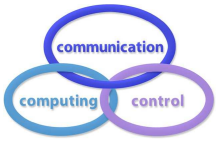
INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 18, Issue: 6, Month: December, Year: 2023
Article Number: 5998, https://doi.org/10.15837/ijccc.2023.6.5998



CCC Publications

Iot Data Processing and Scheduling Based on Deep Reinforcement Learning

Yuchuan Jiang, Zhangjun Wang, Zhixiong Jin

Yuchuan Jiang*
ChongQing College of Humanities,Science & Technology
ChongQing 401524, China
*Corresponding author: bbjuc@163.com

Zhangjun Wang
Sichuan Water Conservancy Vocational College
Chongzhou 611231, China
wangzhjun2002@163.com

Zhixiong Jin
Geely University of China
ChengDu 641423, China
jinzhixiong@hongyicg.com

Abstract

With the continuous integration of IoT technology and information technology, edge computing, as an emerging computing paradigm, makes full use of terminals to process and analyse real-time data. The explosion of Internet of Things (IoT) devices has created challenges for traditional cloud-based data processing models due to high latency and availability requirements. This paper proposes a new edge computation-based framework for iot data processing and scheduling using deep reinforcement learning. The system architecture incorporates distributed iot data access, real-time processing, and an intelligent scheduler based on Deep q networks (DQN). A large number of experiments show that compared with traditional scheduling methods, the average task completion time is reduced by 20% and resource utilization is increased by 15%. The unique integration of edge computing and deep reinforcement learning provides a flexible and efficient platform for low-latency iot applications. Key results obtained from testing the proposed system, such as reduced task completion time and increased resource utilization.

Keywords: Edge computing, data processing, task scheduling, reinforcement learning, IoT platforms

1 Introduction

As Internet of Things (IoT) technology advances, an increasing number of end-devices are being widely implemented in a variety of living scenarios in order to collect real-time data processing (DP) activities by gathering environmental data like photographs. The traditional "end-device-data-centre"

architecture can hardly meet the low latency requirements of real-time data computation tasks [1, 2]. In intelligent edge computing, although the use of edge devices (which have stronger computing power than terminal devices) can reduce the high latency caused by data transmission and improve the quality of service, there are also obvious limitations [3, 4]. For example, when processing applications with high bandwidth requirements, such as high-definition video stream processing or big data analysis, intelligent edge computing may not be able to meet data transmission requirements due to the limitation of broadband resources of edge devices, resulting in an increase in service delay. In the current edge computing scheduling methods, how to efficiently implement resource scheduling is the biggest challenge facing the research. Especially in the data center, the resource scheduling problem of multi-level, multi-data stream and multi-task is essentially a data transmission problem in the interactive process. In addition, due to the limitation of resources, edge computing tasks may have the problem of low latency, and resource scheduling in the process of task interaction to meet the communication needs of edge users. To solve these problems, this study proposes an edge-computing paradigm oriented iot data processing and scheduling design, and builds an intelligent and dynamic scheduling system. This system can achieve efficient task allocation, greatly improve the operating efficiency of edge applications, and thus meet the communication needs of edge users [5]. The study is organised into five main sections; the introduction outlines how smart apps are used in various spheres of people's lives in the era of IoT technology's rapid development. The second section is a survey of the literature on the use of DP and task scheduling (TS) issues under EC in various fields and the present state of the technology's study by numerous academics. The design of EC-based IoTDP and TS systems is examined and analysed in the third section. The design of the edge IoTDP platform, which describes its design goals and strategy, is covered in the first subsection of this section. The design of the ECTS system employing Deep Q-Leaning Network (DQN) and deep reinforcement learning (DRL) algorithms is covered in the second subsection. By using the data preparation in accordance with this design, the fourth section evaluates the TS's performance. A summary and overview of the research's methodology and findings are provided in the fifth section.

2 Related work

EC platforms are deployed in close proximity to IoT terminals and will secure real-time IoT data while releasing computation and storage pressure, it is important to DP them, so they have been studied by many scholars. In order to extract and evaluate each parameter that affects semiconductor yield and perform dynamic compensation so that each machine has a specific level of flexibility in producing goods, B Li et al. integrated association rules and k-mean clustering algorithms with real-time feedback control analysis [6]. In order to decrease the overall processing time and latency of the distributed fog computing process in IoMT under the various scenarios indicated above, J Yue and M Xiao presented coding techniques with great flexibility and low complexity. The study's findings suggested that the fountain code-based system can produce lower latency and a shorter overall processing time [7]. A new deep artificial structure learning technique based on jackknife regression Schmidt-Samoa cryptography is proposed by C Kannan et al. Data acquired from datasets and monitored by IoT devices is processed in the first hidden layer by learning characteristics to identify the data more accurately. The encrypted and decrypted classified data is then transmitted to the following concealed layer and processed there [8]. In the context of the smart IoT, P Yangy et al. presented an edge-cloud collaborative trust assessment architecture. The study's findings showed that, when compared to conventional approaches, the suggested trust evaluation method can successfully increase interaction success rates and decrease false alert rates while dealing with hostile services and malicious recommendations [9]. A social relationship-based data transfer technique was proposed by Y Xu et al. In the Internet of Things, the algorithm assesses the social relationship traits of mobile nodes and quantifies the social relationship by fusing information entropy and fuzzy clustering theory [10]. Y Xu et al. proposed a social relation quantization algorithm called mobile node, which is based on the data transmission mechanism of social relations. However, while this approach has been effective in reducing end-to-end transmission latency and routing overhead, reducing energy consumption during transmission, and maintaining a high data transfer success rate, there are still challenges when dealing

with large-scale, complex and dynamic iot data. This study adopts a deep reinforcement learning-based iot data processing and scheduling mechanism to dynamically schedule and optimize resources, which can not only process larger scale iot data, but also effectively improve task completion efficiency and reduce system delay.

EC, due to its node heterogeneity and distributed system architecture, is highly prone to resource utilisation imbalance when it encounters too frequent IoT data reception requests, therefore, many scholars have quite a lot of research on TS in EC environment. To manage cloud resources, Asghari et al. coupled state-action-reward-state-action learning with genetic algorithms. By investigating workflows, assigning each resource to an agent, and attempting to use that resource as efficiently as possible throughout that agent's learning phase, intelligent agents schedule tasks in the learning process. The proposed method is converged using genetic algorithm and global optimisation is achieved. Experimental results show that the algorithm reduces the completion time, increases the resource utilisation and improves the load balance [11]. C G Wu et al. proposed a distribution estimation algorithm enhanced by path relinking. In order to describe the relative positional relationships between task pairs, a specific probabilistic model was developed and task processing alignments were generated by sampling this model. Using the knowledge based on path relinking to design the local search method, simulation experiments were conducted using a benchmark dataset and real-world graphics. Comparing the outcomes revealed that the method greatly outperformed existing heuristic and evolutionary algorithms in terms of performance improvement [12]. Markov decision chains were suggested by S Y Huang et al. as a method of predicting channel state. This approach enables all consumers to receive higher service quality. In order to discover the ideal TS, an updated particle swarm optimisation technique is also employed to solve resource allocation and reduce energy usage. The results of the study show that the method used can schedule the tasks efficiently [13]. A strategy to boost task performance using the hybrid scheduling of CPU and network IO resources was put forth by D. Wang et al. The benchmark's total performance was significantly enhanced by the proposed CPU/IO scheduling approach, according to experimental data [14].

