http://doi.org/10.35784/iapgos.6996

received: 27.12.2024 | revised: 19.08.2025 | accepted: 09.09.2025 | available online: 30.09.2025

DEVELOPMENT OF A REINFORCEMENT LEARNING-BASED ADAPTIVE SCHEDULING ALGORITHM FOR COMMERCIAL SMART KITCHENS

Karol Kabała[1], Piotr Dziurzanski[2], Agnieszka Konrad[3,4]
[1]Numlabs Ltd, Krakow, Poland, [2]West Pomeranian University of Technology, Faculty of Computer Science and Information Technologies, Szczecin, Poland, [3]Polish Academy of Sciences, Institute of Bioorganic Chemistry, Poznan, Poland, [4]Papukurier Ltd, Poznan, Poland

Abstract. *Reinforcement learning (RL) is a machine learning method in which a model optimizes its decision-making strategy based on rewards or penalties received for the actions it takes in an environment, often simulated. An example of an optimized process could be work scheduling in a restaurant, with the cost function being the absolute error of the difference between the scheduled and actual delivery times of an order. In task planning, RL stands out for its ability to handle problems requiring a complex sequence of actions, where traditional planning algorithms may struggle. RL models can effectively explore the solution space, adjusting their decisions to changing conditions, which enables dynamic and adaptive task execution management. RL is a broad class encompassing various approaches to achieving a goal, and in this research, we focus on selected ones. Three popular RL methods named DQN, SARSA and TD-AC have been implemented and evaluated. The study was conducted in a simulated environment designed to replicate a "delivery-based" restaurant business model. The kitchen simulation model has been developed based on 65,845 recorded food preparation processes performed in 30 restaurants located throughout Poland. A rule-based, queue-driven model (FIFO) served as the baseline for absolute quality comparison of the generated schedules. The results show that, for the defined problem, the quality of the scheduling outcomes varies significantly depending on the choice of learning algorithm. Notably, the hybrid approach performed best under simulation conditions, considerably reducing the total completion time in a scenario reflecting the operations of a small, typical restaurant.*

Keywords: reinforcement learning, scheduling, smart factories, kitchen delivery systems

OPRACOWANIE ADAPTACYJNEGO ALGORYTMU PLANOWANIA OPARTEGO NA UCZENIU PRZEZ WZMACNIANIE DLA INTELIGENTNYCH KUCHNI KOMERCYJNYCH

Streszczenie. *Uczenie przez wzmacnianie (RL) to metoda uczenia maszynowego, w której model optymalizuje swoją strategię decyzyjną w oparciu o nagrody lub kary otrzymywane za działania podejmowane w środowisku, często symulowanym. Przykładem zoptymalizowanego procesu może być planowanie pracy w restauracji, gdzie funkcją kosztu jest bezwzględny błąd różnicy między zaplanowanym a rzeczywistym czasem dostawy zamówienia. W planowaniu zadań, RL wyróżnia się zdolnością do radzenia sobie z problemami wymagającymi złożonej sekwencji działań, gdzie tradycyjne algorytmy planowania mogą mieć trudności. Modele RL mogą efektywnie eksplorować przestrzeń rozwiązań, dostosowując swoje decyzje do zmieniających się warunków, co umożliwia dynamiczne i adaptacyjne zarządzanie realizacją zadań. RL to szeroka klasa obejmująca różne podejścia do osiągnięcia celu, a w tym badaniu skupiamy się na wybranych z nich. Trzy popularne metody RL o nazwach DQN, SARSA i TD-AC zostały zaimplementowane oraz ich efektywność została przebadana eksperymentalnie. Badanie przeprowadzono w symulowanym środowisku zaprojektowanym w celu odtworzenia modelu restauracji opartego na dostawach do klientów zdalnych. Model symulacji kuchni został opracowany w oparciu o 65845 zarejestrowanych procesów przygotowywania dań przeprowadzonych w 30 restauracjach zlokalizowanych w całej Polsce. Model kolejkowy FIFO, oparty na regułach, posłużył jako punkt odniesienia do bezwzględnego porównania jakości wygenerowanych harmonogramów. Wyniki pokazują, że dla zdefiniowanego problemu, jakość wyników planowania różni się znacząco w zależności od wyboru algorytmu uczenia. Warto zauważyć, że podejście hybrydowe działało najlepiej w warunkach symulacji, znacznie skracając całkowity czas realizacji w scenariuszu odzwierciedlającym działalność małej, typowej restauracji.*

Słowa kluczowe: uczenie przez wzmacnianie, planowanie, inteligentne fabryki, systemy dostaw w kuchni

Introduction

The objective of the modern economy, sometimes known as Society 5.0, is to use technology to complete activities more quickly for the benefit of people [1]. In keeping with this approach, process automation and optimisation will benefit even relatively small businesses in the service sector. This contrasts with the more widely known idea of Industry 4.0, which was associated with the production and industrial sectors [12].

