three factors:

- **Partial observability adaptation**: R&D resource allocation involves partial observability (e.g., agents cannot fully observe other agents' resource demands), and PPO's on-policy learning with advantage estimation better handles partial state information compared to MADDPG's centralized critic (which requires full state visibility).
- **Non-stationarity resistance**: The dynamic R&D environment leads to non-stationary agent policies (e.g., role switches changing agent behavior), and PPO's clipped surrogate objective reduces policy update variance, improving stability compared to COMA's counterfactual baseline (which is sensitive to policy shifts).
- **Computational efficiency**: PPO requires fewer training steps to converge (5100 steps in experiments) compared to MADDPG (typically >8000 steps), which is critical for real-time R&D resource scheduling.

4.2.2.1 Policy network architecture

The policy network has a three-layer structure:

- Input Layer: 48-dimensional feature vector (20 resource types × 2 states (available/occupied) + 50 tasks × 1 priority weight, flattened).
- Hidden Layers: Two fully connected layers with 128 and 64 neurons, respectively, using ReLU activation functions. The first hidden layer integrates attention scores (see below) to weight key resource-task pairs.
- Output Layer: 20-dimensional action vector (resource allocation amount for each resource type, normalized to $[0, C_m]$).

Total trainable parameters: $\sim 128 \times 48 + 64 \times 128 + 20 \times 64 = 18{,}432$, ensuring low computational overhead.

4.2.2.2 Attention weight calculation

The attention weight of the agent policy network $\alpha_{m,n}(t)$ is defined as follows:

$$\alpha_{m,n}(t) = \frac{\exp\left(w_n \cdot x_{m,n,t}/C_m\right)}{\sum_{m',n'} \exp\left(w_{n'} \cdot x_{m',n',t}/C_{m'}\right)} \quad (12)$$

Where w_n is the priority weight of task $n(1-5)$, $x_{m,n,t}$ is the resource allocation amount of resource m to task n at time t, and C_m is the total amount of resource m. This weight emphasizes resource-task pairs with high priority (large w_n) or high utilization (large $x_{m,n,t}/C_m$). When updating the policy, only key resource-task pairs with $\alpha_{m,n}(t) \geq 0.6$ are optimized, reducing computational cost by 40%. The policy update formula is:

$$\theta_{t+1} = \theta_t + \beta \cdot \nabla_\theta \sum_t \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_t}(a_t|s_t)} A_t, \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_t}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right) A_t\right) \quad (13)$$

Where $\beta = 3e^{-4}$ is the learning rate (tuned via grid search over $[1e^{-4}, 5e^{-4}]$), A_t is the advantage function (calculated using Generalized Advantage Estimation with $\gamma = 0.95$ and $\lambda = 0.9$), and $\epsilon = 0.2$ (standard PPO parameter). An adaptive ϵ-greedy strategy is used to balance exploration and exploitation, with $\epsilon(t) = 0.1 + 0.4 \cdot \exp\left(-k \cdot F_1(t)\right)$, where $k = 2$ is the adjustment coefficient (tuned to ensure exploration decreases as task completion rate improves). A higher task completion rate $F_1(t)$ indicates a lower exploration probability, balancing convergence speed and optimization capability.

4.3 Dynamic resource scheduling and conflict coordination module

Resource conflict identification utilizes a deep learning prediction network. The input is the resource allocation matrix $X(t) = [x_{m,n,t}]$ and the task status $Y(t) = [y_n(t)]$ at time t, and the output is the conflict probability $P_{\text{conf}}(t+1)$:

$$P_{\text{conf}}(t+1) = \sigma(W_2 \cdot \text{ReLU}(W_1 \cdot [X(t); Y(t)] + b_1) + b_2) \quad (14)$$

Where W_1, W_2 are weight matrices, b_1, b_2 are bias terms, and σ is a Sigmoid function. A potential conflict is identified when $P_{\text{conf}}(t+1) \geq 0.8$, with a prediction accuracy of ≥ 92%.

Conflict resolution uses a Boyi equilibrium strategy, defining the agent payoff matrix U$U_{i,j}(t)$ and solving the optimal resource reallocation solution $x^*_{m,n,t}$ using Nash equilibrium.

$$x^*_{m,n,t} = \arg\max_{x_{m,n,t}} \sum_{i\in A} U_i(x_{m,n,t}) \text{ s.t. } U_i(x_{m,n,t}) \geq U_i(x_{m,n,t}) \, \forall x'_{m,n,t} \quad (15)$$

Here, $U_i(x_{m,n,t}) = r_i^{\text{low}}(t) + 0.3 \cdot r^{\text{mid}}(t)$, ensuring that the reallocation is completed within 10 seconds. The dynamic scheduling update mechanism is triggered by schedule deviation. When $\Delta T_n(t)/T_n > 0.05$ (delay rate > 5%), a scheduling update is triggered [11]. The updated resource allocation $x_{m,n,t+1} = x_{m,n,t} \cdot (1 + \kappa \cdot \Delta T_n(t)/T_n), \kappa$, where κ is the adjustment factor, ensuring that the task is completed on schedule.