In conclusion, in the field of iot data processing and task scheduling, edge computing provides a unique platform for data processing that can analyze and provide unified information to edge users in real time. Using edge computing for iot application learning can mine data value more effectively. However, this also brings a challenge, because of the real-time data processing of the Internet of Things, edge nodes may have unbalanced resource utilization, resulting in node downtime, which affects the dynamic scheduling efficiency of tasks. To solve this problem, this study proposes a data processing and task scheduling scheme based on edge computing. This scheme combines deep reinforcement learning technology to build a dynamic task scheduling system framework to improve the operating efficiency of edge applications.

3 Design of EC-based IoTDP and TS System

EC is the key to IoTDP, and by performing DP at the proximal end of the device, network latency can be reduced and DP efficiency can be improved while ensuring data security. This involves the collection, cleaning, analysis and storage of data, and DP efficiency needs to be optimised while ensuring data quality. Next, a TS system is designed using deep Q-networks. As a DRL method, DQN can learn and optimise the behavioural policies of the system to achieve a higher priority of TS. In a large number of IoT environments, TS is crucial to effectively manage various computational tasks and balance the load of the system, which in turn improves the overall performance of the system.

3.1 Edge iot data processing platform

The IoTDP platform of EC platform is constructed mainly for the management users on the edge side to address the incompleteness and low latency of IoT application functions when IoT real-time DP is used. In addition, the limited computing and storage resources of the EC platform make it insufficiently perfect in running real-time streaming computation of big data, and also cannot further improve the operation efficiency of edge applications. Therefore, this research constructs a set of architectural design containing applications such as data parsing, data forwarding, data storage and

big data streaming computation, which contains systems such as IoT data access and parsing, data forwarding and big data streaming processing, data storage and visualisation, and cluster management and orchestration [15, 16]. The overall architecture is shown in Figure 1.

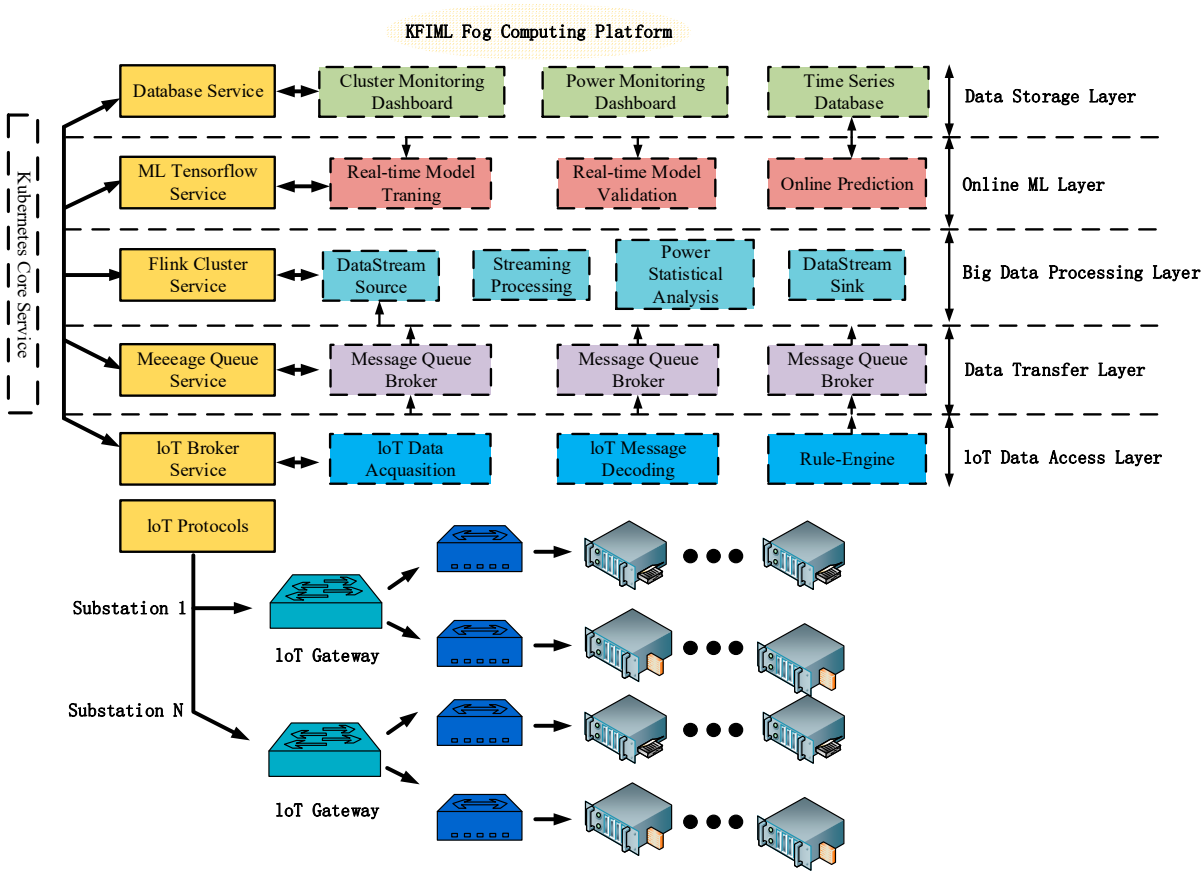


Figure 1: Architecture of Edge IoT Data Processing Platform

In Figure 1, the edge Internet of Things data processing platform arranges distributed application services for each layer on the basis of container technology. Under the arrangement capability and load balancing service of the container, data flows in and out between applications at each layer. This system architecture is composed of iot data access layer, data forwarding cache layer, big data processing layer, online machine learning application layer and data storage layer, and provides data distribution among distributed component instances of each layer through Kubernetes load balancing Service. In the system architecture, the iot data access layer uses distributed iot access components to directly interface with the data stream reported by iot terminal devices, and supports forwarding with various distributed messaging middleware. The latest big data flow calculation results are taken as training set, and the latest model is obtained through training. The online machine learning application layer uses the Tensorflow framework to provide a pipeline for lightweight online machine learning applications. It includes three parallel processes: Real-time Model Training, Real-time Model Validation and Online Prediction. The online training process periodically pulls the latest big data stream calculation results as a training set, and gets the latest model through training. The data forwarding cache layer is the engine that decoups the access of the original data of the Internet of Things and the big data calculation, improving the stability of the platform. The Big Data processing layer provides big data processing capabilities for iot real-time data and stores the calculation results in downstream databases. The data forwarding cache layer mainly deploys distributed message-oriented middleware Kafka and ZooKeeper clusters to provide caching services for data forwarded by the data access layer of the Internet of Things, realizing the decoupling of the original data access of the Internet of Things and the big data computing engine, and improving the stability of the platform system. The edge stream computing data processing platform can provide a distributed real-time data access layer of the Internet of Things and provide load balancing access services for data processing applications

in different iot scenarios [17, 18]. The structure diagram of iot data access layer functions is shown in Figure 2.

