



UNIVERSITY OF LEEDS

This is a repository copy of *Reinforcement Learning for Patient Scheduling with Combinatorial Optimisation*.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/216847/>

Version: Accepted Version

Proceedings Paper:

Liu, X., Zheng, C., Chen, Z. et al. (3 more authors) (2025) Reinforcement Learning for Patient Scheduling with Combinatorial Optimisation. In: Lecture Notes in Computer Science series. 44th SGA International Conference on Artificial Intelligence, AI 2024, 17-19 Dec 2024, Cambridge, UK. Lecture Notes in Computer Science, 15447 . Springer , Cham, Switzerland , pp. 238-243. ISBN 978-3-031-77917-6

https://doi.org/10.1007/978-3-031-77918-3_18

This is an author produced version of a conference paper published in Lecture Notes in Computer Science, made available under the terms of the Creative Commons Attribution License (CC-BY), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Reuse

This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here:

<https://creativecommons.org/licenses/>







Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.



eprints@whiterose.ac.uk
<https://eprints.whiterose.ac.uk/>

Reinforcement Learning for Patient Scheduling with Combinatorial Optimisation

Xi Liu¹ , Changgang Zheng² , Zhen Chen¹ , Yong Liao¹ , Ren Chen³ ,
and Shufan Yang⁴ 

¹ University of Science and Technology of China, China

² University of Oxford, UK

³ Anhui Medical University, China

⁴ University of Leeds, UK

s.f.yang@leeds.ac.uk

Abstract. Patient scheduling is a complex task that plays a crucial role in the quality of care. Effective scheduling management mitigates dissatisfaction among patients and physicians, serving as a crucial indicator. Traditionally, the approach to patient scheduling has been ad hoc, often overlooking key factors that may influence scheduling.

In this paper, we propose a reinforcement learning approach that utilises an early stopping mechanism which balances exploration and exploitation to provide combinatorial optimisation from both theoretical and experimental perspectives. Our study utilised datasets from NHS Scotland and The First Affiliated Hospital of Anhui Medical University to evaluate patient scheduling. Our results demonstrate that our Reinforcement Learning (RL) method with early stopping can successfully conduct preliminary practice on realistic examples of the General Practitioner (GP) Scheduling Problem and hospital scheduling issues.

Keywords: Scheduling; · Reinforcement Learning; · CO Problem

1 Introduction

Patient scheduling emerges as a pivotal component within the realm of healthcare management, involving allocating healthcare resources (like doctors, nurses, equipment, and rooms) to patients based on their needs and the availability of these resources. This intricate scheduling mandates a comprehensive consideration of various factors, including the urgency of medical needs, the availability of medical staff, the duration of appointments, and the operational hours of the healthcare facilities. The inherently unpredictable nature of healthcare demands necessitates a combinatorial optimisation (CO) strategy to enable flexible scheduling. This adaptability is crucial for accommodating emergency situations or unforeseen events, such as the sudden unavailability of medical practitioners. In this paper, we introduce an early stopping method in deep reinforcement learning approach to address the patient scheduling challenge as a combinatorial optimisation problem, aiming to generate optimal scheduling solutions. This

methodology can also be applied to other scheduling problems where objective functions are not well-defined or difficult to model mathematically.

The CO problems [17] widely exist in many tasks in real life such as Traveling Salesman problem (TSP), hospital appointment scheduling [16, 20] and production and transport scheduling [4].

Traditional approaches to solving CO problems mainly conclude three categories: exact methods [15], approximate methods [21], and heuristic methods [1, 22]. Exact methods can find the global optimal solution to the CO problem, but since they do not operate in polynomial time, they are not feasible for large problem sizes. Approximate methods are usually greedy algorithms in nature. Although they operate in polynomial time, they often require skillful algorithm design. Heuristic methods do not have a universal framework [13] and always suffer from two issues: 1) reliance on manual selection of heuristic, and 2) difficulty in determining when to apply heuristic. Metaheuristic methods [10] such as genetic algorithms and evolutionary algorithms can be applied to solve most CO problems, but they usually require a given initial solution and do not guarantee polynomial-time convergence.

