Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

Optimization of Shell and Tube Heat Exchangers Using Reinforcement Learning

Luana P. Queiroz[a,b,c*], Olve R. Bruaset[c], Ana M. Ribeiro[a,b], Idelfonso B. R. Nogueira[c]

[a]LSRE-LCM – Laboratory of Separation and Reaction Engineering - Laboratory of Catalysis and Materials, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, Porto, 4200-465, Portugal
[b]ALiCE – Associate Laboratory in Chemical Engineering, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, Porto, 4200-465, Portugal
[c]Chemical Engineering Department, Norwegian University of Science and Technology, Trondheim, 793101, Norway
* Corresponding Author: up201700139@edu.fe.up.pt.

ABSTRACT

This work presents a model for optimizing shell-and-tube heat exchanger design using Q-learning, a reinforcement learning technique. An agent is trained to interact with a simulated environment of a heat exchange model, iteratively refining design configurations to maximize a reward function. This reward function balances heat exchanger effectiveness and pressure drop, emphasizing designs that minimize pressure drop. Results showed that simpler configurations consistently achieved higher rewards, despite complex designs offering better heat transfer efficiency.

Keywords: heat exchanger, reinforcement learning, design optimization, machine learning

INTRODUCTION

Heat exchangers are essential components across industries, facilitating heat transfer between fluids of differing temperatures [1]. Since their development in the early 20th century for industrial steam generators and power plant condensers, their applications have expanded into refrigeration, automotive cooling, waste heat recovery, and petrochemical processes [2]. The optimization of heat exchangers is directly connected to the energy and operational costs [3]. Thus, linking this equipment to important performance parameters. However, traditional design methods often rely on iterative manual processes that fail to guarantee optimal solutions due to the complexity of influencing parameters [4].

Recent advancements in Scientific Machine Learning (SciML) offer promising avenues for addressing these challenges. By integrating scientific principles with machine learning, SciML has shown potential in optimizing intricate systems like heat exchangers [5]. For instance, reinforcement learning (RL), a branch of SciML, can be utilized to identify optimal designs by exploring vast solution spaces without manual intervention. For example, a Scopus search using "heat exchanger" and "machine learning" revealed 259 studies, the narrower term "q-learning," an RL technique, yielded just five results. Research highlights, such as Wang et al. (2023) [6], demonstrate the potential of machine learning and computational fluid dynamics to enhance heat transfer in finned heat pipe radiators by optimizing parameters like fluid flow and thermal efficiency. By leveraging machine learning models trained on CFD simulation data, the authors identified design configurations that significantly enhanced thermal performance while maintaining manufacturability. This approach highlights the value of integrating machine learning with traditional physics-based methods to tackle the complexities of heat exchanger optimization.

Studies leveraging machine learning for heat exchanger optimization have explored diverse methods. For instance, Nevin et al. (2023) [7] employed Gaussian Process Regression and Random Forest models to enhance performance with turbulator inserts. In contrast, Keramati et al. (2022) [8] applied Deep Reinforcement Learning to optimize heat exchanger geometry, balancing heat transfer efficiency with reduced pressure drop. While these techniques have made notable advances, commercial optimization tools like Aspen EDR and HTRI Heat Exchanger Suite are still widely used in industry. These tools typically rely on meta-heuristics to solve

design problems, which, while effective, are often computationally intensive and require significant time for design iteration [9, 10]. This work differentiates itself by applying Q-learning, a simpler yet efficient reinforcement learning method, to optimize shell and tube heat exchangers, offering a potential for faster design evaluation and reduced computational time compared to traditional methods. The study focuses on maximizing a reward function defined by heat exchanger effectiveness and pressure drop. Despite limited precedent in applying Q-learning to this field, the findings from this exploratory work provide a foundation for further development.

BACKGROUND

Shell and Tube Heat Exchangers

Shell and tube heat exchangers are versatile and widely used in industrial applications due to their scalability, durability, and flexibility. In these exchangers, one fluid flows through a bundle of tubes, while another fluid circulates around the tubes within a shell. This arrangement enables heat transfer, where the hotter fluid warms the cooler one or vice versa. The efficiency of this process depends on factors like the overall heat transfer coefficient, heat transfer surface area, and the temperature gradient between fluids [1].