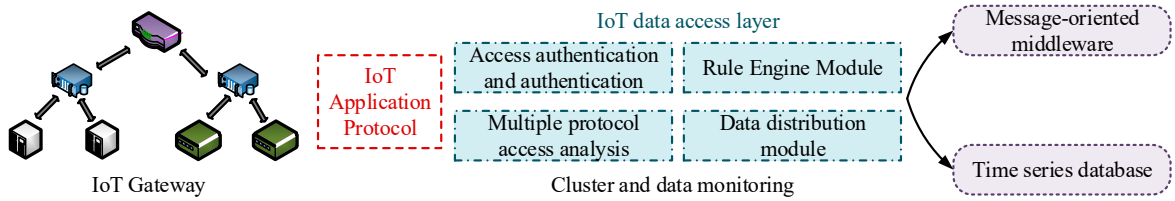


Figure 2: Edge IoT data platform access layer functional diagram

In Figure 2, this IoT data access layer function, mainly consists of six modules: access authentication and forensics, IoT protocol access and data parsing, rule engine, data distribution, and cluster and data monitoring. Data access authentication and forensics are two of them, and their purpose is to confirm the legality of IoT real-time data access by confirming the identity of the IoT client connecting to the access platform. The rule engine, on the other hand, provides processing rules for IoT real-time data streams, and response actions are triggered by configuring the rule engine. And the data dissemination module with distributed messaging middleware such as Kafka, and the cluster and data monitoring module provide visual interfaces to display the metrics information of data sources. In order to meet the user's needs, multiple machine learning algorithms need to be assembled to develop IoT machine learning applications to build a complete set of online machine learning IoT applications. The schematic diagram of the functions of the big DP layer and the online machine learning application layer is shown in Figure 3.

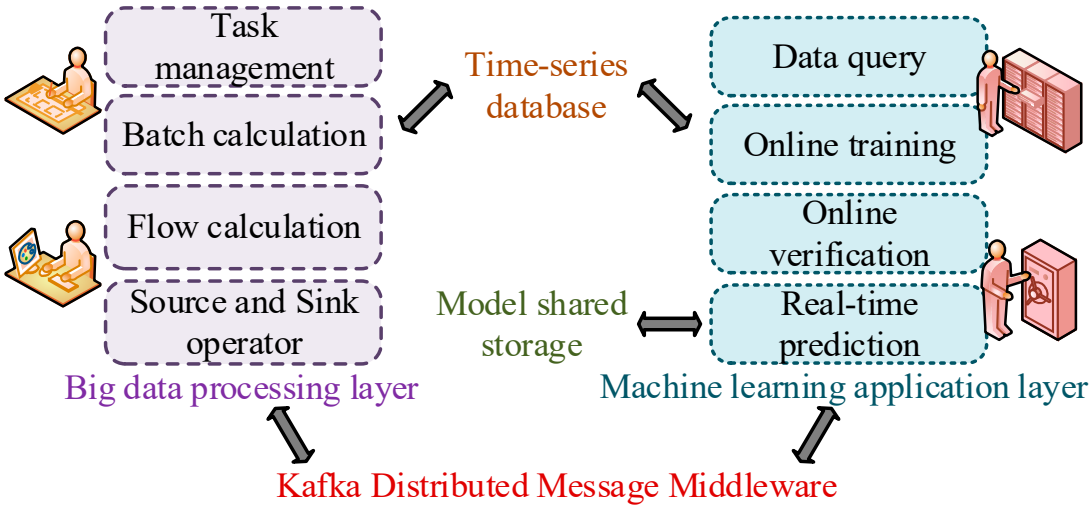


Figure 3: Functional diagram of the processing layer and online machine learning application layer in the IoT data processing platform

In Figure 3, the machine learning application layer is based on Tensorflow.js, a lightweight learning framework in Java language, to develop the application process and algorithms. Distributed messaging middleware, such as Kafka, is connected to the data downlink module, whose function is to forward real-time IoT data to the middleware distributed nodes, and to complete the fast retrieval of real-time data under the guidance of the big data real-time computing engine. In addition, the data results can also be obtained directly from the time series database. Based on the real-time nature of IoT data and the user's demand for machine learning application functions, this application layer contains the processes of online training, online verification and real-time reasoning interacting and outputting in parallel. In addition, online training, online validation and real-time inference are shared under the same NFS directory for model sharing.

3.2 Scheduling system design

When designing a task scheduling system for edge computing applications, managers need to orchestrate the online machine learning tasks running in edge clusters to improve the efficiency of applications. This involves real-time processing of iot data streams and precision design of scheduling algorithms that take into account the execution time of each task and the actual utilization of edge node resources. DQN deep reinforcement learning algorithm is chosen because of its unique advantages in dealing with such problems. Compared with other algorithms, DQN can find the optimal decision strategy through continuous learning and optimization in the face of complex and dynamic scheduling environment, and realize the automation and intelligence of task scheduling [19, 20]. At the same time, the scheduling management information system developed by time series database and Kafka message middleware are used for data communication, which not only decoupling components developed in different languages, but also enhancing the stability of the scheduling system [21]. Figure 4 depicts the architecture diagram of the scheduling system.

