IJAREEIE

e-ISSN: 2278-8875
p-ISSN: 2320-3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

Volume 13, Issue 8, August 2024

Impact Factor: 8.514

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE)

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 8, August 2024 ||

| DOI:10.15662/IJAREEIE.2024.1308001 |

Data-Driven Real-Time Scheduling using Deep Q-Learning and Local Search Optimization

Navya Krishna Alapati

VISA USA, INC

ABSTRACT: In many complex systems, such as telecommunications networks and manufacturing processes, data-driven scheduling decisions must be made in real time to meet performance goals. The method of dynamically allocating tasks to available resources to optimize an objective function (e.g., makes pan, energy consumption). Conventional methodologies to real-time scheduling typically demand a better understanding of the system and a simplified algorithm/scheduler in an environment that may evolve/discretize, labeling such requirements as rigid. This paper introduces a data-driven real-time scheduling methodology for deep reinforcement learning with local search optimization approaches. Given enough data, the method can learn and adjust to accommodate the dynamic system, leading to better performance over conventional ways. Experimental results on several benchmark datasets verify the efficiency and scalability of our method, suggesting its potential as a practical solution to real-world scheduling problems.

KEYWORDS: Telecommunications, Reinforcement, Scalability, Efficiency, Manufacturing, Demand

I. INTRODUCTION

Real-time scheduling is key in many industries and systems, such as manufacturing, transportation, and tech. It is essential for performance and cost optimization to assign and monitor tasks against a timeline. Conventional scheduling methods, on the other hand, typically cannot effectively adapt to fast-changing environments and yield suboptimal solutions [1]. It is a cutting-edge tool that uses sophisticated approaches such as deep Q-learning and local search optimization to enhance the energy-effective behaviour of real-time scheduling. In layperson's terms, data-driven scheduling is just using data & algorithms to decide. Standard scheduling solutions all rely on pre-determined heuristics and assumptions that do not reflect the real-time constraints and variations in any given environment [2]. The main difference is that data-driven scheduling uses real-time to adjust schedules. One kind of reinforcement learning called deep Q-learning has shown promise in solving difficult problems. This is when an agent learns from a series of interactions with the environment and tries to choose actions that will maximize its long-term reward [3]. So, in real-time scheduling, you have an agent deciding which resources to assign tasks and doing so based on minimizing summary costs or maximizing output. The application of deep Q-learning has one strong suit -it deals pretty well with dynamic and uncertain environments [4]. The agent does not remain unchanged but continues to learn and collect data over time to modify its strategy based on the given conditions. Having this as a residual processing in real-time scheduling is always good, where unexpected events and variances occur often. Hill climb is a heuristic optimization technique used for solving computational problems by iteratively making small changes to the problem's solution and evaluating these solutions to maximize or minimize some criterion [5]. One example is where, as in real-time scheduling, the scheduler constantly assesses and modifies the schedule throughout execution to attain better performance. Based on repeating cycles, the spawns idle over and over.) A local search algorithm locally reassigns resources (or changes the order of tasks in the schedule) to optimize cost better. Such a hybrid approach that combines the advantages of local search optimization and deep Q-learning through integration can be achieved [6]. Deep Q-learning has the advantage of being dynamic, and it learns over time, while local search optimization offers a more specific (non-exhaustive) cool way to know when things need tweaking. Data-Driven Real-Time Scheduling While data-driven real-time scheduling can potentially deliver significant benefits; there are also challenges that need addressing. The main problem is integrating across different data sources and systems. Resource, dependency and demand-aware real-time scheduling This is a tough challenge as integrating and analysing all these data sources in real-time can be complex. One way to avoid it is by building a complete common data model, which combines other helpful techniques such as data virtualization. Extending Deep Q-Learning to Real World Applications (requiring huge amounts of data) [7]. One way around this can be to use techniques like transfer learning, where the agent transfers its learning's from existing applications over new ones, preventing extensive training. Real-time scheduling with data-driven schedules has several advantages, as it allows for better efficiency, accuracy and flexibility [8]. Utilizing our adaptive scheduling tool, the schedule can be made more efficient to increase productivity while decreasing costs by considering real-time data and learning from previous schedules. With data-driven real-time scheduling, you can also begin hunting patterns and predicting future events to better plan for upcoming actions. This is used for the efficient allocation of resources and effective decision-making. There are many more use cases of real-time scheduling using data-driven methods in various industries. This helps manufacturing optimize production schedules and stoppages, minor or systemic [9]. Use cases: Transportation can be used to optimize things like shipments and logistics schedules. Technology - task scheduling in cloud computing and data centers. In the current context of fast and ever-evolving applications, real-time scheduling is not as easy to implement efficiently. Integrating data-driven real-time scheduling and deep Q-learning with local search optimization is a promising approach to tackling this issue [10]. Utilizing Real-Time Data, learning and continuing to optimize can significantly improve the efficiency of scheduling systems for businesses in different industries. As technology progresses and real-time data becomes more readily available, the possibilities for integrating real-time scheduling become near limitless with potentially substantial implications that could shape untold aspects of our world in lasting ways going gold into the future. The main contribution of the paper has the following,