One of the not so widely discussed applications of process automation is related to smart kitchen. This sector is typically resistant to process automation [3], even though it is substantial and significant in every economy. For example, according to Polish Central Statistical Office (GUS), there were 83.9 thousand restaurants nationwide in 2022, and their combined revenue was 64.6 billion Polish zloty [18].

The approach described in this paper is a part of a larger project whose scope started with recording and generalisation of processes carried out in commercial kitchens delivering food to customers' premises using a fleet of couriers. The most important research problem concerns the development of decision-making algorithms responsible for dividing the work and synchronising the tasks of kitchens and couriers in such a way that the work of restaurants is managed as closely as possible to the optimum from the point of view of the owners, their staff and customers. A kitchen simulation model has been created to give customers a more reliable delivery time, so-called promise time, at the time of the food ordering. A custom hardware module called the Kitchen Display System (KDS) has been used to capture a significant quantity of food order and preparation procedures to create such a model.

Fig. 1 shows an example of an integration of the hardware and custom software into a data collection system prototype. This system was designed to be extremely straightforward to use and intuitive since it was meant to be used by the actual kitchen staff. The kitchen staff were primarily focused on fulfilling their core duties rather than recording data for later analysis. Hence, a high likelihood of human error has been assumed. For this reason, the kitchen data acquisition system has been equipped with an automatic validation system ensuring the reliability and accuracy of the collected data.

The data collection system was introduced in almost 30 restaurants, spread across Poland and located in big cities, smaller towns, and urban-rural regions. Sushi bars and pizzerias predominated among the eateries. In total, 65,845 food orders and the related food preparation processes have been collected.

After gathering input data, a development of scheduling algorithm has been performed. In contrast to previous approaches suggesting static scheduling [2], based on the gathered data, we concluded that variability of the processes in a commercial kitchen is significant and hence dynamic scheduling methods are more appropriate. After analysis of the current trends in this domain [10], we decided to apply reinforcement learning approach to generate rules for state-space problems. The policy to be obtained is aimed to steer the appropriate course of action for each kitchen state. During the learning process, the learning system gets a reinforcement signal known as reward for a certain action. The reward is generated with the developed kitchen and courier simulation models. Finding an appropriate strategy to maximise the predicted reinforcement over subsequent actions forms the learning system's objective.

In the following section of this paper, we describe in more detail the domain of the problem under analysis. Next, we give a brief overview of the queue-based scheduling, followed by its reinforcement-learning-based counterpart. In that section, we explain how the scheduling job might be designed to use with the analysed problem. Next, we discuss the outcomes of our tests on simulated problem sets. These findings suggest that in life-size problems, reinforcement learning can perform better than a traditional scheduler.

1. Related work

Digital gastronomy seeks to alleviate the restaurant employees' burden by adding new capabilities to traditional cooking [7]. Modern technology is becoming increasingly widespread in restaurants. For example, food preparation, cleaning, and storage equipment can be monitored using IoT devices [5]. For more detailed identification of kitchen activities, Kinect-style cameras can be used [6]. In this paper, we propose reporting the kitchen task processes endpoint directly by staff, using simplified interfaces (an example of them is shown in Fig. 1) and assuming rigid verification of the obtained data. More details on the applied verification stage and its results can be found in [3].

Fig. 1. Software and hardware integration example within a prototype of a data collection system papu.io

The cooking process has been viewed as an optimisation problem in [4]. Because various tasks can be completed in parallel, the cooking time can be shortened. This system used the list scheduling algorithm to minimise meal preparation time and maximise food quality. Paper [2] focused on task scheduling in smart kitchens. It assumed genetic-algorithm-based static scheduling of tasks rather than dynamic, which is proposed in this paper. We claim that dynamicity of the kitchen tasks and the variability of its processing time [3], requires the use of dynamic queuing techniques, as proposed in the following sections. Furthermore, to the best of the authors' knowledge, prior research has not examined the synchronization of kitchen schedules with courier order deliveries. These two novel aspects represent the main contribution of this paper.

2. Decision algorithm for kitchen operation using a queue

The decision model using a queue works based on the First-In-First-Out (FIFO) principle, i.e. the orders that are queued first will be served earliest. This model is in line with the typical kitchen schedule, as confirmed by the restaurants participating in this research. In this simple approach, an order goes into the back of the queue if: (1) the restaurant received it as a new delivery request, or (2) the order has completed one of the preparation phases. In the latter case, the order returns to the queue with a new priority with an updated status.

