Journal of Computers Vol. 35 No. 2, April 2024, pp. 1-16
doi: 10.53106/199115992024043502001

Application of Bayesian Networks and Reinforcement Learning in Intelligent Control Systems in Uncertain Environments

Liefeng Zhu[1], Yongbiao Luo[2*]

[1] School of Mechanical and Electrical Engineering, Shaoxing University, Shaoxing, Zhejiang, 312000, China
923916204@qq.com

[2] College of Yuanpei, Shaoxing University, Shaoxing, Zhejiang, 31200, China
81261161@qq.com

Received 21 June 2023; Revised 9 October 2023; Accepted 11 December 2023

Abstract. Reinforcement learning is a machine learning paradigm that focuses on how an agent can perform actions in an environment to achieve a certain goal. The agent learns through interaction with the environment, observing the state and making decisions to maximize its reward. Reinforcement learning has wide applications in intelligent control systems. However, one limitation of reinforcement learning is the uncertainty in handling the environment model. Usually, reinforcement learning is performed without a clear model, which requires estimating environmental uncertainty and state transitions. Bayesian Networks are effective in modeling uncertainty, which can aid in establishing a probabilistic model of environmental dynamics. This allows for the integration of uncertainty information into the environmental model, leading to a more accurate understanding of the dynamic characteristics of the environment. In this study, we propose a reinforcement learning algorithm based on Bayesian Networks. We utilize optimal generalized residual differentiation, parallel integration causal directional reasoning, and other modeling techniques to address reinforcement learning tasks. The main idea is to utilize the prior distribution to estimate the uncertainty of unknown parameters. Then, the obtained observation information is used to calculate the posterior distribution in order to acquire knowledge. Experiments demonstrate that this approach is feasible in intelligent control systems operating in uncertain environments.

Keywords: bayesian network, reinforcement learning, intelligent control systems, uncertain environments

1 Introduction

Reinforcement learning methods are widely used in various fields, including autonomous driving, gaming, robot control, financial trading, resource management, healthcare, network optimization, intelligent recommendation systems, industrial automation, natural language processing, the Internet of Things, and smart homes. Reinforcement learning aims to learn feedback signals from the environment through interactions with the environment [1-3]. Reinforcement learning methods can be divided into value-based methods, policy-based methods, and Actor-Critic (AC) methods, which combine the two [4]. When the environment model is deterministic, reinforcement learning methods demonstrate good performance and applicability in the aforementioned application areas. Uncertainty refers to situations where information is incomplete or results cannot be accurately predicted. Uncertain intelligent control involves the processing and management of uncertainty in control systems. In the field of autonomous driving, the intelligent control system of a vehicle must be capable of making decisions under constantly changing road conditions, traffic conditions, and environmental factors. In the field of smart homes and the Internet of Things, intelligent control systems need to adapt to user behavior patterns, environmental changes, and equipment failures. When confronted with an uncertain environment, intelligent control systems that integrate deep learning into reinforcement learning demonstrate their effectiveness and feasibility, as long as computational resources can meet the necessary demands. However, for intelligent control systems with limited computational power, deep reinforcement learning suffers from low sample utilization and slow learning speed. This directly leads to a reduction in computational output efficiency when dealing with uncertain environments. Therefore, when intelligently controlling an uncertain environment, it is necessary to have a method that can reduce the dimensionality of the state representation space in reinforcement learning. This method should achieve algorithmic effectiveness while satisfying computational resource constraints.

* Corresponding Author

At present, common reinforcement learning algorithms can be divided into the following three types: value-based algorithms, policy-based algorithms, and Actor-Critic framework-based algorithms. Value-based algorithms and policy-based algorithms are commonly used to calculate the optimal policy and determine the behavior that maximizes the reward value in different states. The most common value-based algorithm is the Q-learning algorithm. It obtains the current state of the environment, selects the reward value that can be obtained by taking the corresponding action, and gradually builds a Q-table (Q(s, a)) to update. The current Q value and the specific process are shown in Fig. 1:

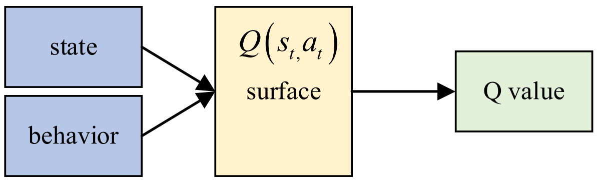

Fig. 1. Flow chart of Q-learning algorithm

Compared to other value-based algorithms, the Q-learning algorithm has the advantage of using the time difference method for offline learning [5]. Additionally, it utilizes the Bellman equation (Bellman) can be used to solve the state value function $V^*(s)$ of the current states. Finally, the cumulative expectation is used to obtain the result $V_\pi(s)$. The specific calculation methods are as follows,

$$V^*(s) = \max_\pi V_\pi(s)\ . \tag{1}$$

$$V_\pi(s) = E\left[\sum_{t=0}^{H} r_t R\left(s_t, a_t, s_t'\right) \mid \pi, s_0 = s\right]. \tag{2}$$

At the same time, the state of the Q-learning algorithm is discontinuous in the action space. Therefore, when the dimension is low, the Q-Table table can easily establish the correspondence between any state value and action value. However, when the action space has high dimensional continuous states, the Q-Table method does not work effectively. The neural network structure proposed by the DQN algorithm is shown in Fig. 2:

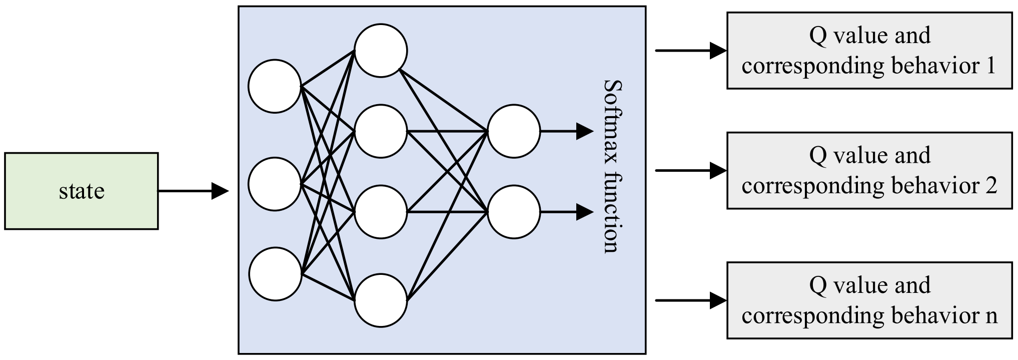
state
Softmax function
Q value and corresponding behavior 1
Q value and corresponding behavior 2
Q value and corresponding behavior n

