Reinforcement Learning based Resource Management for Fog Computing Environment: Literature Review, Challenges, and Open Issues

Hoa Tran-Dang, Sanjay Bhardwaj, Tariq Rahim, Arslan Musaddiq, and Dong-Seong Kim

***Abstract*—In the IoT-based systems, the fog computing allows the fog nodes to offload and process tasks requested from IoT-enabled devices in a distributed manner instead of the centralized cloud servers to reduce the response delay. However, achieving such a benefit is still challenging in the systems with high rate of requests, which imply long queues of tasks in the fog nodes, thus exposing probably an inefficiency in terms of latency to offload the tasks. In addition, a complicated heterogeneous degree in the fog environment introduces an additional issue that many of single fogs can not process heavy tasks due to lack of available resources or limited computing capabilities. Reinforcement learning is a rising component of machine learning, which provides intelligent decision making for agents to response effectively to the dynamics of environment. This vision implies a great potential of application of RL in the concept of fog computing regarding resource allocation for task offloading and execution to achieve the improved performance. This work presents an overview of RL applications to solve the resource allocation related problems in the fog computing environment. The open issues and challenges are explored and discussed for further study.**

***Index Terms*—Fog computing, machine learning, performance improvement, reinforcement learning, resource allocation, task offloading, task scheduling.**

I. INTRODUCTION

A. Context and Motivations

THE Internet of things (IoT) paradigm has been recognized as a key driving force to realize the smart concept in various domains such as smart cities [1], smart grids [2], smart factories [3] since it enables the interconnection and interoperability of IoT-enabled physical and virtual entities to create smart services and informed decision makings for monitoring, control, and management purposes [4], [5]. The underlying principle of realization involve a set of activities that includes collecting, processing, analyzing, and getting insights from IoT data perceived by the IoT devices. Traditionally, the cloud computing platform plays an essential role in the realization process since it provides rich and powerful resources (e.e., storage, computation, networking) to handle an enormous amount of IoT data (big data) efficiently [6]. However, the data traffic has increased exponentially due to the increase of IoT-enabled devices and growth of customized applications, thus leading to congested networks consequently. Some of the leading IoT applications have put higher demand on resource-constrained devices. Additionally, more stringent quality of service (QoS) requirements of IoT service provisioning such as (ultra)low delay expose crucial limitations of the cloud based solutions because the delivery of data from the IoT devices to the centralized cloud computing servers seriously affects the performance in processing, analyzing data and results in network congestion issues and excessive delay as an ultimate consequence. This fact context leads to a strong push of fog computing integration into the IoT-cloud systems since it puts computing, storage, communication, and control closer to the IoT devices to meet the prescribed QoS requirements [7], [8]. Technically, the fog computing platform that is placed between the physical IoT devices and the cloud servers can handle a majority of service requests on behalf of the cloud servers to improve the system performance in terms of service delay, workload balancing, and resource utilization [9].

The mutual benefits gained from the combination of fog and cloud enable the resulting IoT-fog-cloud systems to provide uninterrupted IoT services with various QoS requirements for the end users along the things-to-cloud continuum. However, employing the fog computing raises another concern regarding decisions whether the tasks should be processed in the fog or in the cloud. There are many factors impacting on the offloading decision policies such as offloading criteria, application scenarios [9]. Basically, in the most of existing offloading techniques the tasks are probably offloaded to the best surrogate nodes, which have the most ample resources (e.g., large storage capacity, high speed processing) and reliable communication network conditions in terms of delay, bandwidth between them and their neighbors, the IoT devices, and even the cloud servers. However, such the fog offloading solutions face significant challenges regarding the workload distribution among the complicated heterogeneous fog devices characterized by different computation resources and capabilities. The challenge is further amplified by increasing the rates of service

Manuscript received August 5, 2021; approved for publication by SuKyoung Lee, Division III Editor, October 24, 2021.

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2020-2020-0-01612) and supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation), Priority Research Centers Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2018R1A6A1A03024003), and Korea Research Fellowship Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2020R1I1A1A01073019).

H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D.-S. Kim are with department of IT Convergence Engineering, Kumoh National Institute of Technology, Korea, email: {hoa.tran-dang, arslan, dskim}@kumoh.ac.kr, sanjay.b1976@gmail.com, tariqrahim@ieee.org.