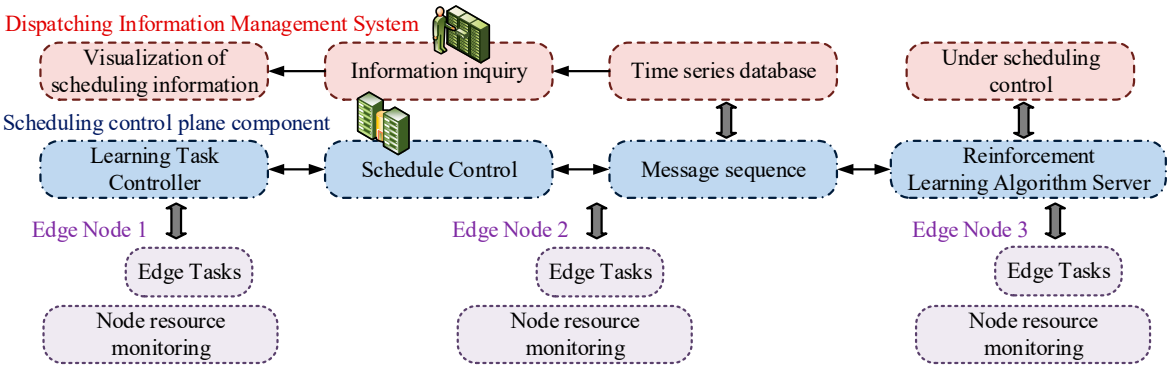


Figure 4: Schematic diagram of scheduling system architecture

In this system, the scheduling control plane component consists of three core components: the edge machine learning task controller, the reinforcement learning algorithm server, and the TS controller. In addition, the scheduling management system provides query and visualisation options for scheduling information and task information. Among them, the Edge Machine Learning Task Controller and the Scheduling Controller are both custom controllers developed on the basis of the Kubebuilder framework to satisfy the user's task configuration scheduling needs in the information management interface. Among them, the TS controller is also a Kubebuilder custom controller that manages the lifecycle of scheduling objects and executes the scheduling process. The interaction between the different components is critical, and they communicate and transfer data through well-defined API interfaces. The Edge Machine Learning Task Controller is mainly responsible for receiving machine learning task requests from users, parsing the task requirements, and then forwarding them to the TS Controller. It also obtains the execution status of the task and ensures that the task is performed according to the user's requirements. The Reinforcement Learning Algorithm Server then acts as the core decision engine to provide decision support for the TS. When the TS controller receives a task request, it asks the reinforcement learning algorithm server how to optimise the scheduling. By learning from historical data and real-time feedback, the reinforcement learning algorithm continuously optimises its decisions to ensure efficient and stable operation of the system. The task scheduling controller dominates the whole task distribution and resource management process [22]. It relies not only on existing resources, but also heavily on suggestions from servers based on reinforcement learning algorithms to effectively determine task allocation and execution. Regularly sync with the edge machine learning task controller to ensure the latest feedback on task status [23, 24]. In order to provide a user-friendly monitoring and management environment, this study designed an intuitive visual interface. On this screen, you can view the task status, system resource usage, and historical task records. When constructing the model, a specific training framework and a set of hyperparameters are used to optimize its performance. These hyperparameters, such as learning rate and reward discount factor, are constantly adjusted and optimized through experiments to achieve the optimal performance of the scheduling system. The

components of the scheduling control plane are represented schematically in Figure 5's diagram.

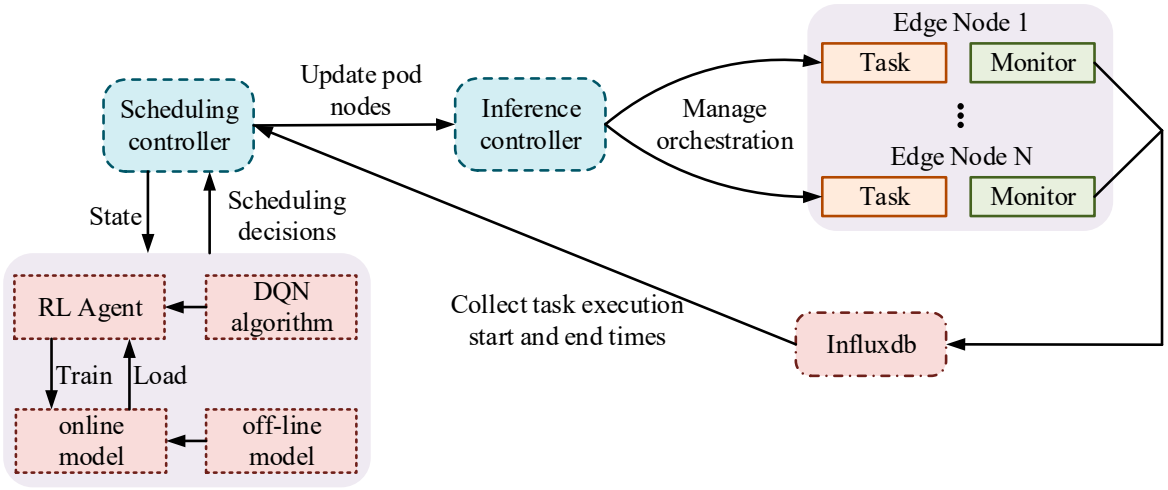


Figure 5: Schematic diagram of scheduling control plane components

In Figure 5, the structure of the scheduling control plane and the relationships between the components are shown. The Edge Machine Learning Task Controller, a component from which several arrows point to the TS Controller, denotes task requests and status updates [25, 26]. The Reinforcement Learning Algorithm Server, indicating its intelligent decision-making function. It is connected to the TS controller with arrows indicating the provision of suggestions and the reception of feedback. The TS controller is located in the centre with multiple arrows pointing to the other two components, indicating its core scheduling function. The visualisation interface is located on the right side with an arrow pointing to it, indicating the provision of information to the user from the TS controller. The components interact with each other through defined API interfaces to enable the system to work together. This schematic provides an intuitive view for the user to help understand how the components work together to achieve an efficient TS. the DQN scheduling algorithm framework, as shown in Figure 6.

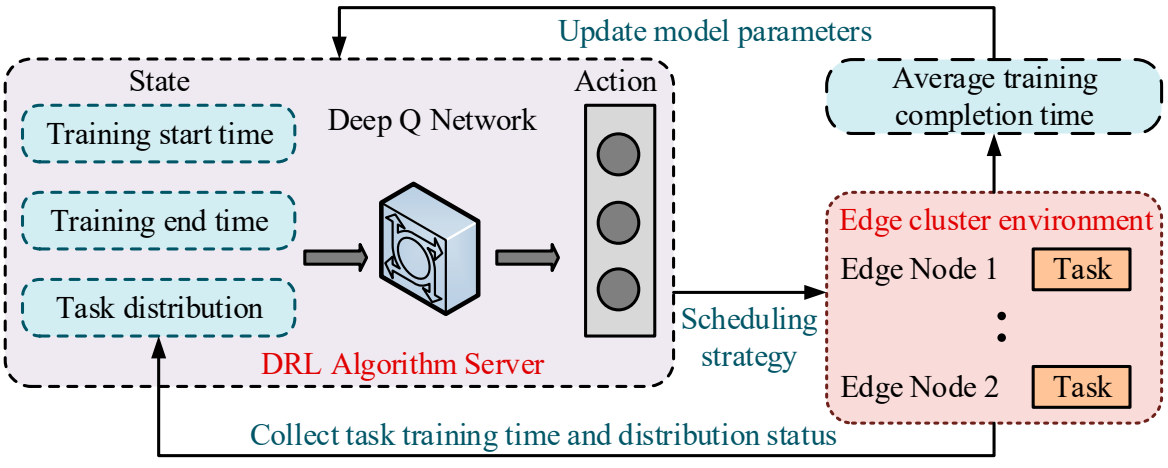


Figure 6: DQN scheduling algorithm framework

In Figure 6, the integration of the DQN algorithm into the TS controller is demonstrated to improve the efficiency and stability of the TS. the TS controller with the DQN module, inside the TS controller, there is a clearly labelled DQN module. This module is used to handle complex scheduling decisions. state inputs, pointing from the state pool of the system (including various resource conditions and task demands) to the DQN module, denote the state of the system as an input to the DQN. Action output indicates that the DQN module will output an optimal action (or a series of actions) for the TS controller to execute. Reward signals, from the system feedback (e.g., task completion time,

resource usage efficiency, etc.) to the DQN module have a reversed arrow, indicated for training and optimising the DQN model. And Data Synchronisation, demonstrates a data synchronisation arrow between the DQN module and the reinforcement learning algorithm server for periodically updating the DQN model. The user interface interacts with the DQN, and the visualisation interface is not only connected to the TS controller, but also has a bi-directional arrow with the DQN module, allowing the user to view or manually adjust certain parameters of the DQN. The DQN module interacts with the rest of the system through an API interface in order to achieve intelligence and automation of the TS. The core of the algorithm is based on empirical recovery, the creation of a network of target values and the establishment of a new exploration mechanism. The target network reveals continuous cyclic stability over the cycle in resolving unbalanced action values. The resulting new error function is generated as shown in equation (1).