Fig. 2. DQN structure

The DQN algorithm [6] is similar to the Q-learning algorithm in that it employs different strategies for updating the action value and selecting the action. This method is generally referred to as the off-policy feature [7]. It is easy for the parameters to fail to converge. Based on this problem, the improvement of the DQN algorithm involves utilizing a neural network to approximate the value function for behavior. Additionally, the target value network and the experience replay unit are used to update the target Q function [8, 9]. The specific algorithm flowchart is shown in Fig. 3:

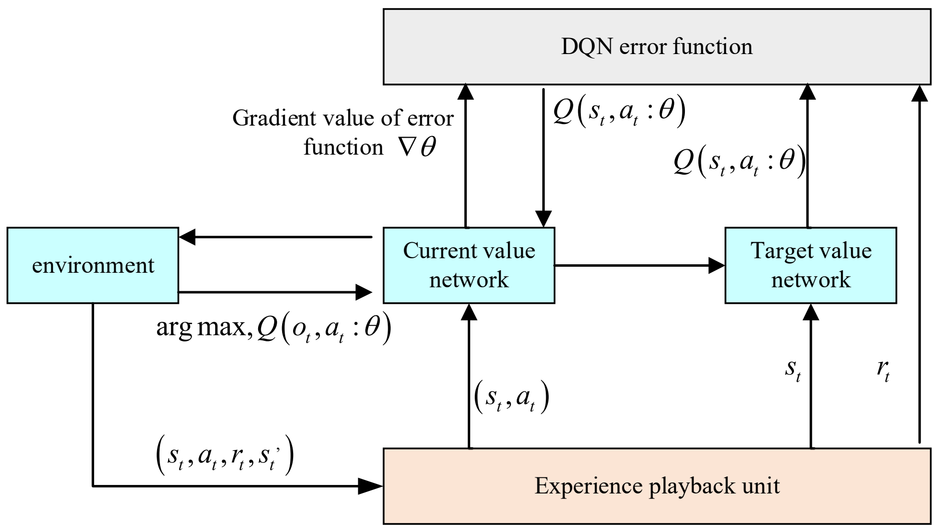

Fig. 3. DQN algorithm flow chart

Although the policy-based method has certain advantages in terms of convergence efficiency and the discovery of stochastic policies, it also has the ability to process high-dimensional action space and continuous action space. However, the neural network formed by this method is prone to being limited to local optima and has a large variance in the evaluation strategy. In recent years, with the increasing computing power of computer computing power, it has gradually become less prominent in the field of study [10]. The reinforcement learning algorithm based on the Actor-Critic framework has become the most popular approach in recent years. The algorithm evolved from the control and behavior training of the intelligent robot [11]. The essence of it is to use the neural network as the value function estimator. The structure is shown in Fig. 4:

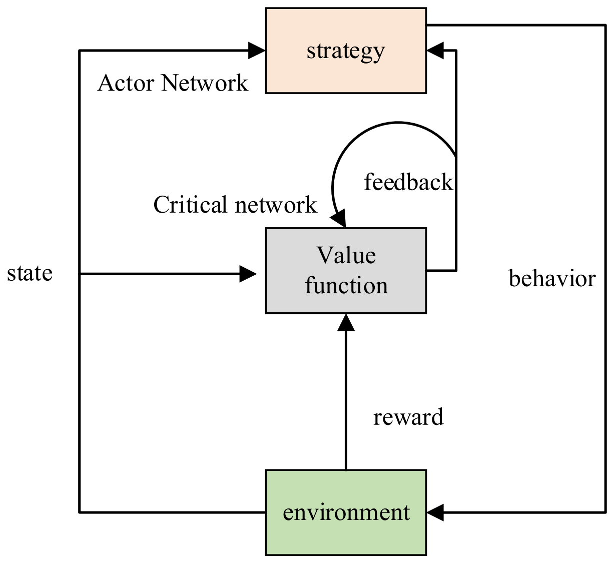

Fig. 4. Basic structure of Actor critical framework

As can be seen from the above figure, the main components of the Actor-Critic framework are the Actor network and the Critic network [12, 13]. Actor networks are developed from gradient policies, which address the inefficiency of Q-learning in selecting action values by determining the most appropriate action based on successive action values. The critic is the predecessor of the equivalent learning algorithm to Q-learning. The network is used for efficient single-step updates. Compared to the traditional gradient strategy algorithm's turn-based update strategy, this approach can effectively enhance the algorithm's learning efficiency.

This paper presents research on intelligent control systems in uncertain environments and highlights several research achievements. By combining Bayesian Networks (BN) with the Actor-Critic framework, the paper demonstrates the ability to provide accurate uncertainty estimates and optimize policies efficiently for intelligent decision-making. This integration can leverage the probability inference and uncertainty information provided by BN as one of the inputs to the Actor-Critic framework, allowing the agent to develop a more comprehensive understanding of the environment and make informed decisions.

The research content mainly consists of four parts. The first part provides an overview of the research status of Reinforcement Learning in Intelligent Control Systems both domestically and internationally. In the second part, an Uncertain Environments decision-making method based on a Bayesian Network model is proposed. In the third part, we propose a region enhancement method for intelligent control systems in uncertain environments. This method is based on Bayesian Networks and Reinforcement Learning. In the fourth part, the method proposed in the study is tested and analyzed. The results show that the method based on Bayesian Networks and Reinforcement Learning has a positive impact on intelligent control systems in uncertain environments.

2 Related Work

2.1 Bayesian Network Model

BN [14], also known as a credibility network, is an uncertain probability inference model based on the Bayesian rule proposed by the founder of the causal inference method in 1988. It is currently the most effective theoretical model for solving problems of uncertainty. The probabilistic reasoning process of the network relies on a rigorous mathematical theory - the Bayesian formula. This formula, also known as the posterior probability formula, is a mathematical method used to solve conditional probability. This method assumes that event A has m states and event B has n states. A_j and B_i represent the states of event A and event B, respectively. Currently, when an event A_j occurs, the probability of event B_i occurring is,

$$P\left(B_i | A_j\right) = \frac{P\left(B_i\right)P\left(A_j | B_i\right)}{P\left(A_j\right)} = \frac{P\left(B_i\right)P\left(A_j | B_i\right)}{\sum^{m}{}_{j=i}\left(B_i\right)P\left(A_j | B_i\right)}. \tag{3}$$

