**Frontiers in Computing and Intelligent Systems**
ISSN: 2832-6024 | Vol. 17, No. 3, 2026

# Optimization of Deep Integration of Data Analysis and Intelligent Scheduling for University Public Resource Allocation Systems

**Yaoen Xu ***

School of Engineering, Shenzhen MSU-BIT University, Shenzhen, Guangdong, 518172, China
*** Corresponding author Email:** 17866985585@163.com

**Abstract:** This paper studies the deep integration and optimization of data analysis and intelligent scheduling for university public resource allocation systems. To address the issues of inaccurate prediction, single algorithm, and disconnection between data and scheduling in traditional rule-driven systems, a data-driven intelligent decision scheme is proposed. The LSTM prediction model is optimized by integrating university-specific contextual features, and a PPO-based deep reinforcement learning algorithm is employed for dynamic multi-objective scheduling and optimization. A closed-loop feedback is constructed to realize real-time linkage between data and scheduling. Experiments show that the proposed method significantly improves resource utilization, reduces conflicts and abnormal occupation, and upgrades resource scheduling from rule-driven to data-driven intelligent decision-making.

**Keywords:** Public Resource Allocation; Data Analysis; Intelligent Scheduling; Data-Driven Decision-Making; Deep Reinforcement Learning.

## 1. Introduction

Currently, the public resource reservation systems used in Chinese universities mainly adopt a front-end and back-end separation architecture of SpringBoot + Vue. This is highly compatible with the management model and usage scenarios of Chinese universities [1], achieving an upgrade from traditional manual management to a digital and intelligent system. The core achievements focus on the standardized architecture, the entire process functions, and the basic intelligent scheduling implementation [2].

However, the existing scheduling algorithms (such as traditional time series algorithms, ARIMA and Prophet) are unable to accurately capture the specific scenarios of universities (including the start and end of semesters, holidays, course schedules, and campus activities, etc.), resulting in large prediction errors. Moreover, these systems have not yet achieved the deep integration of data analysis and intelligent scheduling, and most of the scheduling still relies on "rule-driven" rather than "data-driven intelligent decision-making". This paper aims to address this key issue: how to deeply integrate data analysis and intelligent scheduling through methods such as LSTM + PPO joint training and secondary feedback, and transform the system's scheduling method from "rule-driven" to "data-driven intelligent decision-making".

## 2. Optimized Approach and Analysis

Connect the link from data analysis results → real-time → scheduling instructions, allowing "prediction/ exception/ association" to directly drive the scheduling process, eliminating the need for manual management or rigid rule-based scheduling. Based on the algorithm chain of "data characteristics → analysis model (prediction/ exception/ association) → decision factors → scheduling optimizer → scheduling instructions → execution feedback → model update", add secondary data analysis iterations while maintaining the algorithm's lightness, while balancing speed and accuracy. The data analysis results are transformed into real-time "new data" to participate in algorithm iterations. Through innovative nonlinear mapping, weighted group operations are used as one of the input features of the original data network, but the final instructions are generated by the optimizer through rapid iteration using reinforcement learning, breaking the limitations of a single rule and automatically converting the analysis results into scheduling actions, achieving optimal intelligent decision-making compared to the current situation.

### 2.1. Data Processing and Iterative Secondary Analysis

Firstly, the original data is processed and transformed into structured numerical/label data so that the scheduling algorithm can directly read it. Reference: Optimization Design and Analysis of Real-time System Scheduling Algorithms [3] for factor construction as follows:

- Load prediction factor: [0,1] Hourly congestion degree;
- Abnormal risk factor: [0,1] Probability of user malicious occupation;
- Resource heat factor: [0,1] Resource utilization rate;
- Supply-demand gap factor: [-1,1] Excess/shortage;
- User preference factor: [0,1] Matching degree;
- Cross-campus allocation value factor: [0,1] Allocation benefit;

These factors will be used as the input of the scheduling algorithm and combined with mathematical mapping for scheduling decisions.

Subsequently, referring to the LSTM unit state update equation in [4], the following mapping formula is established:

Scheduling instruction = Scheduling optimizer (W1 × prediction factor + W2 × anomaly factor + W3 × popularity factor + W4 × preference factor)

Among them: The scheduling optimizer can adopt methods such as PPO/NSGA-II/linear programming/constrained programming.

W represents the dynamic weight, which changes dynamically according to the user's demands in different periods (such as exam weeks, holidays, research seasons).