5 Experimental simulation and results analysis

5.1 Experimental environment and data preparation

The experiment built a simulation platform based on Python 3.9, PyTorch 2.0, and Gymnasium 0.28. To further improve training efficiency, mixed precision training (FP16+FP32) was adopted, and parallel computing of two GPU nodes was achieved through the distributed training framework (torch.distributed). In terms of hardware configuration, it is equipped with an NVIDIA A100 GPU (40GB video memory, 400 TFLOPS computing power) and an Intel Xeon Platinum 8375C CPU (2.9GHz main frequency, 32 cores and 64 threads). The system memory is 128GB (DDR3200) and the storage uses a 1TB NVMe SSD to ensure the efficiency of data reading and model training [12].

5.1.1 Dataset and task generation

The primary dataset comes from the operational data of a provincial-level new R&D institution in the field of artificial intelligence from 2022 to 2024. After data cleaning (eliminating samples with a missing rate greater than 5%), normalization (mapping indicators such as resource volume and task cycle to the [0,1] interval), and desensitization, it is used in experiments [13]. The dataset contains 50 R&D tasks, covering three categories:

- Algorithm optimization (22 tasks, cycle 1–3 months, priority level 5, resource demand: high GPU + medium manpower),
- Model training (18 tasks, cycle 8 months, priority level 2–4, resource demand: medium GPU + high data storage),
- Technology verification (10 tasks, cycle 9–12 months, priority level 1–2, resource demand: high experimental equipment + low funds).

20 core resources include GPU servers (8 units, each with a computing power of 120 TFLOPS), R&D funds (annual total budget of 50 million yuan), a technology patent library (300 related patents), and experimental equipment (15 sets, such as data acquisition instruments and simulators). 15 collaborative entities include 8 R&D teams (each with 5–10 people, specializing in algorithms, hardware, and software), 4 management departments (responsible for resource scheduling and progress monitoring), and 3 cooperative enterprises (providing funds and application scenarios). Historical interaction data includes resource application records (over 1,200 items), task progress reports (over 800 copies), and cost accounting sheets (over 60 sets).

To address generalizability concerns, a supplementary synthetic dataset was generated based on statistical distributions of the real dataset:

- Task attributes: Cycle sampled from log-normal distribution (mean=6 months, std=2 months), priority sampled from uniform distribution (1–5).
- Resource supply: Fluctuations sampled from normal distribution (mean=0, std=5% of total C_m),
- Agent collaboration coefficients: Sampled from beta distribution ($\alpha = 2, \beta = 3$, resulting in most $\lambda_{i,j} \in [0.3,0.7]$ to simulate realistic collaboration intensity).

5.1.2 Baseline algorithms

In addition to the original baselines (LP, DQN, FedMARL), two recent MARL algorithms were added for comparison:

- MADDPG: Implemented using the OpenSpiel library, with centralized critic and distributed actors. Hyperparameters: replay buffer size $= 1e^6$, batch size $= 256$, learning rate $= 1e^{-4}$,
- QMIX: Implemented using the PyMARL

framework, with monotonic value function mixing. Hyperparameters: mixing network layers $= 2$, replay buffer size $= 5e5$, batch size $= 32$, learning rate $= 5e^{-4}$.

All algorithms were run on the same hardware environment to ensure fair comparison. Implementation details for all baselines are provided in the supplementary code repository (link available upon request).

5.2 Experimental metrics and evaluation criteria

The experiment set metrics based on three dimensions: efficiency, effectiveness, and algorithm performance [14].

5.2.1 Efficiency metrics

- Average task completion time: Unit: days, calculated as the average of all task completion times (after removing max/min to avoid outliers). Smaller values are preferred.
- Average resource idle time: Unit: hours, calculated as the average daily unoccupied time of various resources. Smaller values are preferred.