D.-S. Kim is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2021.000041

1229-2370/21/$10.00 © 2021 KICS

requests, which probably make the task queues of resource-rich fog nodes longer. As a result, the requirements of latency-sensitive applications can be violated because of excessive waiting time of long queue. Furthermore, the reliance on the remote cloud servers to accomplish the tasks may not help in improving the situation due to high communication delay or networking related disturbance.

Executing the tasks in the fog computing tier requires an efficient resource allocation and management to satisfy the QoS requirements. However, to achieve the objective is facing many critical challenges due to the complex heterogeneity and limitations of fog resources, locality restrictions as well as dynamic nature of resource demands. Most of heuristics existing algorithms are proposed as efficient resource allocation solutions for distributing and executing the tasks in some certain computing scenarios [10], [11]. In other word, they lack a generic framework to study the resource allocation issues in the practical computing context, which encompasses multiple criteria to derive the efficient algorithms such as heterogeneity, QoS management, scalability, mobility, federation, and interoperability [12].

RL has been increasingly studied and applied to effectively solve the resource allocation related issues in many uncertain computing environments [13]. In principle, RL-based algorithms employ the learning processes to learn the dynamic and changing environment to enrich the experiences, thus deriving the best decisions in the long-term operations [14]–[16]. For example, a RL-based task scheduling algorithm has been developed and deployed in the cloud computing scenarios to reduce the average task execution delay, and task congestion level [17]. Many RL-powered algorithms have been proposed to improve the computing performance such as saving the energy consumption in the data centers [18]–[20]. Recently, the collaboration between deep learning and RL can further enhance the capabilities of resulting deep RL (DeepRL) approaches which achieves outstanding performance in complex control fields strongly shows its superiority of decision making in complex and uncertain environment [21], [22]. For example, such a DeepRL algorithm is developed to optimize the task scheduling in the data center [23]. The resource allocation problem is also tackled by using a DeepRL algorithm to achieve the optimal job-virtual machine scheduling in the cloud [24]. The dynamic task scheduling problems are effectively addressed by RL-based techniques in the field of computer science [25]. These primary findings expose an efficient alternative to solve the resource allocation problems in the fog computing using the RL concept. In this regard, this paper provides a significant review to channelize the state-of-the-art RL based methods to solve the resource allocation problems in the fog computing environment.

B. Contributions

The main contributions of paper are summarized as follows:

- We highlight key issues regarding the fog resource management in the fog computing for task offloading and task computation algorithms.
- We examine the RL concept as potential solutions to the resource management issues.
- The state-of-the-art review of existing applications of RL in the fog computing environment is surveyed.
- We explore and discuss the challenges as well as associated open issues when using RL algorithms in the context of fog computing.

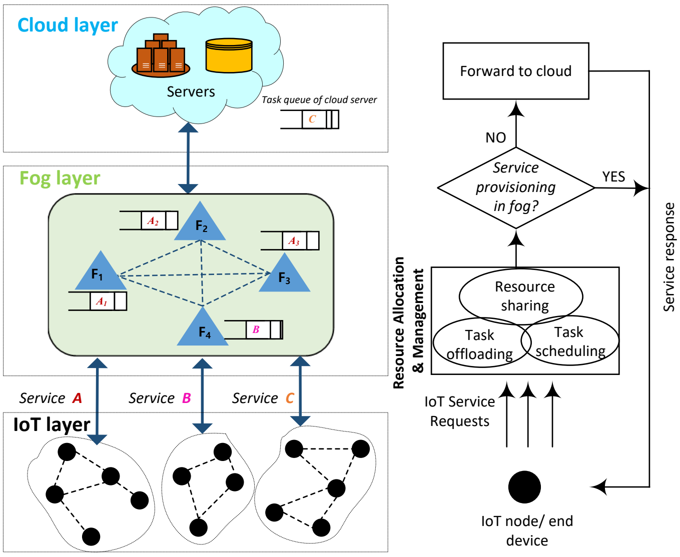

Fig. 1. A typical three-tier architecture of IoT-fog-cloud system, which provides specific kinds of IoT services (e.g., service A, B, C as examples) through either the fog layer or the cloud based on the adaptive resource allocation mechanism.

C. Paper Structure

The rest of paper is structured as follows. Section II present the key concept of fog computing and existing resource management issues of fog nodes. Section III overviews the principle of RL and key algorithms developed and used in the practical applications. Section IV presents a comprehensive review of existing works which apply RL algorithms in the context of fog computing. Section V discuss the challenges and open issues. Finally, Section VI concludes the work.

II. Fog Computing Environment

A. System Model

A fog computing system is usually placed between an IoT layer and a cloud layer to form a three-tier architecture for the resulting IoT-fog-cloud system as illustrated in Fig. 1.