The above formula is not a simple weighted summation plus threshold decision-making method. Instead, it takes the vector composed of each decision factor as the input, and simultaneously utilizes the nonlinear fitting and memory capabilities of LSTM. [5] Through a learnable policy network (such as PPO), it generates a nonlinear mapping innovation for scheduling actions.

This mapping relationship $\pi$:R$n$→A is nonlinear.

Weighted summation cannot capture the nonlinear interaction relationships between factors. However, the output results of this new mapping relationship are generated by the optimizer through reinforcement learning and quadratic iteration, and can handle the complex coupling relationships among multiple factors. It has broken through the limitations of traditional weighted summation, and is the core mechanism for this paper to upgrade from "rule-driven" to "data-driven intelligent decision-making".

The use of the joint learning architecture enables real-time coupling of analysis and scheduling [6], and adopts distributed parallel algorithms to alleviate the system's computing pressure and enhance operational efficiency [7], conducting joint optimization across the entire process.

## 2.2. Prediction Model, Scheduling Model and Their Joint Loss and Feedback Algorithm

By using the quadratic iterative algorithm optimized by the GS-like iterative algorithm [8], within the same time and number of iterations, a result closer to the optimal solution can be achieved. Combined with the traffic occupancy situation and priority to schedule the queue [9], and referring to the secondary iterative MIC algorithm [10], the data interference can be further reduced, enabling the model to be optimized more efficiently.

The joint loss of the model is defined as:

Loss = α × Prediction Error + β × Scheduling Objective (Utilization/Fairness/Conflict Rate)

The "Prediction Error" comes from the load prediction of LSTM, while the "Scheduling Objective" originates from the scheduling benefits of PPO. Through this unified loss function, LSTM no longer merely minimizes the prediction error, but also takes into account the scheduling goals, forcing the prediction model to provide more discriminative features for the scheduler, while optimizing the prediction accuracy and scheduling benefits. LSTM collaborates with PPO for optimization [11], achieving a tightly coupled "analysis-scheduling" and demonstrating the advantages of joint training (joint training) and collaborative task decision-making [12].

The following is the pseudo-code of the feedback algorithm:

```
// 1. Extract features features = get_campus_features(user, resource, time, schedule, event)
// 2. The data analysis model outputs decision factors
    load_score = LSTM(features)   // Load prediction
    risk_score = IsolationForest(...)   // Exception Detection
    prefer_score = CF(...)   // Preference Matching
// 3. Construct the decision vector (connect the key points)
    decision_vector = [load_score, risk_score, prefer_score, gap_score]
// 4.The scheduler directly inputs the decision vector action = PPO_optimizer (decision_vector)
// 5. Output scheduling instructions (executed directly by the system)
    if action == 1: Reserve resources
    if action == 2: Prohibit reservations
    if action == 3: Allocate resources across campuses
    if action == 4: Recommend alternative resources
```

## 2.3. Optimization Results and Outcomes

Combining the above content and using artificial intelligence for auxiliary modeling and real-time performance monitoring [13] can make the public resource scheduling algorithm more closely integrated between data analysis and intelligent scheduling.

