Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS. 2017.Doi Number

Q-learning Algorithms: A Comprehensive Classification and Applications

BEAKCHEOL JANG, (Member, IEEE), MYEONGHWI KIM, GASPARD HARERIMANA, (Member, IEEE) and JONG WOOK KIM, (Member, IEEE)
Department of Computer Science, Sangmyung University, Seoul, South Korea

Corresponding author: Jong Wook Kim (e-mail: jkim@smu.ac.kr)

ABSTRACT Q-learning is arguably one of the most applied representative reinforcement learning approaches and one of the off-policy strategies. Since the emergence of Q-learning, many studies have described its uses in reinforcement learning and artificial intelligence problems. However, there is an information gap as to how these powerful algorithms can be leveraged and incorporated into general artificial intelligence workflow. Early Q-learning algorithms were unsatisfactory in several aspects and covered a narrow range of applications. It has also been observed that sometimes, this rather powerful algorithm learns unrealistically and overestimates the action values hence abating the overall performance. Recently with the general advances of machine learning, more variants of Q-learning like Deep Q-learning which combines basic Q learning with deep neural networks have been discovered and applied extensively. In this paper, we thoroughly explain how Q-learning evolved by unraveling the mathematical complexities behind it as well its flow from reinforcement learning family of algorithms. Improved variants are fully described, and we categorize Q-learning algorithms into single-agent and multi-agent approaches. Finally, we thoroughly investigate up-to-date research trends and key applications that leverage Q-learning algorithms.

INDEX TERMS Reinforcement learning, Q-learning, Single-agent, Multi-agent

I. INTRODUCTION

Recently reinforcement learning [1] has received considerable attention, with many successful applications in various fields such as game theory, operations research, information theory, simulation-based optimization, control theory, and statistics. Reinforcement learning, which is an area of machine learning, is becoming a major tool in computational intelligence as a technique, in which computers make their own choices in a given environment without having a clue of historical or labeled data [2]. Artificial intelligence will continue to drive cross-cutting innovations and the possibilities of future use of reinforcement learning will grow tremendously and new variants of will be introduced [3].

Reinforcement learning is a strong learning algorithm that learns the optimal policy through interaction with the environment without the model of the environment [4]. It uses an agent that learns the value function for a given policy through interaction with the environment to predict an optimal solution and based on the value function, it continuously develops and learns the optimal policy [5]. The most commonly used method in reinforcement learning applications is the Temporal-Difference (TD) learning [6] which exploits a combination of the Monte Carlo [7] method of measuring value through the experience without a model and the advantages of dynamic programming [8], which can estimate the value by using only current estimates. Q-learning uses an off-policy control that separates the deferral policy from the learning policy and updates the action selection using the Bellman optimal equations and the e-greed policy [9]. Unlike other reinforcement learning algorithms, Q-learning has simple Q-functions, hence it has become the foundation of many other reinforcement learning algorithms [10]. However, early Q-learning algorithms were impeded by the reward storage issue [11]. As the number of actions increases, the available storage space becomes insufficient, precluding the solution of the problem. In other words, for complex learning problems with large state-action environments, it is difficult to achieve effective learning. In addition, the state storage space for multi-agent environments becomes larger than that of single-agent environments, and this storage takes up much, if not all, of the computer memory [12]. Hence, the computer cannot provide the correct answer. Many Q-learning algorithms have been developed to solve this problem in various environments.

With the importance of reinforcement learning algorithms, many researchers have presented survey papers as well as classification studies many of which focused only on reinforcement learning in general [13], [14], [15].

One of the most popular algorithms for single-agent environments is deep Q-learning [16] developed at Google in 2016. In this paper we analyze algorithms for solving Q-learning problems in multi-agent environments. Modular Q-learning [17] is a multi-agent Q-learning algorithm in which a single learning problem is divided into several parts and a Q-learning algorithm applied to each. Ant Q-learning [18] is a method in which agents share reward values with each other, like a colony of ants discarding lower reward values and solving problems using higher values. This allows facilitates the action's reward values to be obtained efficiently in a multi-agent environment. Nash Q-learning [19] is a modification of the basic Q-learning algorithm that is suitable for multi-agent environments.

In the early days of reinforcement learning, Q-learning was applied to the domain of process control [20], chemical process, industrial process automatic control, and in the field of airplane control [21]. Currently, Q-learning is used in the field of network management [22] mainly for the optimization of routing and the processing of reception in network communication. With the advent of AlphaGo, active research is underway in the field of game theory [23]. Reinforcement learning through trial and error has characteristics very similar to those of the human learning process [24]. Hence, Q-learning is performing extremely well in the field of robotics. Especially in autonomous vehicles, drone, and humanoid robots [25].

In this paper, we thoroughly explore how Q-learning evolved by unraveling the mathematical complexities behind it as well its flow from reinforcement learning family of algorithms. Improved variants are fully described, and we classified into single-agent and multi-agent approaches. Finally, we extensively investigate up-to-date research trends and key applications that leverage Q-learning algorithms to various domains.

A. Organization of this paper

This paper is structured as follows. Section II introduces background knowledge and genesis of Q-learning. In Section III, we analyze and classify various Q-learning algorithms. In section IV we cover the latest research trends, as well as recent applications. Section V investigate related works on Q-learning. Finally, in Section V, we present our conclusions.

II. BACKGROUND KNOWLEDGE

Reinforcement learning has evolved as shown in Fig.1. The sequential behavior decision problem that is the basis of reinforcement learning is defined by the Markov decision process (MDP) [26] which describes an agent that introduces the concept of the value function for learning, and the value function is linked to the Bellman equation. First, reinforcement learning uses MDP and the value function to construct the Bellman equation, then Q-learning is applied to solve the Bellman equation problem. To maximize the efficiency of reinforcement learning, it is important to choose an efficient algorithm that solves the Bellman equation [27]. This section describes MDP, the value function and the Bellman equation.

A. Markov decision process

MDP is the mathematical definition of the sequential action decision problem. The environment is probabilistic, which means that the state of the transition and the compensation are random after the action is performed. The rules for selecting actions to be performed in a specific state are called policies, and reinforcement learning algorithms can be formulated using MDP. [28].

1) STATE

The state is a set S of agent observable states. State means "observation of your situation" [29].

2) ACTION

An action is a set of possible actions A in a state S. Usually, the actions that an agent can do are the same in all states. Therefore, one set of A is represented [30].

3) STATE TRANSITION PROBABILITY MATRIX

The state transition probability is a numerical representation of the movement of an agent from one state S to another state S 'when taking action A. For MDP, the following states and compensation are dependent only on the current state and actions. Thus, the probability of the next state to be compensated by the next compensation and magnitude is given by [31]. The probability is:

$$P_{ss'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a] \tag{1}$$

where (1) $P_{ss'}^{a}$ is the probability contained in the matrix P of moving to state s' when action a is performed in state s, and t denotes the time.

4) REWARD

The reward is the information given to the agent in the environment so that it can be learned by the agent. When the state is s and the action is *a* at time *t*, the reward that the agent receives is:

$$R_{ss'}^{a} = E[R_{t+1} | S_t = s, A_t = a] \tag{2}$$

where (2) $R_{ss'}^{a}$ is the definition of the reward function. t is the time, and E is the expected value for the reward to be given as action a occurs when it moves from a state to s'. The agent can express the compensation value as an expected value because it can give different reward even if the same action is taken in the same state depending on the environment. When the agent makes an action A in state S, the environment informs the agent of the next state S' in which the agent intends to go into and the reward it will receive. It is at time of t+1 that the environment informs the agent.

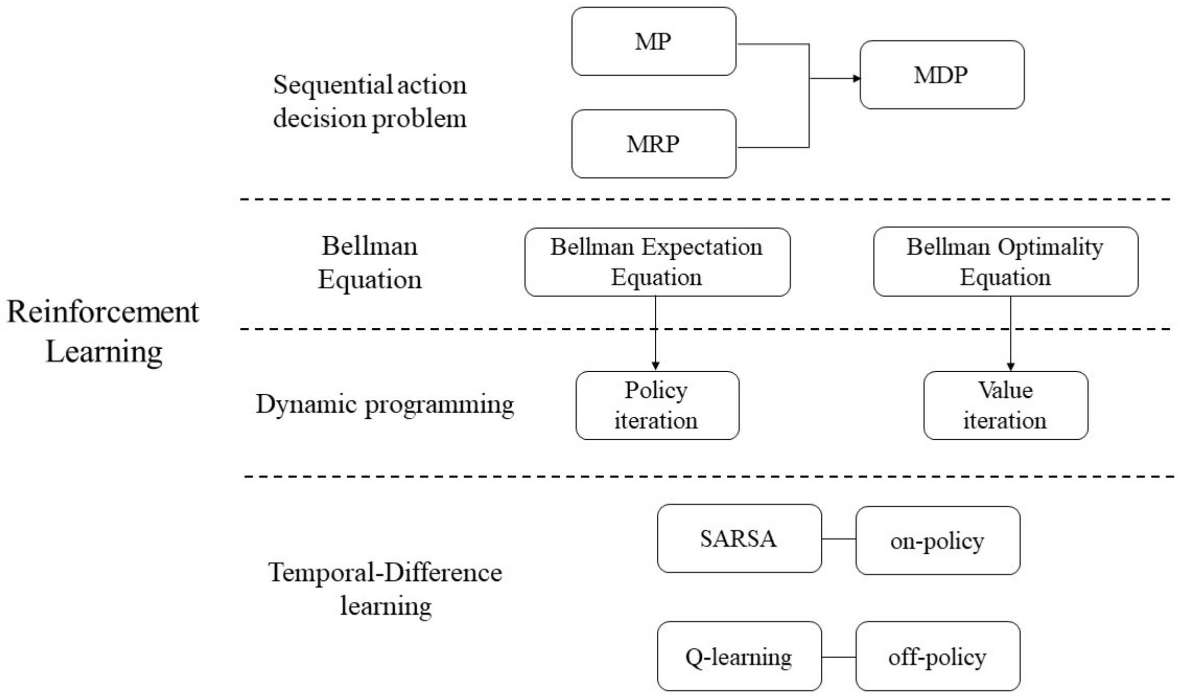

FIGURE 1. The flow of reinforcement learning

Therefore, the compensation to be received by the agent is represented by R_{t+1} .