Among them, $P(B_i)$ is referred to as the prior probability of event B_i, $P(A_j)$ is the prior probability of the event, and the conditional probability $P(B_i | A_j)$ obtained through the Bayesian formula is known as the posterior probability [15]. The BN model is a directed acyclic graph, which is a type of probabilistic graphical model that intuitively describes the causal relationships between random variables. It represents these relationships in the form of nodes and conducts probabilistic inference by calculating the conditional probability of each variable. The specific form is as follows,

$$B = \left\langle G, P \right\rangle. \tag{4}$$

Among them, G represents a directed acyclic graph, which consists of nodes and unidirectional arcs connecting each node. This can be represented by the following two-tuple,

$$G = \left\langle V, E \right\rangle. \tag{5}$$

In the given context, V represents the set of nodes that includes all nodes in Bayesian Network (BN), which are used to represent the variables X_1, X_2, . . . , X_N in the universe of discourse. V represents the set of directed arcs, and its unit variable is the directed $e = (X_1, X_2)$, where $X_1, X_2 \in V$. The ordered pair e describes the causal relationship or conditional probability relationship between variables X_1 and X_2. The network parameter P consists of the probability distribution of all nodes in G, which describes the conditional probability of each node given its parent node. Each node corresponds to a conditional probability table, which can be expressed in the form of $P(X_m \mid P_A(X_m))$, where $P_A(X_m)$ represents the set of parent nodes for variable X_m . When the network structure and conditional probability table of a BN are determined, the construction of the BN is completed. Subsequently, the network can be used for probabilistic reasoning. The process of BN probabilistic reasoning is to estimate the probability distribution of the variable set to be calculated in the current state, given the evidence variable set $E = e$. The calculation process is as follows,

$$p\left(X_i = q \mid E = e\right) = \frac{p\left(X_i = q \mid E = e\right)}{p(E = e)}, i = 1, \cdots, n . \tag{6}$$

2.2 Multi-agent Reinforcement Learning Algorithm

The Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm [16] is an extension of the deterministic policy gradient algorithm in a multi-agent environment. Due to the non-stationarity of the environment, MADDPG adopts a centralized critic and decentralized execution approach for learning and training. The algorithm utilizes a centralized critic network within the AC framework, known as a centralized training-decentralized execution framework. The specific network framework is shown in Fig. 5:

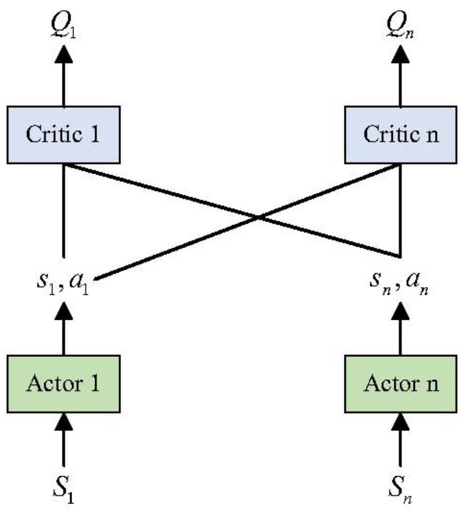

Fig. 5. MADDPG network structure

As can be seen from Fig. 5, each agent maintains a local centralized value network. This network receives the observations and actions of all agents as a joint state-action pair for learning. It can be considered as a centralized brain coordinating all actions of the agent [17]. At the actor, each agent seeks a policy only through its own local observation state. Like the single-agent deterministic policy gradient, the multi-agent policy gradient can also be directly derived using the chain rule. Its gradient is given by,

$$\nabla J\left(\theta_i\right) = E_{s,aD}\left[\nabla_{a_i} Q_i\left(s, a_1, \cdots, a_i, \cdots, a_n\right) \nabla_{\theta_i} u_{\theta_i}\left(s_i\right) \mid_{a_i = u_{\theta_i(s_i)}}\right] . \tag{7}$$

Among them, $\theta_i = [\theta_1, \theta_2, ..., \theta_n]$ are parameters of n agent policies, $s_i = [s_1, s_2, ..., s_n]$ represents the observation vector represents the state., and $Q_i = (s, a_1, ..., a_i, ..., a_n)$ represents the agent centralized state action function. It can be observed from the policy gradient that even though the local policies of all agents are decentralized and executed independently, the gradient boosting direction of the agents' local policies aims to optimize the global goal. This effectively mitigates the non-stationary nature of the environment. This is also the most significant distinction between the centralized training-decentralized execution framework of MADDPG and the single-agent reinforcement learning algorithm. The algorithm under this framework considers the actions of other agents during training and the environmental non-stationarity caused by the actions of other agents. However, it can also be weakened by this. Therefore, the gradient ascent algorithm can be directly used for learning the policy network. The value network of the MADDPG algorithm is essentially the same as that of other value-based algorithms, such as the DQN update method [18]. It utilizes a dual network structure for updating, specifically employing the following formula,

$$\begin{aligned} &J(\varphi) = E_{s,a,r,s' \text{ doneD}}\left[\left(Q_i^{\varphi}\left(s,a_1,\cdots,a_i,\cdots,a_n\right)-y_i\right)^2\right] \\ &y_i = r_i + \gamma \cdot (1- \text{ done }) \cdot Q_i^{\varphi'}\left(s',a_1',\cdots,a_n'\right) \\ &a_j' = u_j'\left(s_j'\right) \end{aligned} \quad . \tag{8}$$

These target networks are similar to the target networks used in other algorithms. They are not updated using gradients, but instead, a smoother moving average method is employed, rather than the periodic update method used in DQN. The MADDPG algorithm [19] is a significant milestone in multi-agent reinforcement learning algorithms. The centralized training-decentralized execution framework proposed by MADDPG greatly improves the effectiveness of the original single-agent reinforcement learning algorithm [20]. This framework has also emerged as the leading multi-agent reinforcement learning algorithm. A Paradigm for Agent Reinforcement Learning Algorithms. However, this framework has the disadvantage of not being able to scale with the number of agents. Since the strategy of the centralized network in this method is determined by the current number of agent strategies, when the number of agents in the environment changes, the action value function learned by the strategy network can no longer accurately represent the joint behavior after the environment changes. The value function of the strategy, therefore, its decentralized execution strategy, cannot be used directly in most cases [21]. In a multi-agent cooperative environment, there is a problem of distributing beliefs among agents. The problem of reliability assignment can be described as follows: in a collaborative environment, the reward received by an agent at each moment is based on the joint action, and the agent cannot accurately assess the impact of its current action on the reward provided by the environment. Therefore, determining the significance of the actions taken by the agent in relation to the joint reward provided by the environment is also a prominent area of research. The existing algorithm primarily utilizes the concept of the COMA algorithm. It calculates a baseline state-action value for the agent's action and determines the action's contribution through the advantage function [22].