The proposed algorithm can be summarised in the following manner. In step 1, the presence of new orders is checked. If there are any, they are added to the end of the queue. If the queue is empty, the system waits for a new order. Otherwise, the oldest item is popped from the queue and the related operation is performed. As a result, the order moves to the next stage, where it is added to the end of the queue. If the order popped from the queue is ready to be dispatched, it is assigned to a courier along with the other orders that were assigned to the courier earlier. Thus, in the case of several orders being at the same state, the priority is given to the orders being in the queue for the longest time. This scheme is summarised in the flow chart presented in Fig. 2.

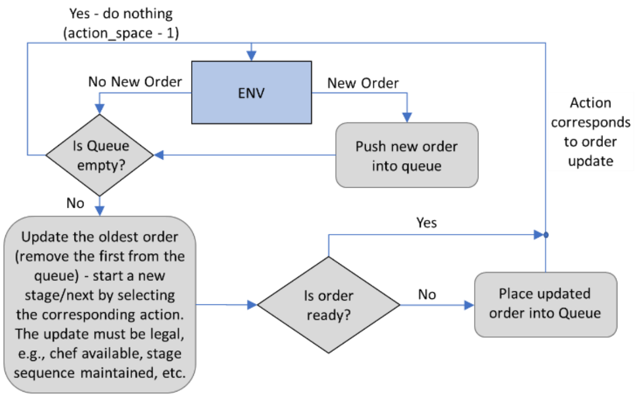

Yes - do nothing
(action_space - 1)
No New Order
ENV
New Order
Action corresponds to order update
Is Queue empty?
Push new order into queue
No
Yes
Update the oldest order (remove the first from the queue) - start a new stage/next by selecting the corresponding action. The update must be legal, e.g., chef available, stage sequence maintained, etc.
Is order ready?
No
Place updated order into Queue

Fig. 2. Flow chart of the queue-based decision-making model

3. Principles of reinforcement learning

Reinforcement Learning (RL) is a machine learning paradigm that uses rewards or penalties for actions performed in an environment to help a model to optimise its decision-making strategy. In this section, a summary of the techniques used in the remaining part of the paper is presented. More information can be found in numerous resources, for example in [11].

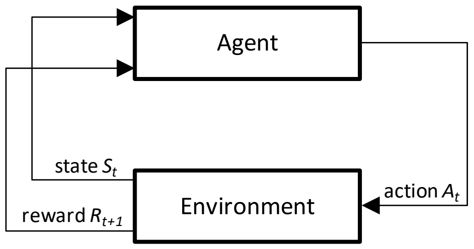


Fig. 3. The agent–environment interaction in an RL paradigm

Fig. 3 shows the primary actors in an RL scheme. The environment is usually described as a classic Markov decision process (MDP). The learning and decision-making processes are performed by the agent. The agent interacts with the environment using states, actions, and rewards. The entire process is performed continually at discrete time points. The agent chooses an action A_t at each time step t after receiving a representation of the state S_t from the environment. In the following time step t+1, the agent obtains a numerical reward R_{t+1} and a new environment state S_{t+1}, whose change has been partially influenced by action A_t. The RL process can be then described with a trajectory built with subsequent states, actions and rewards: S_0, A_0, R_1, S_1, A_1, R_2, … . The transition between state s and s' under action a is described by probability:

$$P_a(s,s') = Pr(S_{t+1} = s' \,|\, S_t = s, A_t = a) \tag{1}$$

The probability of taking action a in state s is named policy

$$\pi(a, s) = Pr(A_t = a \mid S_t = s) \quad (2)$$

The state-value function V_π(s) denotes discounted return from state s under a given policy π. Hence, it can be viewed as an indicator of the value of state s with regards to the optimisation goal. This function is described by formula

$$V_\pi(s) = E_\pi[R_t|s_t = s] = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s] \quad (3)$$

where $0 \le \gamma < 1$ is the discount factor. The value of γ defines the influence of the rewards in the distant future with respect to the rewards in the immediate future. This equation, often referred to as V function, can be immediately applied in a reinforcement learning algorithm, as its goal is to find the policy with maximum expected discounted return.

Another equation often used in reinforcement learning is the so-called Q function, which computes the value of a state s and an action a under policy π:

$$Q_\pi(s, a) = E_\pi[R_t|s_t = s, a_t = a]$$
$$= E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a] \quad (4)$$

$Q(s,a)$, known as Q-values, estimates of how good it is to execute action a in state s. They are used in Q-learning, a fundamental reinforcement learning algorithm, to iteratively improve the learning agent's performance. The temporal difference (TD) update rule is used at every time step to iteratively compute this estimate

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma Q(s', a') - Q(s, a)) \quad (5)$$

where s and s' denote the current state of the agent and the next state, a denotes the current action selected according to a certain policy, a' is the best action to be selected in the next state using the current Q-value estimation, r is the current reward, issued by the environment as the response to action a, γ is the discount factor discussed earlier, and α is the learning rate determining the degree of overriding the old information with the newly acquired one.