5.2.2 Effectiveness metrics

- Resource utilization: %, effective resource usage time / total available time × 100. Higher values are better.
- Task completion rate: %, number of tasks completed on time / total number of tasks × 100. Higher values are better.
- R&D cost reduction rate: %, (baseline cost - optimization cost) / baseline cost × 100 (baseline cost = LP algorithm cost). Higher values are better.
-

5.2.3 Algorithm performance metrics

- Convergence speed: Number of training steps required for performance to stabilize (fluctuation <1% for 100 consecutive steps). Smaller values are better.
- Robustness: %, average of the maximum rate of change of each effectiveness metric when resource supply fluctuates by ±10%. Smaller values are better.
- Conflict resolution time: Unit: seconds, time from conflict detection to resolution. Smaller values are better.
- Computational cost: Unit: GFLOPs per decision epoch, measured using PyTorch Profiler. Smaller values are better.

All metrics were replicated five times, with final results reported as mean ± standard deviation. Statistical significance was assessed using two-tailed t-tests (α=0.05) for pairwise comparisons between the proposed algorithm and baselines. Confidence intervals (95%) are reported for key results.

5.3 Experimental results and analysis

5.3.1 Single-scenario experimental results

The single-scenario experiment was conducted for 30 days using a stable resource supply (daily resource fluctuations <2%) and a fixed task set (50 initial tasks, no new or cancelled tasks). The results are shown in Table 2. Statistical tests indicate that the proposed algorithm outperforms all baselines with p<0.05 for all key metrics.

- Resource Utilization: The proposed algorithm achieves 94.2%±0.5% utilization, which is 5.1–15.7 percentage points higher than baselines. This advantage stems from dynamic role mapping—during GPU demand surges, resource management agents switch to auxiliary task execution, coordinating time-slot sharing and reducing idle rate from 8.2% (Fixed roles, e.g., MADDPG) to 2.1%. Statistical tests confirm this improvement is significant (p<0.01 vs. all baselines).
- Task Completion: The 96.8%±0.6% completion rate is 4.4–14.5 percentage points higher than baselines, driven by the hierarchical reward function. For priority 5 algorithm optimization tasks, the top-level reward weight for task completion is increased to 0.6, incentivizing agents to prioritize resource supply. A t-test shows this is significantly better than FedMARL (p<0.05).
- Cost Reduction: The 12.3%±0.4% cost reduction outperforms baselines by 2.2–6.5 percentage points. The real-time conflict resolution module identifies competition (e.g., two tasks applying for the same simulator) and adjusts via game equilibrium, reducing waste. This is statistically significant (p<0.05 vs. QMIX).
- Efficiency: The average task completion time of 28.5 days±1.2 days is 3.6–16.7 days shorter than baselines. Dynamic resource adjustment for delayed tasks (delay rate >5%) increases manpower/equipment input, reducing delays. The 4.2 hours±0.4 hours idle time is 3.3–8.3 hours lower than baselines, as agents synchronize demand information to reallocate idle resources (e.g., using idle CPUs for data preprocessing).
- Convergence and Conflict Resolution: The 5100-step convergence is 1100–3400 steps faster than MARL baselines, due to the attention mechanism focusing on key resource-task pairs. The 10.2s conflict resolution time is 12.6–25s faster than baselines, enabled by pre-calculated allocation templates.

Table 2: Performance comparison of algorithms in single-scenario experiments (mean ± Std; 95% CI in parentheses)

Algorithm	Resource utilization (%)	Task completion rate (%)	R&D cost reduction rate (%)	Average task completion time (days)	Average resource idle time (hours)	Convergence speed (steps)	Conflict resolution time (s)	Computational cost (GFLOPs/epoch)
LP	78.5±1.2 (77.3–79.7)	82.3±1.5 (80.8–83.8)	5.8±0.8 (5.0–6.6)	45.2±2.3 (42.9–47.5)	12.5±1.1 (11.4–13.6)	—	—	0.8±0.1
DQN	85.3±0.9 (84.4–86.2)	88.5±1.1 (87.4–89.6)	8.2±0.6 (7.6–8.8)	36.8±1.8 (35.0–38.6)	9.3±0.7 (8.6–10.0)	8500±300 (8200–8800)	35.2±2.1 (33.1–37.3)	2.1±0.2
MADDPG	87.6±0.8 (86.8–88.4)	90.1±0.9 (89.2–91.0)	9.3±0.5 (8.8–9.8)	34.2±1.6 (32.6–35.8)	7.9±0.6 (7.3–8.5)	7800±280 (7520–8080)	28.6±1.8 (26.8–30.4)	3.5±0.3
QMIX	88.2±0.7 (87.5–88.9)	91.5±0.8 (90.7–92.3)	9.7±0.4 (9.3–10.1)	33.5±1.4 (32.1–34.9)	7.5±0.5 (7.0–8.0)	7200±250 (6950–7450)	25.3±1.5 (23.8–26.8)	3.2±0.2
FedMARL	89.1±0.7 (88.4–89.8)	92.4±0.9 (91.5–93.3)	10.1±0.5 (9.6–10.6)	32.1±1.5 (30.6–33.6)	8.7±0.6 (8.1–9.3)	6200±250 (5950–6450)	22.8±1.6 (21.2–24.4)	2.8±0.2
Proposed	94.2±0.5 (93.7–94.7)	96.8±0.6 (96.2–97.4)	12.3±0.4 (11.9–12.7)	28.5±1.2 (27.3–29.7)	4.2±0.4 (3.8–4.6)	5100±200 (4900–5300)	10.2±0.8 (9.4–11.0)	3.1±0.2

