The 21st International Conference on
Thermal Science and Engineering of Serbia
SimTerm 2024
Niš, Serbia, Oct 22-25 2024
doi: 10.5937/SimTerm24216Z

Application of Reinforced Learning in Intelligent Buildings

Matija Žuža[a], Milan Ristanović[b], Žarko Ćojbašić[c] , Luka Filipović[d]

[a]*University of Belgrade – Faculty of Mechanical Engineeing, Belgrade, RS, zuza.matija@gmail.com*
[b]*University of Belgrade – Faculty of Mechanical Engineeing, Belgrade, RS, mtodorovic@mas.bg.ac.rs*
[c]*University of Niš – Faculty of Mechanical Engineeing, Niš, RS, zcojba@ni.ac.rs*
[d]*University of Belgrade – Faculty of Mechanical Engineeing, Belgrade, RS, lfilipovic@mas.bg.ac.rs*

Abstract: This paper aims to present the fundamental mathematical framework necessary for understanding reinforcement learning (RL) and provides an overview of RL algorithms, focusing on their application in Heating, Ventilation, and Air Conditioning (HVAC) systems. Specifically, the paper addresses the use of RL in Building Energy Management Systems (BEMS) to tackle the issue of high CO2 emissions resulting from HVAC operation, with RL proposed as a potential solution for reducing emissions by enhancing energy efficiency while maintaining occupant comfort. Additionally, the paper highlights the key advantages and limitations of RL when applied in intelligent buildings. The review bridges theoretical concepts and findings from the literature to identify appropriate algorithms for various problems and highlight research gaps. Furthermore, the future research direction of meta-RL is discussed, which trains agents on diverse tasks, offering strong generalization capabilities, making RL algorithms more adaptable to real-world conditions.

Keywords: Reinforcement learning, HVAC, Intelligent Building, BEMS.

1. Introduction

The Heating, Ventilation, and Air Conditioning (HVAC) system plays a crucial role in maintaining a comfortable and safe indoor environment in residential, commercial, and industrial buildings by regulating temperature, air quality, and domestic hot water. In 2022, HVAC systems represented around 16.4% of global energy consumption, with most of this energy sourced from fossil fuels like coal, oil, and natural gas. As a result, these systems are responsible for 14% of the world's operational CO2 emissions, a significant contributor to global warming.

This heightened awareness of HVAC systems impact on carbon emissions has gained importance with international efforts to combat climate change. The Paris Agreement, established during the 2015 United Nations climate conference, emphasized reducing carbon emissions, prompting many countries to develop building energy regulations. In response, several nations are adopting stringent energy efficiency standards that require the implementation of advanced digital controllers based on MPC and RL algorithms. These modern systems are designed to optimize HVAC performance, reducing energy usage while maintaining occupant comfort.

2. Brief introduction to reinforcement learning

Reinforcement learning (RL) is unsupervised method of machine learning. Key components of reinforcement learning are **agent** and **environment**, see figure 1, where the agent is responsible for taking actions that changes state of the environment. Agent in state s_t in the environment is taking action a_t. When agent took action a_t he changed state of environment, resulting in next state s_{t+1} and gets reward r_{t+1}.

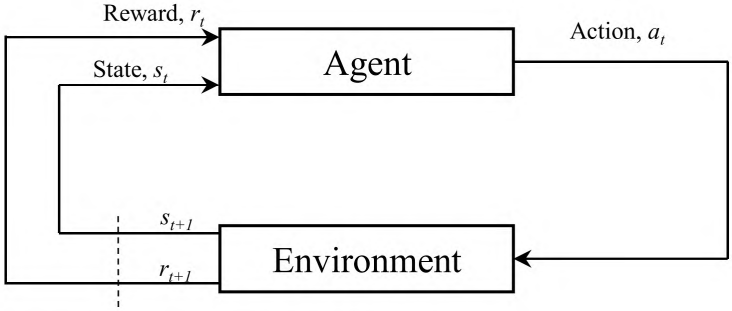

Figure 1. Schematic of Markov Decision Process (MDP)