Most challenging problems in the real world are large-scale and often subject to execution time constraints. Therefore, traditional algorithms encounter difficulties when applied to real-world challenging tasks. Recently, Deep reinforcement learning (DRL) has shown significant potential in overcoming the limitations of traditional approaches [2, 14]. Many CO Problems can be transformed into sequence decision-making problems. For example, the TSP problem is to decide in what order to visit each city, and the shop scheduling problem is to decide in what order to process components on the machine. DRL is a very suitable solution for sequence decision-making. The main difficulty is the definition of the Markov Decision Process (MDP) in the CO problems. A diversity of reinforcement learning (RL) based heuristic method has demonstrated a promising solution as it does not require pre-solved examples of these hard problems [5, 11]. However, so far there is a lack of guidance on how to utilise reinforcement learning (RL) to automatically learn good heuristics for various combinatorial problems since RL relies on estimating Q-value to form policy.

This paper proposes a framework leveraging a Deep-Q Network (DQN) [8] based reinforcement learning methodology to conceptualise healthcare management issues as combinatorial optimisation challenges. By employing real-world datasets, we demonstrate the feasibility of our approach in dynamically allocating hospital resources, outperforming traditional methods that struggle with the complexity and variability inherent in real-world scenarios. Significantly, our method exhibits superior adaptability to temporal changes, a critical attribute for effectively managing scheduling tasks in healthcare settings, where conditions can rapidly change. Furthermore, the utilisation of a synthetic dataset underscores our method's capability to manage problems encompassing a large number of variables and constraints, maintaining computational efficiency even as the scale of the optimisation challenge expands.

2 Problem Formulation

Many CO problems cannot find efficient solutions in polynomial time. As a result, there have been some researchers using machine learning algorithms to solve CO problems in recent years [3, 6, 12]. A particular branch of machine learning, reinforcement learning (RL) is widely used to solve CO problems by modelling the problem as a decision-making process. During this process, an agent interacts with the environment by performing a series of actions to find a solution.

We consider a Markov Decision Process (MDP) to model the CO problem. At each time step, the process is in some state s , and the agent may choose any action a to interact with the environment to get the next state s' . Repeat this process until the maximum cumulative reward is gained. Formally, A Markov decision process is a 4-tuple (S, A, P_a, R_a) , where: State space S is a set of states. Action space A is a set of actions. Reward R is the immediate reward (or expected immediate reward) received after transitioning from state s to state s' , due to action a . Transition probability P : is the probability that action a in state s at time t will lead to state s' at time $t + 1$.

The goal of an agent acting in Markov Decision Process (MDP) is to find a policy function π mapping from states to actions. Solving MDP means finding the optimal policy that maximises the discounted cumulative sum of random rewards formulated as:

$$\pi^* = \arg \max_{\pi} E[\sum_{i=t}^{\infty} \gamma^i R(s_i, a_i)] \quad (1)$$

3 Motivation

When apply reinforcement learning for combinatorial optimisation, the agent must extensively interact with the environment until the algorithm converges and a better policy is obtained. However, those frequent interactions can incur high costs. Batch mode with experience replay provides stability without incurring these high costs. Nevertheless, Batch RL faces a significant issue. As shown in Fig. 1, the proportion of positive data and negative data in the buffer storing the training data directly affects the proportion of the two in the samples, thus affecting the convergence of the model. In extreme cases, this imbalance may prevent the model from converging. In other words, the agent only learns experience with bad behaviors and has little or no experience with good behaviors. This has a great impact on the learning process of the agent. This is ultimately a matter of balance between exploration and exploitation. To achieve this balance, we propose a new algorithm based on the Deep Q-Network [8], which incorporates early stopping and evolutionary methods.

In the past, early stopping was often used to solve the problem of neural network overfitting on training set. With the increase in epochs, the error on the training set and the validation set increases. Early stopping is an effective tool to address

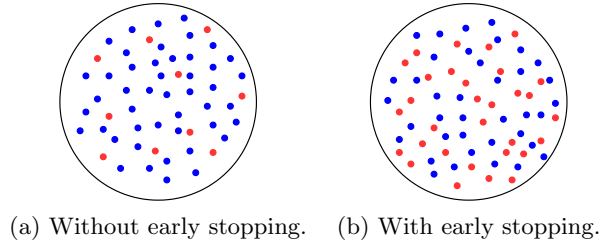


Fig. 1: Data schematic diagram in buffer with/without early stopping. Red dots represent the positive data in the buffer, and blue dots represent the negative data.