In the simplest implementation, the data for Q-learning is stored in tables. For solving larger problems, however, some technique for function approximation is required. In the approaches used in the sequel of this paper, neural networks have been applied for this approximation.

Q-values are often determined using recursive Bellman's equation

$$Q(s, a) = R(s, a) + \gamma \cdot \max_a Q(s', a) \quad (6)$$

where R(s,a) is the immediate reward from the environment after performing action a in state s, and the max function computes the maximal Q-value for next state s' among all possible states a.

During the RL process, one of the crucial decisions is whether to select an action a that has been confirmed during the earlier stages of the process to be beneficial, or to apply not yet well explored actions. This dilemma is known as the exploration vs exploitation trade-off. In popular ε-greedy method, where $0 < \varepsilon < 1$, the action with the highest Q-value at that moment is selected with probability 1-ε (which shall be the majority of cases), whereas a random action is selected uniformly with probability ε.

4. Implementation of reinforcement-learning-based scheduling methods

A decision model agent using the temporal difference learning technique implements three RL methods, DQN, SARSA and TD-AC. The agent allows the users to select an action using the current policy (ε-greedy in the case of DQN and SARSA, for TD-AC randomise the action using a discrete action distribution). The weights of the neural network constituting a model of the system could be optimised. The training is performed inside a learning loop; the agent executes a specified number of episodes using the target environment, after each episode it calls the optimisation method. The evaluation is performed on the environment which returns the sum of the rewards received and the duration of the evaluation.

Temporal Difference (TD) Learning is a set of methods involving iterative improvement of the estimate of the current value function. The most common extensions of this method used for control (in addition to the V (3) or Q (4) function, we also obtain a policy) are: DQN and SARSA. The Actor-Critic method has been chosen. This approach uses the TD Error value obtained by the critic as a regulariser of the so-called baseline used to reduce the variance of the actor's loss function, which is optimised in a similar way to Q-learning. Both the policy (actor) and the critic (V-function approximation) are represented by a neural network.

In the SARSA method, the agent (a neural network in the proposed approach) is trained on-policy, i.e. using the current policy, the action for the next state is selected, thus obtaining the Q value for calculating the loss function. And the policy itself is obtained from the approximated Q-function, namely the action giving the largest Q-function value for the current state is selected. The method uses a dual neural network approach. In addition, the network called target network is a copy of the network used to predict the value of the Q function and is used to increase the stability of the learning procedure by predicting the value of Q for the future state and action (not for the present state of the environment) in the Bellman equation (6) used to calculate the cost function of the neural network. In contrast, the parameters of the target network are not trained, but periodically synchronised with the main network instead.

```
# in contrast to DQN we calculate next_state_values
# using current policy not max operator

state_action_values = self.policy_net(state_batch)
.gather(1, action_batch)
with torch.no_grad():
  next_action_batch = (
    self.policy_net(next_state_batch)
    .max(-1)[1].unsqueeze(1)
  )
  next_state_values = (
    self.target_net(next_state_batch)
    .gather(1, next_action_batch)
    .squeeze(1)
  )

expected_state_action_values = (
  1 - done_batch
) * next_state_values * self.gamma + reward_batch

loss = F.smooth_l1_loss(
  state_action_values.squeeze(1),
  expected_state_action_values
)

self.optimizer.zero_grad()
loss.backward()
self.optimizer.step()
```

Fig. 4. A fragment of the optimise model function of the SARSA method responsible for calculating the cost function

As the approach for the Deep Q-Learning technique, Double-DQN has been chosen as it is less prone to overestimate the action values in noisy environments [13]. Similarly to SARSA, a target network was used when calculating the cost function. However, a difference in learning dynamics is observed, as Q-Learning is an off-policy method. When the next state is processed in DQN, the Q-value is selected by applying the max operator on data sampled from the replay buffer. In contrast, in the SARSA method, this is performed on-policy and using the current policy the action for the next state is selected, thus the Q(next_state, next_action) value for the calculation of the loss function was obtained.

The implemented decision models based on reinforcement learning share most of the hyperparameters of the learning procedure, such as the sizes of hidden network layers, the size of the batch sampled from the transit buffer, the probability of drawing a random action (exploration vs. exploitation) maximum value, minimum value and change step, the number of training episodes, the learning rate of the optimiser, the minimum buffer size before the training commences, and the parameter indicating how much focus is given to rewards from future actions (γ).