3. Modeling of environment with RL

Markov decision process (MDP) is the framework for modeling environment with RL. MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathcal{P}, \Omega, \mathcal{O} \rangle$, where:

- $\mathcal{S}$ is a set of states ($s \in S$),
- $\mathcal{A}$ is a set of actions ($a \in A$),
- $\mathcal{R}$ is a set of rewards ($r \in R$),
- $\mathcal{T}$ is a set of time epochs
- $\mathcal{P}$ is **dynamic**/transition model, it represents the probability that agent change state from $s \rightarrow s'$, and getting a reward after he take an action a in every epoch e, which satisfies Markov property (1).

$$\mathcal{P}(s', r|s, a, e) = \mathbb{P}(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a, T = e), \ \forall s', s, a, r \quad (1)$$

- Ω is a set of obesrvations
- $\mathcal{O}$ is probability distribution of getting observation o after taking action a and visiting state s' (2):

$$\mathcal{O}(o|s', a) = \mathbb{P}(O_{t+1} = o \mid S_{t+1} = s, A_{t+1} = a). \quad (2)$$

3.1 State $\mathcal{S}$

State is representing current state of the environment, giving enough amount of data used for making decisions. In HVAC systems, states can be divided in two categories:

- **Endogenous** elements, like room temperature, energy consumption, air quality in the term of concentration of CO2, relative humidity, and actuators can take actions to change the those states like fan speed and airflow rate, system mode (heating, cooling, ventilation), ventilation levels.
- **Exogenous** elements, like outside temperature, energy prices, using schedule (if occupancy in a building or room changes in a way that is not predictable or controlled by the HVAC system), weather conditions (wind speed, solar radiation, precipitation), *etc.*

This division of states allows us to clearly distinguish between the states that can be directly controlled (endogenous states), and those that are beyond influence of agent actions (exogenous states), thereby enabling more precise and efficient optimization of HVAC systems. The HVAC system can control indoor temperature by adjusting its heating or cooling output. This is a key endogenous parameter that the system regulates to maintain occupant comfort. The speed of the HVAC system's fans or the rate of air circulation is a controllable parameter. Adjusting fan speed influences energy consumption and temperature regulation inside the building. The level of fresh air ventilation is often controllable by the HVAC system, affecting both indoor air quality and energy efficiency. The system can adjust this parameter based on current needs, such as when indoor CO2 levels are high. Variable energy tariffs or real-time electricity prices are typically beyond the control of the system but have a significant impact on how the HVAC system operates, especially if energy cost optimization is part of the goal. Beyond temperature, other weather-related factors like wind speed, solar radiation, and precipitation can affect building thermal dynamics but are not directly controllable by the system.

3.2 Actions $\mathcal{A}$

Action space in MDP is a set of decisions through which controller can influence the environment. In HVAC systems complexity appears in diversity of components that HVAC system contains, where each one of them has different control mechanisms. Those components are:

- condenser – crucial component in refrigeration cycle of chillers and heat pumps,
- the Source component, such as boilers, chillers, and heat pumps, which creates the heating or cooling medium.
- Air Handling Units – AHUs
- Terminal components, like VAV boxes, fan coils, radiators, which offers localised temperature control.

Action space gives as overview of elements that we control, to obtain thermal comfort.

3.3 Rewards $\mathcal{R}$

Reward in HVAC systems is defined as weighted sum of following terms:

$$R_t = \alpha R_{comfort_t} + \beta R_{CO_{2_t}} + \lambda R_{energy_t} + \delta R_{cost_t} + \sigma R_{other_t}, \quad (3)$$

Where it is important to mention that product $\lambda\delta = 0$, which means that negative reward R_{energy_t} and R_{cost_t} can exclusively be derived by one of them. Either from energy consumption (measured in kWh) or the cost of energy consumption (measured in $/€).