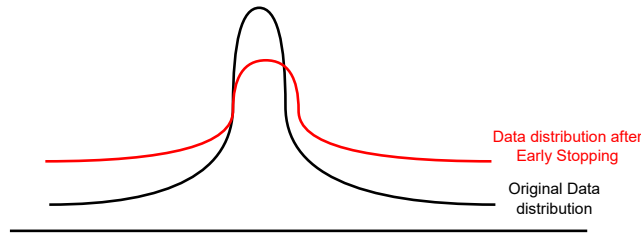


Fig. 2: Data distribution in buffer with/without early stopping. The black curve is the original buffer in DQN, and the red curve represents the data distribution in the buffer with Early Stopping.

the generalisation problem [19]. However, the advantage of early stopping extends beyond this. We can utilise it to adjust the data distribution in the buffer, and seek a balance between exploration and exploitation.

A robust stopping criterion is crucial. All stopping criteria follow a rule: stop when the criteria are satisfied at some time t . In other words, when a value exceeds a certain threshold, the stopping operation is performed. Our criterion involves the number of negative samples: stop if negative samples appear several times when updating buffer data. We set a fixed value k and a counter to count the number of negative samples. When the counter reaches the fixed value k , we stop updating the data in the buffer.

Algorithm 1 DQN with Early Stopping(EDQN)

Require: $C \leftarrow$ a counter, $k \leftarrow$ the early stopping threshold, $Q^\pi \leftarrow$ the policy network,
 $Q^{\pi^*} \leftarrow$ the target network

- 1: Initialize experience replay buffer $\mathcal{B} = \emptyset$
- 2: In each episode:
- 3: Initialize state s_0
- 4: **for** $t = 1 \rightarrow T$ **do**
- 5: select action based on ϵ -greedy
- 6: Observe s_t, r_t, s'_t
- 7: Store the transition (s_t, a_t, r_t, s'_t) in the buffer \mathcal{B}
- 8: **if** r_t is not a good reward **then**
- 9: the value of counter C add one
- 10: **end if**
- 11: Sample experiences $\{(s_t, a_t, r_t, s'_t)\}$ from \mathcal{B} randomly
- 12: Set $y_t = r + \gamma \max_{a'} Q^{\pi^*}(s', a')$
- 13: Update weights of the neural network
- 14: **if** the counter C reaches threshold **then**
- 15: break
- 16: **end if**
- 17: **end for**

We combine DQN [8] and Early Stopping to reduce the times that the agent performs repeated useless actions in a specific state, thereby achieving the purpose of adjusting the data distribution in the buffer as shown in Fig.2. The pseudo-code of DQN with Early Stopping(EDQN) is shown in Algorithm1.

4 Experiment

In this section, we present experimental results from two real-world scheduling tasks.

4.1 System Setup

For each experiment, we first clean and organise the data from real-world tasks, and build simulation environments based on different experimental datasets. We have two real-world tasks: the GP scheduling problem and the hospital patient appointments. From the perspective of the task, the environment we built is a grid world environment, but its basic parameters are different depending on the data. Our datasets are provided by the National Health Service and the First Affiliated Hospital of Anhui Medical University.

4.2 GP Scheduling

In the UK, the National Health Service (NHS) normally assigns patients to a particular General Practitioner (GP), where the dataset is taken from NHS web-

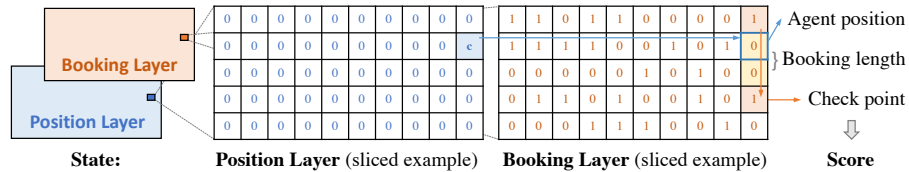


Fig. 3: State structure of Scheduling Problem.

site⁵). Given a certain number of GPs and working hours, the question we need to address is how to arrange GPs for a group of patients rationally and efficiently. More specifically, we want the patient’s appointment slots which may be of variable length to be scheduled at a time that avoids wasting GPs’ resources, just like in Tetris games, where each block is optimally placed.