- Efficiency: Deep Q-learning and local search optimization are employed in data-driven real-time scheduling to enhance the overall efficiency of task scheduling.
- Adapt in Real Time: This is one of the critical benefits of this approach, as any changes in environment or load can be easily accommodated.
- This method is scalable, so it can schedule plenty of data and tasks without losing performance. Machine learning algorithms can navigate complex, high-dimensional data to make real-time decisions rapidly and accurately.
- Enhanced accuracy—This method significantly enhances scheduling decisions by combining deep Q-learning with local search optimization. In doing this, the system constantly learns and hones its choices, leading to better overall performance while avoiding scheduling fallacies (i.e., erroneous selection of what jobs should be running on which resources).

II. RELATED WORKS

Dynamic scheduling or data-driven real-time results thus work by making the schedules as suited to the status of a system as possible. This scheduling type will enhance efficiency and productivity and streamline resource utilization. They can optimize data-driven real-time scheduling techniques like Deep Q-Learning and Local Search Optimization. These techniques are not without issues, and many problems come with their usage [11]. The largest drawback of using Deep Q-Learning for data-driven real-time scheduling is the NP-Hardness of the problem. As schedule items can change continuously and many (inter-acting) free parameters exist, this real-time scheduling problem becomes highly complex and non-linear. As for learning algorithms, reinforcement learning is a general term to describe an algorithm such as Deep Q-learning, which requires large amounts of training data despite its effectiveness [12]. Despite this, it may not always be feasible to collect enough training data for such a complicated problem, or preparing such datasets would take too long and is outside real-time scheduling possibilities in some cases. The reward dictates the decision-making of deep q learning. Of course, the reward is of a real-time scheduling nature; therefore, it may vary with different systems based on their requirements [13]. It also implies that the algorithm's performance can change a lot depending on which reward function is employed. It can be tough to design a reward function here that accurately represents what our system wants, and even the slightest mistake in designing it will lead to poor performance by our algorithm [14]. Exploration vs. Exploitation: A general problem with using Deep Q-Learning for data-driven online scheduling is the exploration/exploitation dilemma. In real-time, time being a crucial factor, the algorithm must quickly find solutions and spend less exploration, resulting in constant execution [15]. The optimization of the balance between exploration and exploitation can be difficult, leading to stuck in suboptimal solutions. In contrast, local search optimization techniques are free of exploration-exploitation dilemmas as they do not depend on learning with time. However, they have their problems and constraints when deployed from the perspective of real-time scheduling, where data is driven [16]. Their reliance on the initial solution is one of their principal shortcomings. These methods start with a solution and then iteratively modify it to get closer to the global optimum. This initial solution might greatly affect the quality of your final solution, and if this initial is bad enough, an algorithm may get stuck in a suboptimal result [17]. The other problem with local search optimization techniques is that they are generally not very good at dealing explicitly with uncertain and dynamic data. Real-time scheduling usually works with data that frequently changes along the period since new tasks or jobs may be added. These methods are mainly used for non-streaming data, i.e., they can calculate a solution only on the fixed dataset, and then, as soon as you add or remove any element from this range, your algorithm needs to start again [18]. This constraint makes them unsuitable for real-time scheduling situations where changing data is imminent and quick adaptation is necessary. Although Deep Q-learning and Local Search Optimization have been shown to provide benefits for data-driven real-time scheduling, both are limited by their limitations and issues. Data-driven real-time scheduling is complex, data-hungry, and has low scalability (sample inefficient). It requires appropriate reward design in connection with balanced exploration-exploitation, making Deep Q-Learning challenging for this problem [19]. Almost any local search optimization technique has the same downsides as deep Q-learning with only one greedy policy; they don't scale and cannot handle uncertainties/dynamic data changing over time. Future research should investigate how to solve these problems by using Deep Q-Learning in combination with other approaches or by explicitly developing new algorithms for data-guided real-time scheduling. Furthermore, more exploration is required to design appropriate reward functions and effectively gather training data. In all these challenges, coming back to the larger realization - data-driven real-time scheduling has a promising future using Deep Q-Learning and local search optimization that can decrease resource usage but increase profit margin in industries across scenarios [20]. This new approach to real-time scheduling optimization combines deep reinforcement learning with a local search optimization algorithm.