Here is the pseudo-code of the program：

```
# Global Configuration
MAX_ITER = 3
REAL_TIME_THRESHOLD = 100
DECISION_DIM = 6

# Module 1: Real-time Data Analysis
# Input: Raw streaming data → Output: Standardized new data

def real_time_analysis(stream_data):
    """ Real-time analysis, output new data for system feedback """
    # 1. Extract scenario features (campus as example)
    features = extract_features(
        stream_data.user_id,
        stream_data.resource_id,
        stream_data.time_slot,
        stream_data.campus_id,
        stream_data.schedule,
        stream_data.event
    )

    # 2. Prediction using lightweight LSTM model
    load_score = lightweight_LSTM_infer(features)

    # 3. Anomaly detection
    risk_score = isolation_forest_infer(features)

    # 4. Supply-demand calculation
    gap_score = calculate_ supply_ demand_ gap (stream_ data)

    # 5. Resource heat and user preference
    heat_score = compute_ resource_ heat (stream_ data. resource_ id)
    prefer_score = user_ preference_ match (stream_ data. user_ id)

    # 6. Dynamic weight (adaptive for exam weeks /
```

```
research seasons)
        dynamic_weight = adaptive_ weight (stream_ data. time_slot)

        # Generate standardized new data; analysis results become new data fed back to the system
        new_data = {
            "load_score":      load_score,      # 0~1
            "risk_score":      risk_score,      # 0~1
            "gap_score":       gap_score,       # -1~1
            "heat_score":      heat_score,      # 0~1
            "prefer_score":    prefer_score,    # 0~1
            "dynamic_weight": dynamic_weight
        }

        # Write new data to system / cache
        save_new_data_to_system(new_data)
        return new_data

    # Module 2: Second-iteration Optimization
    # Input: System state and new data from first analysis; Output: Approximate optimal instructions

    def second_iteration_optimize(system_state, new_data):
        """ Second lightweight iteration to quickly approach optimal scheduling """
        best_schedule = None
        best_reward = -float("inf")

        # Fixed small number of iterations
        for iter in range(MAX_ITER):
            # 1. Generate scheduling candidates based on new data
            candidate = generate_ schedule_ candidate (system_state, new_data)

            # 2. Multi-objective evaluation based on factors
            current_reward = evaluate_multi_objective(
                candidate,
                new_data["load_score"],
                new_data["risk_score"],
                new_data["gap_score"],
                new_data["dynamic_weight"]
            )

            # 3. Keep the best candidate
            if current_reward > best_reward:
                best_reward = current_reward
                best_schedule = candidate

        return best_schedule


    # Module 3: Real-time Scheduling Execution

    def real_time_schedule_execute(stream_data):
        """ Main pipeline: Real-time analysis → New data → Second iteration → Execution """
        # 1 First-round real-time analysis → generate and feed back new data
        new_data = real_time_analysis(stream_data)

        # 2 Get current global system state
        system_state = get_system_real_time_state()

        # 3 Second-iteration optimization → approximate optimal schedule
        optimal_schedule = second_ iteration_ optimize (system_ state, new_data)

        # 4 Dispatch scheduling commands in real time
        dispatch_schedule_command(optimal_schedule)

        # 5 Feed back execution results to update the model
        feedback_to_model(optimal_schedule, new_data)

        return optimal_schedule


    # Module 4: Real-time Trigger Entry

    def event_trigger_entry(event_type, stream_data):
        """ Trigger types: New reservation / Cancellation / Check-in / Overtime / Periodic 15min """
        if event_type in ["reserve", "cancel", "check_in", "overtime"]:
            # Event-triggered: execute immediately
            real_time_schedule_execute(stream_data)
        elif event_type == "periodic":
            # Periodic global rescheduling
            batch_real_time_schedule()
```

Subsequently, a simple simulation experiment model was established, which mainly relied on PPO deep reinforcement learning [14] and combined with a simplified LSTM for joint training. First, data was randomly generated based on the given feature parameters, and then the joint model was trained under different load conditions to simulate the acceptance rate, conflict rate, and utilization rate of resources in the real scenario. After referring to common algorithms, iterative algorithms [15], and the performance of PPO's algorithm [16][17], an appropriate comparison baseline was set. The data obtained from the experiment was statistically tested and compared with it.

A total of four simulation program files were used in the simulation experiment: generate_data.py: generates the data source and simulates the real appointment logs; lstm_model.py: this file contains the LSTM network structure, training function, and a global inference interface; env.py: simulation environment simulation, which can formalize the scheduling problem as an MDP; train.py: experiment process management, used for training, evaluation, comparison, and result display.

The final results are shown in the following Table 1:

Through analysis, it can be concluded that intelligent scheduling is most useful when resources are under stress but not extremely saturated. The joint training of PPO and LSTM shows the most significant advantage when resources are moderately saturated. In scenarios of abundant resources and severe saturation, it performs on par with the Accept-All strategy. In the simulation experiments, the end-to-end joint training of PPO and LSTM is greatly influenced by the design of the LSTM reward function. The third set of data at a load of 3000 demonstrates that although extremely low conflicts are achieved under high penalties, the acceptance rate is sacrificed. The actual deployment needs to be balanced according to user preferences.

**Table 1.** Performance Comparison of Different Scheduling Methods Under Various Reservation Loads

| **Load (Reservation Number)** | **scheduling method** | **Acceptance rate (%)** | Conflict rate (%) | Utilization rate (%) | Conflict Rate Optimization | remark |
|---|---|---|---|---|---|---|
| **1800** | Accept-All | 82.3 | 17.7 | 84.9 | - | Low load |
| 1800 | PPO+LSTM | 82.3 | 17.7 | 84.9 | 0% | |
| 2400 | Accept-All | 66.8 | 33.2 | 91.9 | - | medium load |
| **2400** | PPO+LSTM | 66.8 | 33.2 | 91.9 | 0% | |
| 3000 | Accept-All | 56.5 | 43.5 | 97.1 | - | Moderate saturation |
| 3000 | PPO+LSTM | 55.7 | 27.6 | 95.8 | 36.6% | optimum balance |
| 3000 | PPO+LSTM (HighPenalty) | 31.2 | 0.9 | 53.6 | 97.9% | conservative strategy |
| 3500 | Accept-All | 49.7 | 50.3 | 99.7 | - | High saturation |
| 3500 | PPO+LSTM | 49.7 | 50.3 | 99.7 | 0% | |
| 5000 | Accept-All | 34.9 | 60.1 | 100 | - | Extremely saturated |
| 5000 | PPO+LSTM | 34.9 | 60.1 | 100 | 0% | |