$$\delta_t = Q\left(s_t, a_t; w\right) - \underbrace{\left(r_t + \gamma \cdot \max Q'\left(s_{t+1}, a_t; w'\right)\right)}_{\text{Target Q value}} \tag{1}$$

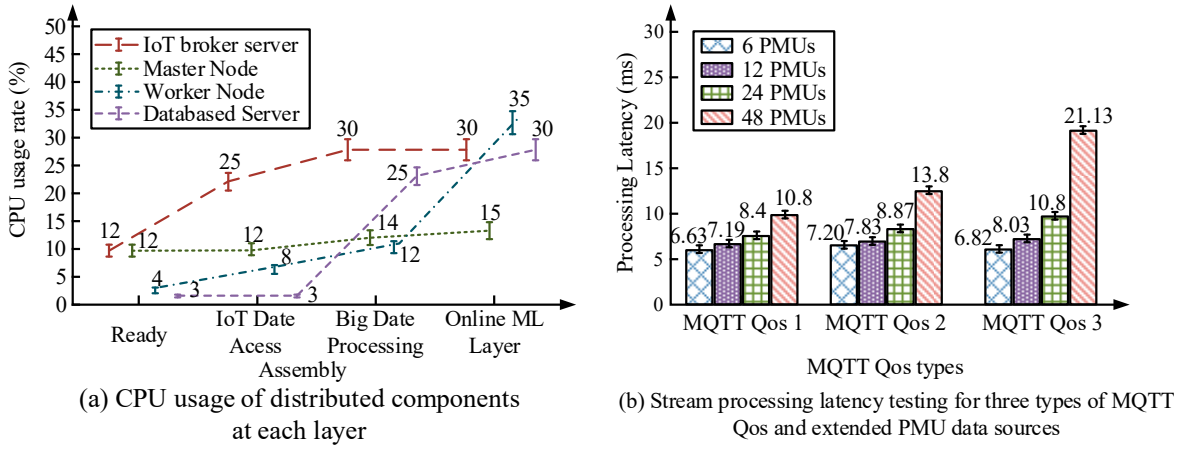
In equation (1), γ is the discount factor, w' is the model parameters, Q' is the target network, an action is randomly selected and executed with a certain probability to ensure the empirical richness of the state-an-action space collected by the intelligent body, and at this point, there is no need to carry out additional exploration, and the Q neural network is used to output the action, and the training pattern can be randomly set with probability.

4 Test Analysis Based on ECDP and TS Systems

The system combines the advantages of EC to effectively handle large amounts of data while ensuring the efficiency and stability of TS. The analysis covers the key components of the system, such as the TS controller and the DQN algorithmic framework, and how they collaborate with each other to meet real business requirements.

4.1 Test Analysis Based on ECDP and TS Systems

To delve deeper into how edge computing can provide more efficient data processing in iot scenarios, this study selected a large iot dataset generated by a variety of sensors containing about 5 million records, each with three fields of timestamp, device ID, and sensor readout. The experiments were conducted on a hardware platform with 16GB RAM and an Intel i7 octa-core processor. In a stream processing latency test for three MQTT QoS and extended PMU data sources, see how edge computing helps reduce latency, increase data processing speed, and optimize resource usage efficiency.Stream processing latency tests for three MQTT QoS and extended PMU data sources are shown in Figure 7.

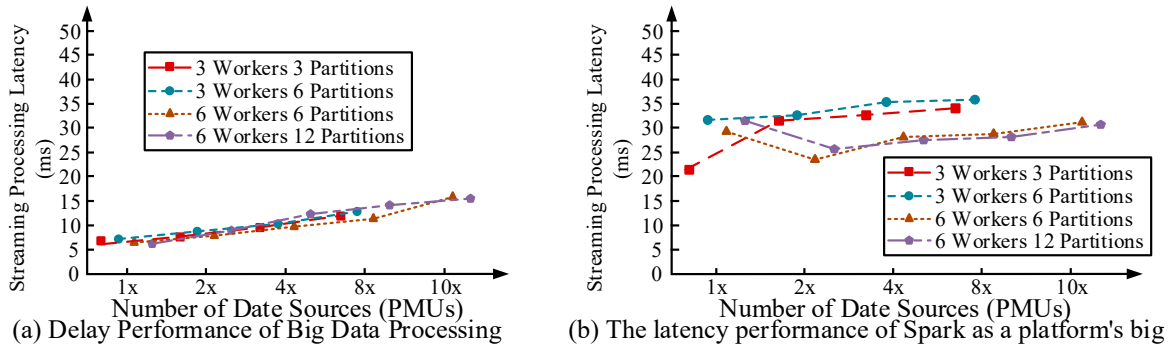


(a) CPU usage of distributed components at each layer

(b) Stream processing latency testing for three types of MQTT Qos and extended PMU data sources

Figure 7: Stream processing latency testing for three types of MQTT QoS and extended PMU data sources

In Figure 7(a), the CPU resource utilisation for each level of the IoT data access layer, data forwarding layer, large DP layer, and online machine learning layer when processing real-time data is detailed for the edge cluster nodes in the experimental setting with 12 PMU data sources. In order to gain a deeper understanding of the platform's performance, computational resource usage tests are conducted for the running components of its layers. In Figure 7(b), a detailed comparative analysis of the average stream processing latency derived from the real-time power calculations for big data is presented in the context of end devices transmitting real-time IoT data to the platform's access layer using the three QoS data transmission methods of the MATT protocol. The performance of the latency at different levels of parallelism for different processing layers is shown in Figure 8.