3 Methodology

3.1 Bayesian Network Structure Generation Based on Reinforcement Learning

Bayesian network learning is first transformed into its corresponding directed graph learning problem, and the directed graph is represented by a symmetric graph adjacency matrix [23]. In terms of implementation ideas, this section utilizes policy gradients and stochastic optimization methods to train the weights of the neural network. The final output is the graph that achieves the highest return among all the graphs generated during the training process, specifically, the directed Bayesian structure. In the specific operation process, the data is first input into the Actor network. The matrix is then converted into the next transformation behavior by re-encoding in the Actor network. After that, it is input into the Critical network and the decoder. The decoder establishes a relationship between variables and generates a graph adjacency matrix. It then passes the matrix into the scoring function to calculate the score of the behavior. The score is then fed back to the Critic network, which predicts the next behavior based on the behavior and score [24]. The specific process is shown in Fig. 6 below:

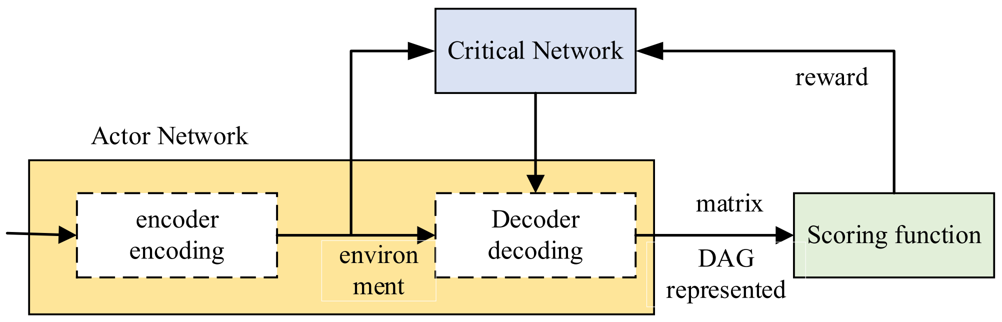

Fig. 6. Actor critical reinforcement learning framework for causal discovery of functions

As shown in Fig. 6, to apply reinforcement learning to causal discovery, we utilize an encoder-decoder neural network model. This model generates directed graphs based on observed data. These directed graphs are subsequently used to compute a reward, which includes a penalty term to ensure acyclicity. We believe that using it with causal independence constraints will be more beneficial for mining causal relationships between variables. The decoder network generates the graph adjacency matrix element-wise by establishing the relationship between two encoders for outputting semantic and network features. First, a single-layer decoder can be represented by the following function,

$$g_{ij}\left(W_1, W_2, u\right) = u^T \,\text{tan(h}\, W_1 enc_i + W_2 enc_j \big) . \quad \textbf{(9)}$$

Bayesian Information Criterion (BIC) scoring is used in this section because it is not only consistent throughout, but also exhibits specific characteristics in the decomposed local area. The BIC score for a given directed graph G is as follows,

$$S_{BIC}(g) = -2\log p(X;\hat{\theta}, g) + d_{\theta} \log_{m} . \quad \textbf{(10)}$$

In the formula, $\hat{\theta}$ is the maximum likelihood estimator, and d_θ represents the dimension of the parameter. This paper assumes independent and identically distributed (IID) Gaussian additive noise between the variables. Therefore, if we apply a linear model to each causal relationship and let be an estimate of *kix*, where *kix* represents the *i* entry in the kth observation sample, then the final BIC score is,

$$S_{BIC}(g) = \sum_{i=1}^{d} \left(m \log\left(GRSS_i / m \right)\right) + \#(\ edges\ ) \log m . \quad \textbf{(11)}$$

where ${}^{1}GRSS_i = \sum_{k=1}{}^{m} \sum_{l=1}{}^{n} \left(x_k^i - \hat{x}_k^i \right)^2 / ni$ represents the generalization residual sum of squares for the I-th variable is calculated using the generalization base of the residual sum of squares, and l represents the generalization group number being calculated. This calculation is done while using the log-likelihood objective in the generalization scoring method of the sample. During each training, the training set is divided into n parts, with each part randomly selecting (n-1)/n of the sample size from the original training set. The pair of #(edges) log m represents the penalty term for the number of edges in graph g. In the training process of real data, the reward value calculated by the scoring function is greatly affected by the insufficient sample size, abnormal data, and missing data in the sample set. The issue can be effectively resolved by expanding the sample set through the generalization idea. After completing the graph scoring, it is necessary to address the issue of loops in the Bayesian network, as loops are not allowed. Since the algorithm described in this paper aims to learn a directed graph, it is possible for a cycle to exist in the Bayesian network when three or more variables have a cyclic relationship. In such cases, we can determine the presence of a cycle by examining the variable relationships in the adjacency matrix. Therefore, when a loop occurs, the evaluation and termination of the loop is performed by establishing the causal relationship among the relevant variables [25]. The specific measures in this section are divided into two parts. First, after each edge addition behavior occurs, if it is determined that a loop appears in the network, the corresponding arc will be assigned a direction, and the acyclicity will be checked through the GES display. Second, to prevent the occurrence of the next cycle, a large penalty term is added to the case where the cycle is generated. This implicitly enforces the acyclicity of the scoring function and allows the generated graph to change by one at each iteration above the edge. This section of the content utilizes the research findings of et al., who argue that if and only if,

$$h(A) := \mathrm{trace}\left(e^{A}\right) - d = 0\,. \tag{12}$$

If only $h(A)$ were used, we would need very large penalty weights to ensure acyclicity. To this end, we add a second penalty term, the indicator function, which is acyclic to induce precise causality in local regions. Therefore, the final reward function includes two parts: the scoring function and the acyclic constraint reward. The specific function is expressed as follows,

$$\text{reward} := -\left[S(g) + \lambda_1 \mathrm{I}(g \notin \mathrm{DAGs}) + \lambda_2 h(A)\right]. \tag{13}$$

where $\mathrm{I}(g \notin \mathrm{DAGs})$ represents the indicator function, and $\lambda_1 \geq 0$, $\lambda_2 \geq 0$ are two non-negative penalty parameters. It can be seen from the above formula that the larger λ_1 and λ_2 are, the more likely it is for the generated graph with higher reward to be acyclic. To maximize returns, the equation above can be transformed equivalently into,

$$\frac{\min}{g}\left[S(g) + \lambda_1 \mathrm{I}(g \notin \mathrm{DAGs}) + \lambda_2 h(A)\right]. \tag{14}$$

On the initial penalty weight setting, we found that if it is set too large, the reward fed back to the Critic network has little effect on it, thus limiting the learning effect of the Actor-Critic framework. So, in practice. We use the update rule on the LaGrange multiplier adopted in NOTEARS to start the penalty weight from a small value, and then gradually increase it to obtain the optimal directed graph network as much as possible while ensuring the learning effect.