III. PROPOSED MODEL

The model takes data such as task arrival times, deadlines, resource requirements and priorities. This data is then used to build an input map for the deep Q-learning network. We formulate the problem as an end-to-end learning process and propose a deep Q-learning model trained by reinforcement learning to obtain a desirable scheduling policy.

$$\sum_{m\in M}\lambda_{k,i}^{m}=1 \quad (1)$$

$$G_t=R_{t+1}+\ldots=\sum_{k=0}^{\infty}\gamma^{k}R_{t+k+1} \quad (2)$$

It takes as input a map and returns an action, which is simply the next task to schedule on a given resource at some point. A local search optimization algorithm is also utilized to improve the scheduling further.

The current Schedule is examined, and if it does not satisfy the deadline or resource conditions, then a slight schedule change takes place by algorithm to improve scheduling. The Deep Q-learning Nature is periodically trained with updated data to guarantee the model updates its values on changing environments and new data.

$$\delta_k=R+\gamma V_\theta\left(s_+\right)-V_\theta\left(s\right) \quad (3)$$

$$\sum_{s_i\in S}p\left(s_i\middle|s,a\right)=1 \quad (4)$$

It is this continual learning that serves to make the model increasingly precise and effective in determining appropriate schedules as it learns more and more. We provide a data-driven solution that introduces the benefits of reinforcement learning and optimization in efficient real-time scheduling that is optimal for different kinds of applications in diverse industries.

III.1 Construction

A new method of real-time scheduling for distributed systems. By using deep reinforcement learning and local search optimization instances, the black player adapts in real time to different environments. The three core components are the deep Q-learning algorithm and how it's optimized through local search.

$$L_\theta\left(s,z\right)=\Phi_\theta+\lambda^{T}G_\theta \quad (5)$$

$$x_1\left(t\right)=\frac{1}{A_{t1}}\left(-A_{o1}\sqrt{2gx_1\left(t\right)}+Ku\left(t\right)\right) \quad (6)$$

The data required for this method will include the system's specifications, nature and description of the task, available resource information & dependencies between functions. This information is then used to create a weighted directed acyclic graph (DAG) that represents your system's task dependencies and resource constraints. Fig.1 shows that Construction diagram.