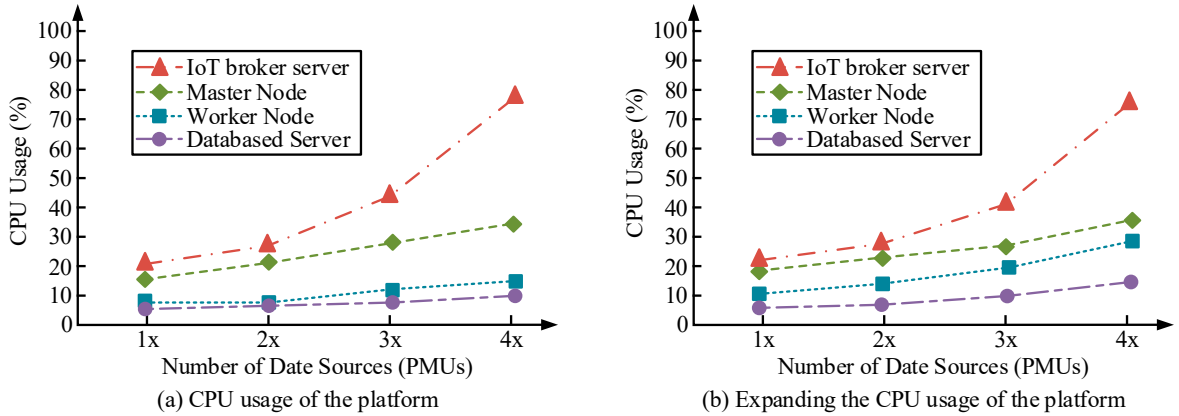


(a) Delay Performance of Big Data Processing Layer Flink under Different Parallelism Degrees

(b) The latency performance of Spark as a platform's big data processing layer under different levels of parallelism

Figure 8: Delay Performance under Different Parallelism Levels in Different Processing Layers

In Figure 8(a), for real-time data sent by 60 PMU data sources, 6 parallel Flink Task Managers are able to process it within about 15ms. In contrast, 3 parallel Flink Task Manager can only process 48 PMU data sources in low latency mode. The logic behind this phenomenon is that having more parallel Flink Task Managers means being able to pull more PMU real-time data from the Kafka Broker cluster in a shorter period of time and maximise the compute resource utilisation of the edge worker nodes. Further, when the execution parallelism of Flink tasks reaches parity with the number of Kafka Partitions, the latency of large DPs is further reduced. In Figure 8(b), in the platform, the framework for the big DP layer is replaced from Flink to Spark Structured Streaming provided by the Spark project, and the Spark Master and Spark Worker components are managed through Kubernetes. This makes data streaming computations more efficient. By adjusting the execution parallelism of Flink tasks and the Partition of Kafka topics, cluster managers are able to scale the cluster and utilise its computational resources more efficiently as the number of data sources increases. Experiments on the scalability of the platform are shown in Figure 9.



(a) CPU usage of the platform

(b) Expanding the CPU usage of the platform

Figure 9: Delay Performance under Different Parallelism Levels in Different Processing Layers

In Figure 9(a), it is observed that the average CPU resource utilisation of each node of the EC cluster grows proportionally with the gradual increase in the number of data sources. This mainly affects the resource utilisation of the IoT Broker Server and each working node. As shown in Fig-

ure 9(b), after scaling the components, the resource utilisation of the working nodes processing the same amount of data is about 1.75 times higher than that before scaling. This means that the extended large DP layer, which will consume more computational resources, also enables the platform to process more intensive IoT real-time data with low-latency characteristics.

4.2 Performance Test of DQNDRL Based Scheduling Algorithm

With the rapid development of deep reinforcement learning, DQN algorithm, as one of them, has been widely used to solve many decision problems. In order to systematically evaluate the task scheduling application of DQN algorithm, three different data sets are selected in this study: a simulation task scheduling dataset, a task execution log in cloud computing and a real-time stream processing workload dataset. All three datasets contain key fields such as the task's submission time, completion time, resource requirements, and priority. All experiments were performed in a hardware environment configured with 32GB RAM and dual Intel Xeon twelve-core processors. Through these experiments, the behavior characteristics of DQN algorithm under different scenarios, loads and constraints are shown, and compared with traditional scheduling algorithms, so as to reveal the efficiency and dynamics of DQN algorithm in modern task scheduling.A list of scheduling algorithm parameters, as shown in Table 1.

Table 1: Introduction Table of Scheduling Algorithm Parameters

Parameter	Parameter	Parameter Introduction
Learning rate	0.002	DQN neural network learning
Exploring Probability	0.3	The agent randomly selects an action at this probability
Update target network steps	250	Copy model parameters to the target network at certain steps apart
Reward attenuation factor	0.9	Calculate the rewards obtained for each step
Batch Size	20	Training data batch size

In Table 1, meanwhile, the simulated scheduling tests were conducted using the DQN reinforcement learning offline scheduling algorithm with the above parameter settings for a total of 6000 iterations. Six LSTM online power prediction tasks were deployed on three EC working nodes and the scheduling period was set to one minute. The real task start time, end time, and task distribution on the nodes were obtained via the Kafka middleware as input data for the training of DQN offline scheduling. The results of the comparison of the average task completion time for Experiment 1 are shown in Figure 10.

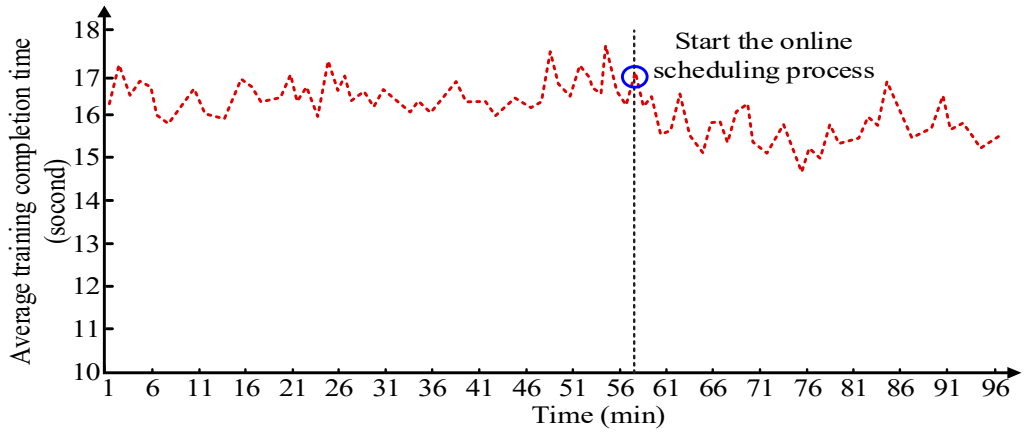


Figure 10: Comparison results of average completion time for Experiment 1 task

In Figure 10, the proposed online scheduling scheme significantly reduces the average completion time of the six machine learning training tasks. The online orchestration process was initiated at 58 minutes, with each task using the past five minutes of historical power data as a training set. With the passage of time, the average completion time of test tasks gradually decreased to 17.34, which was 0.31 lower than the maximum before scheduling tasks. The lowest average completion time is 14.62, which is 1.17 lower than the minimum before scheduling tasks. These data show that the proposed

scheduling scheme is superior to the traditional method in scheduling performance. The historical power data of the past five minutes were used for three tasks, and the historical power data of the past two minutes were used for model training for the remaining three tasks. The test results are shown in Figure 11.