To optimize heat transfer, shell and tube exchangers incorporate numerous small tubes within a single shell. Baffles inside the shell direct the fluid in a serpentine path, creating turbulence that enhances heat transfer by breaking up thermal boundary layers on the tube surfaces [1]. Figure 1 illustrates a typical shell and tube heat exchanger with a one-shell pass and one-tube pass configuration, showing the internal structure and flow arrangement.

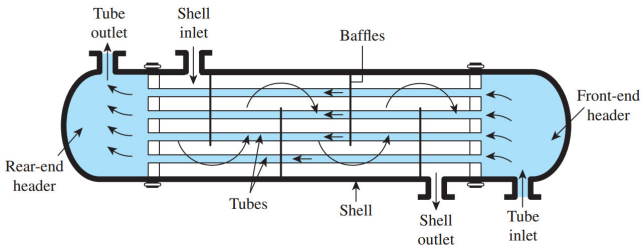

Figure 1. Shell and tube heat exchanger with one-shell pass and one-tube pass scheme.[1]

The flexibility of shell and tube heat exchangers allows for various configurations tailored to specific operational needs. These designs, combined with their ability to withstand high pressure and temperature, make them ideal for chemical processing, power generation, and petrochemical production [1]. This work employs the effectiveness-NTU (ε-NTU) method to evaluate the performance and design of such heat exchangers.

Performance-indicating Parameters

The heat transfer rate (q) in a shell and tube exchanger is often calculated using the rate equation:

$$q = UA_s\Delta T_{lm} \tag{1}$$

where *U* is the overall heat transfer coefficient, A_s is the total surface area for heat transfer, and ΔT_{lm} is the log mean temperature difference (LMTD). However, when only inlet temperatures are available, the effectiveness-NTU method simplifies the process by introducing dimensionless parameters like NTU (number of transfer units) and heat capacity ratios to directly estimate the exchanger's effectiveness (ε) [10].

The relationship for effectiveness in a single-shell pass shell and tube exchanger is:

$$\varepsilon_1 = \frac{2}{1 + C_r + \sqrt{1 + C_r^{\,2}}\,\frac{1 + exp\,(-\Gamma)}{1 - exp\,(-\Gamma)}} \tag{2}$$

where C_r is the heat capacity ratio $C_r = C_{min}/C_{max}$, and $\Gamma = NTU\sqrt{1 + C_r^{\,2}}$. For exchangers with multiple shell passes, the generalized effectiveness equation is:

$$\varepsilon = \frac{\left(\frac{1-\varepsilon_1 C_r}{1-\varepsilon_1}\right)^n - 1}{\left(\frac{1-\varepsilon_1 C_r}{1-\varepsilon_1}\right)^n - C_r} \tag{3}$$

High effectiveness values indicate efficient heat transfer, making these parameters crucial for assessing exchanger performance. Additional considerations, such as temperature profiles and pressure drop, provide a comprehensive evaluation.

The pressure drop (ΔP) is critical for evaluating performance, calculated using the *Darcy-Weisbach* equation [12]:

$$\Delta P = f\frac{v^2}{2}\frac{L}{d_i} \tag{4}$$

Here, ΔP is the pressure drop across the tubes; f is the friction factor; and L is the total length of the pipes, calculated as the length of each pipe (l) multiplied by the number of shell passes (n): L = l · n. The friction factor is determined using the *Colebrook* equation [12]:

$$\frac{1}{\sqrt{f}} = -2\log\left(\frac{\frac{e}{d_i}}{3.7} + \frac{2.51}{Re\sqrt{f}}\right) \tag{5}$$

where *e* is the absolute roughness of the pipes, and *Re* is the Reynolds number. The Reynolds number describes the flow patterns of a fluid, indicating whether the flow is laminar or turbulent. For fluid flow within a pipe, it is defined as [13]:

$$Re = \frac{v d_i \rho}{\mu} \tag{6}$$

where μ is the fluid viscosity.

These equations provide the foundation for evaluating heat exchanger performance by calculating key metrics such as effectiveness, temperature profiles, and pressure drop. In this work, these parameters play a central role in assessing and optimizing heat exchanger designs through the application of Q-learning.