The first layer include all IoT-enabled devices that generate IoT data, and/or primarily process the data, and periodically reports the raw and/or pre-processed data to the fog or cloud for further advanced processing, and computing (i.e., data streaming applications). The fog computing devices (e.g., routers, switches, gateways, and cloudlets for mobile computing services) and servers distributed in the fog layer and cloud layer, respectively are responsible for receiving, processing,

TABLE I
RESOURCE STATE TABLE OF NEIGHBORS OF FOG NODE F_1.

Node ID	Fog specification & Resource Status				
	M	f	Processor	RTT	W
F_2	200	10	CPU	2.5	350.2
F_3	100	5	CPU & GPU	3.1	500
F_4	400	2.5	GPU	4.8	239.1

and responding to the IoT computing service requests sent from the IoT layer. In contrast to the powerful servers in the cloud tier, the fog devices have limited resources. In addition, they are heterogeneous in terms of computation, storage, and communication capability. Therefore, these resources require an efficient resource allocation strategy to improve the performance of fog computing tier in serving and delivering the services.

Depending on the specific objectives and scale of application system, the fog layer can be structured by three main architecture: centralized, distributed and hierarchical form. In the first architecture, the computing system comprises of a fog controller and a set of fog nodes [26]. The controller is responsible for information aggregation and decision-makings, whereas the nodes work directly to serve as the supportive computing devices. Such the architecture is widely applicable in the software-defined networks. The second one is referred as to a network of fogs which forms connectivity among fogs in a distributed manner. Task scheduling and resource allocation in the task offloading processes can be decided by fogs through distributed manners. Whereas in the hierarchical architecture, there exists clusters in which each cluster operates according to the master-slave operations. Specially, a fog node serves as a master to control and manage the operations of associated fog nodes known as slaves. The master is able to know the states of slaves regarding the resources, capacity, thus it can derive optimal resource allocation and task scheduling in its cluster. In addition, a federation is enabled among the master fogs for further resource and load sharing.

Regardless architecture, the systems rely on available resource tables that contain the updated resource states of fogs in the systems to facilitate the resource allocation processes. Depending on the specific architecture of fog systems, the tables are maintained by different responsible fogs. For example, the controllers and fog masters are able to know the resource state of all fog devices, and in their clusters, respectively in the first two architecture. Whereas, in the distributed system each fog maintains it own neighbor resource table containing the updated information about the available resources of its colony [27], [28]. These tables are shared among the member of their colony to support the primary host to make offloading decisions, which ultimately aim at selecting the offloadees, the hosts, and the collaborative fogs. Table I shows an example of neighbor resource table stored by the fog node F_1, which records the resource states of neighbors with respect to residual memory (M), clock frequency (f), processor (i.e., CPU or GPU support), round-trip time (RTT), and expected waiting time in queue (W).

Each computing task k can be described with a tuple (A_k, B_k), where A_k and B_k are the vectors representing the input data features and required computing resources, respectively. Basically, A_k can include following features: Total size (in bits or bytes), splittable or non-splitable, number of data types. The sizes of input data of tasks can be ranged from kilobytes to tera-bytes depending the specific applications [29]. Based on this feature, the tasks can be classified into light, medium, and heavy tasks as studied in many of existing works [28], [30] for further analyzing the impact of task sizes on the performance of fog computing systems. The divisibility of task (i.e., input data) is also investigated in the literature. Accordingly, the whole input data can only be processed by a single fog device as it is unable to be splitted into data subsets. Whereas, several existing works assumed that the task can be divided into subtasks with smaller data sizes. Such the task division is employed to get benefit from parallel computing since the substasks can be processed by different fog devices simultaneously. Fig. 2 illustrates main subtasks for computing the input data a as it can be divided into three independent subsets $\{a_1, a_2, a_3\}$.

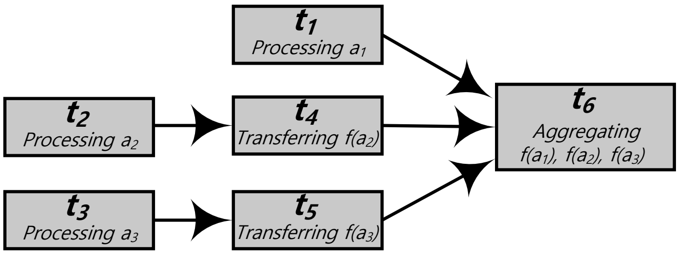

Fig. 2. Based on the data fragmentation concept, processing the input data a includes a set of tasks, which distributively process the data subsets a_1, a_2, a_3 before completing the whole task by jointly processing the output data $f(a_1), f(a_2)$, and $f(a_3)$ to achieve $f(a)$.