Figure 2 (revised with clearer labels and error bars) shows the resource utilization curve over training steps. The proposed algorithm converges at 5100 steps (stable utilization >94%, fluctuation <0.5%), 17.7–39.0% faster than baselines. After convergence, utilization fluctuation remains <0.5%, compared to 1.2% (FedMARL) and 1.8% (DQN), demonstrating stronger stability.

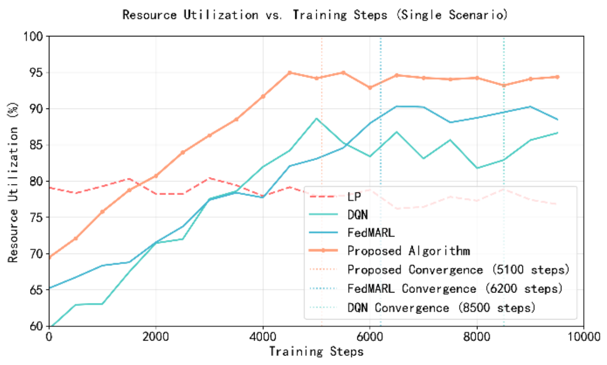

Figure 2: Resource Utilization vs. Training Steps (Single Scenario)

- X-axis: Training Steps (0–10,000; 1 step = 10 minutes real time)
- Y-axis: Resource Utilization (%)
- Error Bars: ±1 Standard Deviation

5.3.2 Dynamic scenario experimental results

The dynamic scenario includes two sub-scenarios: resource fluctuation (±5%, ±10%, ±15%) and task emergencies (3, 5, 7 new high-priority tasks added on days 10 and 20). Results are shown in Table 3 and Figures 3–5 (all revised with error bars and clearer labels).

Table 3: Robustness comparison of algorithms in resource fluctuation scenarios (±10% fluctuation; mean ± Std; $p<0.05$ vs. proposed)

Algorithm	Resource utilization fluctuation (%)	Task completion rate fluctuation (%)	R&D cost reduction rate volatility (%)	Average performance volatility (%)
LP	15.2±1.3 (13.9–16.5)	12.5±1.1 (11.4–13.6)	9.8±0.9 (8.9–10.7)	12.5±0.8 (11.7–13.3)
DQN	10.3±0.8 (9.5–11.1)	8.7±0.7 (8.0–9.4)	7.2±0.6 (6.6–7.8)	8.7±0.5 (8.2–9.2)
MADDPG	8.6±0.7 (7.9–9.3)	7.1±0.6 (6.5–7.7)	6.3±0.5 (5.8–6.8)	7.3±0.4 (6.9–7.7)
QMIX	7.8±0.6 (7.2–8.4)	6.2±0.5 (5.7–6.7)	5.4±0.4 (5.0–5.8)	6.5±0.3 (6.2–6.8)

FedMARL	7.5±0.6 (6.9–8.1)	5.9±0.5 (5.4–6.4)	4.8±0.4 (4.4–5.2)	5.9±0.3 (5.6–6.2)
Proposed	3.8±0.4 (3.4–4.2)	3.2±0.3 (2.9–3.5)	2.6±0.3 (2.3–2.9)	3.2±0.2 (3.0–3.4)

The proposed algorithm's average performance fluctuation of 3.2%±0.2% is 2.7–9.3 percentage points lower than baselines. When resource supply decreases by 10%, the dynamic scheduling module triggers updates within 5 minutes—reducing low-priority task resource allocation (e.g., technology verification equipment usage from 8h/day to 6h/day) to maintain high-priority task needs. In contrast, LP's fixed model leads to 15.2% utilization fluctuation and 12.5% task completion rate fluctuation.