3.2 Bayesian Network Structure Optimization Based on Causal Direction Judgment

However, further processing is required since it may lead to the appearance of false edges during the reinforcement learning process. After obtaining the initial Bayesian network with the highest score, this section will perform a secondary optimization of its structure using the following methods. The specific process is shown in Fig. 7:

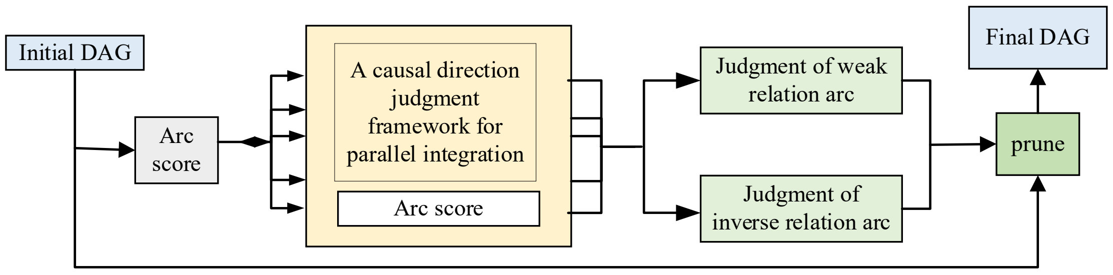
Initial DAG
Arc score
A causal direction judgment framework for parallel integration
Arc score
Judgment of weak relation arc
Judgment of inverse relation arc
prune
Final DAG

Fig. 7. Bayesian network structure optimization process

As shown in the Fig. 7 above, arc splitting and pruning are performed on the graph. In the arc processing step, each arc in the DAG needs to be split and input into the parallel integrated causal direction judgment framework for arc scoring. The orientation method of a single arc is the same as binary causal inference, which can be directly determined using the method described in Chapter 2 of this paper. The pruning process involves using a greedy method to remove estimated edges based on regression performance or a score function. Secondly, after completing the arc scoring operation, the number of weak relationship arcs and inverse relationship arcs to be deleted is determined based on the obtained score, in proportion to the total number of arcs. The weak relationship arcs and inverse relationship arcs are identified by considering the absolute value and minimum value of the score. Relationship Arc. Finally, the final Directed Acyclic Graph (DAG) is obtained by performing pruning based on the weak relationship arcs, inverse relationship arcs identified using the above method, and arcs with high penalty values determined through the greedy algorithm. Compared to the traditional pruning method based on regression performance or scoring function, this method has the following advantages. First, this method considers the Bayesian network from a different perspective, including the specific scoring function utilized in this section. It is a scoring method based on the BIC score. The essence of its thinking is to achieve the appropriate relationship between all nodes in a greedy manner. Theoretically, there is no problem with this method. However, the essence of the Bayesian network lies in the relationship between variables, which is not as rigid as a mathematical formula. The pruning method proposed in this section not only considers the fitting relationship between variables but also takes into account the transfer of information entropy between variables. It identifies the relationship with the least transfer of information entropy and the arc whose transfer direction of information entropy is opposite to the prediction. That is, the weak relation arc and the inverse relation arc. The concept of information entropy is analogous to the causal relationship between variables from the very beginning, and many methods for judging causal direction are derived. Therefore, the pruning method provides assistance in constructing Bayesian networks from multiple perspectives. In the specific process of pruning, it is necessary to ensure the integrity of the graph by preventing the generation of any new subgraphs after pruning. Therefore, after completing the entire process of optimizing the Bayesian network structure, it is still necessary to apply the graph traversal method. Determine if any unexpected subgraphs appear.

3.3 Multi-agent Reinforcement Learning Algorithm under Global Observation

In the reinforcement learning algorithm, the critic-actor algorithm framework necessitates the use of a policy network and an evaluation network. The policy network takes in the agent's state information and outputs the agent's current policy, while the evaluation network takes in the agent's state. Action pair, output the evaluation value of the agent's current action. When there are multiple agents in the environment, the centralized evaluation network needs to input the state and action pairs of all the agents, while the policy network needs to input the current observation information of all agents in the environment. Using the method of direct splicing, as described above, although this approach is simple, the neural network trained using this method can only be applied in an environment with a fixed number of agents.

Furthermore, the information of other agents may not be relevant to local agents. The aggregation method is useful, but it increases the convergence cost when the information from other agents has little impact on the optimization of the local agent's decision. The attention mechanism can be adapted for unstructured information fusion, making it highly suitable for merging observation information from each agent in a multi-agent environment. In the evaluation network and the policy network, the input information is different, which results in slight differences in the information fusion methods between the two modules, even when the attention mechanism is used to replace the traditional information fusion method simultaneously. In this paper, the policy network utilizes the attention mechanism's aggregation method. The local information o_i of the agent in the environment is represented by nodes in the time series. By considering all agents locally as the entire input sequence, the attention mechanism can efficiently aggregate environmental information. Specifically, each agent can use their own information o_i to aggregate the local information o_j of other agents, forming a new feature vector representation of agent information. This vector can then be used to represent the agent's observation information on the environment at every moment in the substitute environment. In the multi-agent environment, the attention mechanism is used as follows: at each moment, a weight value w_{ij} is calculated for other agents based on the local information of the agent. The calculation method is as follows,

$$w_{ij} = \frac{\exp\left(Q \cdot o_i \cdot \left(K \cdot o_j\right)^T / \kappa\right)}{\sum \exp\left(Q \cdot o_i \cdot \left(K \cdot o_j\right)^T / \kappa\right)}. \tag{15}$$

Among them, *Q* and *K* are the feature transformation matrix of weight sharing between different agents, which will perform feature transformation on the state information of the agent, and κ is a normalization factor to ensure that even in the case of multi-layer attention, There is no problem that the data is too large or too small after calculation. Finally, the weight value is transformed through SoftMax transformation, so that the sum of the obtained weight values of all agents is 1. After the attention weight value is obtained, the environmental information *i* aggregated by the agent *i* is obtained by weighting the weight value h_i ,

$$h_i = \sum w_{ij} v_j . \tag{16}$$

Where v_j is the feature representation vector of agent j's local information o_j after the feature transformation using the shared parameter transformation matrix *V* namely,

$$v_j = V \cdot o_j . \tag{17}$$

In the calculation of attention, this paper does not use the form of position encoding, because in the multi-agent collaborative environment, the considered agents are homogeneous agents, and it is hoped that the aggregated information of each agent does not follow the agent's change according to the input order. That is, the input order of the agent is expected to be ignored, and therefore, the positional encoding trick is not used for the agent's attention mechanism. The use of this attention mechanism not only changes the traditional method of integrating the information of other agents, but also can use the local information of the agent to filter the local information of other agents in the environment, so that the agent will be the focus of attention at the current moment. The information of other agents is assigned a larger attention value, and a smaller attention value is assigned to the agent that has little influence on the agent at the current moment, which can intelligently learn the attention relationship between the agents. Finally, the feature representation obtained after screening the local information of other agents in the environment through the attention mechanism is fused with the local information of the agent itself. This fusion process can be performed using a neural network. In this case, the final agent's environmental observation m_i at each moment is,

$$m_i = f\left(o_i, h_i\right). \tag{18}$$

In this way, agents in the environment can still retain the local information of other agents. Fig. 8 shows the fusion of feature information in the agent using the attention mechanism.