Positive rewards R_{comfort_t} and $R_{\text{CO}_{2_t}}$, are gained if thermal comfort and optimal level of CO2 is obtained. And R_{other_t} specifies other rewards in HVAC systems, *e.g.* indoor relative humidity comfort or the management of the battery state-of-charge, particularly in PV systems (photovoltaic systems), it encourages charging the battery during off-peak hours when electricity prices are lower [7,8].

3.4 Probabilities $\mathcal{P}$ and $\mathcal{O}$

Those probabilities are essential in defining the MDP. They determine type of environments for our MDP problem, and it is derived as follows:

- partially observable Markov decision processes (POMDP),
- standard stationary Markov decision processes (SSMDP),
- standard non-stationary Markov decision processes (SNSMDP).

POMDP is the situation where the sensors are noisy or where certain aspects of environment are hidden from agent. When environment is not observable problem can be modeled through the definition of Ω and $\mathcal{O}$ while in case of observable environment, which is modeled with equations (4) and (5):

$$\Omega = \mathcal{S}, \tag{4}$$

$$\mathcal{O}(s'|s,a) = \begin{cases} 1, & if \quad s' = s \\ 0, & otherwise \end{cases} \tag{5}$$

Standard categories can be defined.

3.5 Environment problem

The main issue in implementing RL algorithms in intelligent buildings is the problem of the environment. The problem appears in lack of implementation of RL algorithms in real buildings. Main cause of the lack of implementation in real buildings is the disturbence of occupants comfort during the training procees, which can be etrimely long. The solution is to use simulations, with tools like EnergyPlus, TRNSYS, Dymola, *etc.*, see figure 2.

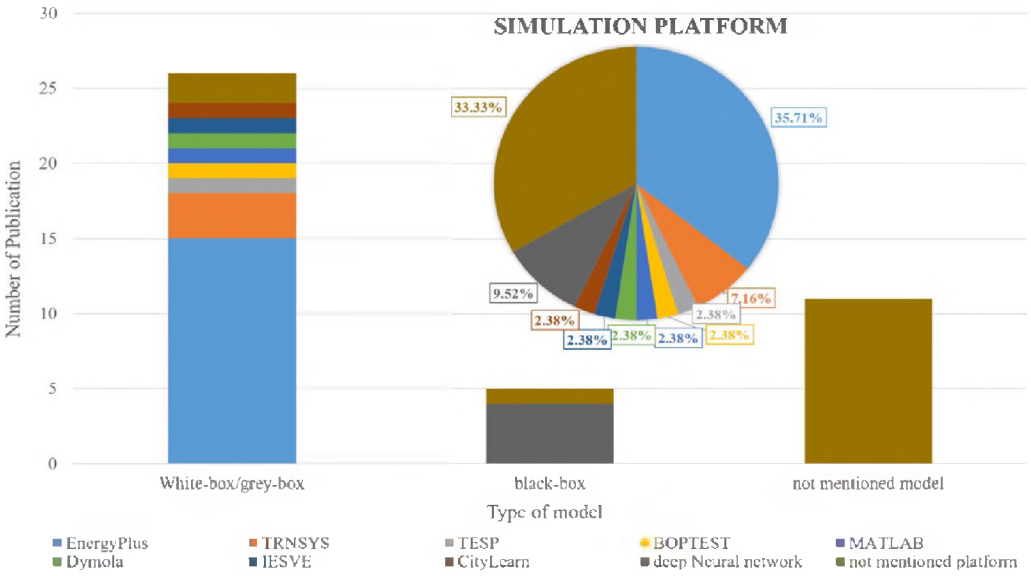

Figure 2. Setup of the simulation platform utilized for training the RL controller.

4. Classification of reinforcement learning algorithms

In each time stamp $t \in \mathbb{N}$ the agent interacts with the environment (building) receiving a state $S_t \in \mathcal{S}$ consisting of endogenous and exogenous HVAC factors, and selects action $A_t \in \mathcal{A}$ accordingly. In the next time step

$t+1$ and as a result of its action A_t, a scalar reward $R_{t+1} \in \mathcal{R}$ is given to the agent that has entered a new state S_{t+1} until the agent reaches the terminal state.