Reinforcement Learning

Reinforcement learning (RL) is a machine learning technique where an agent interacts with an artificial environment to maximize cumulative rewards, based on the Markov Decision Process (MDP) [14]. The MDP provides a mathematical framework for decision-making, particularly suited to dynamic programming and optimization challenges [15].

In an MDP, the agent observes the current state S_t , selects an action A_t, and receives a reward R_t+1 alongside a new state at each time step [16]. Transitions between states follow the Markov Property, where the next state depends only on the current state and action. A visual representation of this process is illustrated in Figure 2.

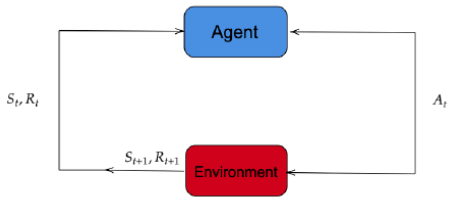

Figure 2. A simplified representation of MDP.

The agent's goal is to maximize total return over time by optimizing its policy π (a|s), which determines the probability of taking action a in state s. Value functions are key to RL, estimating the expected return for a policy π. Iterative evaluation and policy improvement enable the agent to identify the optimal policy $\pi *$, maximizing returns.

While models of the environment can simplify this process, RL often relies on experience, where episodes of states, actions, and rewards guide learning [16].

Q-Learning

Q-learning, introduced by Watkins (1989), is a reinforcement learning algorithm based on Temporal Difference (TD) learning. It estimates an optimal decision strategy for infinite-horizon problems without requiring an environmental model [17, 18]. Q-learning adopts an off-policy approach within TD learning, meaning that the optimal value function derives from actions chosen independently of the current policy π (a|s) [16]. The updating rule proposed by *Watkins* (1989) is [17]:

$$Q\ (S_t, A_t) \leftarrow\ Q\ (S_t, A_t) + \alpha[T_{t+1} + \gamma\ max_a Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (7)$$

Here, α is the learning rate, γ is the discount factor, and $max_a Q(S_{t+1}, A_{t+1})$ represents the estimated optimal future reward. A Q-table, as shown in Table 1, stores these values to guide the agent's decisions.

Table 1: A simple schematic of a Q-table.

	A_1	A_2	...	A_n
S_1	$Q\ (S_1, A_1)$	$Q\ (S_1, A_2)$	...	$Q\ (S_1, A_n)$
S_2	$Q\ (S_1, A_1)$	$Q\ (S_1, A_2)$	...	$Q\ (S_1, A_n)$
...	...	...	....	...
S_m	$Q\ (S_m, A_1)$	$Q\ (S_m, A_2)$	....	$Q\ (S_m, A_n)$

The agent balances exploration (trying new actions) and exploitation (choosing the best-known actions) using the ϵ-greedy method. Exploration probability ϵ decreases over time, transitioning from exploration to exploitation.

Q-learning's adaptability and effectiveness make it a powerful tool for solving dynamic decision-making problems, enabling agents to learn optimal strategies through continuous experience.

METHODOLOGY

The development and execution of the Python program for this work were conducted on the following computer setup:

- Hardware:
  - Processor: 1.4 GHz quad-core Intel Core i5
  - Memory (RAM): 8 GB 2133 MHz LPDDR3
  - Storage: 256 GB
- Operating System: macOS Sonoma V. 14.4.1
- Development Environment:
  - IDE/Editor: Virtual Studio Code V. 1.88.0
  - Python Version: Python 3.9.18
  - Key Libraries and Frameworks: NumPy, Matplotlib, SciPy, Gym, Pandas, Os

Database

To enable Q-learning and train the model, a database containing the necessary input parameters for the heat exchanger had to be provided. Since experimental data was not available, this database was artificially created using randomly selected values with reasonable intervals. For simplicity, all fluids were assumed to be water. The required input data included inlet mass flow ($\dot{m}$), overall heat transfer coefficient (U), heat capacities (Cp), and inlet temperatures (T) for both the hot and cold fluids. The intervals set for each parameter were chosen based on plausible values, as shown in Table 2.

Table 2: Model input parameters with corresponding intervals.