The above enumerated RL algorithms have been implemented in the Python language with PyTorch [17] a popular machine learning library, using Jupyter notebook [16]. For example, the fragment of the implementation of the SARSA method is presented in Fig. 4. The remaining tools used for the development and the evaluation have been enumerated in Fig. 5. Among them, the following two tools used to monitor and automate experimental runs are worth mentioning. MLFlow [15] has been used to monitor the models during training. This tool has been also used for visualization of metrics, logging of model parameters and manually produced graphs. The metrics have been logged each episode using the library interface and visualizations have been created automatically from them. Data Version Control (DVC) [14] has been used for recording the trained models and the scenarios used, as well as the outputs produced by the model. It has been also applied to easily automate the process of running and reproducing the experiments. By defining the structure of the experiments and the set of parameters for each stage of the experiment in two yaml files, it allowed us to to track changes to the parameters and models, using a git-inspired interface.

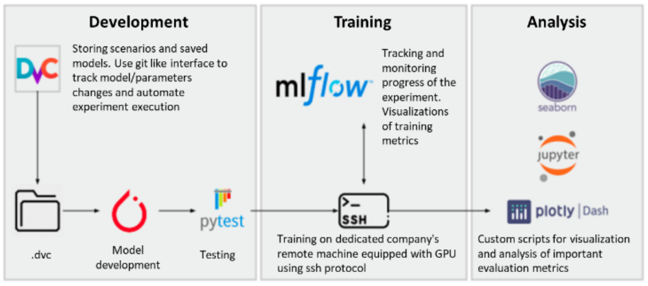

Development
Storing scenarios and saved models. Use git like interface to track model/parameters changes and automate experiment execution
.dvc
Model development
pytest
Testing
Training
Tracking and monitoring progress of the experiment. Visualizations of training metrics
mlflow
SSH
Training on dedicated company's remote machine equipped with GPU using ssh protocol
Analysis
seaborn
jupyter
plotly | Dash
Custom scripts for visualization and analysis of important evaluation metrics

Fig. 5. The relationship between the used tools and the pipeline of experiments on decision models

5. Evaluation of the developed decision-making models

All decision models described in the previous section were pre-tested during the development process using test scenarios with the following parameters:

- Number of cooks, kitchen desks and ovens: 2,
- Number of couriers: 2,
- Number of orders processed in 8 h: 5.

The state and action spaces for the parameters thus defined in this case were both equal to 51.

This approach was tested with various hyper-parameters. Three popular optimisers have been used: Adam, SGD, and RMSProp. The sub-module features were tested with 1-3 layers with neuron numbers ranging from 256 to 64. The tested values for the learning rate were set in the range (1e-5, 0.1). The discounted rate tested values was set in the range $\gamma \in (0.9, 0.99)$.

The approach did not achieve the expected results. The sum of rewards even on small test scenarios reached negative values where, for example, the DQN reaches a value of about 300.

In Fig. 6 left, a graph of the sum of rewards received for each episode is shown, while Fig. 6 right shows the number of simulation steps needed to complete an episode. The sum of rewards received by the TD-AC agent during the learning process does not increase with successive episodes (and even deteriorates), which resembles the performance of a randomised policy.