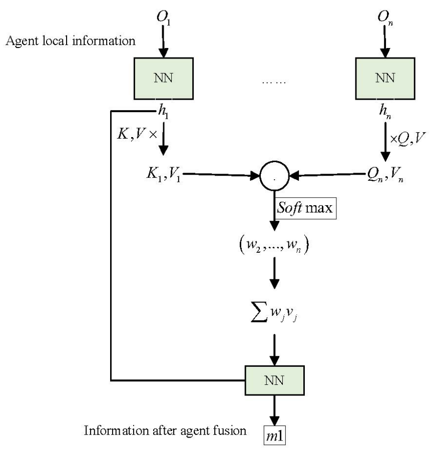

Fig. 8. Attention mechanism model

3.4 Multi-agent Intelligent Control System in Uncertain Environments

Bayesian reinforcement learning methods are used as optimization decision methods for multi-agent intelligent control systems facing uncertain environments. The unknown environmental parameters are modeled using a priori probability distribution, and the posterior distribution is solved using Bayesian technology and observed data to estimate the uncertainty of the parameter vector. Prior distributions can model environmental model parameters, value functions, strategies, and their gradients. They ensure that the Bayesian reinforcement learning method maintains a probability distribution and converges to the optimal strategy. The main steps are as follows:

Step 1: Build the Bayesian network structure among multiple agents, define variables and dependencies, identify the relevant variables for multi-agent problems, and infer the conditional probability relationships between them.

Step 2: Encode the state of the multi-agent Bayesian network as the state representation for reinforcement learning. Apply reinforcement learning to causal discovery using the encoder-decoder neural network model. This model generates a directed graph based on observation data and defines the action space to adjust the network structure.

Step 3: Take the Bayesian Information Criterion (BIC) as the reward function to evaluate the quality of the Bayesian network structure. Assume that there is independent and identically distributed (IID) Gaussian additive noise between variables. Apply a linear model to each causal relationship and prune the edges of the directed graph based on the regression performance or scoring function.

Step 4: Strengthen the training of the learning agent by incorporating the index function as a secondary penalty term. This will help prevent local optimization by utilizing scoring functions and acyclic constraint rewards. Specifically, calculate the penalty weight starting from a smaller value using the update rules of Lagrange multipliers mentioned in the notes, and then gradually increase the penalty weight. Under the premise of ensuring effective learning, strive to obtain the optimal directed graph network as much as possible.

Step 5: In this step, the cooperation and competition among agents are facilitated by the exchange of local information. The local information of agents in the environment corresponds to the nodes in the time series. To aggregate the environmental information and provide feedback for the decision-making process of agents, an attention mechanism is employed.

Step 6: The agent interacts with the environment, observes the status, takes actions, and continuously updates the structure or parameters of the Bayesian network according to the reward.

Step 7: Using the results of reinforcement learning, the conditional probability table of the Bayesian network is optimized to accurately describe the state transition and environmental uncertainty.

Step 8: Utilize the Bayesian network optimized by reinforcement learning as the decision-making strategy for multi-agents in an intelligent control system. This approach aims to strike a balance between the cooperation and competition among the multi-agents.

To summarize, by utilizing reinforcement learning to optimize Bayesian networks, each agent can achieve the global optimal solution when confronted with decisions in uncertain environments.

4 Experiments

4.1 Implementation Details

To prove the effectiveness of the Bayesian network model generation method based on reinforcement learning proposed in this paper, a CPU will be used in this chapter is Inter(R) Core (TM) i9-10900K 3.70 GHz, memory is 128 GB, operating system It is Windows 10 Enterprise Edition, the graphics card is GeForce RTX 3080, and the video memory is 10GB for the experiment. The compiler used is Spider, the language used is python3.6, and the neural network construction framework is built through TensorFlow version 1.13.1. Part of the code uses the code in the Trustworthy AI toolbox written by Huawei Noah's Ark Lab and the Causal Discovery Toolbox.

4.2 Bayesian Reinforcement Learning Search Experiment

To better verify the effect of the experiment, this paper uses a Bayesian network with 12 variables to change the source. The network has a totally of 2132 different DAG possibilities. It is almost impossible to obtain the DAG with the best score by traversing, and the reinforcement learning-based Bayesian network model generation method based on this paper also considers the addition of the variable's environment to it, and the result is better than the result obtained by directly scoring the DAG. At the same time, the DAG used in this paper uses a 12×12 upper triangular matrix as the binary adjacency matrix of the graph, the cause variable is sampled independently from the Bernoulli distribution of $N = 1$, and the edge weights use a uniform distribution U(-2,0.5) Assign the edge weight to U(0.5,2) to obtain the weight matrix $ddW \times \in R$, and finally obtain 5000 samples from the sampling model $Td\times = W\times + N \in R$ through the Gaussian noise model and the non-Gaussian noise generation method proposed by ICA-Lingam in the paper, and conduct 5 independent tests. The experiment obtains its maximum value, minimum value median, and upper and lower quartiles, and compares it with the traditional method and the non-pruning method. The results are shown in Fig. 9 and Fig. 10 below:

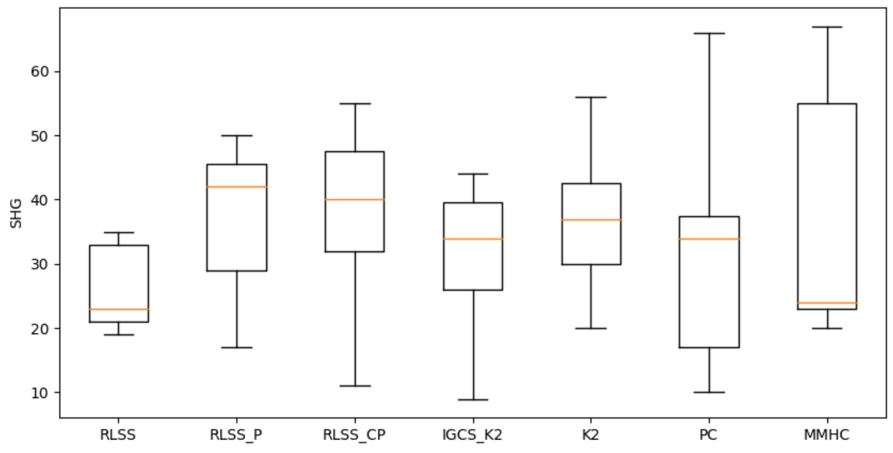

Fig. 9. SHD comparison between the method in this chapter and the traditional method under non-Gaussian noise