5) DISCOUNT FACTOR

The concept of a discount factor was introduced in response to problems arising from compensation operations. After the agent acts in each state, it gets compensation. As time goes, the value of reward decreases, introducing the concept of depreciation. Depreciation has a value between 0 and 1, and the amount of compensation the agent receives over time is reduced. [31].

6) POLICY

When an agent arrives at a certain state, it determines the action using the policy

$$\pi(a \mid s) = P[A_t = a \mid S_t = s] \tag{3}$$

where (3) π is the probability of policy that the agent chooses a in state at time t. Finally, reinforcement learning learns better policies than the current one to obtain an optimal policy [32].

B. Value function

For the agent to calculate the reward that he will receive in the future it has to consider which action he will perform. The criterion that determines which policy is a better policy is the value function. The value function is the sum of the rewards that are expected to be received when following the policy from the current state [32] as follows:

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \tag{4}$$

where (4) the expectation equation Vπ (s) is the expected value Eπ, Rt +1 is the reward value to be awarded next and l is the discount factor. (4) provides the state value function that computes the sum of the rewards to be received when the state is given. It allows the agent to determine a better state.

Next, there is the action value function that considers the state and action. the agent uses the Q-function as a criterion for selecting the action. The Q-function is defined as follows

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1} \mid S_t = s, A_{t+1} = a) \tag{5}$$

The relationship between the Q- function and the value function is expressed as the following equation.

$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s) q_\pi(s,a) \tag{6}$$

For all actions, the value of the Q-function plus the policy is added together. The Q-function and the value function are expressed as Bellman equations. The Bellman equation is an equation representing the relationship between the value function of the current state and the value function of the next state.

C. Bellman equation

1) BELLMAN EXPECTATION EQUATION

The value function represents the expected value of a state. The value function of a state is the sum of the reward to be received when the agent moves to the next state and is affected by the current agent's policy. The Bellman equation that reflects the policy, expresses the relationship between the `value function of the present state and the value function of the next state [32], [33].

$$v_{\pi'}(s) = \sum_{a\in A} \pi(a \mid s)(R_{t+1} + \gamma \sum_{s'\in S} P^{a}_{ss'} v_{\pi}(s')) \qquad (7)$$

(7) is the Bellman expectation equation. $\sum_{a\in A} \pi(a|s)$ is the probability policy to do the action. $\sum_{s'\in S} P^{a}_{ss'}$ is the state transition probability matrix. As in (4) and (5), R_{t+1} is the reward, γ is the discount factor

2) BELLMAN OPTIMALITY EQUATION

Reinforcement learning is to find the optimal policy in the problem defined by the MDP. The policy is determined by the value function, and the policy that gives the greatest expectation for all policies is the optimal policy. The Bellman optimal equation is the policy that receives the optimal value using the value function. The following is the Bellman optimal equation.

$$v_{*}(s) = \max_{a} E_{\pi}[R_{t+1} + \gamma v_{*}(S_{t+1}) \mid S_t = s] \qquad (8)$$

where (8) $\max_{a} E_{\pi}$ the maximum expected value among the policies that agents can receive. Reinforcement learning calculates the problem defined by MDP using the Bellman expectation equation and the Bellman optimal equations.

III. CLASSIFICATION OF Q-LEARNING ALGORITHMS

In this section, we describe Q-learning algorithms and classify them as single-agent and multi-agent algorithms. We fully describe the most popular of them and Fig. 2 provides an extensive classification.

A. SINGLE-AGENT

1) BASIC Q-LEARNING

In contrast to previous algorithms which did not differentiate behavior from learning, *Q*-learning uses an off-policy method to separate the acting policy from the learning policy. As a result, even if the action selected in the next state was mediocre, the information was not included in the updating of the Q-function of the current state, and the dilemma is that it is a wrong choice [32]. However, since Q-learning uses off-policy, it solves the dilemma. Equation for the Q-value is as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max Q(s',a') - Q(s,a)] \qquad (9)$$

where (9) α is the learning rate and has a value between 0 and 1. R is a reward and is the reduction rate of the reward as time passes.

The Q-value Q (S, A) of the action for the current state S is updated with the sum of existing value Q (S, A) and the equation which determines the best action in the current state. Q-learning is continued by updating the Q-value for each state continuously using the above equation. Before starting Q-learning, rewards are present in the Q-table. If an agent selects an action through a policy in the starting state, then it moves to the next state using (1). This process is repeated several times so that the overall Q-value converges to a specific value where the Q-table is used to solve a given problem [33]. Q-learning combines dynamic programming and Monte Carlo methods, which have been used to solve the Bellman equation. This approach has become the basis of many reinforcement learning algorithms because unlike other methods, Q-learning is simple and exhibits an excellent learning ability in single-agent environments. However, in Q-learning, a value is updated only once per action. Therefore, it is difficult to effectively solve complicated problems in a large state-action environment because these many states- actions might not been experienced previously. Moreover, because the Q-table for rewards is preset, a considerably large amount of storage memory is required [34]. In a multi-agent environment with two or more agents, a large state-action memory is required, which leads to problems. For this reason, basic Q-learning algorithms are disadvantageous because they cannot accomplish effective learning in a multi-agent environment.

2) DEEP Q-LEARNING

Google Deep Mind developed deep Q-learning, which combines Convolution Neural Networks (CNN) with basic Q-learning. Q-learning employs an approximation function using a CNN when it becomes difficult to express the value function for every state [16] Deep Q-learning combines two approaches in addition to the value approximation using a CNN [35] . One is an experience replay, and the other is the target Q technique. The value approximation using a neural network is highly unstable, and the experience replay stabilizes this.

In the experience replay approach, all states, actions, and rewards are affected by previous states. That is, there exist correlations between states, actions, and rewards. Owing to these correlations, the approximation function cannot learn in a stable manner. The experience replay stores the experience in a buffer and randomly extracts the learning data, which eliminates correlations [36].

The target Q technique prepares the target network and Q network separately. It obtains the target value using the target network and causes the Q network to learn based on the target value, which reduces correlations [52].

The key idea behind deep Q-learning is that it uses the experiential replay to combine Q-learning with an artificial neural network (CNN) [52]. The agent generates samples (s, a, r, s’) interacting with the environment. Various environments and samples are possible. If the agent learns from the samples created according to the situation, then the learning may flow in an unusual direction owing to the correlation between the samples. To solve this problem, deep Q-learning collects many samples. When CNN learns, samples that are stored in the memory are arranged randomly and extracted as often as possible. However, using too much memory can cause the learning speed to decrease.

3) HIERARCHICAL Q-LEARNING

Hierarchical Q-Learning is designed to solve the problems that arise when the state-action space of Q-learning increases [53].