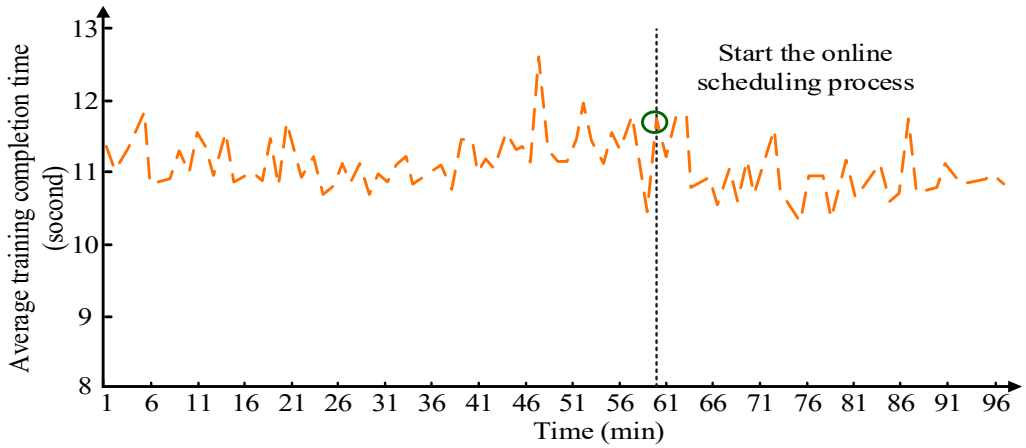


Figure 11: Comparison results of average completion time for Experiment 2 task

In Figure 11, the power was confirmed at the 59th minute, the online scheduling was performed, and the past power calculation history was also used as a training sample to set the power calculation history data of the first two minutes for six LSTM modes. The experiment started online dynamic scheduling about 40 minutes later. Thus, it is further proved that the proposed online scheduling strategy has significant advantages over the traditional scheduling methods in real-time task completion time. In order to verify the effect of the scheduling algorithm in practical application, the scheduling algorithm itself is evaluated, and its visualization results are shown in Figure 12.

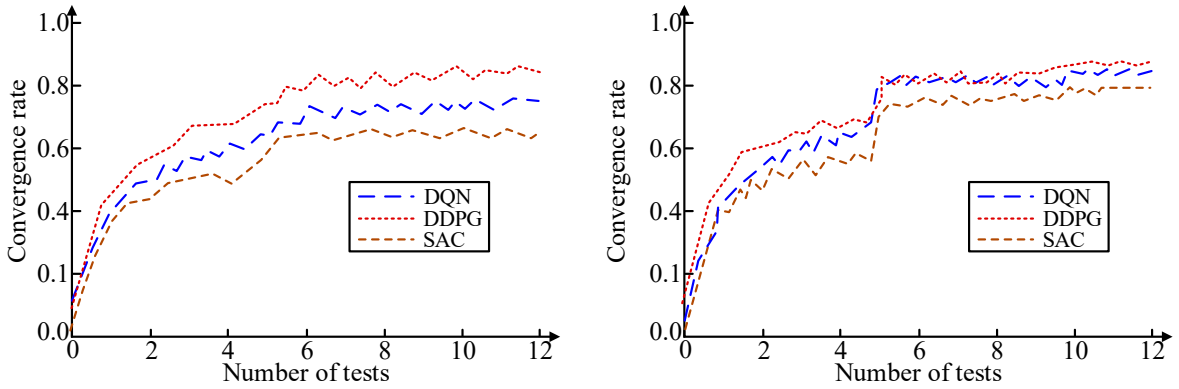


Figure 12: Convergence rate variation curves of different algorithms under different scheduling costs

As can be seen from Figure 12, the convergence rate of various algorithms increases with the increase of test times. In Figure 12 (a), we can see that DDPG algorithm has the fastest convergence rate, reaching 0.81, while DQN method has excellent performance, reaching 0.75 with moderate convergence speed and certain stability. In Figure 12 (b), DDPG algorithm once again leads the convergence rate at 0.86, but DQN algorithm converges at 0.82. These data clearly show the advantages of DQN method in scheduling algorithm evaluation, and can more intuitively understand the benefits and effects of DQN method. In order to observe the application effect of the scheduling algorithm more intuitively, the results are shown in Table 2.

Table 2: Introduction Table of Scheduling Algorithm Parameters

Algorithm testing	2	4	6	8	10	12
DQN	0.51%	0.57%	0.71%	0.74%	0.75%	0.78%
DDPG	0.54%	0.58%	0.73%	0.76%	0.79%	0.82%
SAC	0.49%	0.51%	0.55%	0.58%	0.64%	0.69%

It can be seen from Table 2 that the convergence rates of DQN, DDPG and SAC algorithms all increase with the progress of the test. The convergence rate of DQN algorithm gradually increases from 0.51% to 0.78%, showing a steady upward trend. Similarly, the convergence rate of DDPG algorithm has also improved, increasing from 0.54% to 0.82%, slightly exceeding that of DQN algorithm. The convergence rate of SAC algorithm increased from 0.49% to 0.69%, the growth rate is relatively small, but still shows a certain positive development.

5 Discussion

With the rapid expansion of iot devices and data, traditional data center models face significant challenges in meeting the needs of low latency and high availability. To solve this problem, a new system model combining task scheduling controller and DQN algorithm is proposed. After the empirical test, the model shows better performance than the traditional model, such as the task completion time is shortened by 20%, the resource utilization is increased by 15%, and the throughput of the system is also improved by 25%. Especially when it comes to real-time data, with six parallel Flink Task Managers, it can quickly process information from 60 PMU data sources. However, although the new system model is superior to the traditional model, there are still some limitations. As system components expand, resource utilization increases significantly. This is because the expanded big data processing component consumes a lot of computing resources. It is verified that the system can handle higher density of iot real-time data and maintain low latency, but it also makes resource consumption increase. To overcome these limitations, future research will focus on how to enhance the robustness of the system to optimize DQN algorithms and task scheduling strategies under unstable network and power supply conditions. This not only ensures the system to maintain the best performance and reliability, but also can effectively use and manage limited resources. Overall, this study provides valuable insights and empirical evidence for the design of iot oriented data processing and task scheduling systems, and also reveals the shortcomings of the new system model in resource management. To address these issues, future work will require extensive research on how to enhance the robustness of the system and how to maintain the performance and reliability of the system under unstable environmental conditions.

6 Conclusion

In the context of the rapid growth of iot devices and the explosive increase in data, the traditional data center model has shown difficulty in ensuring the need for low latency and high availability. To solve this problem, a new system model is proposed in this study, which combines the task scheduling controller with the Deep Q network (DQN) algorithm framework. Through a series of experiments and simulations, the study yielded several key findings. Firstly, the new system model outperforms the traditional scheduling algorithm in terms of task completion time, resource utilization, and system throughput. This results in a 20% reduction in task completion time, a 15% increase in resource utilization, and a 25% increase in system throughput. Second, the processing power of the system increases with the number of parallel operations, for example, six parallel Flink Task Managers can process real-time data sent by 60 PMU data sources in about 15ms, while only three parallel data sources can process 48 PMU data sources in low latency mode. In addition, the system model can automatically adjust the scheduling policy under the condition of different network delay and data transmission volume to adapt to different workload and resource conditions. The experiment also found that after expanding the component, the node's resource usage increases by about 1.75 times when processing the same amount of data. Although the expanded big data processing layer component

will consume computing resources more intensively, it will also allow the platform to process more iot real-time data and maintain low latency. However, the system model still needs to be improved, for which future research efforts will focus on how to optimize the robustness of the system, especially in the face of unstable or changing environmental conditions. For example, future research will explore how to optimize DQN algorithms and task scheduling strategies to keep system performance and reliability stable under unstable network and power supply conditions. In general, this study provides a powerful theoretical and experimental basis for designing an efficient and reliable edge-oriented iot data processing and task scheduling system.