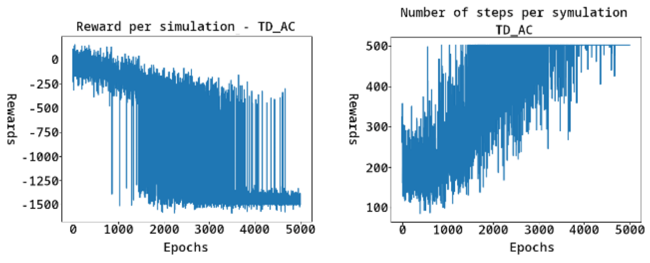


Fig. 6. TD-AC agent during the learning process

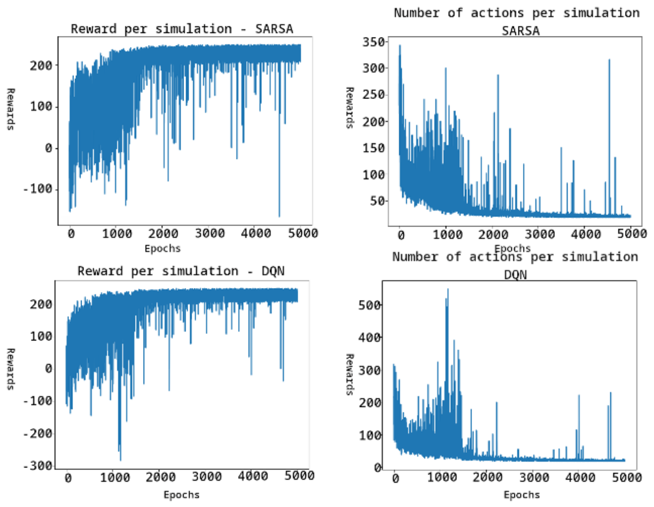


Fig. 7. SARSA and DQN agents during the learning process

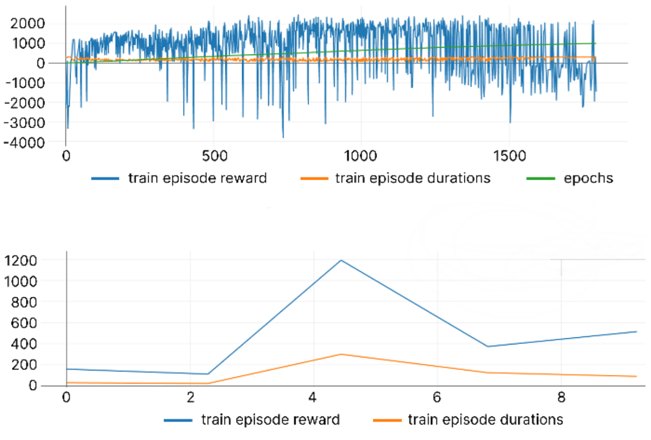


Fig. 8. DQN agent performance graphs on a full simulator for a scenario with 30 orders executed per day

The learning procedure for the SARSA or DQN methods for such simple scenarios, on the other hand, yields positive results, as shown in Fig. 7. The sum of rewards increases with training, the agent requires fewer steps to complete a single simulation episode, and the agent can correctly execute test scenarios (scenarios generated for identical parameters to the scenarios used during training which the agent encounters for the first time during evaluation).

As TD-AC did not give satisfactory results even for small scenarios, and because DQN is a more popular method, i.e. there are many more scientific publications around this method compared to the SARSA method [8], the focus was on just one specific solution. Therefore, in the rest of this section, the focus is exclusively on the Q-Learning-based method together with the method using the queue as a baseline.

Even the Q-Learning-based model alone is unable to solve test scenarios for the full kitchen simulator. Although the total reward received during the learning procedure increases accordingly, its variance continues to be high even in the final phase.

This results in poor performance during evaluation on scenarios that were not available during training. Therefore, a hybrid model was proposed as the final solution.

In the upper graph in Fig. 8, the blue line shows the total rewards received during the episode, whereas the orange lines visualises the number of steps needed to solve it. Fig 8 bottom shows the characteristics of the agent during the evaluation, i.e. each breakpoint is one of the five test scenarios (the colours of the breakpoints have the same meaning as for the upper graph). In comparison, the model using the queue achieves rewards of circa 1200 for all test scenarios, as shown in Fig. 9.

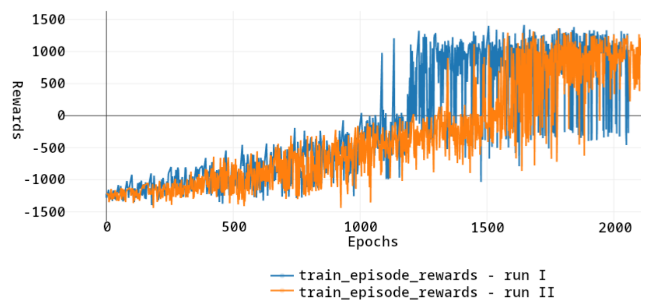


Fig. 9. Evaluation results of the queue-based decision-making model on a full simulator for a scenario with 30 orders executed per day

A hybrid approach, schematically shown in Fig. 10, was also analysed. An agent trained using reinforcement learning was used to handle the kitchen, whereas an agent using a queue was in charge of the courier module. The latter manages consecutive events, i.e. assigning orders to a courier and dispatching orders, in a FIFO-compliant manner. In addition, an attempt was made to use a popular extension of the DQN concept: Dueling DQN, PER DQN, and NoisyNet DQN. However, these extensions have not improved the final performance, as the model still performed worse than the baseline during the evaluation.

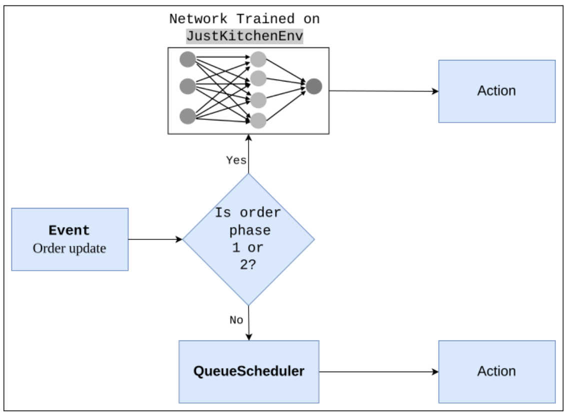


Fig. 10. The hybrid model idea

The final evaluation of the proposed models was carried out only for the hybrid model and the queue. The scenarios for which model comparisons were carried out were generated considering the following parameters:

- Number of cooks, kitchen desks and ovens: 2,
- Number of couriers: 1,
- Number of orders processed in 8 h: 30.