At every time stamp, the agent interacts with the environment according to a function-based policy, which is defined as follows:

Definition 4.1[9] A policy π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}(A_t = a \mid S_t = s), \tag{6}$$

Where the goal is to maximize the **return**, which is defined as:

Definition 4.2[9] The return G_t is a total discounted reward form time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{7}$$

Where $\gamma \in [0,1]$ is discount and represents the present value of future rewards.

In the contest of HVAC control problem, a γ close to 0 is typically used when the goal is to achieve optimal occupant comfort with minimal energy consumption at each timestep. In contrast, a γ close to 1 allows for occasional comfort violations or even excessive energy consumption during some time-steps, in order to achieve long-term energy savings as a final result at the end of the day, month, or year (depending on the episode length).

Categorization of RL algorithms can be show in the terms of model of the environment. According to that algorithms are divided into two categories:

- model-based algorithms
- model-free algorithms

In model-based algorithms, the dynamic of the world ($\mathcal{P}$) is known. On the other hand, model-free algorithms does not need to know the real-world dynamics $\mathcal{P}$, and instead approximate the optimal policy trough trail-and-error experiences gathered from the real interaction with the environment in less cases, or with simulations.

Despite previous division, another can be made. The algorithm's strategies for formulating the optimal decision-making policy can be segmented into three principal approaches:

- value-based
- policy-based
- actor-critic

4.1 Value Based

This methodology is applicable in finite action space. The key concept in this approach is the estimation of state-action vale function according to a specific policy. The definition of value function for the policy π is as follows:

Definition 4.1.1 [9] The state value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π.

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] \tag{8}$$

Definition 4.1.2 [9] The state-action value function $q_\pi(s,a)$ of an MDP is the expected return starting from state s, taking action a, and then following policy π.

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \tag{9}$$

The Bellman equation for v_π and q_π states that each value has its own fundamental recursive relationship [**9**].

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(s|a) \sum_{s',r} \mathcal{P}(s', r|s, a)[r + \gamma v(s')] \tag{10}$$

$$q_\pi(s,a) = \sum_{s',r} \mathcal{P}(s', r|s, a)[r + \gamma v(s')] \tag{11}$$

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(s|a) q_\pi(s,a) \tag{12}$$

According to the definition, the optimal policy π^* is the policy that maximizes value function or action-value function. Having that in mind, the optimal state-value function and optimal action-value function can be defined.

Definition 4.1.3 [9] The optimal state-value function $v_\pi(s)$ is the maximum value function over all policies.

$$v_*(s) = \max_\pi v_\pi(s) \tag{13}$$

Definition 4.1.4 [9] The optimal action-value function $q_\pi(s,a)$ is the maximum action-value function over all policies.

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{14}$$

According to the equations (10), (11), (13), (14) Bellman optimality equations can be defined as:

$$v_*(s) = \max_{a\in\mathcal{A}} \sum_{s',r} \mathcal{P}(s',r|s,a)[r + \gamma v_*(s')] \tag{15}$$

$$q_*(s,a) = \sum_{s',r} \mathcal{P}(s',r|s,a)\,[r + \gamma \max_{a'\in\mathcal{A}} q_*(s',a')] \tag{16}$$

It is not requisite for optimal policy to be singular. Rather, a general formulation for any optimal policy, denoted as π_* can be stated as follows:

$$\pi_*: \mathcal{S}\times\mathcal{A} \to [0,1], (s,a) \to \pi_*(s,a) = 0 \text{ if } a \notin \Sigma(s) \tag{17}$$

$$\Sigma(s) = \arg\max_{a\in\mathcal{A}} \sum_{s',r} \mathcal{P}(s',r|s,a)[r + \gamma \backslash v_*(s') = \arg\max_{a\in\mathcal{A}} q_*(s,a) \tag{18}$$

The key elements for devising an optimal policy are v_* and q_*.