Parameter	Interval
U [W/m^2K]	[400, 1500]
$\dot{m}_{hot}$ [kg/s]	[0.2, 1.5]
$\dot{m}_{cold}$ [kg/s]	[0.2, 1.5]
$C_{p.hot}$ [kJ/kgK]	[3.8, 4.5]
$C_{p.cold}$ [kJ/kgK]	[3.8, 4.5]
T_{hot} [°C]	[70, 200]
T_{cold} [°C]	[10, 30]

The database was created using a for-loop that generated 100 examples, which were stored in a DataFrame1. These data points were then randomly selected by the program during each training step.

Heat Exchanger Calculations

A core component of the program calculates key performance parameters:

Effectiveness (ε): Calculated using the ε-NTU method, considering the number of shell and tube passes. The program implements the necessary equations for single and multiple shell passes.

Pressure Drop (ΔP): Calculated using the Darcy-Weisbach equation, with the friction factor determined using the Colebrook equation and the Reynolds number.

Temperature Profiles: Calculated by solving discretized versions of the energy balance PDEs for the tube and shell sides, including a stabilizing thermal diffusion term.

The calculate_performance() function takes input parameters (from the database and chosen action) and predefined fluid/tube properties (Table 3) to compute these performance metrics. The hot fluid was consistently assigned to the tube side and the cold fluid to the shell side.

Table 3: Predefined parameters for the shell and tube side.

Parameter	Tube side	Shell side
Type of fluid	Water	Water
Density, ρ $[kg/m^3]$	997	997
Dynamic viscosity, μ $[Pa \cdot s]$	$10.52 \cdot 10^{-3}$	$10.52 \cdot 10^{-3}$
Thermal conductivity, κ $[W/m \cdot K]$	0.61	0.61
Internal diameter of tubes, d_i $[m]$	$20 \cdot 10^{-3}$	
Outer diameter of tubes, d_o $[m]$	$22 \cdot 10^{-3}$	
Tube length, l $[m]$	1	

Q-Learning Implementation

The Q-learning algorithm was implemented through a structured framework consisting of an environment, an agent class, and a systematic training process:

Environment Class The reward function combines the effectiveness factor (ε) and pressure drop (ΔP) to evaluate the performance of the heat exchanger. By adjusting the pressure drop to a scale of [0, 22] psi, the model is incentivized to maximize ε and minimize ΔP, promoting optimal design outcomes with high efficiency and low energy loss. The step() function calculates the next state, reward, and a stop criterion based on maximum steps per episode and steps since reward increase. The reset() function initializes the environment for each new data point.

$$reward = \varepsilon + \frac{1}{\Delta P_{psi}} \quad (8)$$

Table 4: Total number of configurations based on the action space.

Tube passes / Shell passes (n)	2n	4n	6n
1	(1,2)	(1,4)	(1,6)
2	(2,4)	(2,8)	(2,12)
3	(3,6)	(3,12)	(3,18)

Agent Class Implements the Q-learning agent. It initializes with a learning rate (0.1), discount factor (0.9), and epsilon value (0.1). The choose_action() function uses the ε-greedy policy for action selection. The update() function updates the Q-table based on the Q-learning update rule.

Training Process (training.py): The training process follows these steps:

1. Read heat exchanger parameters from the database.
2. Initialize the agent and environment.
3. Iterate through episodes (50) and data points (10 per episode).
4. For each data point:
   - Randomly select heat exchanger parameters.
   - Reset the environment.
   - While the stop criterion is not met:
     - Agent chooses an action using ε-greedy.
     - Environment performs the action and returns the next state, reward, etc.
     - Agent updates the Q-table.
5. Store best-performing data points, cumulative rewards, and trained agents.

RESULTS

The implementation of the program required approximately 48 hours on the specified computer, averaging under one hour per episode. The average reward per episode is shown in Figure 3.

Performance varied significantly across episodes due to the reset of training for each new episode, which provided insight into diverse configurations. The training aimed to optimize performance for specific parameters, achieving near-optimal designs through 5000 simulation steps per episode.

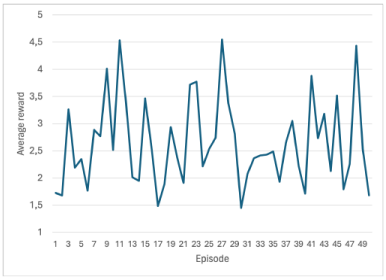

Figure 3. Average reward for each episode of the training procedure.