Ten scenarios were generated. Half of them is used for training and the other half for evaluation. The training parameters were selected using the Ray Tune library [9] and take the following learning rate, γ and epochs values: 0.000174331, 0.933775, and 2000, respectively.

For a total number of 30 orders, the sum of rewards received during training converges to a positive value, similar to the sum received using the queue-based model. The final reward was equal to 1352.25 and the duration was equal to 196.

The variance at the end of training decreased significantly, so it should be concluded that the model succeeded in finding a policy that works in the setting of the kitchen module.

For the hybrid model prepared in this way, an evaluation and milestone verification were carried out on the target simulator.

The hybrid model using the procedure when evaluating on test scenarios for both total rewards (more is better) and average order delivery time (less is better) performs better than the queue-only model, as shown in Figure 9. The obtained average delivery time for the hybrid approach is equal to 30.886 ± 2.185 min, whereas for the solution with queue only is equal to 37.443 ± 6.864 min.

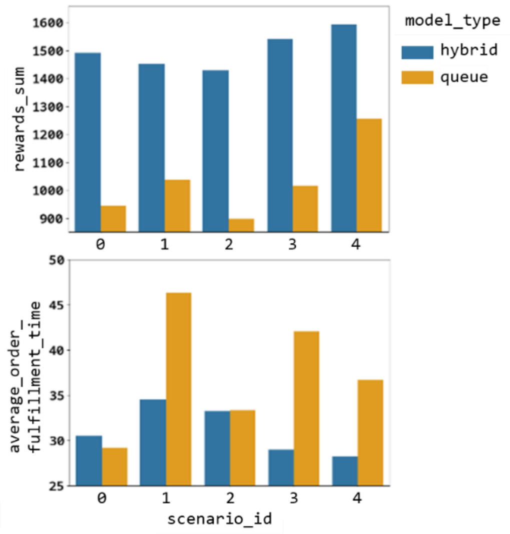


Fig. 11. Comparison of the hybrid model with the model using the queue

Each point in Fig. 11 represents one of the test scenarios. The upper graph shows the total rewards received for each model, while the one shown below visualises the average turnaround time for an order during the execution of a given scenario.

The data in Figure 11 clearly demonstrate that the hybrid model, with incoming orders, fulfills them as quickly as possible, both for the stages performed by the kitchen (the DQN model) and with the delivery process (the use of the queue). An example of the time taken to complete the various stages of orders for one of the test scenarios by the hybrid model is shown in Fig. 12.

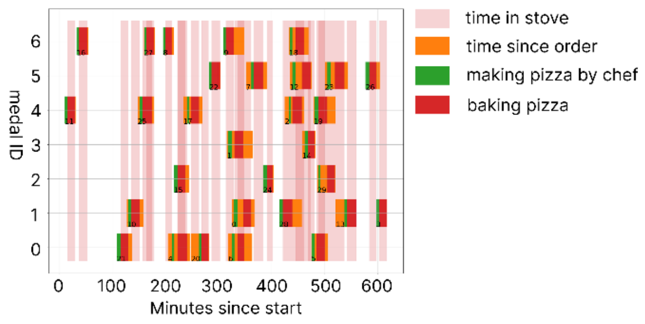


Fig. 12. Diagram of the time taken to complete the various stages of orders for one of the test scenarios by the hybrid model

6. Conclusion

This work presents the findings of a study on scheduling tasks using various RL techniques. The study was conducted in a simulated environment designed to replicate a "delivery-based" restaurant business model. A rule-based, queue-driven model (FIFO) served as the baseline for absolute quality comparison of the generated schedules. The results show that, for the defined problem, the quality of the scheduling outcomes varies significantly depending on the choice of learning algorithm.

Notably, the hybrid approach performed best under simulation conditions, considerably reducing the total completion time in a scenario reflecting the operations of a small restaurant.

However, certain limitations of the obtained results should be noted. These results do not account for operational errors by staff or the complexities involved in larger restaurant setups, which may include numerous couriers, chefs, and multiple orders to fulfill. Such conditions greatly expand the state space that the agent must explore during training. Therefore, future research will be extended to address these challenges.