All value-based RL algorithms employs the framework of generalized policy iteration (GPI) to ascertain v_* (or q_*) and π_*. This technique intertwines two key processes, (1) the policy improvement, where the policy is iteratively refined based on the current value/action-value function:

$$\pi'(s) = \arg\max_a q_\pi(s,a) \forall s \Rightarrow v_\pi(s) \le q_\pi(s,\pi'(s))\ \forall s \Rightarrow v_\pi(s) \le v_{\pi'}(s) \forall s \Rightarrow \pi \le \pi' \tag{19}$$

And (2) the policy evaluation, where the approximated value/action-value function incrementally aligns with its true value, whether wholly (in dynamic programming case) or in part (Monte-Carlo and TD(0)), for the designated policy. Upon convergence of these evaluation and improvement phases, both the value/action-value function and the policy are deemed optimal.

In a value-based RL framework, the approximation methods for v_* (or q_*) can be classified into three primary categories, each with its unique frequency of updates in GPI. These three categories are:

- Dynamic programming,
- Monte Carlo methods,
- Temporal-difference (TD) learning.

Dynamic programming approach is limited due to necessity of having the model of the environment which is usually not the case. Dynamic programming algorithm is shown in Algorithm 1. Within dynamic programming two algorithms are essential:

- Value iteration,
- Policy iteration.

Algorithm 1 Value iteration

1: **Parameter:** a small threshold $\theta > 0$
2: **Initialize:** $v(s)\ \forall s \in \mathcal{S}^+$ arbitrarily except $V(\text{terminal}) = 0$
3: **repeat**
4: $\quad\Delta \leftarrow 0$
5: $\quad$**for** each $s \in \mathcal{S}$ **do**
6: $\quad\quad v \leftarrow V(s)$

7: $V(s) \leftarrow \max_a \sum_{s',r} \mathcal{P}(s', r|s, a)[r + \gamma V(s')]$

8: $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

9: **end for**

10: **until** $\Delta < \theta$

11: **Output:** deterministic policy π such that:

$\pi(s) = \arg\max_a \sum_{s',r} \mathcal{P}(s', r|s, a)[r + \gamma V(s')]$

Monte Carlo MC is model-free algorithm; it prioritizes the approximation of action-value function Q over the state-value function V during policy evaluation. The policy evaluation does not entail rigorous approximation of Q like dynamic programming does for V. For policy improvement, an ε-greedy strategy is employed to ensure comprehensive exploration of states throughout environmental interaction. One of the important algorithms, GLIE Monte-Carlo Control (GLIE MC) (Algorithm 2) samples episodes and updates the visitation count $N(s_t, a_t)$ for each state–action pair and adjusts the action-value function Q based on the incremental returns G_t; it then progressively refines the policy using an ε-greedy approach that becomes less exploratory as the number of encounters increases. As more episodes are processed, the estimated Q converges to the optimal action-value function Q_* under the optimal policy π_*.

Various components of HVAC system leads to huge state and action spaces, which represent a challenge for storing action-value for every state in conventional lookup table. Also, ensuring that policy explores every state sufficient number of times require large number of episodes. To address this problem, function approximations can be employed. Types of function approximations are linear approximators, which are rarely used, and nonlinear approximators like neural networks (NN). NN is trying to estimate action-value function for each state-action pair, denoted as $\hat{Q}(s, a; w)$, where NN is parametrized by w. Those weights are adjusted vie stochastic gradient descent (SGA) to minimize a designed loss function L:

$$L(w) = \mathbb{E}_\pi\left[\left(G_t - \hat{Q}(s, a; w)\right)^2\right] \tag{20}$$

$$\Delta w = \frac{1}{2}\alpha \nabla_w L(w) = \alpha\left(G_t - \hat{Q}(s, a; w)\right)\nabla_w \hat{Q}(s, a; w) \tag{21}$$

Algorithm 2 GLIE MC

1: Sample k^{th} episode using $\pi: \{s_1, a_1, r_2, \ldots, s_T\} \sim \pi$

2: **Initialize:** $v(s)\ \forall s \in \mathcal{S}^+$ arbitrarily except $V(\text{terminal}) = 0$