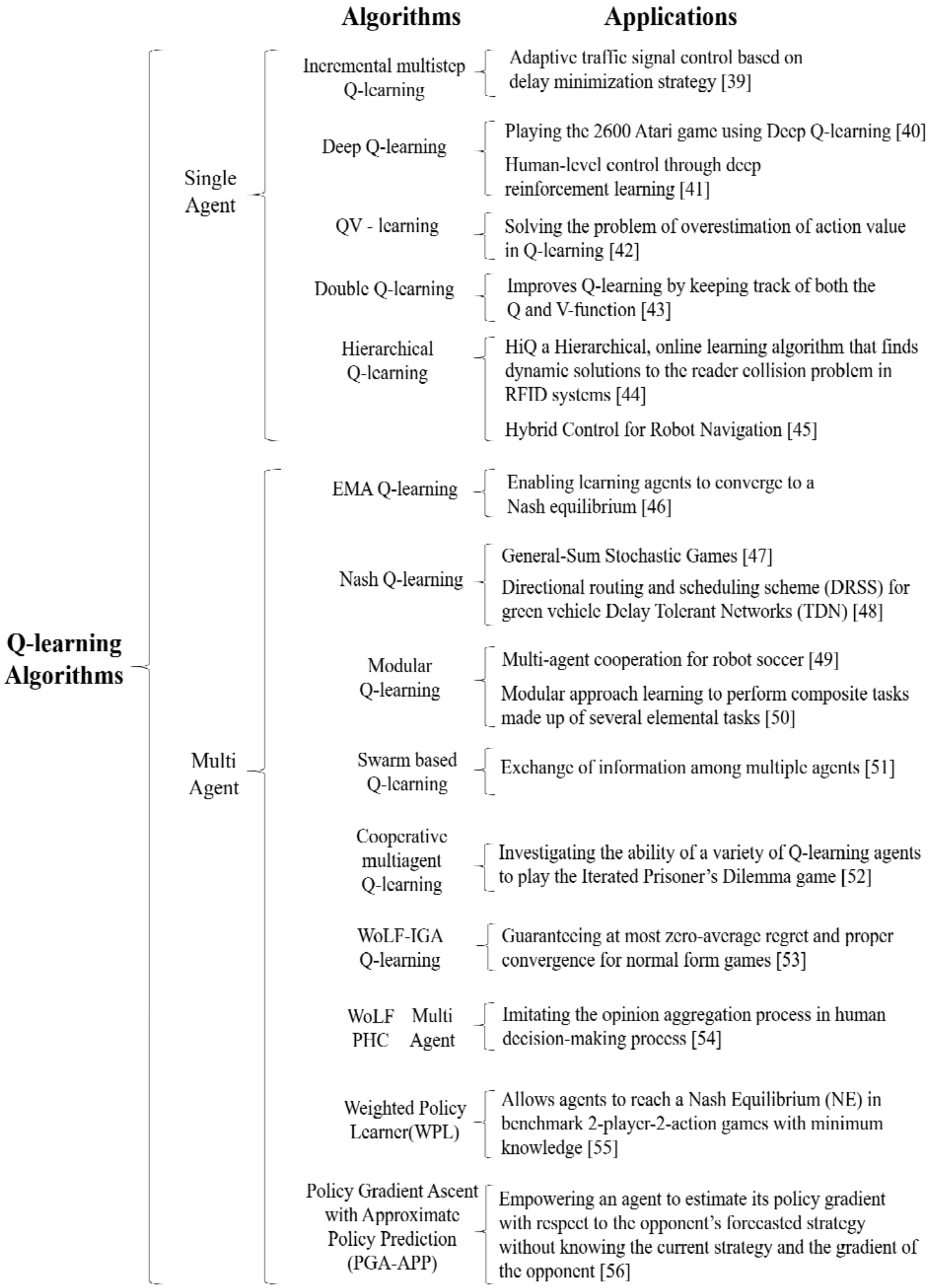
Algorithms
Applications
Q-learning Algorithms
Single Agent
Incremental multistep Q-learning
Adaptive traffic signal control based on delay minimization strategy [39]
Deep Q-learning
Playing the 2600 Atari game using Deep Q-learning [40]
Human-level control through deep reinforcement learning [41]
QV - learning
Solving the problem of overestimation of action value in Q-learning [42]
Double Q-learning
Improves Q-learning by keeping track of both the Q and V-function [43]
Hierarchical Q-learning
HiQ a Hierarchical, online learning algorithm that finds dynamic solutions to the reader collision problem in RFID systems [44]
Hybrid Control for Robot Navigation [45]
Multi Agent
EMA Q-learning
Enabling learning agents to converge to a Nash equilibrium [46]
Nash Q-learning
General-Sum Stochastic Games [47]
Directional routing and scheduling scheme (DRSS) for green vehicle Delay Tolerant Networks (TDN) [48]
Modular Q-learning
Multi-agent cooperation for robot soccer [49]
Modular approach learning to perform composite tasks made up of several elemental tasks [50]
Swarm based Q-learning
Exchange of information among multiple agents [51]
Cooperative multiagent Q-learning
Investigating the ability of a variety of Q-learning agents to play the Iterated Prisoner's Dilemma game [52]
WoLF-IGA Q-learning
Guaranteeing at most zero-average regret and proper convergence for normal form games [53]
WoLF PHC Multi Agent
Imitating the opinion aggregation process in human decision-making process [54]
Weighted Policy Learner(WPL)
Allows agents to reach a Nash Equilibrium (NE) in benchmark 2-player-2-action games with minimum knowledge [55]
Policy Gradient Ascent with Approximate Policy Prediction (PGA-APP)
Empowering an agent to estimate its policy gradient with respect to the opponent's forecasted strategy without knowing the current strategy and the gradient of the opponent [56]

FIGURE 2. **Classification of Q-learning algorithms**

Hierarchical Q-Learning improves basic Q-learning by adding hierarchical processing to the existing Q-learning system. The idea of Hierarchical Q-learning began with a method designed for the hybrid control of a robot navigation system. The main concept behind hierarchical Q-learning is the concept of the abstract action, which divides the action of the agent into a higher level and lower level [54].

For example, when the actions that the agent can choose are up, down, left, and right, and the goal is to reach the target point, the movement to the target point is contained in the higher level, and the movements of up, down, left, and right make up the lower level. This hierarchical division of the agent's action is called the abstract action [55]. Conventional Q-learning encounters many problems in solving complex environments. The hierarchical Q-learning algorithm solves complex problems using the abstract action, which speeds up the processing time for complex problems.

4) DOUBLE Q-LEARNING

Double Q-learning was developed to solve the problem that Q- learning does not perform well in a stochastic environment. In a stochastic environment [56], Q-learning is biased because the action value of the agent is overestimated. Conventional Q-learning does not search for any new optimal value after a certain time, but repeatedly selects the highest value among existing values.

Hasselt developed double Q-learning, which solves this problem of Q-learning [57], [58]. Double Q-learning divides the valuation function of Q-learning that determines the action to prevent the deviation of the value in the Q-learning algorithm. The existing algorithm is the same as Q-learning. Equation (9) is divided into two equations, and the value is selectively and randomly derived. The algorithm is as follows [56]:

$$Q^{A}(s,a) \leftarrow Q^{A}(s,a) + \alpha(s,a)(R + \gamma Q^{B}(s',a') - Q^{A}(s,a)) \quad (10)$$

$$Q^{B}(s,a) \leftarrow Q^{B}(s,a) + \alpha(s,a)(R + \gamma Q^{A}(s',a') - Q^{B}(s,a)) \quad (11)$$

Double Q-learning has two Q-functions and Each Q-function is updated with the value of another Q-function. The two Q-functions are important to learn from a separate set of experiences, but both value functions are used to choose the action.

Double Q-learning has been actively developed and combined with deep Q-learning to develop double deep Q-learning. Double deep Q-learning has also improved the performance of deep Q- learning by preventing optimistic predictions and divergences of Q-values that express future values.

5) OTHERS

In addition, there are various algorithms that utilize Q-learning in a single-agent environment. Typical examples are incremental multistep Q-learning [59], asynchronous stochastic approximation Q-learning [60], and Bayesian Q-learning. Incremental multistep Q-learning is a combination of Q-learning and Temporal-Difference learning, which is efficient for delayed reinforcement learning. The incremental multistep Q-learning algorithm performs significantly better than basic Q-learning in terms of the number of tasks. It can also serve as a basis for developing various multiple time-scale learning mechanisms, which are essential for applications of reinforcement learning to real-world problems [61]. Asynchronous stochastic approximation Q-learning analyzes the characteristics of convergence of the Q-learning algorithm. Bayesian Q-learning uses a Bayesian approach to obtain a Q-value. Thus, an agent can make decisions based on accurate information [62].

B. MULTI-AGENT

1) MODULAR Q-LEARNING

Modular Q-learning was introduced to overcome the problem of basic Q-learning's inefficiency in multi-agent systems [63]. As the number of agents increases, the number of dimensions of the state space for each agent increases exponentially in [63], [64]. This may cause an explosion in the amount of memory and number of states. Modular Q-learning solves the large state-space problem of Q-learning by decomposing a large problem to be learned into smaller problems and applying Q-learning to each sub-problem. In the action selection stage of the agent, each learning module provides Q-values for actions of the current state. A mediator module selects the best action to be taken by executing the action of the learning module. As a result, a reward value is derived from the environment and stored in each module, and the Q-function value is newly updated, as follows:

$$a \leftarrow \mathrm{argmax} \sum_{i \in l}^{n} Q_i(s,a) \quad (12)$$

where *a* is an action and denotes the Q-value. It is difficult to guarantee the convergence of Q-values in all states through infinite repetitions, which yields a function that produces the optimal result for the Q-value and chooses the action that maximizes the function. Modular Q-learning solves the problems of existing Q-learning approaches by not applying Q-learning directly to the multi-agent system, but rather dividing the system into modules for each agent, performing Q-learning on individual modules, and collecting the learning results to determine the optimal action. However, modular Q-learning uses fixed modules assigned by the engineer, and when combining them it also uses a simple fixed method known as the greatest mass (GM) approach. Therefore, the main disadvantage of this approach is that it is difficult to efficiently learn in an environment that changes rapidly. Various algorithms have been developed to solve the problem of modular Q-learning [65].