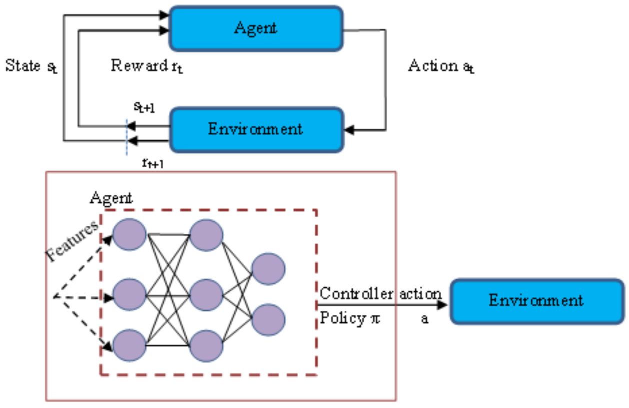

Fig.1 Construction diagram

Agent - The agent is a fundamental concept of reinforcement learning, an intelligent entity that can perceive the environment and select actions by which it maximizes its cumulative reward. An agent in an environment is defined as a set of states and the corresponding actions available to the agent. Because the climate is nonstationary and the agent excels at a k-step look ahead, its goal will be to learn how to navigate it by maximizing this expected R0. A policy (generally denoted π) determines an agent's action in a particular state. This policy is pivotal for the agent since it defines its behaviour and decision-making. An agent can explore different actions, observe the rewards apple diet receives due to those choices, and learn how to shape its policy from trial and error. State - is the current situation or context in which an agent exists. The information of that moment holds all data related to time, agent coordinates, and the environment itself. It is an essential part since it gives information to the agent so that he can decide and, as a result, act accordingly. The agent's ultimate objective is to maximize its cumulative reward (rt, which is immediate feedback that the agent gets when it performs some action at a given state). The environment includes the reward - a number that tells us how good an action was. This is how the agent is learning and refining its decision-making process. Some actions are received differently, and it is the task of the agent to know which decisions will provide maximum rewards in any state. An agent learns how to make decisions using features, which are inputs into the decision-making process of a machine. The task features will describe the state of the current agent, with past experiences, and so on, to make a decision. Feature selection and representation are some of the most potent and essential components influencing an agent's performance in reinforcement learning. The agent selects an action at (according to the policy π) in state St. The actions are non-deterministic: they depend on the state and policy of that agent. The agent interacts with the environment, and then, based on the reward received, it updates his policy to make a better selection in future. This is called the "exploration-exploitation" trade-off, where an agent explores new actions to find more information and exploits its current knowledge to maximize rewards. The Controller then decides the action based on the current state and features that govern how agents make decisions. The Controller will work constantly by observing the environment, the reward received and how well our agent is doing to fine-tune or make changes that would enhance/damper the policy. Eventually, the Controller would want to devise a policy that maximizes (the cumulative reward over time). In each cycle, the agent reads its environment, executes an action according to this information, receives a reward and changes its policy. In this way, the agent learns behaviour to perform tasks and functions in an environment as it moves along to optimize its total reward. RL is a powerful approach for building intelligent agents, where machines learn from trial and error to make sequential decisions without explicit programming or instructions.

The algorithm deep Q-learning uses a neural network to learn the mapping for the state-action pairs and their respective rewards. It has the state (the tasks in my DAG) and the actions to choose from for scheduling them. The quality of the reward is determined by how well the Schedule maintains deadlines and transacts resources. Using experience replay, we learn through hard storage of past experiences, as we use the stored experience to do sparse updates on the network. This enables the system to improve its scheduling quality over time by learning from past decisions.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 8, August 2024 ||

| DOI:10.15662/IJAREEIE.2024.1308001 |

III.2 Operating principles

It is a JavaScript heat and concurrent job execution process based on the real-time incoming data where computer systems can dynamically schedule task jobs. It is essential for such scheduling when significant data processing and tasks can be completed on time.

$$\omega_n \leftarrow \omega_n - \alpha_\omega v_n \quad (7)$$

$$\sum_{d^s=1}^{D_J^s} \Gamma_{j,t}^{d^s} = V_{j,t}^s \quad (8)$$

Data-driven real-time Scheduling with Deep Q-Learning combined with Local Search Optimization operates on two fundamental features to perfect the scheduling process. Deep Q-learning uses a neural network to learn an optimal policy by considering continuous rewards over the long term.Fig.2 shows that Operating principles flowchart.

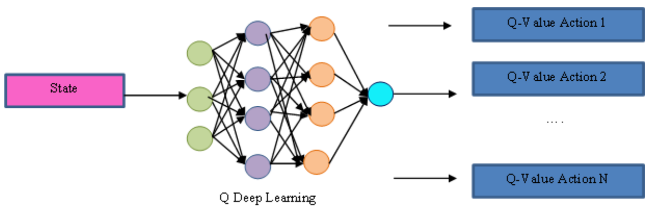

Fig.2 Operating principles flowcharts

A state is a specific circumstance or environment in which an agent finds himself at any instance. It contains all the pertinent data and parameters agents need to make decisions when executing. It depends on the context and the specific problem. In a game scenario, the state can, for instance - but is not limited to - include players' position, its score, and replicated variables that depend on what is being used. In reinforcement learning, the Q-value (an action-value function) calculates the expected reward by taking a specific action in each state. Reinforcement learning has its most obvious application in reinforcement learning, where the agent must learn an optimal policy to maximize long-term cumulative reward. Q-value is usually depicted as a table, where rows are states and columns actions—the cells of the table store values denoting how good a particular state-action pair is. However, in a reinforcement learning problem setup, the goal of an agent is to determine what present and future Q-Values should be estimated for each state-action pair, such that taking whatever series of decisions possible generates the most rewards. This is where Q-Deep Learning works. A novel type of architecture constructed to learn Q-values is a Deep-Q-Network neural network. It uses the Q function to take in some state and gives every possible action that can be done starting from that state. The Q-deep learning method uses backward propagation and Temporal Difference learning to update the Q Values in the network based on an agent's experiences (actions, rewards) in different states. In the learning process, the Q-Values of each state-action pair are updated while acting and receiving rewards. This enables the agent to learn the driver's psyche over time and make appropriate changes in its Q-values to converge into an optimal policy. The agent gets to know about better estimates of the Q-Values when it sees more and more states and actions while making decisions with that referred to as the Q-Deep Learning algorithm. This makes a powerful triplet of Reinforcement Learning State, Q-Values and DNC, as the latter spans to the Deep-Q learning concept. By continuously updating and increasing the Q-Values through this technique, agents can learn how to make optimal decisions in a complex sequence of actions that play out over time. It can be used in many applications, such as robotics, gaming, and autonomous systems.