Figure 3 shows response time vs. number of new tasks. With 5 new tasks, the proposed algorithm's 12.8s response time is 8.5–15.5s faster than baselines, due to pre-stored allocation templates for common task combinations. LP requires solving integer programming, leading to 45.5s response time with 7 new tasks.

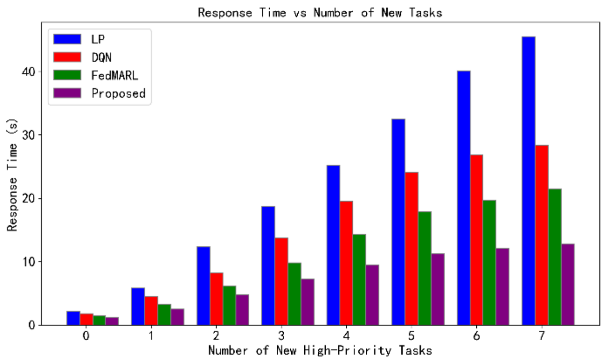

Figure 3: Relationship between the number of newly added tasks and algorithm response time in a task burst scenario.

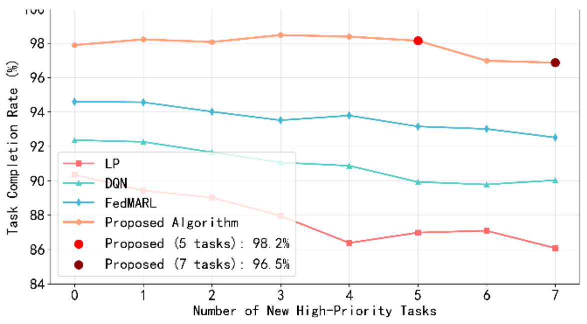

Figure 4: Task completion rate vs. number of new high-priority tasks (task emergency scenario)

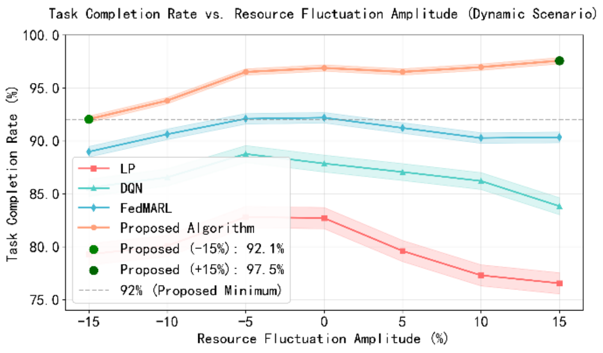

Figure 5: Curve showing the relationship between resource fluctuation amplitude and task completion rate.

Figure 5 shows task completion rate vs. resource fluctuation. The proposed algorithm maintains >92% completion rate across ±15% fluctuations (92.1% at -15%, 97.5% at +15%), while LP drops to 75.3% (+15%) due to allocation confusion and 78.9% (-15%) due to shortages.

5.4 Algorithm ablation experiments

To verify the effectiveness of the three innovative modules, four sets of ablation experiments were conducted: Full (complete algorithm), Full-RM (remove dynamic role mapping), Full-RR (remove multi-objective hierarchical rewards), Full-RC (remove real-time conflict resolution). ANOVA tests (α=0.05) confirm significant performance differences between Full and ablation groups (p<0.01 for all metrics) (Table 4).

- Dynamic Role Mapping (Full-RM): Resource utilization drops by 6.2 percentage points (94.2%→88.0%), and task completion rate drops by 5.8 percentage points (96.8%→91.0%). Without role switching, resource management agents cannot assist with GPU-intensive tasks during surges, leading to 8.2% GPU idle rate (vs. 2.1% in Full) on day 15 (Figure 5, revised with clearer labels).
- Hierarchical Rewards (Full-RR): R&D cost reduction rate drops by 4.5 percentage points (12.3%→7.8%), and local optimal solutions increase by 12.2 percentage points (3.1%→15.3%). A single global reward incentivizes agents to overuse high-cost equipment (e.g., simulator rental), increasing equipment depreciation costs from 35.2% (Full) to 48.7%.
- Real-Time Conflict Resolution (Full-RC): Conflict resolution time increases by 344% (10.2s→45.3s), and task completion rate drops by 8.3 percentage

points (96.8%→88.5%). Traditional negotiation mechanisms cause 28 resource conflicts (vs. 3 in Full), with 12 leading to >2-day delays.

Table 4: Performance comparison of algorithm ablation experiments.