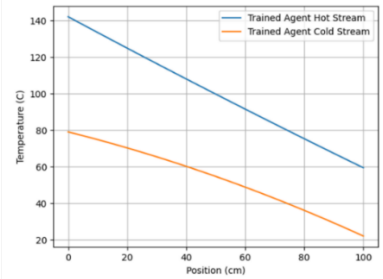

Figure 4. Temperature profile for the best performing epoch within data point 10, episode 27.

Best and Worst Performing Episodes

Episode 27 achieved the highest reward, with an average score of 4.549. Within this episode, data point 10 reached a peak reward of 10.450. The details are presented in Table 5, and the corresponding temperature profile is illustrated in Figure 4.

Table 5: Properties of the best performing epoch within data point 10, episode 27.

Properties	Hot fluid	Cold fluid
Reward	10.450	
Effectiveness factor, ε [-]	0.687	
Pressure drop, ΔP [psi]	0.706	
Overall heat transfer coefficient, U [W/m^2K]	683	
Inlet mass flow, $\dot{m}$ [kg/s]	0.218	0.358
Heat capacity, C_p [kJ/kgK]	4.392	3.891
Inlet temperature, T_{in} [ºC]	142	22
Action, (nº shell passes, nº tube passes)	(1,2)	

The heat exchanger achieved 69% effectiveness with minimal pressure drop (0.7 psi), using a configuration of one shell pass and two tube passes. Although higher effectiveness might be expected for better performance, the reward function heavily penalizes high-pressure drops, making this configuration optimal. The temperature profiles indicate consistent energy conservation principles, with the hot stream cooling from 142°C to 59°C and the cold stream heating from 22°C to 79°C.

In contrast, episode 30 recorded the lowest reward, averaging 1.450. Data point 2, with a reward of 0.780, stands out as the poorest performer due to a high-pressure drop of 190 psi despite achieving 75% effectiveness. This pressure drop renders the configuration impractical for real-world applications. The suboptimal results highlight the importance of balancing effectiveness and pressure drop for optimal design.

Optimal Design Insights

Analysis of the top-performing epochs identified the (1, 2) configuration as the most effective design. This configuration consistently achieved superior rewards due to a favorable balance between moderate effectiveness (~69%) and minimal pressure drop (~0.7 psi). Although designs with additional shell or tube passes offered higher effectiveness, they incurred significantly higher pressure drops, reducing their overall performance scores.

Comparative Analysis

Simpler configurations like (1, 2) outperformed more complex designs, demonstrating the trade-off between heat transfer efficiency and system practicality. The reward system penalized configurations with excessive pressure drops, reinforcing the preference for designs that optimize both criteria.

CONCLUSION

This work proposed a novel framework for optimizing heat exchanger design using Q-learning, showcasing the integration of Scientific Machine Learning (SciML) with heat exchanger optimization. The model was built upon fundamental heat transfer principles and the ε-NTU method, enabling a thorough performance evaluation based on effectiveness and pressure drop across the heat exchanger.

Q-learning was chosen for its capability to operate without a precise system model, relying instead on trial-and-error exploration to navigate the design space. The model's training process incorporated an environment, an agent, and a structured training sequence, with the agent iteratively maximizing the reward function by exploring various design configurations.

The findings revealed that the simplest design was identified as optimal within the model's constraints, primarily due to the reward function's emphasis on minimizing pressure drop over maximizing heat transfer effectiveness. This outcome underscores the critical role of reward function design in guiding optimization priorities. A comparative analysis demonstrated that while more intricate designs improved heat transfer efficiency, they yielded lower reward scores due to higher pressure drops.

Future improvements to the framework should focus on enhancing its versatility and realism. These enhancements include incorporating a broader range of fluid types, integrating more detailed and accurate heat transfer equations, expanding the diversity of design configurations, and refining the reward function to balance multiple performance parameters more effectively. Additionally, leveraging real-world datasets for training and validation will increase the model's reliability and applicability, paving the way for more robust and practical optimization outcomes.

ACKNOWLEDGEMENTS

This work was supported by national funds through FCT/MCTES (PIDDAC) UI/BD/154743/2023; FCT/MCTES (PIDDAC): LSRE-LCM, UIDB/50020/2020 (DOI: 10.54499/UIDB/50020/2020) and UIDP/50020/2020 (DOI: 10.54499/UIDP/50020/2020); and ALiCE, LA/P/0045/2020 (DOI: 10.54499/LA/P/0045/2020).