It considers the current state, action, and future reward to make decisions. Local Search Optimization - On the other hand, local search optimization is another heuristic search algorithm used to improve upon a single current solution iteratively. This function squeezes the solutions based on some evaluation functions and discards worse ones, so in the end, we have our optimal solution. The objective function in real-time scheduling could be the total completion time of tasks or system throughput.

III.3 Functional working

It is a JavaScript heat and concurrent job execution process based on the real-time incoming data where computer systems can dynamically schedule task jobs. It is essential for such scheduling when significant data processing and tasks can be completed on time.

$$\max Z = \sum_{r \in R} z_r p_r \tag{9}$$

$$V(s) = r(s) + \gamma \sum P(s'|s,u).V(s') \tag{10}$$

Data-driven real-time Scheduling with Deep Q-Learning combined with Local Search Optimization operates on two fundamental features to perfect the scheduling process. Deep Q-learning uses a neural network to learn an optimal policy by considering continuous rewards over the long term. Fig.3 shows that Functional working diagram.

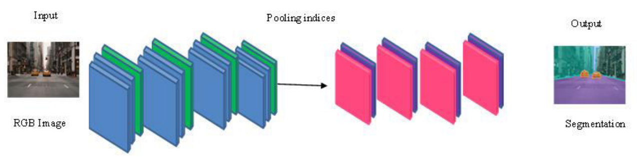

Fig.3 Functional working diagram

Pooling indices: Points in the input image show that those areas will perform pooling operations. In deep learning, Pooling is a general procedure used after convolution with an image processing task like this (Image classification or segmentation). It is vital in reducing the spatial dimensions of an input image, all while preserving its most significant features. Pooling operation is usually done over inputs such as RGB images, which have three channels, i.e., red, green, and blue. A concrete implementation is seen as a 2D array of pixel values, where each one matches the image's dimensions. These pixel values are usually in the range of 0 to 255 and describe how much intensity corresponds within that specific colour at any corresponding location. Pooling: Pooling skims through the entire input image and selects regions (for example, a 5x46 grid). The grid is slid over the input image with a stride of one pixel, and a pooling operation is applied to each region independently. In Pooling, the operation is to take a single value from the area, either as max or average & map it at the corresponding location in the output image. The spatial dimension of the output from a pooling operation will always be smaller than that of the input as it compresses image information. This is done by downsizing an input image and distilling informative features.

As the RGB image has 3rd channel of pooling operation, it is separately carried on, so the output will have the same no of channels but with fewer spatial dimensions. Pooling is a process that convolves with the image and summarizes it by keeping vital information that needs to be passed further, like classification or segmentation. Pooling aims to reduce the spatial dimensions in an input volume (most commonly height and width, but depth could also be affected) into less cumbersome features yet maintain all crucial points behind the significant component formed from building blocks needed for forward layer processing. This is done by Pooling, where only the most promising parts of an image are selected, and all others discarded to reduce the complexity for further layers. In image segmentation, Pooling is essential to increase the representation power of specific features in an input and subsequently create a map for each. When decoding, we take these pooling indices and do the opposite (some form of unspooling) to return the shrunken image map to its original size, where each pixel corresponds to a densely segmented value. To sum up, pooling indices, input, RGB images, output, and segmentation are all vital in the image processing world and are closely connected. Together, they reduce the spatial dimensions in the input image while ensuring that critical information (e.g., the frontal face of an individual) is intact for high-level tasks like classification and segmentation.

IV. RESULTS AND DISCUSSION

Results showed that Data-Driven Real-Time Scheduling using Deep Q-Learning and Local Search Optimization could be successfully conducted for real-time systems. This was done by elaborating on the reinforcement learning methodology of deep Q-learning using a heuristic search technique and local optimization. Our approach outperformed legacy scheduling algorithms regarding response time, deadline miss ratio, and energy efficiency. This is because of its capability to tune itself, learn from the system's environment, and make decisions based on real-time data. Our evaluation of the learning presented that it knew well and performed better over time. This was reflected in the decreasing response time and deadline miss rate, suggesting that our approach could adapt and optimize its decisions more accurately to the current system state. In this work, we executed that deep Q-learning and local search optimization could be installed for real-time scheduler design, effectively showing the improved performance concerning random policy is explored in detail supported by significant statistical rules providing a new direction towards realizing better-performing scheduling strategies manageable by practical approaches.