Task Description. The General Practitioner (GP) Scheduling Problem involves resource allocation in which GPs may already have pre-booked appointments, leaving limited availability for new patients’ appointments. Patients have varying demands for appointments of different durations, and the challenge lies in maximising the utilisation of the available time slots to accommodate as many patients as possible while allowing GPs to allocate blocked time slots for other activities like training. To define the specific environment, we can refer to four scenarios of face-to-face consultations, home visits, telephone consultations and video consultations. The relevant data of these four scenarios are shown in Table 1.

Table 1: Appointments in the England by NHS statistics for one month.

scenarios	face-to-face	home visits	telephone	video
Number of appointments	15404951	171669	9184791	115725
Proportion of appointments	0.6192	0.0069	0.3692	0.0047
slots requirements	1	3	2	1

Environment settings. In our experiment setting, there are 100 GPs, each working for 8 hours, divided into 32 slots of 15 minutes each. We divide the consultation time according to different scenarios, as detailed in Table 1. ‘3’ indicated that the appointment will need 3 consecutive timeslots. The problem can be abstracted as finding a series of suitable positions in a 32×100 grid world, minimising gaps and meeting the requirements as efficiently as possible. Here reducing the number of gaps implies finding suitable locations to accommodate longer time slots when needed.

In the GP environment, the agent needs to have a certain understanding of the current overall appointments and the agent’s position. Therefore, the state of the size $2 \times 32 \times 100$ has two layers, one layer named the booking layer

⁵ <https://digital.nhs.uk/data-and-information/publications/statistical/appointments-in-general-practice/march-2022>

represents the occupied situation, and the other named position layer indicates the agent position. As shown in Fig. 3, in the booking layers, '1' represents that the position has been occupied, and '0' represents that it is empty. In the position layer, the position of the agent is constant C , and all other positions are 0. The booking length is the number of slots required by a patient. Action determines the direction the agent moves in the grid world. The action space is a finite and discrete set, containing four actions: up, down, left, and right. The rewards are designed based on the current agent's position and the status of the upcoming reservation location. As shown in equation (2), if the current agent's position and the upcoming reservation location are empty, in other words, the values of these positions are '0', the agent will obtain the reward of +0.5. if these positions are not empty, the reward of -0.1 will be given. Then we also consider extreme situations. When the agent reaches the boundary and the next action does not cause it to leave the current position, this situation is very undesirable. In addition, the agent is not encouraged to repeatedly jump between two adjacent locations. In both cases, the agent receives a reward of -0.5.

$$reward = \begin{cases} +0.5, & \text{if the position is empty} \\ -0.1, & \text{if the position is not empty} \\ -0.5, & \text{if the agent reaches the} \\ & \text{boundary or goes back} \end{cases} \quad (2)$$

4.3 Hospital Patient Appointments Scheduling

Unlike in the UK, patients get web-based appointments with specialists in UK. The hospital encompasses various specialised departments, including internal medicine, surgery, among others. Patients can seek treatment in different departments based on their medical conditions. These systems aim to reduce wait times and improve efficiency by allowing patients to book appointments online. However, due to high demand, getting a specialised appointment is limited, resulting in patients potentially missing valuable treatment time. An optimised booking system to utilise the booking will reduce the chance of missing treatment. Table 2 illustrates the appointment data that collected from the First Affiliated Hospital of Anhui Medical University. This table indicates that the departments of Internal Medicine and Surgery have the highest patient visit numbers. Therefore, our experiment is based on Internal Medicine and Surgery departments scheduling needs.