In particular, the outputs of substasks (i.e., $f(a_1)$, $f(a_2)$, and $f(a_3)$) are collected and jointly processed in the final stage to achieve the desired outcome (i.e., $f(a)$). This mechanism is called partial offloading as investigated in the existing works such as [31]. The input data of a certain computing task can include multiple types of data such as text, image, video, and audio as studied in [27], [32]. Regarding the required computing resource, there are many attributes included in D_k to process the task. Some of existing works just only consider B_k as the central processing units (CPU cycles) [33]. In another scenarios, GPU and memory requirements are considered during resource allocation for executing heavy and complex task such as the machine learning algorithms [34].

B. Resource Allocation Problems in Fog Computing Environment

The fog computing technology provides additional resources for executing the tasks and correspondingly providing various services with improved quality in the computing systems. However, the nature of fog computing environment exposes major challenges regarding the resource allocations for task execution to achieve the objective. This section explores and

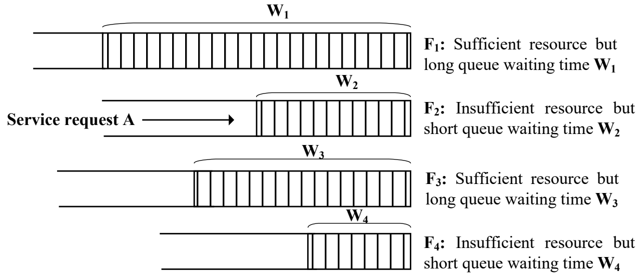
W₁
F₁: Sufficient resource but long queue waiting time W₁
W₂
Service request A
F₂: Insufficient resource but short queue waiting time W₂
W₃
F₃: Sufficient resource but long queue waiting time W₃
W₄
F₄: Insufficient resource but short queue waiting time W₄

Fig. 3. The heterogeneity and unbalanced workload of fog environment in the IoT-fog-cloud systems expose issues in providing IoT services.

discusses such key challenges in the fog computing, which urge a need to develop alternative solutions beyond the existing heuristics propositions.

Firstly, the fog computing environment is complex and dynamically changing. Basically, the fog computing devices like gateways, hubs, and switches are heterogeneous in terms of computation, storage, and communication capability. In many of use cases, the complicated heterogeneity of fog environment represents a critical challenge for these systems to achieve the performance improvement objective. For instance, the presence of limited resource fogs results in the imbalance of workload distribution among the fogs, which in turn impacts negatively on the performance of system in terms of delay. Fig. 3 illustrates such typical issue in the fog computing environment systems, in which a fog node F_2 is unable to process the whole input data of service request A due to lack of resources. Meanwhile, offloading the task to the fog neighbors F_1 and F_3 may lead to an extensive delay since there are high workloads in queues of these fog nodes.

In addition, the high rate of requests potentially prolong the task queues in the powerful fog nodes since the limited resource fog nodes with respect to the computational capability, and storage may be unable to process the whole input data of service. Furthermore, constrained by the cloud rental cost, the other objective of proposed framework is to maximize the usage of fog resources, thus minimizing the reliance on the cloud. All these perspectives lead to a direction to explore the task division concept, that potentially can help in reducing the task execution delay through parallel subtask executions in the limited resources fogs. However, dividing tasks may not be effective in the large-scale systems or high task request rate since it may increase the resource contention among the fogs, thus increasing the time and computation complexity of algorithms [27].

Secondly, the fog computing resource is dynamically changing due to mobility. In many practical applications, the presence of mobile fog nodes such as drones [35] and vehicles [36], [37] leads a dynamic change of fog computing resources over time. Moreover, leaving out and joining in the fog computing systems by fog devices are accounted for as causes for this change. Therefore, the resource allocation strategies must be designed in an adaptive and flexible way to cope with this situation.