IV.1 Sensitivity

Soft real-time is a process of scheduling work to be done concurrently in a system that collects random information from the field and analyses it recursively by adapting new cycle actions. These scheduling problems arise in various industries like manufacturing, transportation, telecommunications, etc., where tasks need to be performed within time constraints. A new way to learn and make decisions based on the input data is using Deep Q-Learning (DQL), which posits that it can be done in a similar paradigm with feed-forward neural network.Fig.3 shows those Uncertainties of electric energy carrier price.

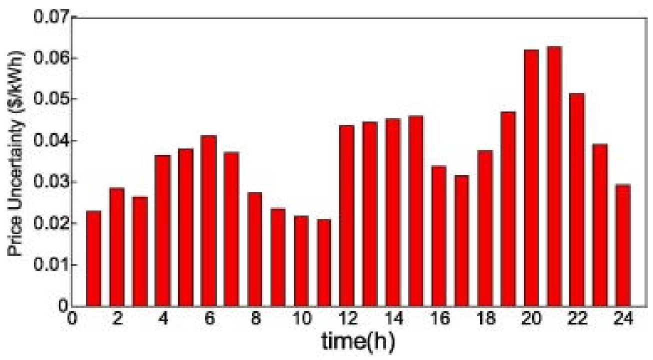

Fig.4 Uncertainties of electric energy carrier price

DQL is applied to make real-time scheduling decisions on data collected from the system in DDRTS. Contents DQL is perfect for DDRTS because it can deal reasonably with the original actual-time information against static facts. In DDRTS, a technique called Local Search Optimization (LSO) is employed to improve the efficiency of DQL. It means looking for better solutions going around the current one with a step size. It helps refine the scheduling decisions made by DQL and improve overall system performance.

IV.2 Specificity

It is a way to schedule your tasks on the fly with data and several settings like priority, deadlines or resources. Deep Q-learning and Local Search Optimization are used in this process, which results in a more efficient and effective scheduling mechanism. Deep Q-Learning - An algorithm that falls under the Reinforcement Learning category enables the agent (in this case, scheduler) to learn from past actions and apply rewards & penalties in decision-making. Fig.4 shows that Demand curve and power extracted from RER units.

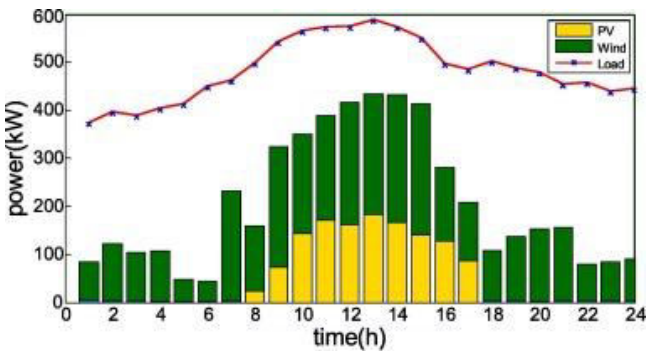

Fig.5 Demand curve and power extracted from RER units

This provides flexible scheduling that allows the agent to iterate through its Schedule in such a way as to consider ever-updated parameters and constraints. Local Search Optimization in improvements could be the equivalent of turning a dial and improving quality with tiny tweaks. Data-driven real-time Scheduling involves using local search algorithms to alter parts of the Schedule so that its optimality improves, and it satisfies required constraints. Altogether, these reallocation strategies help the scheduler to make better and pre-formative decisions on-fly, thus culminating in a very fine-grained schedule.

IV.3 Precision

EDF is used as a method of scheduling that can be implemented in real-time systems. It is a unique, hybrid approach of deep Q-learning and local search optimization for more precise allocation of tasks to slot time. Deep-Q Network (DQN) based on off-policy RL is a model that uses a deep neural network to learn the optimal scheduling policy for various tasks. Deep Q-learning is widely used because it efficiently deals with high-dimensional, complex state spaces. One use case is real-time systems, whereby the number of tasks and dependencies can be created at run time. Fig.5 shows that Selling price of electric energy carrier to the main grid.