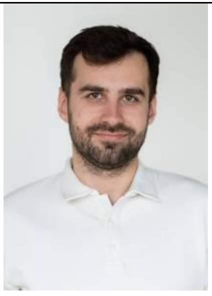
References

[1] Deguchi A. et al.: What is society 5.0. Society 5.0, 2020, 1–24.
[2] Dziurzanski P., Zhao S., Indrusiak L. S.: Integrated Process Planning and Scheduling in Commercial Smart Kitchens. arXiv preprint arXiv:1910.03322, 2019.
[3] Dziurzanski P., Kabala K., Konrad A.: A stochastic interval algebra for smart factory process. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska – IAPGOS 1, 2025, 33–38.
[4] Hamada R., et al.: Cooking navi: assistant for daily cooking in kitchen. 13th annual ACM international conference on Multimedia. 2005.
[5] Kiesel J.: The internet of things in restaurants. 2018 [https://www.manufacturingtomorrow.com/article/2017/03/the-internet-of-things-in-restaurants/9261].
[6] Lei J., Ren X., Fox D.: Fine-grained kitchen activity recognition using rgb-d. ACM Conference on Ubiquitous Computing. 2012.
[7] Mizrahi M., et al.: Digital gastronomy: Methods & recipes for hybrid cooking. 29th Annual Symposium on User Interface Software and Technology. 2016.
[8] Mohan A., Zhang A., Lindauer M.: Structure in Deep Reinforcement Learning: A Survey and Open Problems. Journal of Artificial Intelligence Research 79, 2024, 1167–1236.
[9] Moritz P., et al.: Ray: A distributed framework for emerging {AI} applications. 13th USENIX symposium on operating systems design and implementation – OSDI 18. 2018.
[10] Padakandla S.: A survey of reinforcement learning algorithms for dynamically varying environments. ACM Computing Surveys – CSUR 54(6), 2021, 1–25.
[11] Sutton R. S., Barto A. G.: Reinforcement learning: An introduction. MIT press, 2018.
[12] Ustundag A., Cevikcan E.: Industry 4.0: managing the digital transformation. Springer, 2017.
[13] Van Hasselt H., Guez A., Silver D.: Deep reinforcement learning with double q-learning. AAAI conference on artificial intelligence 30(1), 2016.
[14] Iterative.ai. Data Version Control (DVC). Iterative.ai, 2025 [http://dvc.org] (accessed: 21.08.2025).
[15] MLflow Project (part of LF Projects, LLC). MLflow. MLflow Project, 2025 [http://mlflow.org] (accessed: 21.08.2025).
[16] Project Jupyter. Project Jupyter. Project Jupyter, 2025 [http://jupyter.org] (accessed: 21.08.2025).
[17] PyTorch Foundation. PyTorch. PyTorch Foundation, 2025 [http://pytorch.org] (accessed: 21.08.2025).
[18] Rynek wewnętrzny w 2022 r., Analizy statystyczne. Główny Urząd Statystyczny. 2023.

M.Sc. Eng. Karol Kabala
e-mail: k.kabala@numlabs.com

A graduate with a master's degree in automation and robotics from AGH University of Science and Technology in Kraków. For the past 5 years, he has been associated with Numlabs, a company that conducts research and development projects in the fields of information technology and artificial intelligence. His areas of interest include the application of optimization techniques and computer vision in industry 4.0 and remote sensing.

https://orcid.org/0000-0003-0380-9270



D.Sc., Ph.D. Eng. Piotr Dziurzanski
e-mail: piotr.dziurzanski@zut.edu.pl

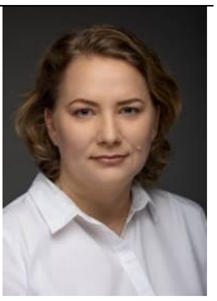
He received the Ph.D. and D.Sc. degrees in computer engineering from the West Pomeranian University of Technology in Szczecin, Poland, in 2003 and 2022, respectively. Currently, he is an associate professor at that university, and a visiting researcher at the University of York, UK. In the past, he worked at the University of York, UK, and Staffordshire University, UK. He was involved in 2 EU-funded research projects and a national project funded by EPSRC (UK). He was also PI or CI in 3 national projects funded by the National Science Centre in Poland. His main research interest is related to resource allocation, cloud computing, parallel processing, and reconfigurable hardware.

https://orcid.org/0000-0001-9542-652X

M.Sc. Agnieszka Konrad
e-mail: aga.konrad@gmail.com

Agnieszka Konrad graduated with a degree in International Relations from Poznań University of Economics (Poland) and holds a degree in European Economy & Management from Abertay University Dundee (Scotland). Currently, she serves as the Deputy Director for Collaboration at the Institute of Bioorganic Chemistry, Polish Academy of Sciences in Poznań. Previously, she managed Project Support Office focusing on enhancing the administration and coordination of research projects. Her professional experience includes collaboration on numerous projects at the national and international levels.
Her interests include research collaboration and the development of strategic partnerships within the scientific community.

https://orcid.org/0000-0002-7900-3153



Article realized as part of the project No. POIR.01.01.01-00-0799/21 co-financed by the European Union from the European Regional Development Fund under the Smart Growth Programme. The project was implemented by Papukurier Sp. z o.o. as part of the National Centre for Research and Development's Fast Track call.