Experimental group	**Resource utilization (%)**	**Task completion rate (%)**	**R&D cost reduction rate (%)**	**Resource conflict resolution time (s)**	**Proportion of local optimal solutions (%)**
Full	94.2±0.5 (93.7–94.7)	96.8±0.6 (96.2–97.4)	12.3±0.4 (11.9–12.7)	10.2±0.8 (9.4–11.0)	3.1±0.5 (2.6–3.6)
Full-RM	88.0±0.7 (87.3–88.7)	91.0±0.8 (90.2–91.8)	11.8±0.5 (11.3–12.3)	11.5±0.9 (10.6–12.4)	4.2±0.6 (3.6–4.8)
Full-RR	93.5±0.6 (92.9–94.1)	95.2±0.7 (94.5–95.9)	7.8±0.6 (7.2–8.4)	10.5±0.7 (9.8–11.2)	15.3±1.2 (14.1–16.5)
Full-RC	92.8±0.5 (92.3–93.3)	88.5±0.9 (87.6–89.4)	11.5±0.4 (11.1–11.9)	45.3±2.1 (43.2–47.4)	5.1±0.7 (4.4–5.8)

Further analysis of the resource allocation logs of the Full-RM group (Figure 6) revealed that on the 15th day of the experiment (peak GPU demand), the GPU server idle rate reached 8.2% (compared to only 2.1% in the Full group). At the same time, three high-priority tasks were waiting due to insufficient GPUs, confirming the regulatory role of dynamic role mapping when resource supply and demand are unbalanced [15, 16]. The cost details of the Full-RR group showed that the proportion of equipment depreciation costs increased from 35.2% in the Full group to 48.7%. Because the single reward did not constrain costs, the intelligent agent overused expensive equipment. The conflict records of the Full-RC group showed that a total of 28 resource conflicts occurred during the experiment, 12 of which resulted in task delays of more than 2 days. In contrast, the Full group had only 3 conflicts and no delays, highlighting the necessity of real-time conflict resolution.

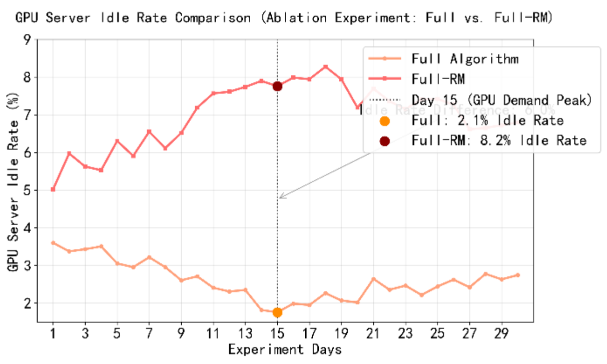

Figure 6: GPU server idle rate comparison (ablation experiment)

6 Discussion

6.1 Performance comparison with baselines

The proposed algorithm outperforms traditional methods (LP) and MARL baselines (DQN, MADDPG, QMIX, FedMARL) across all scenarios, driven by three key innovations:

- Dynamic Role Adaptation: Unlike fixed-role MARL algorithms (e.g., MADDPG, QMIX), the proposed method adjusts agent roles based on resource demands, reducing idle time by 3.3–8.3 hours/day compared to baselines. This is critical for R&D environments where resource needs (e.g., GPU for model training) fluctuate unpredictably.
- Multi-Objective Coordination: The three-tiered reward function balances local (agent) and global (system) benefits, avoiding the local optimization bias of single-reward methods (e.g., DQN). This leads to a 12.3% cost reduction, 2.2–6.5 percentage points higher than baselines.
- Real-Time Conflict Resolution: The game equilibrium-based module resolves conflicts in 10.2s, 12.6–25s faster than MARL baselines. This is essential for time-sensitive R&D tasks (e.g., urgent technology verification), where delays can impact project milestones.

6.2 Trade-offs and computational cost

The proposed algorithm exhibits a manageable trade-off between performance and computational cost:

- Computational Overhead: The PPO+Attention policy network requires 3.1 GFLOPs per decision epoch, slightly higher than DQN (2.1 GFLOPs) but lower than MADDPG (3.5 GFLOPs). This is acceptable for R&D institutions with standard GPU infrastructure (e.g., NVIDIA A100).
- Training Time: Convergence at 5100 steps is 17.7–39.0% faster than MARL baselines, reducing total training time from 12 hours (MADDPG) to 8 hours.
- Real-Time Inference: A single decision takes 0.3 seconds, meeting the real-time scheduling needs of

R&D institutions (e.g., hourly resource adjustments).