# 3. Conclusion

The proposed PPO scheduler in this paper can effectively balance conflicts and acceptance rates in scenarios of moderately saturated resources, outperforming the simple all-acceptance strategy. The experiments reveal the importance of the applicable range of the algorithm (with moderate load) and the significance of reward parameter tuning, providing a quantitative basis for the intelligent upgrade of university resource scheduling systems.

# References


[1] Wang, L., & Lu, K. (2026). A reinforcement learning framework for modeling the reinforcement of cultural inertia in the allocation of public welfare resources. Scientific Reports, [https:// doi.org/ 10. 1038/ s41598] (https://doi. org/10. 1038/ s41598) -026-50004-4.

[2] Yang, C. (2025). Analysis of real-time and accurate balance strategies for main grid dispatching based on big data technology. Electronic Technology, 54(7), 298-299.

[3] Song, Y. (2019). Optimization design analysis of real-time system scheduling algorithms. Computer Products and Circulation, (12), 93-94.

[4] Chen, B., Pan, T., & Xie, Z. (2026). Research on intelligent management and data-driven decision-making in hydraulic engineering. Science and Technology Innovation, (1), 154-157.

[5] Ding, B. (2020). Research on image and video retrieval algorithms based on nonlinear hashing (Master's thesis, Beijing University of Posts and Telecommunications).

[6] Song, J., Tan, L., Cao, Z., & Wang, S. (2010). A new real-time scheduling algorithm. Computer Technology and Development, 20(12), 73-76.

[7] Wang, B., & Lei, L. (2013). A distributed parallel algorithm optimized by big data analysis. Computer and Digital Engineering, 41(11), 1720-1724, 1861.

[8] Xing, Y., Li, S., Zhai, C., & Wang, D. (2008). Iterative learning control algorithm based on quadratic optimization. Journal of Liaoning University of Petrochemical Technology, 28(3), 53-55.

[9] Chen, J., Wang, J., Hu, J., & Xu, Y. (2025). Priority queue traffic scheduling algorithm for diversified network demands. Computer Engineering, 51(11), 152-161.

[10] Lu, P., Song, C., Bai, L., & Song, C. (2013). A frequency-homogeneous interference cancellation algorithm based on secondary iteration. Communication Technology, 46(8), 1-4.

[11] Liu, J., & Peng, P. (2026). A multi-UAV cooperative task decision-making method based on improved PPO algorithm. Journal of Weaponry and Equipment Engineering, 47(4), 9-15, 48.

[12] Chi, X., Zhang, J., & Yu, H. (2026). Control-communication co-optimization for wireless cloud robotic system via multi-agent transfer reinforcement learning. IEEE/CAA Journal of Automatica Sinica, 13(2), 311-326. [https://doi.org/ 10.1109/ JAS. 2026. 1241228] (https://doi.org/10. 1109/ JAS. 2026. 1241228).

[13] Lu, Y., & Yu, H. (2025). Artificial intelligence-driven dynamic decision-making in supply chain: Modeling and analysis based on complex system theory. Supply Chain Management, 6(12), 97-112.

[14] Wang, X., Gui, J., Li, J., & Song, Q. (2023). Comparative study on iterative pure phase hologram generation algorithms. Laser and Optoelectronics Progress, 60(6), 29-36.

[15] Feng, H., & Ren, F. (2025). Application and performance comparison study of numerical analysis algorithms in computer experiments. Information and Computers, 37(1), 26-28.

[16] Mustafa, E., Shuja, J., Rehman, F., & Gumbley, S. J. (2025). Computation offloading in vehicular communications using PPO-based deep reinforcement learning. The Journal of Supercomputing, 81, 547.

[17] Qi, Y., Wang, C., & Liu, J. (2026). PPO-RR: Virtual machine scheduling algorithm and deep reinforcement learning method for green computing. Intelligent Decision Technologies, 20(1), 392-413.