Thirdly, the resource requirements for executing the tasks also change dynamically due to a various types of tasks. The demand for resources varies according to specific applications, different time periods, and environmental condition. Generally, there are three major computing problems involving the resource allocation in fog computing systems which include resource sharing, task scheduling, and task offloading [12]. Basically, resource sharing is referred as to methods to share available resources among the fog devices to execute the computation demands of tasks [38], [39]. This algorithm is essential in the fog computing environment, where the heterogeneity of fog resources stress the need of multiple computing devices to complete a single task. As the fog computing system is considered as a pool with a set of resource types such as storage, CPU, and GPU the resource sharing requires a cooperative among the nodes to execute the computational task demands. Therefore, mechanisms for fog-to-fog collaboration must be established within the fog computing domains to facilitate the resource sharing [40]. However, to enable such the cooperative fog-to-fog is a challenging job since the practical devices are different in terms of hardware, software, and functionalities such as gateways, routers, and hubs. Specially, task scheduling is usually performed without sufficient information support. Practically, there are no patterns to predict the arriving task profiles characterized by number and size of tasks as well as arrival rate. Therefore, the algorithm has to schedule the tasks at once without any prior experience and prepared information. In addition, the scheduling algorithm needs automatically optimize resource utilization based on the changing demand.Task scheduling is a problem, which have a high impact on the performance of computing system, especially in terms of task execution delay. Generally, scheduling the tasks involves assigning which resources process which tasks within which expected time period. In the large-scale systems including IoT layer, edge and fog layer, and cloud layer possible resources for computation execution include IoT devices, edge devices, fog devices, and also server in the cloud tier. The heterogeneity of fog resources in term of computation capabilities directly lead to the imbalance of workload distributed among the nodes. Concretely, more loads may be carried by powerful devices such as fast processing speed from greedy perspectives. As there exists a lack of management the performance of computing systems can be degraded in the long run since a large available resource in the limited resource fogs are underutilized. Balancing the workload among the computing devices is a challenging problem to ensure a stable operation of computing systems in a long run. There are many factors needed to be considered during designing and developing efficient task offloading algorithms such as resource states of fog devices, and task requirements, which dynamically change over time.

In order to solve the above problems, a lot of related studies are carried out. Most of them focus on some specific scenarios or rely on the acquired details of the incoming tasks in advance to derive efficient heuristics algorithms. In addition, most of the previous studies are single objective optimization-oriented (i.e., delay, energy consumption, and resource utilization). In particular, none of them have a generic framework to model and examine the above mentioned challenges for further de-

TABLE II
SYMBOLS USED IN THE WORK.

Symbol	Description
t	Time step t
a_t	Action that agent can take at t
s_t	State that agent is be in at t
r_t	Immediate reward for agent to take a_t
A	Action space, $A = \{a_t\}$
S	State space, $S = \{s_t\}$
R	Return or cumulative reward
Q	Q value (function)
π	Policy
γ	Discount factor
P	Transition probability

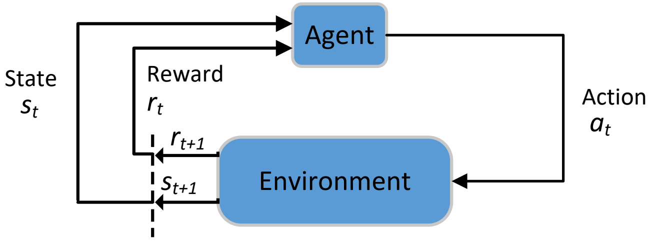

Fig. 4. The fundamental operation of RL with agent, environment, reward, state and action.

signing appropriate algorithms. In the next sections, we briefly introduce the principal concept of RL and conduct an analysis review on application of RL-based methods for solving the resource allocations in the fog computing environment.

III. Reinforcement Learning

A. Basic Concept

RL is a method of supervised and unsupervised learning. It is not exclusively supervised because it does not depend just on a training data, but it is not unsupervised because rewards or punishment are given to the agent in return of the optimization. In the RL algorithms, the agents are able to identify the best as well right actions in any situations so as to achieve the overall goal based on the rewards and punishments. In other words, RL can be in simplest terms defined as the "science of decision making" [41].

For the sake of clarity, Table II provides symbols mostly used in the RL frameworks.

In the standard RL model, the agent interacts with its environment to learn the characteristics of environment. The data perceived through the environment sensing serves as input for the agents for decision makings. The actions taken by the agent results in the change of environment and which is further communicated back to the agent for entire the process to start over again. Fig. 4 illustrate the basic operation of RL.

At every time step t of interaction, the agent makes a partial observation of the state (s_t) of the world and then decides to take an action (a_t) [42]. The environment changes when the agent acts on it or it may change on its own. The agent also perceives a reward (r_t) from the environment, which tells it how good or bad the current sate is. The ultimate goal of the agent is to maximize its cumulative reward, called as return. Therefore, RL methods are the ways that the agent can learn behaviors to achieve its goal [43]. The followings define fundamental concepts and notations used to describe the RL problems and algorithms.