3: **for** each $s \in \mathcal{S}$ **do**

4: $N(s_t, a_t) \leftarrow N(s_t, a_t) + 1$

5: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{1}{N(s_t, a_t)}\left(G_t - Q(s_t, a_t)\right)$

6: **end for**

7: Improve policy based on new action-value function:

$\varepsilon \leftarrow \frac{1}{k}$

$\pi \leftarrow \text{greedy}(Q)$

Temporal-difference, TD(0) is a model free algorithm. Similar to Monte Carlo methods, it focuses on approximating the action-value function Q, and it uses ε − greedy tactic for policy improvement. TD methods does not wait until the end of the episode to adjust the policy, this approach operates on a one-step basis for evaluation and improvement. Temporal difference methods use $r + \gamma Q(s', a')$ as a target, opposite to G_t in MC methods. Two most popular algorithms are:

- State, action, reward, next state, next action (SARSA),
- Q-learning.

SARSA is on-policy algorithm, which means that the agent is choosing next action a' according to the same exploration-exploitation tradeoff the agent is using (*e.g.* ε-greedy), the policy is improving gradually as agent explores the environment, which benefits with safe learning and simplicity as well as easy implementation, but convergence is slow and it is policy-dependent.

Q-learning is off-policy algorithm, which means that the agent is choosing next action a' according to the some other policy, called behavioral policy μ. In Q-Learning, the algorithm updates the Q-value based on the best possible action at the next state even if the agent did not take that action. It is exploration independent, which implies that Q-learning is less sensitive to exploration-exploitation tradeoff. One disadvantage of this algorithm is exploration challenge, improper exploration strategies can lead to suboptimal results, and the other is instability with function approximation. Ref. [2] employs a DQN algorithm to regulate chilled water temperature setpoints, aiming to minimize energy consumption while preserving thermal comfort in a single zone. Consequently, this approach exhibited improved performance in energy efficiency and control efficacy, surpassing rule-based control and traditional RL, and closely matching the performance of MPC.

Algorithm 3 SARSA

Initialize: $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily, and $Q(\text{terminal-state},\cdot) = 0$
for each episode **do**
    Initialize s
    Chose a from s using policy derived from $Q(e.g.,\varepsilon\text{-greedy})$
    repeat for each step of episode
        Take action a, observe r, s'
        Chose a' from s' using policy derived form $Q(e.g.,\varepsilon\text{-greedy})$
        $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$
        $s \leftarrow s'; a \leftarrow a';$
    until s is terminal
end for

4.2 Policy Based

In this methodology, policy is optimized directly to ensure reward improvement and training stability, it bypasses the need of value-function estimation. In this approach policy is parametrized with parameter θ, and generally is a NN with weights θ. Here the objective is to train neural policy π_θ, with data gathered from environment, where parameter θ is update through stochastic gradient ascent (SGA) to find the maximum of objective function J:

$$J(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) r_s^a \tag{22}$$

$$\Delta\theta = \alpha \nabla_\theta J(\theta) = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t \tag{23}$$

Where d^{π_θ} is stationary distribution of Markov chain for π_θ, and v_t is G_t for episodic cases or r_t for step updates.

Algorithm 4 REINFORCE

Initialize: policy parameters θ arbitrarily
for each episode $\{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
    for $t = 1$ to $T - 1$ **do**
        $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$
    end for
end for
return θ

4.3 Actor-Critic

The Actor-Critic method is a popular reinforcement learning (RL) algorithm that combines elements of both policy-based and value-based approaches. In the context of HVAC systems, this method can be used to optimize the control of heating, ventilation, and air conditioning by learning both how to act (through a policy) and how to evaluate those actions (through a value function). It is particularly useful for complex, continuous control problems like HVAC, where there may be many states and actions, and efficient exploration is required. This method consists of two components:

- Actor: actor decides what action to take in a given state. It is responsible for the policy $\pi(s,a)$, which maps states to the probability distribution over actions. In continuous environments, the policy can directly output the action (such as adjusting the temperature setpoint or fan speed in HVAC systems).
- Critic: The critic estimates the value of the state (or state-action pair) by evaluating how good it is to be in a particular state, following the current policy. This is done through a value function $V(s)$ or an action-value function $Q(s,a)$. The critic provides feedback to the actor on how good its actions are and helps guide policy updates.