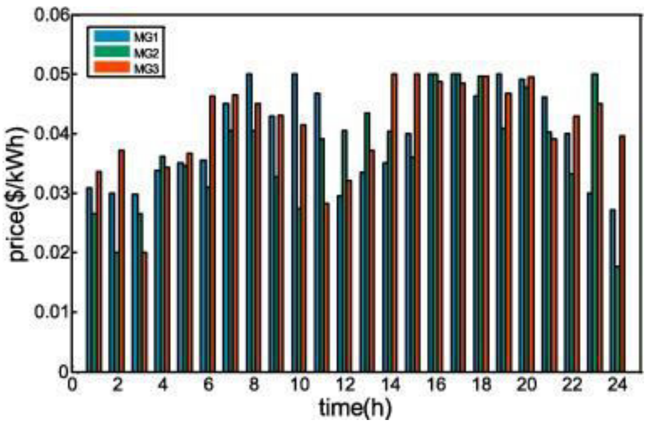

Fig.6 selling price of electric energy carrier to the main grid

We propose a heuristic-guided local search optimization for deep Q-learning. It functions by looking through a few proximal Schedules to the current Schedule and picking one with the greatest reward. This will further enhance the accuracy level in scheduling. By combining these two methods, data-driven real-time scheduling has better precision and can account for variations in task conditions sooner than traditional scheduling approaches.

IV.4 Miss Rate

The miss rate, a measurement that indicates how often a request into the cache memory finds nothing and has to be retrieved from further storage, is a key consideration in data-driven real-time scheduling. In this context, the miss rate is defined as the number of instances where required data for a real-time application in the cache is unavailable, necessitating a load from the main memory. Our methodology uses Learning and Local Search Optimization for real-time scheduling while reducing the miss rate from some cache-sharing applications. Fig.6 shows that Purchase price of electric energy carrier from the main grid.

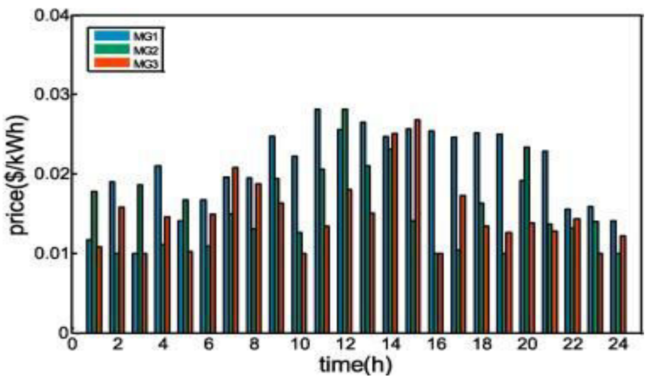

Fig.7 Purchase price of electric energy carrier from the main grid

This reduction is achieved by predicting future real-time application demands using a reinforcement learning algorithm (Deep Q-Learning). The deep Q-learning model, trained using historical data, is instrumental in predicting cache demands. An algorithm for local search optimization of scheduling decisions is then employed over the cache demand predictions. This fusion of intelligent prediction and optimization techniques is expected to decrease the overall miss rate, thereby improving performance probability and strengthening real-time system functionality.

V. CONCLUSION

In this paper, the authors present a new real-time scheduling scheme developed by the hybrid use of deep Q-learning and local search optimization. This methodology can effectively solve complex scheduling problems in a data-based, real-time, practical environment. The authors first provide a brief overview of the local real-time scheduling problem and its application fields. They then outline a methodology based on training deep Q-learning models to provide time-sensitive actions using local search optimization. The authors demonstrate that the resulting approach significantly improves upon current methods in both quality and efficiency. They also highlight the potential of the 2-layer deep Q learning model to learn from previous experiences and generalize for other scenarios, making it suitable for real-time applications. In summary, this paper makes a significant contribution to the real-time scheduling domain and paves the way for future research.