Task Description. Hospital patient appointment scheduling is similar to the General Practitioner (GP) Scheduling Problem, which seeks to optimise the allocation of hospital resources. Considering the specific nature of specialist services in Chinese hospitals and the particular conditions at The First Affiliated Hospital of Anhui Medical University, we constructed two analogous environments using data from the Internal Medicine and Surgery departments, respectively. In each environment, every doctor works for 8-hours, divided into 96 five-minute slots. Since the diagnosis and treatment time varies for each patient, we cat-

Table 2: Summary of data from the First Affiliated Hospital of An Medical University. The table summarised the data collected from various departments as the First Affiliated Hospital of an Medical University. DTCM refers to the Department of Traditional Chinese Medicine; OG denotes the Department of Obstetrics and Gynaecology; and AD stands for the Anaesthesia Department.

Department	Internal Medicine	Surgical	Paediatrics	Dermatology
Number of appointments	3220	3097	1694	1986
Proportion of appointments	0.2439	0.2346	0.1283	0.1504
Number of departments	16	13	7	3
Department	Ophthalmology	Otolaryngology	Stomatology	DTCM
Number of appointments	715	843	1019	228
Proportion of appointments	0.0542	0.0638	0.0772	0.0173
Number of departments	2	2	4	4
Department	OG	AD	Haematology	
Number of appointments	228	79	92	
Proportion of appointments	0.0173	0.0060	0.0070	
Number of departments	8	1	1	

egorise patient needs into three types based on duration: 1 slot, 2 slots, and 3 slots. The objective for each appointment is to find a continuous sequence of slots that fulfils these requirements. Therefore, the problem transforms into identifying a series of positions in a grid, which corresponds to the number of clinics multiplied by 96.

Environment settings. In the Internal Medicine environment, the state size is $2 \times 96 \times 16$, corresponding to 16 clinics in the Internal Medicine department. Two layers represent the state, and its structure is similar to the state structure in Fig. 3. The first layer represents occupancy, while the second represents the agent’s position. The checkpoint plays an crucial role in reward design because it can be utilised to calculate the resource utilisation rate. The action space contains four actions, up, down, left, and right. The reward design is based on the current position of the agent and its nearby positions. If the agent cannot start a reservation from its current position, the reward of -0.1 will be given. Conversely, if the position is suitable, it is essential to verify that the upper and lower positions, labelled as ‘Check Point’ as in Fig.3 satisfy the necessary criteria.

For every step, if the values of both positions are 1, set the *reward* = 0.5. If one of the values is 1, set the *reward* = 0.25. If both values are 0, set the *reward* = 0.1. Here, our default border position value is 1. A reward of -0.5 is given in the following two situations: The first is that the agent reaches the boundary and the next action does not cause it to leave the current position, and the second is that the agent repeatedly jumps between two adjacent locations. In the Surgical environment, the difference from the Internal Medicine environment is the size of the state. The size of the state is $2 \times 96 \times 13$ in the Surgical environment because the number of Surgical clinics is 13.

4.4 Results and Evaluation

We compared the performance of Deep Q-Network (DQN) and DQN with Early Stopping (EDQN) on GP scheduling and hospital patient appointment data. In our experiments, the neural network structure of the policy comprises three convolutional layers and two linear layers which use the Kaiming initialisation method [7] to initialise parameters. The hyperparameter settings include a learning rate $lr = 1e - 4$, $\epsilon = 0.3$, $\epsilon_{decay} = 0.995$ and dynamic γ . At the beginning of exploration, the agent does not fully understand the environment. As the agent further explores the environment, it is more in line with the learning process to take the long-term future benefits into account in the value generated by the current behaviour. Therefore, it is more appropriate to use a dynamic γ . The dynamic γ is formulated as

$$\gamma = 1 - 0.9 \times (1 - \gamma) \quad (3)$$

Here, we set the initial $\gamma = 0.1$, and the maximum value of γ does not exceed 0.99.

Before the experiments, we need to establish the basic parameters of the environment. These include the number of pre-booking slots, the constant C representing the agent’s position in the position layer, and the patient demands. For the GP scheduling problem, we set the number of pre-booking slots to 750, and $C = 100$. The early stopping rule is that the agent obtains 10 negative rewards of -0.5 in total. For hospital patient appointments, we set the number of pre-booking slot to 100, and $C = 100$. The early stopping rule specifies that the agent receives 10 consecutive negative rewards of -0.5.