2) ANT Q-LEARNING

Ant Q-learning combines an ant system (AS) with Q-learning. AS is an algorithmic representation of ants choosing their paths back to their nest after finding food [18]. Ants secrete pheromones as they walk. They ignore weak acidity paths, and the path with the highest acidity is determined as the final path [66]. Ant Q-learning extends existing ASs. Unlike in the usual Q-learning method, learning here is performed using a set of cooperating agents. Cooperating agents exchange AQ-values with each other. The goal of Ant-Q is to learn an AQ-value that can achieve a stochastically superior target value [67].

Unlike basic Q-learning, ant Q-learning learns using several agents. The advantage of ant Q-learning is that it is possible to effectively find the value of the reward for a certain action in a multi-agent environment because agents in ant Q-learning cooperate with each other. The disadvantage of ant Q-learning is that its result can become stuck in a local minimum because agents only choose the shortest path [68].

3) NASH Q-LEARNING

Nash Q-learning is a variant of the Q-learning algorithm that is suitable for multi-agent environments [19]. In a multi-agent environment, all actions of all agents should be considered. When there are n agents, the Q-value is Q (S, A1, A2, . . . An) instead of Q (S, A). Taking this into consideration, the function of Nash Q-Learning is obtained by modifying equation (9) [42] as follows:

$$\begin{aligned} Q_t(s,a_1,a_1 \cdots a_n) &= (1-a_{t-1})Q_t(s,a_1,a_1 \cdots a_n) \\ &+ a_{t-1}[\gamma_{t-1} + \beta \mathrm{Nash}Q_{t-1}(s')] \end{aligned} \tag{13}$$

where s is the current state, a is the action of the nth agent, and t represents the time. To obtain the Nash Q-value, the learning rate must first be determined, and the rate of decrease for reward should also be determined. Furthermore, represents the reward value at time t. The value of β is between 0 and 1. Nash $Q_{t+1}(s')$ is defined as follows:

$$\mathrm{Nash}\ Q_{t\text{-}1}(s') = \pi_1(s') \cdot \pi_1(s') \cdot \ldots \pi_{t-1}(s') \tag{14}$$

where (14) is the reward value determined by the Nash theory when an agent takes an action in state s’. In a multi-agent environment, the information on different agents is not shared between the agents. Thus, agents must derive information about other agents by themselves.

The Q-values of other agents is obtained through learning [43]. Nash Q-learning predicts the actions of other agents and allows agents to determine actions that maximize the sum of the reward values of actions. The advantage of Nash Q-Learning is that its complexity is relatively low because it does not require any additional inference algorithm but uses the same algorithm to predict the actions of other agents. However, this approach is computationally intensive. The big drawback is that it requires a lot of time owing to a large amount of computation.

3) SWARM-BASED Q-LEARNING

In a typical Q-learning algorithm, if the learning problem is complex, it takes a lot of time to find the optimal answer. In addition, in a multi-agent environment, the answer often cannot be found or takes a lot of time. Swarm-based Q-learning uses Particle Swarm Optimization (PSO) to find the optimal solution. PSO can quickly find a globally optimal solution for multiple module functions with a wide solution space. There are some studies that improve the performance of reinforcement learning by combining PSO with Q-learning, Salsa, and ant colony. In this paper, we discuss swarm-based Q-learning [69].

In the existing multi-agent reinforcement learning, general Q-learning is used for each agent to search the optimum answer through individual learning, and information-based learning was performed based on the information exchanged between agents [70]. Swarm-based Q-learning solves the problem by combining the above two methods. In previous algorithms for multi-agents, each agent learned individually using a general Q-learning algorithm [71] whereas swarm-based Q-learning exchanges information regularly during learning for each agent and learns based on the exchanged information. The Q-value of each agent is updated based on the update equation of PSO, and the agent can select the optimal policy because it learns based on the exchanged information. Swarm-based Q-learning also sets up the agent in advance to save time in complex environments. The swarm-based Q-learning algorithm selects a good Q-value, and the agent updates the information using the good Q-value. PSO is based on social behavior, and each agent updates its own candidate solution using each optimal solution and the optimal solution of all agents.

4) OTHERS

In addition, there are various algorithms that utilize Q-learning in a multi-agent environment. The Self-Other-Modeling (SOM) method, agents use their own policies to predict and update the actions of other agents [72]. SOM Q-learning predicts hidden states of other agents from their actions.

Like existing hierarchical Q-learning, one task is divided into several tasks and is then divided into hierarchical tasks [73]. At the same time, this method increases the number of episodes using Stochastic Temporal Grammar (STG) [74]. The concept of STG is that there exists a temporal relationship between two other tasks, and STG summarizes time shifts among various tasks using probabilistic grammatical models to capture time relations. STGs interact with hierarchical Q-learning algorithms using modified switch and guidance policies. However, STGs rely on human guidelines and require more time and effort at each training phase [74].

Finally, [75] applied Deep Q-learning to multi-agent environments. Deep-neural-network-based algorithms have contributed greatly to extending single-agent reinforcement learning to multi-agent reinforcement learning [76]. The scenario of the collaboration and competition is designed by changing the reward for each agent, and the single-agent environment is extended to the multi-agent environment with an emphasis on the overall observation of the discrete space and each agent. Trust Region Policy Optimization (TRPO) has also been extended to multi-agent environments using parameter sharing [77]. Like these, there have been many studies on reinforcement learning in multi-agent environments.

IV. RESEARCH TREND AND KEY APPLICATIONS

With current innovative environment the momentum of new trends in the use of Q-learning is so extensive-learning is currently applied in many intelligent systems like operations research, robotics and industrial process control. In this chapter we explore these rich areas of applications and more recent innovations that involve Q-learning.

A. RESEARCH TRENDS

In this section, we investigate the latest research trends mainly to improve some aspects of Q-learning algorithms.

In reinforcement learning, it is well known that in some stochastic environments, a bias in the estimation error can gradually increase the approximation error leading to large overestimations of the true action values. A Weighted Estimator (WE) method [78] has been studied to reduce this variance and to randomly process many variables natively.

Similarly, a corrupt reward MDP (CRMDP) [79] was developed to overcome the possibility of impairment of the actual reward function. A reward for existing Q-learning can be compromised by bugs or malfunctions. The reward function may also be compromised by improperly modifying the reward mechanism by the agent. CRMDP solved the problem of reward and corruption in various agents by extending MDP with a corrupt reward function and defining formalities and measurement methods.

There is a reward shaping [80] study to overcome the time-consuming disadvantages of Q-learning using delayed feedback or reward. Reward shaping is a method of obtaining results faster by integrating domain knowledge into Q-learning. Reward shaping was applied to multi-agent as well as single-agent systems. Similarly, a method for handling false information in a single-agent system and plan-based reward shaping for solving conflict in a multi-agent system has been developed. It is based on the abstract MDP method and reward shaping, ignoring the inaccurate part of the agent's knowledge and, as a result, enables more accurate learning [81].

There is also new research that greatly improves the reward policy of the multi-agent environment by difference reward and potential reward formation. Differential reward Counterfactual as Potential (CaP) was used, and the potential-based reward was applied to various multi-agent systems. Differential Reward Incorporating Potential (DRIP) formed a differential reward system basing on the potential reward. Combining these two approaches yielded superior results than the agent using only the difference reward [82].

The existing model-free algorithm is often unable to converge to the optimal policy owing to the perturbation of the parameters. The model-free algorithm allows the learning process to be performed more reliably and quickly using Constant Shift Values (CSV). It has been generalized to handle large-scale work and its superiority has been proved through a comparison with a representative MDP [83]. In addition, research on the new Inverse Reinforcement Learning (IRL) is underway by setting a different function of the reverse learning reinforcement learning that is effective in explaining the behavior of a professional by observing a series of demonstrations different from the existing IRL algorithm [84].

Off-Environment Reinforcement learning (OFFER) has been developed to simultaneously optimize policy and proposal distribution for environmental variables in areas with abnormal Significant Rare Events (SRE) in the physical environment that do not appear in simulations [85].

In addition, robust adversarial reinforcement learning (RARL) [86] was developed to overcome the gap between simulations and real environments and the scarcity of data, and to apply it to unstable systems. In addition, through the experiments of the ATARI game and PAC-MAN, HRA obtained better results than humans. However, it has the disadvantage that its performance is not confirmed in other environments except for the specific area [87]. Similarly, PBRS-MAXQ is proposed as a new algorithm by integrating Potential Based Reward Shaping (PBRS) and Hierarchical Reinforcement Learning (HRL) [88].

P-MARL focused on the environment that had a significant impact on agent decisions. P-MARL leverages information about future changes in the environment to reach successful solutions in grid scenarios [89]. In addition, it can usefully interact in real environments, reducing human supervision costs and being applied to state-of-the-art RL systems [90]. Based on human psychology, both non specialists and experts are effective, and research on putting human knowledge into Q-learning agents for speed improvement is underway [91].

Finally, many studies have been conducted to improve the performance of Q-learning in multi-agent systems.

A new architecture, FeUdal Network (FuNs) [92], which uses MANAGER and WORKER modules, was developed by applying hierarchical Q-learning. FuNs was able to solve various problems by separating the multilevel end-to-end learning. In addition, FuNs is efficient for transfer and multitasking learning and can be used to learn new and complex technologies. There is also a HAMQ-INT [93] algorithm with excellent performance in the taxi domain, and the much more complex RoboCup Keep away domain, which utilizes hierarchical Q- learning. HAMQ- INT automatically discovers and utilizes internal transitions within Hierarchies of Abstract Machines (HAM) to verify performance in the benchmark taxi domain RoboCup Keep away domain.