REFERENCES

1. Balaji C, Srinivasan B, Gedupudi S. “Chapter 7 - Heat exchangers”. In: Heat Transfer Engineering: Fundamentals and Techniques. Academic Press, 2021, pp. 199–231. https://doi.org/10.1016/B978-0-12-818503-2.00007-1
2. Theodore L. Heat transfer applications for the practicing engineer. John Wiley & Sons, 2011, pp. 3–5. ISBN: 978-0-47-064372-3.
3. Caputo AC, Pelagagge PM, Salini P. Heat exchanger design based on economic optimisation. *App Therm Eng* 28:1151-1159 (2008) https://doi.org/10.1016/j.applthermaleng.2007.08.010
4. Saxena R, Yadav S. Designing Steps for a Heat Exchanger. *Int J Eng Res Tech* 2:943-959 (2013).
5. Iwema J. “Scientific Machine Learning”. In: Wageningen University & Research (Jan. 2023). URL: https://sciml.wur.nl/reviews/sciml/sciml.html
6. Wang Y, Ma Y, Chao H. Machine learning and computational fluid dynamics-based optimization of finned heat pipe radiator performance. *J Build Eng* 7 8:107612 (2023) https://doi.org/10.1016/j.jobe.2023.107612
7. Celik N, Tasar B, Kapan S, Tanyildizi V. Performance optimization of a heat exchanger with coiled-wire turbulator insert by using various machine learning methods. *Int J Therm Sci* 192:108439 (2023). https://doi.org/10.1016/j.ijthermalsci.2023.108439
8. Keramati H, Hamdullahpur F, Barzegari M. Deep reinforcement learning for heat exchanger shape optimization. *Int J Heat Mass Transfer* 194:123112 (2022)https://doi.org/10.1016/j.ijheatmasstransfer.2022.123112
9. Handibag R, Potdar DU, Jadhav A. Thermal design of tube and shell heat exchanger and verification by HTRI software. *Int J Eng Res Tech* 9:525-530 (2020).
10. Janaun J, Kamin NH, Wong KH, Tham HJ, Kong VV, Farajpourlar M. Design and simulation of heat exchangers using Aspen HYSYS, and Aspen exchanger design and rating for paddy drying application. In *IOP Conference Series: Earth and Environmental Science* 36:1:012056 (2020)
11. Incropera FP et al. Fundamentals of Heat and Mass Transfer. 6th ed. John Wiley & Sons, 2007, pp. 686–699. ISBN: 978-0-471-45728-2.
12. Mohammed Rabeeh V, Vysakh S. Design of Shell and Tube Heat Exchanger Using MATLAB and Finding the Steady State Time Using Energy Balance Equation. *Int J Adv Mech Eng* 4:95-100 (2014), pp. 95–100. ISSN: 2250-3234. URL: https://www.ripublication.com/ijame-spl/ijamev4n1spl_12.pdf
13. Papaevangelou G, Evangelides C, C Tzimopoulos C. A new explicit relation for friction coefficient f in the Darcy-Weisbach equation. In: *Proc 10th Conf Protect Restor Environ*. 166:6-9 (2010).
14. Niranjan K. “Elements of Fluid Flow”. In: Engineering Principles for Food Process and Product Realization. Cham: Springer International Publishing, 2022, pp. 23–50. https://doi.org/10.1007/978-3-031-07570-4_2
15. Yamin Kao et al. “A dynamic approach to support outbreak management using reinforcement learning and semi-connected SEIQR models”. In: BMC Public Health 194 (2024). https://doi.org/10.1186/s12889-024-18251-0
16. Feinberg EA, Shwartz A. Handbook of Markov decision processes: methods and applications. Vol. 40. Springer Science & Business Media, 2012.
17. Sutton RS, Barto AG. Reinforcement learning: An introduction. 2nd ed. MIT press, 2018, pp. 47–68, 119–138. ISBN: 978-02-62039-24-6.
18. Watkins CJCH. “Learning from delayed rewards”. (May 1989)
19. Clifton J, Laber E. Q-learning: Theory and applications. *Annual Review of Statistics and Its Application* 7 (2020), pp. 279–301.