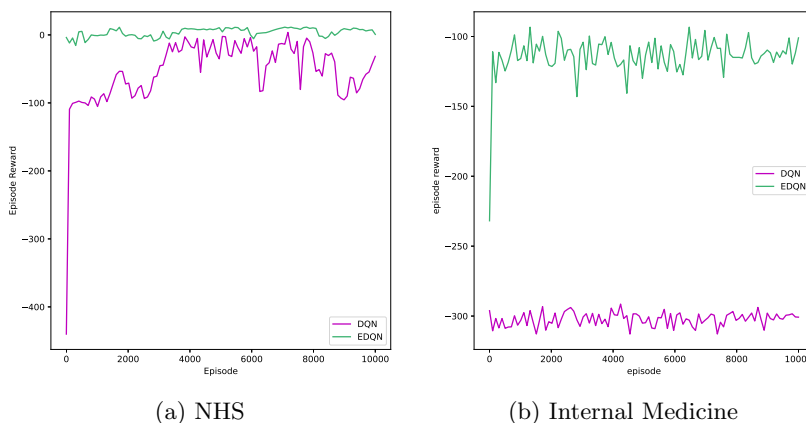


Fig. 4: Hospital patient appointment

The experimental results are presented in Fig.4. Due to Early Stopping, the rewards obtained by EDQN were higher than those obtained by DQN from the beginning. Because the reward we set were relatively small, the curve of EDQN

does not show an obvious upward trend (Fig.4(a)). Since there is no significant difference in appointment data between Internal Medicine and Surgical department in hospitals, the experimental results are similar. The performance in Internal Medicine is shown in Fig.4(b), and the performance in Surgical department shows the same trend. It is obvious that EDQN is better than DQN in the performance of GP appointment and hospital appointment data. This is because we eliminate redundant data and increase the agent’s learning of sparse experience. Additionally, We utilised the Genetic Algorithm [18] with crossover rate of 0.5 and mutation rate of 0.01 to solve the scheduling problems on GP appointment and Hospital appointment scheduling data. Although the evolution method is effective, it is slower than DQN at larger scales.

5 Related Work

The first attempt to utilise the neural networks to solve the classic combinatorial optimisation problem, Traveling Salesman Problem (TSP), was proposed by Hopfield & Tank [9]. However, their approach only covered 30 cities and was trained to solve one instance, which has few advantages over traditional solving methods. In recent years, there has been much research on the application of deep reinforcement learning to combinatorial optimization problems.

Bello et al. [2] combined reinforcement learning and recurrent neural network (RNN) to build a framework, Neural Combinatorial Optimization, for solving combinatorial optimization problems. The framework conducted practical exploration from two aspects: RL pre-training and active search. However, the experimental environment was limited to TSP problems. Kool et al. [14] designed a new attention model and used REINFORCE with a simple greedy rollout baseline to train the model. This approach can be extended to other problems such as VRP.

In addition to the above combination of reinforcement learning and neural networks, combining reinforcement learning with heuristic algorithms to solve CO problems has been a research hotspot in recent years. Nathan et al. [5] proposed Poppy to improve the exploration of solution space of combinatorial optimization (CO) problems. It combined reinforcement learning with evolutionary algorithms and used the encoder-decoder architecture to efficiently train populations. The construction method Poppy has sample-efficient reinforcement learning strategy updates while exploring the evolutionary population level. kallestad et al. [11] proposed a selection hyperheuristic framework that replaced the adaptive layer of Adaptive Large Neighborhood Search with an RL agent for solving the CO problem. Although it is quite versatile in solving CO problems, there are still some challenges in correctly handling large pool heuristics.