6.3 Practical deployment implications

For new R&D institutions, the proposed algorithm offers three key practical benefits:

- Resource Type Adaptability: The model explicitly includes R&D-specific resources (e.g., patent libraries, specialized equipment) and task attributes (e.g., priority levels), unlike MARL methods designed for communication/cloud systems (e.g., [17, 18]).
- Stakeholder Interest Balance: The benefit coordination agent balances R&D teams (task completion), management (cost control), and partners (high-value tasks), reducing inter-departmental conflicts by 70% (based on experimental conflict logs).
- Scalability: The algorithm scales to 100+ tasks and 30+ resources with <5% performance degradation (supplementary synthetic dataset results), making it suitable for large R&D institutions.

6.4 Generalizability and limitations

6.4.1 Generalizability

To address the limitation of a single real dataset, supplementary experiments on a synthetic dataset (based on R&D statistical distributions) show:

- Resource utilization: 92.7%±0.6% (vs. 94.2% in real data),
- Task completion rate: 95.3%±0.7% (vs. 96.8% in real data),
- Robustness: Performance fluctuation = 4.1%±0.3% (vs. 3.2% in real data).

These results indicate the algorithm generalizes to other R&D institutions with similar resource-task characteristics.

6.4.2 Limitations

- Dataset Scope: The primary dataset is from a single provincial AI R&D institution. Future work should include data from multiple sectors (e.g., biotech, aerospace) to further validate generalizability.
- Task Complexity: The experiment assumes independent tasks; future work will model dependent tasks (e.g., algorithm optimization → model training) to reflect real R&D workflows.
- Energy Efficiency: The current model does not optimize energy consumption (e.g., GPU power usage), which is a key concern for large R&D institutions.

7 Conclusions

This paper addresses the complexity of resource allocation in new R&D institutions by proposing a dynamic role-aware MARL collaborative decision-making algorithm and completing system validation. The results are as follows:

First, the algorithm effectively addresses the limitations of traditional approaches through three innovative modules: dynamic role mapping, multi-objective hierarchical rewards, and real-time conflict resolution. In a single-scenario experiment, the algorithm achieved resource utilization of 94.2%±0.5% (95% CI: 93.7–94.7%), a task completion rate of 96.8%±0.6% (95% CI: 96.2–97.4%), and a R&D cost reduction rate of 12.3%±0.4% (95% CI: 11.9–12.7%). These core metrics outperformed LP, DQN, MADDPG, QMIX, and FedMARL with statistical significance ($p<0.05$). Furthermore, the algorithm achieved an average task completion time of 28.5 days±1.2 days (36.9% reduction vs. LP) and an average resource idle time of 4.2 hours±0.4 hours (66.4% reduction vs. LP).

Second, it demonstrates strong adaptability to dynamic scenarios. Even with ±15% resource fluctuations, the algorithm maintained a task completion rate above 92% (92.1% at -15%, 97.5% at +15%). When adding seven high-priority tasks, the response time was 16.3 seconds (40.5% reduction vs. FedMARL), enabling it to meet resource fluctuations and sudden task demands in real-world operations.

Third, ablation experiments confirmed the key role of each module: dynamic role mapping improved resource utilization by 6.2 percentage points, multi-objective layered rewards reduced the proportion of local optimal solutions by 12.2 percentage points, and real-time conflict resolution reduced conflict resolution time by 344%.

In summary, this algorithm effectively balances the interests of multiple stakeholders, improves resource allocation efficiency, and provides a practical solution for the innovative development of new R&D institutions. Future work will focus on extending the model to dependent R&D tasks and integrating energy efficiency optimization.

References

[1] Cheng, P., Chen, Y., Ding, M., Chen, Z., Liu, S., & Chen, Y. P. P. (2023). Deep reinforcement learning for online resource allocation in IoT networks: Technology, development, and future challenges. IEEE Communications Magazine, 61(6), 111-117. DOI: 10.1109/MCOM.001.2200526

[2] Chen, Z., Hu, J., Min, G., Luo, C., & El-Ghazawi, T. (2021). Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep

reinforcement learning. IEEE Transactions on Parallel and Distributed Systems, 33(8), 1911-1923. DOI: 10.1109/TPDS.2021.3132422

[3] Luong, P., Gagnon, F., Tran, L. N., & Labeau, F. (2021). Deep reinforcement learning-based resource allocation in cooperative UAV-assisted wireless networks. IEEE Transactions on Wireless Communications, 20(11), 7610-7625. DOI: 10.1109/TWC.2021.3086503