In addition, there is Deep Multi-Agent Q-learning [94], which combines the experience replay to solve the problem of multiple agents and converts the success of deep learning in single-agent Q-learning into multi-agent settings. In deep multi-agent Q-learning, the importance sampling and the value function were adjusted to successfully combine experiential regeneration. This is utilized in a wide range of nonstationary educational problems such as classification.

Next, there is the Group-LASSO Fitted Q-Iteration (GL-FQI) [95]. which improves performance by simultaneously learning multiple tasks and using similarity in multitask Q-learning. GL-FQI is made by extending the Group-LASSO and FQI algorithms and shares a useful set of functions that improve the performance of single-task learning.

There is also a resource abstraction [96] that provides an autonomous and decentralized solution by applying multi-agent Q-learning to very complex, large-scale real congestion statements. In addition, researchers developed a swarmMDP framework for multi-agent systems by applying reverse reinforcement learning [97].

B. RECENT ADVANCES

Owing to the effectiveness of the Q-learning algorithm, it has been applied to various domains such as industrial processes, network process, game theory, robotics, operation research, control theory, and image recognition. In this section, we describe various applications that leverage the recent advances of Q-learning. Table I summarizes key areas that currently utilize Q-learning techniques

1) CONTROL OF INDUSTRIAL PROCESS

The field of process control, which represents the beginning of reinforcement learning, is still one of the most active application areas of Q-learning. This is because the Q learning methods that mimic the way humans are trained through trial and error can be akin to industrial process control [98]. Q-learning was adopted to improve the performance of the on-line learning control system [99]. The online learning control system using Q-learning has strengthened measures for judging incorrect points from an external environment and improving future performance and has become a successful candidate for the design of online learning control [100] [101].

2) COMPUTER NETWORKING

There has been much research to apply Q-learning in the field network process control. Wireless sensor networks should monitor rapidly changing dynamic behaviors that are caused by external factors or by system designers. To improve the adaptability to changing situations and eliminate the need for system redesign, Q-learning is used to improve performance [102]. Network control using the Q-learning algorithm has inspired many practical solutions that maximize resource utilization and extend network life. In recent years, an antisystem has been applied to pre-networking to enable monitoring and object tracking in a wide range of environments [103]. In addition, the combination of Mocha, a robust system recognition optimization method for system problems [104], and the combination of an online control system and a wireless sensor network has enhanced the performance of wireless sensor network applications [105].

3) GAME THEORY

In game theory, Google Deep Mind has played a significant role. Deep Mind applied deep Q-learning to games helping arcane games to find optimal moves by themselves, and as a result the system was able to outperform humans [106], [107]. Based on this, much research has been carried out in game theory. In an online multiplayer game, it is possible to learn in real time using the data measured along the trajectory of a player, thereby optimizing the game's performance and developing a human-agent feedback loop.

Furthermore, in stochastic cooperative game theory, a Q-learning algorithm is used to maximize the total profit of the system. Recently, Q-learning algorithms have been adopted in mobile application games. As mobile applications become more popular, they have suffered from limited resources like channels hence causing, delay [108]. The combination of game theory and Q-learning has enabled the efficient distribution of resources, yielding improved performance. In addition, research is being conducted to improve performance by utilizing Q-learning in large-scale games with insecure information [109], [110].

4) ROBOTICS

Robotics is the most active field to which Q-learning is applied. Q-learning in robotics provides frameworks and toolkits for designing sophisticated and difficult engineering behavior. Through Q-learning, autonomous robots have achieved considerable growth in behavioral technology with minimal human intervention. However, many studies are in progress to overcome the complicated problems of the existing Q-learning algorithm and its inability to operate with multiple agents.

By seamlessly exchanging information between tasks, a fully integrated approach within the reinforcement learning framework has greatly enhanced robot control capabilities [111]. In recent years, Q-learning has been applied to study robots' emotions and to facilitate mobile robots operating in people's living environments. Robot problems have also influenced the development of Q-learning. The combination of robotics and reinforcement learning has tremendous potential in future research [112], [113].

5) OPERATIONS RESEARCH

Operation Research (OR) is a discipline that deals with the application of advanced analytical methods to make better decisions using math, business, and computer science. Results of OR problems are used in a wide range of engineering management and public systems. In this section, we investigate various studies that utilize Q-learning for the latest OR problem solving. [114] improved the performance of the scheduling method that solves Dynamic Job Shop Scheduling (DJSS) problem considering random work and machine failure by using Q-learning. DJSS focused on selecting an appropriate scheduling method or optimization parameter.

TABLE I
RECENT Q-LEARNING APPLICATIONS

Area	Application	Reference
Control of industrial process	● Improving the performance of the on-line learning control system. ● Optimizing temperature control and power consumption.	[98] [99] [100] [101]
Computer Networking	● Improving the adaptability of Wireless Sensor Networks to changing situations and eliminating the need for system redesign ● Allows wireless nodes to observe and collect information in a dynamic local operating environment for complex routing decisions	[102] [103] [104] [105]
Game theory	● Automatically learning arcane games and beating human gamers (By Google Deep Mind) ● Maximizing the total profit of the system in stochastic cooperative game theory. ● Efficient distribution of resources mobile based games	[106] [107] [108] [109]
Robotics	● Providing frameworks and toolkits for designing sophisticated behavioral aspects. ● Studying robots' emotions and to facilitate mobile robots operating in people's living environments.	[110] [111] [112]
Operations Research	● Improving the performance of the scheduling method that solves Dynamic Job Scheduling (DJSS) ● Load management problems dynamically to adjust the electricity demand in response to grid signals ● Optimization problems requiring device placement	[113] [114] [115] [116] [117] [118]
Artificial intelligence	● Quadrotor control	[119] [120]
	● Image processing and classification	[121] [122] [123]
	● Information theory and natural language processing	[124] [125] [126] [127] [128]

Q-learning has also been used in demand management problems [115]. Load management problems dynamically adjust the electricity demand in response to grid signals to reduce Demand-Side Management (DSM) for preliminary markets, frequency recovery, and expensive household usage. For efficiency of the management problem, it is necessary to disperse peak loads to other load times. As efficiency increases, operational costs are diminished, and the number of blackouts is reduced [115]. This gives consumers and producers the ability to manage with minimal effort. The Q-learning approach enables retailers to quickly identify real-time information they need and provides demand management capabilities by making reliable decisions about trusted customers without classifying future customers from the appropriate clusters. Building load management using reinforcement learning has chosen intuitive clustering technology based on learner's results and improved elasticity of demand, learner's load scheduling, and consumer targeting decomposition techniques. It is also used as a tool for market research [116].

Q-learning has also been used for optimization problems for device placement [117]. Device placement can be grouped into learning to divide a graph across available devices. It makes traditional graph partitioning into a natural baseline. Adaptation methods for optimizing the arrangement of devices for neural networks have been studied [118]. This arranges devices by using the sequence placement model and considering computation in a neural network.

As a result, this approach learns characteristics of the environment including complex tradeoffs between computation and communication in hardware, and overcomes the placement designed by the human expert and highly optimized algorithmic solvers in a variety of tasks including image classification, language modeling, and machine translation. This model has been trained to optimize the execution time of the neural network [119].

TABLE Ⅱ
RELATED PAPERS AND THEIR LIMITATIONS ADDRESSED BY THE CURRENT WORK

Study	Year	Scope	Limitations
[130]	2017	The paper discusses and classifies various Reinforcement learning with special focus on deep Q-learning. They fully describe current research and challenges as well as recent benchmarks for Deep reinforcement learning.	Though the paper tries to explain the learning behavior of Deep Q networks, they do not provide the mathematical background as well as many variants that are there in many applications. Except highlighting some applications like the Atari game, the paper does not dive deeper into the descriptions off various applications of deep Q-learning.
[131]	2017	This paper provides a complete review of Reinforcement algorithms applied to control solutions. The authors discuss Q-learning and use the Bellman equation to fully describe RL. They use off-policy and on-policy RL procedures to utilize while solving various control problems.	The paper concentrates on recipes needed to address various control problems and Q-learning comes as one of the solutions but the paper does not classify the algorithms and does not provide current research trends.
[132]	2015	This study focusses on safe RL and concludes that many of the RL of the approaches utilize model-free RL algorithms like Q-learning.	The paper does not provide an extensive classification as well as recent advances that are key to novel applications.
[133]	2010	The paper analyses RL for risk sensitive applications. They put a special emphasis on Q-learning and actor-critic algorithms. The paper provides mathematical background and some applications of Q-learning to risk sensitive environments.	The paper does not fully describe Q-learning and its applications. It does not classify its variants into single and double agents and does not describe future research and trends.
[134]	2009	The paper describes the genesis of RL with emphasis on RL with Q-values. The paper describes various extensions of the fundamental RL ideas to diverse problem domains and algorithm. The paper covers some RL case studies in operations research and management.	Owing to the recent advances of Q-learning from the time the paper was written the paper does not thoroughly classify the algorithms and how they can be leveraged. Also, various applications were not yet envisaged by the time.
Current Study	2019	Owing to the vast applications that use Q-learning, the current paper focuses on these algorithms. The aim is to classify them as based on the number and behaviors of their agents. The paper also covers fully the recent advances as well as a range of applications that are currently leveraging these powerful algorithms.	The current research does not experimentally describe the technical details of any of the variants of Q-learning. As no other research has tried to classify these algorithms the paper only focuses on single and double agent Q-learning variants. However, there are other types that utilize multiple agents for many applications.