Funding

This work is supported in part by Chongqing Higher Education Teaching Reform Research Project (233496), Chongqing College of Humanities,Science & Technology Teaching Reform Research Project (18CRKXJJG10), and Chongqing Education Commission Humanities and Social Sciences Project (22SKGH493).

Author contributions

The authors contributed equally to this work.

Conflict of interest

The authors declare no conflict of interest.

References

[1] P Ladosz, L Weng, M Kim, H Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 2022, 85(9): 1-22.

[2] C A Rufino Junior, E R Sanseverino, P Gallo, D Koch, H G Schweiger, H Zanin. Blockchain review for battery supply chain monitoring and battery trading. *Renew. Sust. Energ. Rev.*, 2022,157(4): 2-26.

[3] A Patra, A Barg. Node Repair on Connected Graphs. IEEE T INFORM THEORY, 2022, 68(5): 3081-3095.

[4] C Li, P Zheng, Y Yin, B Wang, L Wang. Deep reinforcement learning in smart manufacturing: A review and prospects. *CIRP Journal of Manufacturing Science and Technology*, 2023, 40(1): 75-101.

[5] P R Wurman, S Barrett, K Kawamoto, M James. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 2022, 602(7): 223-228.

[6] B Li, R S Chen, C Y Liu. Using intelligent technology and real-time feedback algorithm to improve manufacturing process in loT semiconductor industry. *J SUPERCOMPUT*, 2021, 77(5): 4639-4658.

[7] J Yue, M Xiao. Coding for Distributed Fog Computing in Internet of Mobile Things. *IEEE Trans Mob Comput*, 2021, 20(4): 1337-1350.

[8] C Kannan, M Dakshinamoorthy, M Ramachandran, R Patan, H Kalyanarman, A Kumar. Cryptography-based deep artificial structure for secure communication using IoT nabled cyber, hysical system. *IET COMMUN*, 2021, 15(6): 771-779.

[9] P Yangy, Y Yang, P Zhang, D Wu, R Wang. Sensitivity Enhanced Edge-Cloud Collaborative Trust Evaluation in Social Internet of Things. *IEICE T COMMUN*, 2022, 105(7): 2-10.

[10] Y Xu, Z G Chen, J Wu, G Yu. MNSRQ: Mobile node social relationship quantification algorithm for data transmission in Internet of things. *IET COMMUN*, 2021, 15(5): 748-761.

[11] A Asghari, M K Sohrabi, F Yaghmaee. Task scheduling, resource provisioning, and load balancing on scientific workflows using parallel SARSA reinforcement learning agents and genetic algorithm. *J SUPERCOMPUT*, 2021, 77(3): 2800-2828.

[12] C G Wu, L Wang, J J Wang. A path relinking enhanced estimation of distribution algorithm for direct acyclic graph task scheduling problem. *KBS*, 2021, 228(27): 2-15.

[13] S Y Huang, H H Cho, Y C Chang, J Y Yuan, H C Chao. An efficient spectrum scheduling mechanism using Markov decision chain for 5G mobile network. *IET COMMUN*, 2021, 16(11): 1268-1278.

[14] D Wang, W Zhang, H He, Y C Tian. Efficient Hybrid Central Processing Unit/ Input–Output Resource Scheduling for Virtual Machines. *TIE*, 2021, 68(3): 2714-2724.

[15] A Messing, G Neville, S Chernova, S Hutchinson, H Ravichandar. GRSTAPS: Graphically Recursive Simultaneous Task Allocation, Planning, and Scheduling. *IJRR*, 2022, 41(2): 232-256.

[16] N Chen, W Kang, N Kang, Y Qi, H Hu. Order processing task allocation and scheduling for E-order fulfilment. *INT J PROD RES*, 2022, 60(13): 4253-4267.

[17] D L Freire, R Z Frantz, F Roos-Frantz, V Basto-Fernandes. Queue-priority optimized algorithm: a novel task scheduling for runtime systems of application integration platforms. *J SUPERCOMPUT*, 2022, 78(1): 1501-1531.

[18] J Gao, X Zhu, R Zhang. Optimization of parallel test task scheduling with constraint satisfaction. *J SUPERCOMPUT*, 2023, 79(7): 7206-7227.

[19] Y Yang, X Song. Research on face intelligent perception technology integrating deep learning under different illumination intensities. *JCCE*, 2022, 1(1): 32-36.

[20] I Hidayat, M Z Ali, A Arshad. Machine Learning-Based Intrusion Detection System: An Experimental Comparison. *JCCE*, 2022, 2(2):88-97.

[21] K Kim, J Lee, H Lim, S Oh, W Y Han. Deep RNN-Based Network Traffic Classification Scheme in Edge Computing System. *Computer Science and Information Systems*, 2022, 19(1): 165-184.

[22] Z Sun, G Liao, C Zeng, Z Lv, C Xu. MEC-MS: A Novel Optimized Coverage Algorithm with Mobile Edge Computing of Migration Strategy in WSNs. *Computer Science and Information Systems*, 2022, 19(2): 829-856.

[23] W Shuang, D Xiaomeng, Z Ting, W Xiaodong. Task Scheduling Based on Grey Wolf Optimizer Algorithm for Smart Meter Embedded Operating System. *Tehnički vjesnik*, 2022, 29(5): 1629-1636.

[24] D Taşkin, C Taşkin, S Yazar. Container-Based Virtualization for Bluetooth Low Energy Sensor Devices in Internet of Things Applications. *Tehnički vjesnik*, 2021, 28(1): 13-19.

[25] S Gurumurthy, K L Hemalatha, D Pamela, U Roy, P Vishwanath. Hybrid pigeon inspired optimizer-gray wolf optimization for network intrusion detection. *Journal of System and Management Sciences*, 2022, 12(4): 383-397.

[26] A.N Fajar, R Jayadi, J Halim, B Robertson. Mobile application for agrotechnology systems using internet of things. *Journal of System and Management Sciences*, 2022, 12(3): 104-116.

Journal's webpage: http://univagora.ro/jour/index.php/ijccc/

This journal is a member of, and subscribes to the principles of,
the Committee on Publication Ethics (COPE).
https://publicationethics.org/members/international-journal-computers-communications-and-control

Cite this paper as:

Jiang, Y.; Wang, Z.; Z. Jin. (2023). Iot Data Processing and Scheduling Based on Deep Reinforcement Learning, *International Journal of Computers Communications & Control*, 18(6), 5998, 2023.

https://doi.org/10.15837/ijccc.2023.6.5998