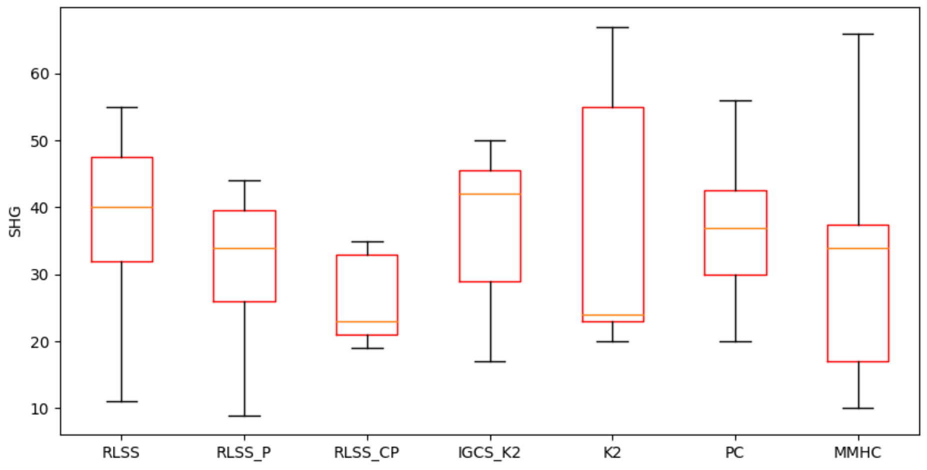

Fig. 10. SHD comparison between the method in this chapter and the traditional method under Gaussian noise

In the above figure, RLSS is a new method proposed in this paper to use reinforcement learning to complete Bayesian structure search. The RLSS_P method adds the traditional GES-based pruning method to this method and prunes it. The threshold is set to the default value of 0.3 described in the paper, and the RLSS_CP method is a Bayesian network generated by using the Bayesian structure optimization method mentioned in Section 4.3.2 based on this method to perform weak relationship arcs and inverses. The number of arcs for relational arc judgment is 1/20 of the total number of arcs. At the same time, the comparison methods used in the above experiment are all common methods in the field of Bayesian structure construction mentioned in Chapter 3, and the selection of specific parameters is the same as that in Chapter 3, while the traditional K2 score search strategy Because the order of input nodes is different from the maximum number of parent nodes, the final result is different, so in this experiment, the maximum number of parent nodes is set as the optimal value, and the final K2 score is obtained by averaging 100 times. strategy results. As can be seen from the above figure, the Bayesian structure obtained by the method proposed in this paper is closer to the real Bayesian network than the traditional Bayesian structure generation method, even if some traditional methods have obtained the optimal theoretically impossible. After hyperparameters, the structural accuracy is still far lower than the method proposed in this paper. However, we can also find that the method proposed in this paper has a large variance. Through observation, it can be found that the reason for this problem is that it falls into a local optimum under 20,000 algorithm traversals. The problem of rising can be alleviated, but the theoretical optimal solution cannot be obtained in a predictable number of times, which is also our follow-up research direction.

4.3 Application Analysis of Multi-Agent Reinforcement Learning Algorithm under Global Observation

The experiments compare the overall return convergence curves of the MADDPG algorithm, and the algorithm proposed in this paper during the training process.

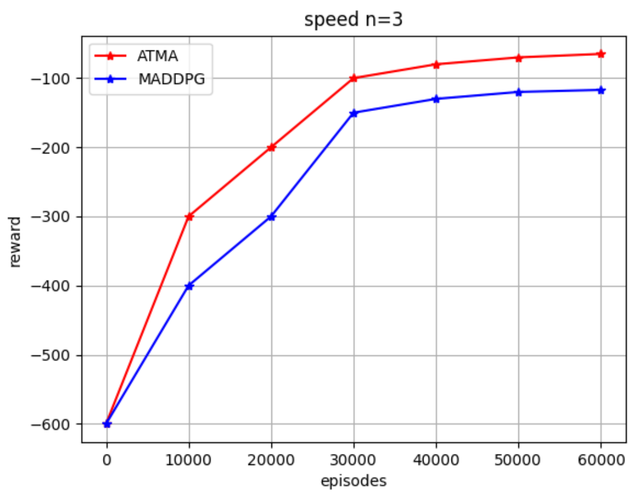

Fig. 11. Cumulative return in spread environment

The changes in cumulative return of the algorithm during training in the spread environment are shown in Fig. 11. It can be observed that in the distributed environment, when the number of agents is three, the algorithm proposed in this paper converges faster than the MADDPG algorithm. However, the MADDPG algorithm exhibits significant fluctuations in the initial stage and gradually increases the cumulative return of the algorithm until it reaches its peak. The algorithm in this paper converges to the maximum value at around 5000 episodes, and there may be some fluctuations during the training process. However, despite the large fluctuations in both algorithms, our method demonstrates superior convergence compared to MADDPG in this environment.

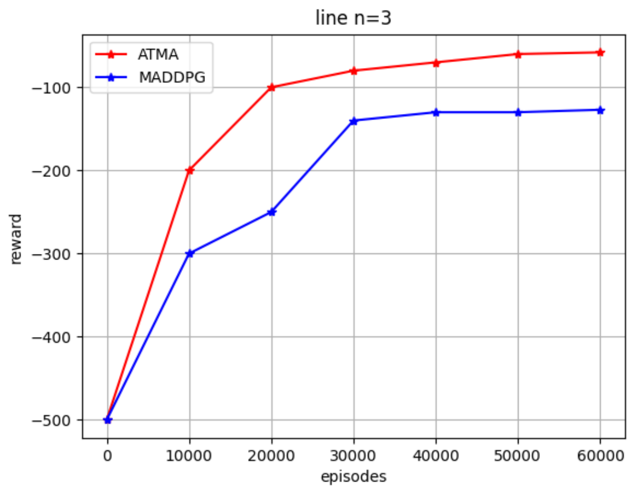

Fig. 12. Cumulative return in line environment

The performance of the algorithm in the linear environment is shown in Fig. 12. The convergence process of the two algorithms in the linear environment is the same as that in the spread environment. At the beginning, due to the random exploration of the environment for a period, the cumulative returns of both parties will exhibit lower range volatility. When the exploration of the environment is completed, the convergence speed of the two sides is not significantly different. However, the method proposed in this paper can achieve a relatively good result.

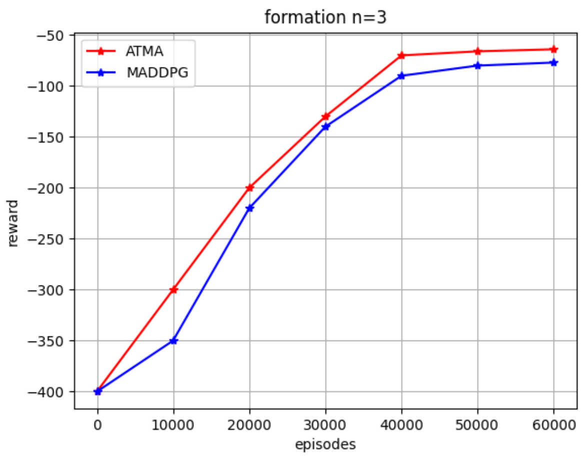

Fig. 13. Cumulative return of formation environment

Fig. 13 shows the convergence of the algorithm in the formation environment. As depicted in the figure, the convergence effect of the two algorithms in the formation environment is the closest. Both the MADDPG algorithm and the algorithm proposed in this paper can converge to the peak value in a shorter time step, and the convergence effect of both algorithms is similar. In this environment, the algorithm demonstrates more stable training compared to the first two environments. The training results of the algorithm in this environment exhibit less fluctuation.