6) ARTIFICIAL INTELLIGENCE

In addition to the abovementioned fields, many studies using Q-learning have been conducted and are being applied to various fields. Q-learning is used in artificial intelligence quadrotor control [120]. Recently, as the development and distribution of quadrotors have accelerated, many global companies such as Google and Amazon have conducted research for the commercial use of quadrotors. Quadrotors have been used and verified in many areas such as surveillance, navigation and rescue, wildlife protection, and unmanned mail delivery. Its core technique is to accurately recognize and track targets, which is indispensable in the application of quadrotors. Q-learning has contributed significantly to the development of drones as a key technology in quadrotor control. Many techniques for controlling quadrotors using Q-learning have been studied. The quadrotor is expected to be widely used not only in the military and commercial industries but also in the private sector [121].

Q-learning is also used in the field of image classification. Image classification is one of the most fundamental research problems in computer vision. In existing image classification, Convolutional Neural Networks (CNN) made great achievements in single-label image classification [122]. In multi-label image analysis, however, computation cost and spatial dependence, and modeling between localized regions, are neglected or oversimplified.

Multi-label image classification is more useful than single-label image classification because the actual image is typically annotated to multiple labels, and modeling large semantic information is essential for high-level image analysis tasks. Q-learning has made a great contribution in the analysis of multilevel image classification [123], and a representative example is a new method for accurate real-time 3D anatomical landmark detection in Computed Tomography (CT) scans.

By combining deep Q-learning concepts with multiscale image analysis, an artificial agent learned the optimal strategy for finding anatomical structures [124].

Finally, Q-learning is applied in information theory, and related studies are underway. Recently, Q-learning and information theory have been applied to various fields such as pattern recognition, natural language processing, abnormality detection, and information theory [125], [126], [127]. In addition, a framework has been developed to generate a satisfactory response based on user's utterance using reinforcement learning in a voice interaction system [128], and a high-resolution prediction system for local rainfall based on deep learning has been developed [129].

V. RELATED WORKS

Notwithstanding a rich applications perspective of Q-learning, we did not come across any paper that bestowed its scope solely on Q-learning and its applications. However various researches tried to touch on these algorithms in a general scope of reinforcement learning hence falling short of various details that Q-learning is built upon. Table Ⅱ summarizes the key related limitations that our paper tries to address. We also reveal the limitations of our paper to encourage possible further studies.

VI. CONCLUSION

Q-learning algorithms are off policy reinforcement learning algorithms that try to perform the most profitable action given the current state. However, these powerful set of algorithms are not fully exploited at their full potential. In this paper we covered all variants of Q-learning algorithms, which are a representative algorithm under reinforcement learning. We distinctively categorized Q-learning algorithms into single-agent and multi-agent and described them thoroughly. With the introduction of a Convolutional Neural Networks, deep Q-learning came as an improved version of basic Q-learning. Double Q-learning solves the basic flaw of basic Q-learning which is the over estimation of the reward using a maximum function. Modular Q-learning is widely utilized in the field of robotics and Nash Q-learning is applied in complex areas such as stochastic games. We also analyzed recent research trend of Q-learning and thoroughly investigated how Q-learning is used in various areas. The improved algorithms might perform poorly while solving simple problems in a simple environment, but they outperform basic Q-learning algorithms when the problem at hand is complex and under a sophisticated environment. As the importance of reinforcement learning increases with artificial intelligence being incorporated in almost all aspects of computing, Q-learning will continue to drive the innovations and development of intelligent systems.

REFERENCES

[1] David Chapman and Leslie Pack Kaelbling. 1991. Input Generalization in Delayed Reinforcement Learning: An Algorithm and Performance Comparisons. In IJCAI, 726–731

[2] Michael I. Jordan and Tom M. Mitchell. 2015. Machine learning: Trends, perspectives, and prospects. Science 349, 6245 (2015), 255–260.

[3] E. Parisotto, S. Ghosh, S. B. Yalamanchi, V. Chinnaobireddy, Y. Wu, and R. Salakhutdinov, "Concurrent Meta Reinforcement Learning," arXiv:1903.02710 [cs], Mar. 2019.

[4] Jens Kober, J. Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. The International Journal of Robotics Research 32, 11 (2013), 1238–1274.

[5] Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. Communications of the ACM 38, 3 (1995), 58–68.

[6] J. A. Boyan, "Technical Update: Least-Squares Temporal Difference Learning," p. 14.

[7] W. R. Gilks, S. Richardson, and D. Spiegelhalter, Markov chain Monte Carlo in practice. Chapman and Hall/CRC, 1995.

[8] M. L. Puterman, Markov Decision Processes.: Discrete Stochastic Dynamic Programming. John Wiley & Sons, 2014.

[9] P. Dayan and C. Watkins, "Q-learning," Machine learning, vol. 8, no. 3, pp. 279–292, 1992.

[10] Richard Dearden, Nir Friedman, and Stuart Russell. 1998. Bayesian Q-learning. In AAAI/IAAI, 761–768

[11] Alessandro Lazaric. 2012. Transfer in reinforcement learning: a framework and a survey. In Reinforcement Learning. Springer, 143–173.

[12] D.Kwak and B.-T. Zhang, "Introspective reinforcement learning: learning in sparse reward environment," Korean Institute of Information Scientists and Engineers, pp. 808–810, 2017

[13] Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. arXiv preprint arXiv:1707.08114 (2017).

[14] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Robotics and Automation (ICRA), 2017 IEEE International Conference on, 3389–3396.

[15] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. 2015. Bayesian reinforcement learning: A survey. Foundations and Trends® in Machine Learning 8, 5–6 (2015),

359–483.

[16] Hester T, Vecerik M, Pietquin O, Lanctot M, Schaul T, Piot B, et al. Deep q-learning from demonstrations. Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[17] Chen K. Tham and Richard W. Prager. 1994. A modular q-learning architecture for manipulator task decomposition. In Machine Learning Proceedings 1994. Elsevier, 309–317.

[18] Marco Dorigo and Luca Maria Gambardella. 1996. A study of some properties of Ant-Q. In International Conference on Parallel Problem Solving from Nature, 656–665.

[19] Yang L, Sun Q, Ma D, Wei Q. Nash Q-learning based equilibrium transfer for integrated energy management game with We-Energy. Neurocomputing. 2019

[20] Yi Jiang, Jialu Fan, Tianyou Chai, Jinna Li, and Frank L. Lewis. 2018. Data-driven flotation industrial process operational optimal control based on reinforcement learning. IEEE Transactions on Industrial Informatics 14, 5 (2018), 1974–1989

[21] Said G. Khan, Guido Herrmann, Frank L. Lewis, Tony Pipe, and Chris Melhuish. 2012. Reinforcement learning and optimal adaptive control: An overview and implementation examples. Annual Reviews in Control 36, 1 (April 2012), 42–59.

[22] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato, and Hwee-Pink Tan. 2014. Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications. IEEE Communications Surveys & Tutorials 16, 4 (2014), 1996–2018.

[23] Kyriakos G. Vamvoudakis, Hamidreza Modares, Bahare Kiumarsi, and Frank L. Lewis. 2017. Game theory-based control system algorithms with real-time reinforcement learning: How to solve multiplayer games online. IEEE Control Systems 37, 1 (2017), 33–52. 2017 35

[24] Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. 2018. Emotion in reinforcement learning agents and robots: a survey. Machine Learning 107, 2 (February 2018), 443–480

[25] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. 2018. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. arXiv:1802.09464

[26] Chelsea C. White and Douglas J. White. 1989. Markov decision processes. European Journal of Operational Research 39, 1 (March 1989), 1–16

[27] Richard S Sutton. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. 7.

[28] Eyal Even-Dar and Yishay Mansour. 2001. Learning Rates for Q-Learning. In Computational Learning Theory, David Helmbold and Bob Williamson (eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 589–604.

[29] Anthony Rocco Cassandra. 1998. Exact and Approximate Algorithms for Partially Observable Markov Decision Processes. Brown University, Providence, RI, USA.

[30] Michael L. Littman. 2001. Value-function reinforcement learning in Markov games. Cognitive Systems Research 2, 1 (April 2001), 55–66.

[31] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. 7.

[32] Thomas G. Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research 13, (2000), 227–303.

[33] Marina Irodova and Robert H Sloan. Reinforcement Learning and Function Approximation. 6.

[34] Lu Shoufeng, Liu Ximin, and Dai Shiqiang. 2008. Q-Learning for adaptive traffic signal control based on delay minimization strategy. In Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on, 687–691.

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602 [cs] (December 2013). Retrieved January 2, 2019 .