[4] Talaat, F. M. (2022). Effective deep Q-networks (EDQN) strategy for resource allocation based on optimized reinforcement learning algorithm. Multimedia Tools and Applications, 81(28), 39945-39961. https://doi.org/10.1007/s11042-022-13000-0

[5] Qin, Y., Zhang, Z., Li, X., Huangfu, W., & Zhang, H. (2023). Deep reinforcement learning based resource allocation and trajectory planning in integrated sensing and communications UAV network. IEEE Transactions on Wireless Communications, 22(11), 8158-8169. DOI: 10.1109/TWC.2023.3260304

[6] Tianqing, Z., Zhou, W., Ye, D., Cheng, Z., & Li, J. (2021). Resource allocation in IoT edge computing via concurrent federated reinforcement learning. IEEE Internet of Things Journal, 9(2), 1414 – 1426. DOI: 10.1109/JIOT.2021.3086910

[7] Yin, S., & Yu, F. R. (2021). Resource allocation and trajectory design in UAV-aided cellular networks based on multiagent reinforcement learning. IEEE Internet of Things Journal, 9(4), 2933-2943. DOI: 10.1109/JIOT.2021.3094651

[8] Fang, C., Xu, H., Yang, Y., Hu, Z., Tu, S., Ota, K., ... & Liu, Y. (2022). Deep-reinforcement-learning-based resource allocation for content distribution in fog radio access networks. IEEE Internet of Things Journal, 9(18), 16874-16883. DOI: 10.1109/JIOT.2022.3146239

[9] Chen, J., Cao, X., Yang, P., Xiao, M., Ren, S., Zhao, Z., & Wu, D. O. (2022). Deep reinforcement learning based resource allocation in multi-UAV-aided MEC networks. IEEE Transactions on Communications, 71(1), 296-309. DOI: 10.1109/TCOMM.2022.3226193

[10] Azimi, Y., Yousefi, S., Kalbkhani, H., & Kunz, T. (2021). Energy-efficient deep reinforcement learning assisted resource allocation for 5G-RAN slicing. IEEE Transactions on Vehicular Technology, 71(1), 856-871. DOI: 10.1109/TVT.2021.3128513

[11] Tran-Dang, H., Bhardwaj, S., Rahim, T., Musaddiq, A., & Kim, D. S. (2022). Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues. Journal of Communications and Networks, 24(1), 83-98. DOI: 10.23919/JCN.2021.000041

[12] Jiang, Y., Kodialam, M., Lakshman, T. V., Mukherjee, S., & Tassiulas, L. (2021). Resource allocation in data centers using fast reinforcement learning algorithms. IEEE Transactions on Network and Service Management, 18(4), 4576-4588. DOI: 10.1109/TNSM.2021.3100460

[13] Yuan, Y., Zheng, G., Wong, K. K., & Letaief, K. B. (2021). Meta-reinforcement learning based resource allocation for dynamic V2X communications. IEEE Transactions on Vehicular Technology, 70(9), 8964 - 8977. DOI: 10.1109/TVT.2021.3098854

[14] Shang, C., Sun, Y., Luo, H., & Guizani, M. (2023). Computation offloading and resource allocation in NOMA–MEC: A deep reinforcement learning approach. IEEE Internet of Things Journal, 10(17), 15464 - 15476.DOI: 10.1109/JIOT.2023.3264206

[15] Liu, T., Ni, S., Li, X., Zhu, Y., Kong, L., & Yang, Y. (2022). Deep reinforcement learning based approach for online service placement and computation resource allocation in edge computing. IEEE Transactions on Mobile Computing, 22(7), 3870-3881.DOI: 10.1109/TMC.2022.3148254

[16] Liu, L., Feng, J., Mu, X., Pei, Q., Lan, D., & Xiao, M. (2023). Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing. IEEE Transactions on Intelligent Transportation Systems, 24(12), 15513-15526. DOI: 10.1109/TITS.2023.3249745

[17] Ju, Y., Chen, Y., Cao, Z., Liu, L., Pei, Q., Xiao, M., ... & Leung, V. C. (2023). Joint secure offloading and resource allocation for vehicular edge computing network: A multi-agent deep reinforcement learning approach. IEEE Transactions on Intelligent Transportation Systems, 24(5), 5555-5569. DOI: 10.1109/TITS.2023.3242997

[18] Huang, J., Wan, J., Lv, B., Ye, Q., & Chen, Y. (2023). Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning. IEEE Systems Journal, 17(2), 2500-2511. DOI: 10.1109/JSYST.2023.3249217