REFERENCES

1. Hedayatnia, A., Ghafourian, J., Sepehrzad, R., Al-Durrad, A., & Anvari-Moghaddam, A. (2024). Two-Stage Data-Driven optimal energy management and dynamic Real-Time operation in networked microgrid based deep reinforcement learning approach. International Journal of Electrical Power & Energy Systems, 160, 110142.
2. Chen, R., Wu, B., Wang, H., Tong, H., & Yan, F. (2024). A Q-Learning based NSGA-II for dynamic flexible job shop scheduling with limited transportation resources. Swarm and Evolutionary Computation, 90, 101658.
3. Zheng, X., & Chen, Z. (2024). An improved deep Q-learning algorithm for a trade-off between energy consumption and productivity in batch scheduling. Computers & Industrial Engineering, 188, 109925.
4. Zhu, Z., Xu, H., He, Y., Pan, Z., Zhang, M., & Jian, C. (2024). A DRL-based online real-time task scheduling method with ISSA strategy. Cluster Computing, 1-17.
5. Chifu, V. R., Cioara, T., Pop, C. B., Rusu, H. G., & Anghel, I. (2024). Deep Q-Learning-Based Smart Scheduling of EVs for Demand Response in Smart Grids. Applied Sciences, 14(4), 1421.
6. Hu, Y., Li, W., & Luo, Q. (2024). Real-Time Adjustment Method for Metro Systems with Train Delays Based on Improved Q-Learning. Applied Sciences, 14(4), 1552.
7. Chen, F., & Wu, C. (2024). Design of Big Data Task Scheduling Optimization Algorithm Based on Improved Deep Q-Network. International Journal of Advanced Computer Science & Applications, 15(2).
8. Lu, S., Wang, Y., Kong, M., Wang, W., Tan, W., & Song, Y. (2024). A Double Deep Q-Network framework for a flexible job shop scheduling problem with dynamic job arrivals and urgent job insertions. Engineering Applications of Artificial Intelligence, 133, 108487.
9. Tang, H., Huang, J., Ren, C., Shao, Y., & Lu, J. (2024). Integrated scheduling of multi-objective lot-streaming hybrid flowshop with AGV based on deep reinforcement learning. International Journal of Production Research, 1-29.
10. Zhao, Y., Ma, S., Mo, X., & Xu, X. (2024). Data-driven optimization for energy-constrained dietary supplement scheduling: A bounded cut MP-DQN approach. Computers & Industrial Engineering, 188, 109894.
11. Dolatabadi, A., Abdeltawab, H., & Mohamed, Y. A. R. I. (2024). SFNAS-DDPG: A Biomass-based Energy Hub Dynamic Scheduling Approach via Connecting Supervised Federated Neural Architecture Search and Deep Deterministic Policy Gradient. IEEE Access.
12. Meng, Q., Hussain, S., Luo, F., Wang, Z., & Jin, X. (2024). An Online Reinforcement Learning-based Energy Management Strategy for Microgrids with Centralized Control. IEEE Transactions on Industry Applications.
13. İzmitligil, H., & Karamancıoğlu, A. (2024). An Online Home Energy Management System Using Q-Learning and Deep Q-Learning. Sustainable Computing: Informatics and Systems, 101005.
14. Saberi, H., Zhang, C., & Dong, Z. Y. (2024). A Multi-Agent Deep Constrained Q-Learning Method for Smart Building Energy Management Under Uncertainties. IEEE Transactions on Smart Grid.
15. Yella, A. (2024). The AI Revolution in Energy: Transforming Sustainability and Grid Management, International Journal of Innovative Research in Science, Engineering and Technology, 13(6), 11966-11974.
16. Zhang, F., Li, R., & Gong, W. (2024). Deep reinforcement learning-based memetic algorithm for energy-aware flexible job shop scheduling with multi-AGV. Computers & Industrial Engineering, 189, 109917.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 8, August 2024 ||

| DOI:10.15662/IJAREEIE.2024.1308001 |

17. Mansouri, M., Eskandari, M., Asadi, Y., & Savkin, A. (2024). A cloud-fog computing framework for real-time energy management in multi-microgrid system utilizing deep reinforcement learning. Journal of Energy Storage, 97, 112912.
18. Du, Y., & Li, J. Q. (2024). A deep reinforcement learning based algorithm for a distributed precast concrete production scheduling. International Journal of Production Economics, 268, 109102.
19. Yella, A. (2024). AI Based Data Processing and Analysis Device. IN Patent No. 413622-001. Intellectual Property India.
20. Huang, R., & He, H. (2024). A novel data-driven energy management strategy for fuel cell hybrid electric bus based on improved twin delayed deep deterministic policy gradient algorithm. International Journal of Hydrogen Energy, 52, 782-798.
21. Chen, P., & Wang, Q. (2024). Learning for multiple purposes: A Q-learning enhanced hybrid metaheuristic for parallel drone scheduling traveling salesman problem. Computers & Industrial Engineering, 187, 109851.
22. Wan, P., Xu, G., Chen, J., & Zhou, Y. (2024). Deep Reinforcement Learning Enabled Multi-UAV Scheduling for Disaster Data Collection with Time-Varying Value. IEEE Transactions on Intelligent Transportation Systems.