6 Conclusion and Future Work

Patient scheduling is a critical component of healthcare management, involving the allocation of resources such as doctors, nurses, equipment, and rooms in accordance with patient needs and the availability of these resources. This paper presents an innovative approach, treating healthcare resource allocation as a combinatorial optimisation problem. We propose a reinforcement learning method, enhanced with an early stopping mechanism, to aim for optimal resource utilisation and increased adaptability across various real-world environments. Our work demonstrates the application of the Deep Q-network (DQN) reinforcement learning algorithm to tackle real-world scheduling challenges, notably the General Practitioner (GP) problem and hospital patient appointments. A significant innovation of our approach is the use of early stopping to fine-tune data distribution in the replay buffer, resulting in improved efficiency and accuracy in practical scheduling tasks. While the presented solution may not achieve theoretical optimality, it provides a highly efficient method for large-scale applications. This study focuses on discrete action spaces due to their relevance in the majority of practical scheduling scenarios addressed. Future research could extend our methodology to tasks involving continuous action spaces, such as dynamic resource allocation and real-time scheduling, significantly enhancing its applicability and potential impact in the field.

References

1. Barrett, T., Clements, W., Foerster, J., Lvovsky, A.: Exploratory combinatorial optimization with reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 3243–3250 (2020)
2. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940 (2016)
3. Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: a methodological tour d’horizon. *European Journal of Operational Research* **290**(2), 405–421 (2021)
4. Ehm, J., Freitag, M.: The benefit of integrating production and transport scheduling. *Procedia CIRP* **41**, 585–590 (2016)
5. Grinsztajn, N., Furelos-Blanco, D., Barrett, T.D.: Population-based reinforcement learning for combinatorial optimization. arXiv preprint arXiv:2210.03475 (2022)
6. Guo, T., Han, C., Tang, S., Ding, M.: Solving combinatorial problems with machine learning methods. *Nonlinear Combinatorial Optimization* pp. 207–229 (2019)
7. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
8. Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al.: Deep q-learning from demonstrations. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
9. Hopfield, J.J., Tank, D.W.: “neural” computation of decisions in optimization problems. *Biological cybernetics* **52**(3), 141–152 (1985)
10. Hussain, K., Mohd Salleh, M.N., Cheng, S., Shi, Y.: Metaheuristic research: a comprehensive survey. *Artificial intelligence review* **52**, 2191–2233 (2019)

11. Kallestad, J., Hasibi, R., Hemmati, A., Sörensen, K.: A general deep reinforcement learning hyperheuristic framework for solving combinatorial optimization problems. *European Journal of Operational Research* **309**(1), 446–468 (2023)
12. Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A.M., Talbi, E.G.: Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research* **296**(2), 393–422 (2022)
13. Kool, W., van Hoof, H., Gromicho, J., Welling, M.: Deep policy dynamic programming for vehicle routing problems. In: *International conference on integration of constraint programming, artificial intelligence, and operations research*. pp. 190–213. Springer (2022)
14. Kool, W., Van Hoof, H., Welling, M.: Attention, learn to solve routing problems! arXiv preprint arXiv:1803.08475 (2018)
15. Laporte, G.: The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research* **59**(2), 231–247 (1992)
16. Li, N., Li, X., Forero, P.: Physician scheduling for outpatient department with nonhomogeneous patient arrival and priority queue. *Flexible Services and Manufacturing Journal* pp. 1–37 (2021)
17. Ma, L., Li, J., Lin, Q., Gong, M., Coello, C.A.C., Ming, Z.: Cost-aware robust control of signed networks by using a memetic algorithm. *IEEE transactions on cybernetics* **50**(10), 4430–4443 (2019)
18. Mirjalili, S., Mirjalili, S.: Genetic algorithm. *Evolutionary Algorithms and Neural Networks: Theory and Applications* pp. 43–55 (2019)
19. Prechelt, L.: Early stopping-but when? In: *Neural Networks* (1996)
20. Shehadeh, K.S., Cohn, A.E., Jiang, R.: A distributionally robust optimization approach for outpatient colonoscopy scheduling. *European Journal of Operational Research* **283**(2), 549–561 (2020)
21. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. *Advances in neural information processing systems* **28** (2015)
22. Wu, Y., Song, W., Cao, Z., Zhang, J., Lim, A.: Learning improvement heuristics for solving routing problems. *IEEE transactions on neural networks and learning systems* **33**(9), 5057–5069 (2021)