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Georg Ostrovski. 2015. Human-level control through deep reinforcement learning. Nature 518, 7540 (2015), 529.

[37] Hado V. Hasselt. 2010. Double Q-learning. In Advances in Neural Information

[38] Marco A. Wiering. 2005. QV (lambda)-learning: A new on-policy reinforcement learning algrithm. In Proceedings of the 7th European Workshop on Reinforcement Learning, 17–18.

[39] Junius Ho, Daniel W. Engels, and Sanjay E. Sarma. 2006. HiQ: a hierarchical Q-learning algorithm to solve the reader collision problem. In Applications and the Internet Workshops, 2006. SAINT Workshops 2006. International Symposium on, 4–pp.

[40] C. Chen, H.-X. Li, and D. Dong, "Hybrid control for robot navigation-a hierarchical Q-learning algorithm," IEEE Robotics & Automation Magazine, vol. 15, no. 2, 2008.

[41] Mostafa D. Awheda and Howard M. Schwartz. 2013. Exponential moving average Q-learning algorithm. In Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2013 IEEE Symposium on, 31–38

[42] Junling Hu and Michael P. Wellman. 2003. Nash Q-learning for general-sum stochastic games. Journal of machine learning research 4, Nov (2003), 1039–1069.

[43] Yuanyuan Zeng, Kai Xiang, Deshi Li, and Athanasios V. Vasilakos. 2013. Directional routing and scheduling for green vehicular delay tolerant networks. Wireless networks 19, 2 (2013), 161–173.

[44] Kui-Hong Park, Yong-Jae Kim, and Jong-Hwan Kim. 2001. Modular Q-learning based multi-agent cooperation for robot soccer. Robotics and Autonomous Systems 35, 2 (2001), 109–122.

[45] Tong Zhou, Bing-Rong Hong, Chao-Xia Shi, and Hong-Yu Zhou. 2005. Cooperative behavior acquisition based modular Q learning in multi-agent system. In Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on, 205–210.

[46] Hitoshi Iima and Yasuaki Kuroe. 2007. Swarm reinforcement learning algorithms-exchange of information among multiple agents. In SICE, 2007 Annual Conference, 2779–2784.

[47] Tuomas W. Sandholm and Robert H. Crites. 1995. On multiagent Q-learning in a semi-competitive domain. In International Joint Conference on Artificial Intelligence, 191–205.

[48] Michael Bowling. 2005. Convergence and no-regret in multiagent learning. In Advances in neural information processing systems, 209–216.

[49] Chao Yu, Minjie Zhang, Fenghui Ren, and Xudong Luo. 2013. Emergence of social norms through collective learning in networked agent societies. In Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, 475–482.

[50] Sherief Abdallah and Victor Lesser. 2008. Non-linear dynamics in multiagent reinforcement learning algorithms. In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3, 1321–1324.

[51] Chongjie Zhang and Victor R. Lesser. 2010. Multi-Agent Learning with Policy Prediction. In AAAI. Adithya M. Devraj and Sean P. Meyn. 2017. Fastest Convergence for Q-learning. arXiv:1707.03770 [cs, math] (July 2017). Retrieved January 2, 2019

[52] Qingchen Zhang, Man Lin, Laurence T. Yang, Zhikui Chen, and Peng Li. 2017. Energy-efficient scheduling for real-time systems based on deep Q-learning model. IEEE Transactions on Sustainable Computing (2017).

[53] Andrew G. Barto and Sridhar Mahadevan. 2003. Recent advances in hierarchical reinforcement learning. Discrete event dynamic

systems 13, 1–2 (2003), 41–77.

[54] ChunLin Chen, DaoYi Dong, Han-Xiong Li, and Tzyh-Jong Tarn. 2011. Hybrid MDP based integrated hierarchical Q-learning. Science China Information Sciences 54, 11 (November 2011), 2279–2294.

[55] Daniel Rasmussen, Aaron Voelker, and Chris Eliasmith. 2017. A neural model of hierarchical reinforcement

[56] Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep Reinforcement Learning with Double Q-learning. arXiv:1509.06461 [cs] (September 2015). Retrieved January 2, 2019

[57] Christopher Schulze and Marcus Schulze. 2018. ViZDoom: DRQN with Prioritized Experience Replay, Double-Q Learning, & Snapshot Ensembling. arXiv:1801.01000 [cs] (January 2018). Retrieved January 2, 2019

[58] Hado V. Hasselt. 2010. Double Q-learning. In Advances in Neural Information Processing Systems 23, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel and A. Culotta (eds.). Curran Associates, Inc., 2613–2621. Retrieved January 2, 2019

[59] Jing Peng and Ronald J. Williams. 1994. Incremental Multi-Step Q-Learning. In Machine Learning Proceedings 1994, William W. Cohen and Haym Hirsh (eds.). Morgan Kaufmann, San Francisco (CA), 226–232

[60] John N. Tsitsiklis. 1994. Asynchronous stochastic approximation and Q-learning. Mach Learn 16, 3 (September 1994), 185–202

[61] Tao Yu, Bin Zhou, Ka Wing Chan, Liang Chen, and Bo Yang. 2011. Stochastic Optimal Relaxed Automatic Generation Control in Non-Markov Environment Based on Multi-Step $ Q ($\backslash$lambda) $ Learning. IEEE Transactions on Power Systems 26, 3 (2011), 1272–1282.

[62] Malcolm Strens. A Bayesian Framework for Reinforcement Learning. 8.

[63] Norihiko Ono and Kenji Fukumoto. 1996. Multi-agent Reinforcement Learning: A Modular Approach. (1996), 7.

[64] Takayuki Kohri, Kei Matsubayashi, and Mario Tokoro. 1997. An adaptive architecture for modular Q-learning. In IJCAI (2), 820–825

[65] Norihiko Ono and Kenji Fukumoto. 1997. A modular approach to multi-agent reinforcement learning. In Distributed Artificial Intelligence Meets Machine Learning Learning in Multi-Agent Environments (Lecture Notes in Computer Science), 25–39.

[66] Hyun Kim and Tae-Choong Chung. 2011. Solving the Gale-Shapley Problem by Ant-Q learning. The KIPS Transactions:PartB 18B, 3 (2011), 165–172

[67] Luca M. Gambardella and Marco Dorigo. 1995. Ant-Q: A Reinforcement Learning approach to the traveling salesman problem. In Machine Learning Proceedings 1995, Armand Prieditis and Stuart Russell (eds.). Morgan Kaufmann, San Francisco (CA), 252–260.

[68] Chia-Feng Juang and Chun-Ming Lu. 2009. Ant colony optimization incorporated with fuzzy Q-learning for reinforcement fuzzy control. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 39, 3 (2009), 597–608.

[69] Hitoshi Iima and Yasuaki Kuroe. 2008. Swarm reinforcement learning algorithms based on sarsa method. In SICE Annual Conference, 2008, 2045–2049.

[70] Wei Lu, Yunlong Zhang, and Yuanchang Xie. 2011. A multi-agent adaptive traffic signal control system using swarm intelligence and neuro-fuzzy reinforcement learning. In Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on, 233–238

[71] Marco Dorigo and Mauro Birattari. 2007. Swarm intelligence. Scholarpedia 2, 9 (September 2007), 1462.

[72] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. 2018. Modeling Others using Oneself in Multi-Agent Reinforcement Learning. arXiv:1802.09640 [cs] (February 2018). Retrieved January 2, 2019

[73] Tianmin Shu, Caiming Xiong, and Richard Socher. 2017. Hierarchical and Interpretable Skill Acquisition in Multi-task Reinforcement Learning. (December 2017). Retrieved January 2,

[74] Charles Gretton. Gradient-Based Relational Reinforcement Learning of Temporally Extended Policies. 8.

[75] Xiangxiang Chu and Hangjun Ye. 2017. Parameter Sharing Deep Deterministic Policy Gradient for Cooperative Multi-agent Reinforcement Learning. arXiv:1710.00336 [cs] (October 2017). Retrieved January 2, 2019

[76] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2015. Dueling Network Architectures for Deep Reinforcement Learning. arXiv:1511.06581 [cs] (November 2015). Retrieved January 2, 2019

[77] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon and R. Garnett (eds.). Curran Associates, Inc., 2137–2145. Retrieved January 2, 2019

[78] Carlo D'Eramo, Alessandro Nuara, Matteo Pirotta, and Marcello Restelli. 2017. Estimating the maximum expected value in continuous reinforcement learning problems. In 31st AAAI Conference on Artificial Intelligence, AAAI 2017, 1840–1846.

[79] Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg. 2017. Reinforcement Learning with a Corrupted Reward Channel. arXiv:1705.08417 [cs, stat] (May 2017). Retrieved January 2, 2019

[80] Marek Grześ. 2017. Reward Shaping in Episodic Reinforcement Learning. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS '17), 565–573. Retrieved January 2, 2019

[81] Kyriakos Efthymiadis and Daniel Kudenko. 2015. Knowledge Revision for Reinforcement Learning with Abstract MDPs. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15), 763–770. Retrieved January 2, 2019

[82] Sam Devlin, Logan Yliniemi, Daniel Kudenko, and Kagan Tumer. 2014. Potential-based Difference Rewards for Multiagent Reinforcement Learning. In Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS '14), 165–172. Retrieved January 2, 2019

[83] Shangdong Yang, Yang Gao, Bo An, Hao Wang, and Xingguo Chen. 2016. Efficient Average Reward Reinforcement Learning Using Constant Shifting Values. In AAAI, 2258–2264.