The actor uses the critic's feedback to improve the policy over time, and the critic evaluates the policy to ensure it improves based on the rewards received from the environment.

Actor-Critic directly parametrizes policy, employing the approximate gradient theorem (24)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\hat{Q}(s,a;w)\right] \tag{24}$$

This strategy seeks the optimal policy by training the neural policy, π_θ, and the critic network $\hat{Q}(\cdot,\cdot,w)$ using data collected by the interaction with the environment. Parameter optimization are implemented in two phases:

- Parameters θ are fine-tuned with SGA to increase objective function J,
- Parameters w are tuned via SGD to decrease loss function L.

$$\Delta\theta = \alpha\nabla_\theta \log \pi_\theta(s,a)\,\hat{Q}(s,a;w) \tag{25}$$

$$\Delta w = \alpha\left(r + \gamma\,\hat{Q}(s',a';w) - \hat{Q}(s,a;w)\right)\nabla_w\hat{Q}(s,a;w) \tag{26}$$

4.4 Overview of implemented algorithms

Figure 3 show that the current trend in literature is the adoption of value-based approaches. The most popular algorithm is DQN, which implies of the need for algorithms that can deal with large or continuous state spaces. Actor-Critic algorithm are also widely used, due to need to control continuous state/action spaces. The leading algorithms in this method are deep deterministic policy gradient (DDPG) and soft actor critic (SAC). The policy-based algorithm are not in use in current literature. It is important to note that model-free algorithms are widely in use (87.75%) in contrast to model-based algorithms (12.25%). This is due to large state/action spaces, and exogenous state parameters that model environment.

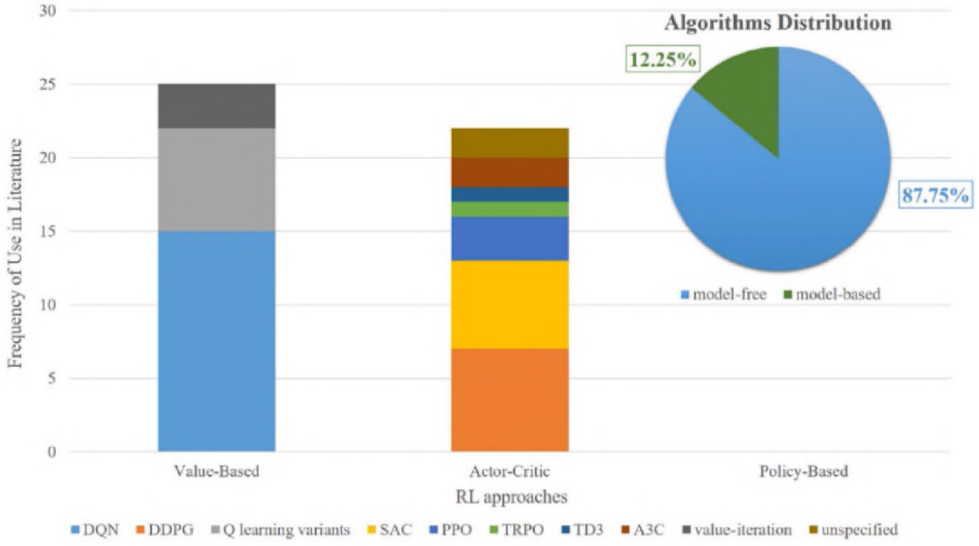

Figure 3. Usage of algorithm in current literature for HVAC control.