5 Conclusion

5.1 Conclusion

This paper proposes an optimization method for generating Bayesian network models based on reinforcement learning. The method utilizes reinforcement learning to search for the optimal generalization residual score, thereby improving the efficiency of constructing the optimal Bayesian network structure. Moreover, we designed a parallel integrated Bayesian network and achieved higher precision performance compared to the basic structure. In addition, we apply the reinforcement learning model to research on multi-agent systems. We propose an ATMA algorithm in a global observation environment and demonstrate the utilization of the attention mechanism for information fusion within the AC framework. Finally, through simulation experiments, the ATMA algorithm has been found to have a faster convergence speed than the MADDPG algorithm in three typical multi-agent cooperative environments. Additionally, the ATMA algorithm is able to converge to a higher cumulative reward value compared to the MADDPG algorithm. The results show that the method based on Bayesian Networks and Reinforcement Learning has a positive impact on intelligent control systems in uncertain environments.

5.2 Prospect

Due to the limitations of their own knowledge and ability, this research still has some shortcomings, that need to be improved. The specific contents are as follows:

(1) Development of probabilistic models, further research and enhancement of Bayesian networks or other probabilistic models to more accurately capture uncertainty and dynamic characteristics within the environment.

(2) Research and application of efficient approximate inference methods to handle large-scale and complex Bayesian networks, thus enhancing learning efficiency and speed.

(3) Exploring the use of Bayesian neural networks to handle uncertainty and enhance the generalization capabilities of neural network models, integrating Bayesian concepts with deep learning.

References

[1] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, J. Qin, A survey on application of machine learning for Internet of Things, International Journal of Machine Learning and Cybernetics 9(8)(2018) 1399-1417.

[2] K.H. Abdulkareem, M.A. Mohammed, A. Salim, M. Arif, O. Geman, D. Gupta, A. Khanna, Realizing an effective COVID-19 diagnosis system based on machine learning and IOT in smart hospital environment, IEEE Internet of Things Journal 8(21)(2021) 15919-15928.

[3] G. Gao, J. Li, Y. Wen, DeepComfort: Energy-efficient thermal comfort control in buildings via reinforcement learning, IEEE Internet of Things Journal 7(9)(2020) 8472-8484.

[4] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, Z. Sun, A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways, IEEE Transactions on Systems, Man, and Cybernetics: Systems 50(10)(2018) 3884-3897.

[5] F. Tang, Y. Kawamoto, N. Kato, J. Liu, Future intelligent and secure vehicular network toward 6G: Machine-learning approaches, Proceedings of the IEEE 108(2)(2019) 292-307.

[6] A. Feriani, E. Hossain, Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial, IEEE Communications Surveys & Tutorials 23(2)(2021) 1226-1252.

[7] J.P. Usuga Cadavid, S. Lamouri, B. Grabot, R. Pellerin, A. Fortin, Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0, Journal of Intelligent Manufacturing 31(6)(2020) 1531-1558.

[8] E. Adi, A. Anwar, Z. Baig, S. Zeadally, Machine learning and data analytics for the IoT, Neural Computing and Applications 32(20)(2020) 16205-16233.

[9] M.I. Alipio, A.E.M.D. Cruz, J.D.A. Doria, R.M.S. Fruto, On the design of Nutrient Film Technique hydroponics farm for smart agriculture, Engineering in Agriculture, Environment and Food 12(3)(2019) 315-324.

[10] D.A. Hashimoto, E. Witkowski, L. Gao, O. Meireles, G. Rosman, Artificial intelligence in anesthesiology: current techniques, clinical applications, and limitations, Anesthesiology 132(2)(2020) 379-394.

[11] J. Wu, X.Y. Chen, H. Zhang, L.D. Xiong, H. Lei, S.H. Deng, Hyperparameter optimization for machine learning models based on Bayesian optimization, Journal of Electronic Science and Technology 17(1)(2019) 26-40.

[12] A.M.U.D. Khanday, S.T. Rabani, Q.R. Khan, N. Rouf, M. Mohi Ud Din, Machine learning based approaches for detecting COVID-19 using clinical text data, International Journal of Information Technology 12(3)(2020) 731-739.

[13] T.K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, N. Kato, Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective, IEEE Communications Surveys & Tutorials 22(1) (2019) 38-67.

[14] J. Xie, F.R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, Y. Liu, A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges, IEEE Communications Surveys & Tutorials 21(1)(2018) 393-430.

[15] S. Tanwar, J. Vora, S. Kaneriya, S. Tyagi, N. Kumar, V. Sharma, I. You, Human arthritis analysis in fog computing environment using Bayesian network classifier and thread protocol, IEEE Consumer Electronics Magazine 9(1)(2019) 88-94.

[16] L.J. Muhammad, E.A. Algehyne, S.S. Usman, A. Ahmad, C. Chakraborty, I.A. Mohammed, Supervised machine learning models for prediction of COVID-19 infection using epidemiology dataset, SN computer science 2(1)(2021) 1-13.

[17] Y. Sun, J. Cheng, G. Zhang, H. Xu, Mapless motion planning system for an autonomous underwater vehicle using policy gradient-based deep reinforcement learning, Journal of Intelligent & Robotic Systems 96(3)(2019) 591-601.

[18] K. Guo, Z. Yang, C.H. Yu, M.J. Buehler, Artificial intelligence and machine learning in design of mechanical materials, Materials Horizons 8(4)(2021) 1153-1172.

[19] N.I. Mowla, N.H. Tran, I. Doh, K. Chae, AFRL: Adaptive federated reinforcement learning for intelligent jamming defense in FANET, Journal of Communications and Networks 22(3)(2020) 244-258.

[20] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.C. Chen, L. Hanzo, Thirty years of machine learning: The road to Pareto-optimal wireless networks, IEEE Communications Surveys & Tutorials 22(3)(2020) 1472-1514.

[21] I.H. Sarker, Machine learning: Algorithms, real-world applications and research directions. SN Computer Science 2(3) (2021) 1-21.

[22] L. Cheng, T. Yu, A new generation of AI: A review and perspective on machine learning technologies applied to smart energy and electric power systems, International Journal of Energy Research 43(6)(2019) 1928-1973.

[23] J. Du, C. Jiang, J. Wang, Y. Ren, M. Debbah, Machine learning for 6G wireless networks: Carrying forward enhanced bandwidth, massive access, and ultrareliable/low-latency service, IEEE Vehicular Technology Magazine 15(4)(2020) 122-134.

[24] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, X. Shen, Deep reinforcement learning for autonomous internet of things: Model, applications and challenges, IEEE Communications Surveys & Tutorials 22(3)(2020) 1722-1760.

[25] F. Samie, L. Bauer, J. Henkel, From cloud down to things: An overview of machine learning in internet of things, IEEE Internet of Things Journal 6(3)(2019) 4921-4934.