[84] Alberto Maria Metelli, Matteo Pirotta, and Marcello Restelli. 2017. Compatible Reward Inverse Reinforcement Learning. In Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett (eds.). Curran Associates, Inc., 2050–2059. Retrieved January 2, 2019

[85] Kamil Andrzej Ciosek and Shimon Whiteson. 2017. OFFER: Off-Environment Reinforcement Learning. In AAAI, 1819–1825

[86] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. 2017. Robust Adversarial Reinforcement Learning. arXiv:1703.02702 [cs] (March 2017). Retrieved January 2, 2019

[87] Harm Van Seijen, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang. 2017. Hybrid Reward Architecture for Reinforcement Learning. In Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett (eds.). Curran Associates, Inc., 5392–5402. Retrieved January 2, 2019

[88] Y. Gao and F. Toni. 2015. Potential based reward shaping for hierarchical reinforcement learning. 3504–3510.

[89] Andrei Marinescu, Ivana Dusparic, Adam Taylor, Vinny Cahill, and Siobhán Clarke. 2015. P-MARL: Prediction-Based Multi-

Agent Reinforcement Learning for Non-Stationary Environments. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15), 1897–1898. Retrieved January 2, 2019

[90] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett (eds.). Curran Associates, Inc., 4299–4307. Retrieved January 2, 2019

[91] Ariel Rosenfeld, Moshe Cohen, Matthew E. Taylor, and Sarit Kraus. 2018. Leveraging human knowledge in tabular reinforcement learning: A study of human subjects. arXiv:1805.05769 [cs] (May 2018). Retrieved January 2, 2019

[92] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. FeUdal Networks for Hierarchical Reinforcement Learning. arXiv:1703.01161 [cs] (March 2017). Retrieved January 2, 2019

[93] Aijun Bai and Stuart Russell. 2017. Efficient reinforcement learning with hierarchies of machines by leveraging internal transitions. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI, 19–25.

[94] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. arXiv:1702.08887 [cs] (February 2017). Retrieved January 2, 2019

[95] Daniele Calandriello, Alessandro Lazaric, and Marcello Restelli. 2014. Sparse multi-task reinforcement learning. In Advances in Neural Information Processing Systems, 819–827

[96] Kleanthis Malialis, Sam Devlin, and Daniel Kudenko. 2016. Resource Abstraction for Reinforcement Learning in Multiagent Congestion Problems. In Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems (AAMAS '16), 503–511. Retrieved January 2, 2019

[97] Adrian Šošić, Wasiur R. KhudaBukhsh, Abdelhak M. Zoubir, and Heinz Koeppl. 2017. Inverse Reinforcement Learning in Swarm Systems. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS '17), 1413–1421. Retrieved January 2, 2019

[98] C. A. Coker, Motor learning and control for practitioners. Routledge, 2017.

[99] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," IEEE transactions on neural networks and learning systems, vol. 29, no. 6, pp. 2042–2062, 2018.

[100] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," IEEE Transactions on Neural networks, vol. 12, no. 2, pp. 264–276, 2001.

[101] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al. Continuous control with deep reinforcement learning. arXiv preprint arXiv:150902971. 2015.

[102] H. A. Al-Rawi, M. A. Ng, and K.-L. A. Yau, "Application of reinforcement learning to routing in distributed wireless networks: a review," Artificial Intelligence Review, vol. 43, no. 3, pp. 381–416, 2015.

[103] R. GhasemAghaei, M. A. Rahman, W. Gueaieb, and A. El Saddik, "Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks," in Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE, 2007, pp. 1–6.

[104] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in Advances in Neural Information Processing Systems, 2017, pp. 4424–4434.

[105] M. I. Khan, K. Xia, A. Ali, and N. Aslam, "Energy-aware task scheduling by a true online reinforcement learning in wireless sensor networks," International Journal of Sensor Networks, vol. 25, no. 4, pp. 244–258, 2017.

[106] K. Madani and M. Hooshyar, "A game theory–reinforcement learning (GT–RL) method to develop optimal operation policies for multi-operator reservoir systems," Journal of hydrology, vol. 519, pp. 732–742, 2014.

[107]. Gao and L. Pavel, "On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning," arXiv preprint arXiv:1704.00805, 2017.

[108] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile Edge Computation Offloading Using Game Theory and Reinforcement Learning," arXiv preprint arXiv:1711.09012, 2017.

[109] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, 2017, pp. 464–473.

[110] C.-Y. Wei, Y.-T. Hong, and C.-J. Lu, "Online reinforcement learning in stochastic games, Advances in Neural Information Processing Systems, pp. 4994–5004, 2017.

[111] A. Barreto, W. Dabney, R. Munos, J.J. Hunt, T. Schaul, H.P. van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," Advances in Neural Information Processing Systems, pp. 4055–4065, 2017.

[112] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," arXiv preprint arXiv:1511.03791, 2015.

[113] Y. Zhu, R. Mottaghi, E. Kolve, J.J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning. In Robotics and Automation (ICRA), IEEE International Conference on, pp. 3357–3364, 2017.

[114] J. Shahrabi, M.A. Adibi, and M. Mahootchi, "A reinforcement learning approach to parameter estimation in dynamic job shop scheduling," Computers & Industrial Engineering, vol. 110, pp. 75–82, 2017.

[115] S. Ahmed and F. Bouffard, "Building load management clusters using reinforcement learning," 2017.

[116] E. Mocanu, P.H. Nguyen, W.L. Kling, and M. Gibescu, "Unsupervised energy prediction in a Smart Grid context using reinforcement cross-building transfer learning, Energy and Buildings, vol. 116, pp. 646–655, 2016.

[117] A. Mirhoseini, A. Goldie, H. Pham, B. Steiner, Q.V. Le, and J. Dean, "A hierarchical model for device placement."

[118] A. Mirhoseini, H. Pham, Q.V. Le, B. Steiner, R. Larsen, Y. Zhou, and J. Dean, "Device placement optimization with reinforcement learning," arXiv preprint arXiv:1706.04972, 2017.

[119] M. Qiao, H. Zhao, S. Huang, L. Zhou, and S. Wang," Optimal channel selection based on online decision and offline learning in multichannel wireless sensor networks," Wireless Communications and Mobile Computing, 2017

[120] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," IEEE Robotics and Automation Letters, 2(4), pp. 2096–2103, 2017.

[121] R. Polvara, M. Patacchiola, S. Sharma, J. Wan, A. Manning, R. Sutton, and A. Cangelosi, "Autonomous quadrotor landing using deep reinforcement learning," arXiv preprint arXiv:1709.03339, 2017.

[122] T. Chen, Z. Wang, G. Li, and L. Lin, "Recurrent attentional reinforcement learning for multi-label image recognition. arXiv preprint arXiv:1712.07465, 2017.

[123] D. Burke, D. Jenkus, I. Qiqieh, R. Sha, S. Das, and A. Yakovlev, "Special session paper: Significance-driven adaptive approximate computing for energy-efficient image processing applications," In Hardware/Software Codesign and System Synthesis (CODES+ ISSS), International Conference on, pp. 1–2, 2017.

[124] F.C. Ghesu, B. Georgescu, Y. Zheng, S. Grbic, A. Maier, J. Hornegger, and D. Comaniciu, "Multi-scale deep reinforcement learning for real-time 3D-landmark detection in CT scans," IEEE

Transactions on Pattern Analysis and Machine Intelligence, 2017.

[125] A. Achille and S. Soatto, "Information dropout: Learning optimal representations through noisy computation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018.

[126] G. Williams, N. Wagener, B. Goldfain, P. Drews, J.M. Rehg, B. Boots, and E.A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," In Robotics and Automation (ICRA), IEEE International Conference on, pp. 1714–1721, 2017.

[127] J.T. Wilkes and C.R. Gallistel, "Information theory, memory, prediction, and timing in associative learning," Computational Models of Brain and Behavior, pp. 481–492, 2017.

[128] Y. An, Y. Wang, and H. Meng, "Multi-task deep learning for user intention understanding in speech interaction systems," 2017.

[129] X. Shi, Z. Gao, L. Lausen, H. Wang, D.Y. Yeung, W.K. Wong, and W.C. Woo, "Deep learning for precipitation nowcasting: A benchmark and a new model," Advances in Neural Information Processing Systems," pp. 5622–5632, 2017.

[130] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," arXiv preprint arXiv:1708.05866, 2017.

[131] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," IEEE transactions on neural networks and learning systems, vol. 29, no. 6, pp. 2042–2062, 2017.

[132] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," Journal of Machine Learning Research, vol. 16, no. 1, pp. 1437–1480, 2015.

[133] V. S. Borkar, "Learning algorithms for risk-sensitive control," in Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems–MTNS, 2010, vol. 5.

[134] A. Gosavi, "Reinforcement learning: A tutorial survey and recent advances," INFORMS Journal on Computing, vol. 21, no. 2, pp. 178–192, 2009.