5. Conclusion

This paper shows basic concepts of reinforcement learning and presents the key challenges on the way of implementing RL algorithms to the real world problems. It gives a brief overview of current algorithm implementation in HVAC systems. Incorporating RL controllers into HVAC systems has proven to significantly enhance energy efficiency and cost reduction, outperforming traditional control methods. Particularly, RL controllers exhibit a reduction in energy usage by 27%–30% compared to rule-based controllers, while still maintaining optimal thermal comfort levels [2]. They also offer considerable cost savings, up to 39.6% over standard controllers, while preserving thermal comfort [3]. Alongside heating energy savings ranging from 5%–12%, with the added benefit of improved interior climate control [4]. Additionally, these controllers achieve 8% energy savings relative to rule-based algorithms, ensuring comfort for both indoor environments and domestic hot water [5]. Furthermore, RL controllers are effective in reducing electricity costs and peak energy demands by 23% in comparison to manually designed rule-based systems, without compromising on thermal comfort [6]. Theoretically, agents trained with (DQN, QPG, SAC, *etc.*) algorithms may not perform well during real deployment due to the change in the environment model to SNSMDP. The only solution to fix the disturbance in the agent's performance is to undergo periodic extensive retraining, which is computationally expensive. In future works, the focus should be on addressing the high computational cost associated with retraining the RL agent before it is deployed in an actual building for HVAC system control, which is modeled as an SNSMDP. In this context, the use of a meta-RL technique is proposed. This approach has been theoretically proven to facilitate rapid adaptation to the disturbances generated by the nature of SNSMDP during real-world deployment. Meta-RL can improve adaptability and reduce the computational cost associated with retraining. Future research should focus on applying Meta-RL to HVAC system control.

Acknowledgements

The results shown here are the result of research supported by the Ministry of Science, Technological Development and Innovation of the RS under the Agreement on financing the scientific research work of teaching staff at accredited higher education institutions in 2024, no. 451-03-65/2024-03/200105 of February 5, 2024.

References

[1] K. He, Q. Fu, Y. Lu, Y. Wang, J. Luo, H. Wu, J. Chen, Predictive control optimization of chiller plants based on deep reinforcement learning, J. Build. Eng. (ISSN: 2352-7102) 76 (2023) 107158
[2] Z. Zou, X. Yu, S. Ergan, Towards optimal control of air handling units using deep reinforcement learning and recurrent neural network, Build. Environ. 168 (2020) 106535.
[3] S. Touzani, A.K. Prakash, Z. Wang, S. Agarwal, M. Pritoni, M. Kiran, R. Brown, J. Granderson, Controlling distributed energy resources via deep reinforcement learning for load flexibility and energy efficiency, Appl. Energy 304 (2021) 117733.
[4] S. Brandi, M.S. Piscitelli, M. Martellacci, A. Capozzoli, Deep reinforcement learning to optimise indoor temperature control and heating energy consumption in buildings, Energy Build. 224 (2020) 110225.
[5] P. Lissa, C. Deane, M. Schukat, F. Seri, M. Keane, E. Barrett, Deep reinforcement learning for home energy management system control, Energy AI 3 (2021) 100043.
[6] G. Pinto, D. Deltetto, A. Capozzoli, Data-driven district energy management with surrogate models and deep reinforcement learning, Appl. Energy 304 (2021) 117642.
[7] S. Touzani, A.K. Prakash, Z. Wang, S. Agarwal, M. Pritoni, M. Kiran, R. Brown, J. Granderson, Controlling distributed energy resources via deep reinforcement learning for load flexibility and energy efficiency, Appl. Energy 304 (2021) 117733.
[8] L. Yu, W. Xie, D. Xie, Y. Zou, D. Zhang, Z. Sun, L. Zhang, Y. Zhang, T. Jiang, Deep reinforcement learning for smart home energy management, IEEE Internet Things J. 7 (4) (2020) 2751–2762, http://dx.doi.org/10.1109/JIOT.2019.2957289.
[9] D. Silver. Lecture 2: Markov decision processes. UCL. Retrieved from www0. cs. ucl. ac. uk/staff/d. silver/web/Teaching files/MDP. pdf, 2015.