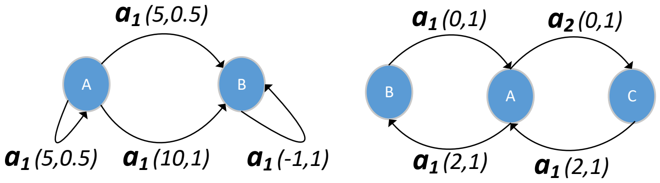

Fig. 5. Two simple MDP's that illustrate the underlying concepts and algorithms. [46]

1) State Space: Interaction with the dynamic environment trains the RL system by trail and error as it learns mapping from situations to actions. The RL system must be able to perfectly observe the information provided by the environment which influences the action to be taken, thus the true states of the environment affects the action taken by the RL system. In the most of environment, the state transition is following a Markovian property that, the current state s_t provides enough information obtain an optimal decision. Therefore, a selection of an action a will have same probability distribution over next states when this action is taken in the same state [44]. Markov decision process (MDP) is a mathematical framework used to model decision making situations in the RL problem. MDPs consists of states, action, transition between states and a reward function. The system is Markovian if the results of an action does not depend on the previous action and historical already visited states, but only on the current state, i.e. $P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}) = P(s_{t+1}|s_t, a_t)$ [45]. Often MDPs are depicted as a state transition graph where the nodes correspond to states and (directed) edges denote transitions. Two simple MDP's as illustrated in Fig. 5 show the underlying concepts and algorithms, in which each transition is labeled with the action taken and a pair of numbers. First number is immediate reward and second number represents the transition probability.

2) Action Space: An action taken by the agent is completely dependent on the environment, therefore different environment results in different actions. The set of all valid actions in a given environment is called as an action space abbreviated as S [47]. Some environment such as Atari and Go have discrete action spaces, where only finite number of actions are available to the agent [48]. Other environments have continuous action spaces, such as, where an agent controls a robot in a physical world [49].

3) Reward and Return: As the agent perform an action a_t, it will immediately get a reward r_t. The immediate reward r_t is quantified by a numerical value (either in positive or negative) to evaluate the desirability of action that the agent took. Therefore, the goal of agent is to maximize the total amount of rewards or the cumulative reward instead of immediate rewards.

The return or commutative reward at time step t is defined as:

$$R_t = \sum_{k=0}^{\infty} r_{t+k+1}. \tag{1}$$

However, to account for the importance of immediate and future rewards, the return is discounted by a discount factor γ ($0 \leq \gamma \leq 1$), thus:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{2}$$

4) Policy: Policy is defined as the rule used by the agent to decide which action to take in each state. In other words, a policy is a function mapping states to actions. Therefore, the ultimate objective of RL algorithms is to derive the optimal policy that maximizes the return.

5) State Value and State-action Value Function: A state value function ($V^{\pi}(s)$) is used to specify how good it is for an agent to be in a particular state (s) with a policy π. In the mathematical formulation, $V^{\pi}(s)$ is defined as:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s]. \tag{3}$$

A state-action value function or Q-function ($Q^{\pi}(s,a)$) is used to specify how good it is for an agent to perform a particular action (a) in a sate (s) following a policy π. The mathematical formulation of Q function is as follow:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a]. \tag{4}$$

Since the environment is stochastic, there is a probability denoted as $\pi(a|s)$ for a policy π to take an action a given the current state s. Certainly, $\sum_{a \in A} \pi(a|s) = 1$. The relationship between $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ is expressed by:

$$V^{\pi}(s) = \sum_{a} \pi(a|s) Q^{\pi}(s,a). \tag{5}$$

By denoting $P^a_{ss'}$ as transition probability to transit from a state s to a state s' as performing an action a, the relationship between $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ is also expressed by:

$$Q^{\pi}(s,a) = r(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} V_{\pi}(s'), \tag{6}$$

where $r(s,a)$ is the immediate reward achieved after taking action a given the current state s.

Consequently, the state values can be formulated as:

$$V_{\pi}(s) = \sum_{a \in A} \pi(a|s)(r(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} V_{\pi}(s'). \tag{7}$$

Similar for Q functions, we have:

$$Q_{\pi}(s,a) = r(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} \sum_{a' \in A} \pi(a'|s') Q_{\pi}(s',a'), \tag{8}$$

where a' is next possible action of a.

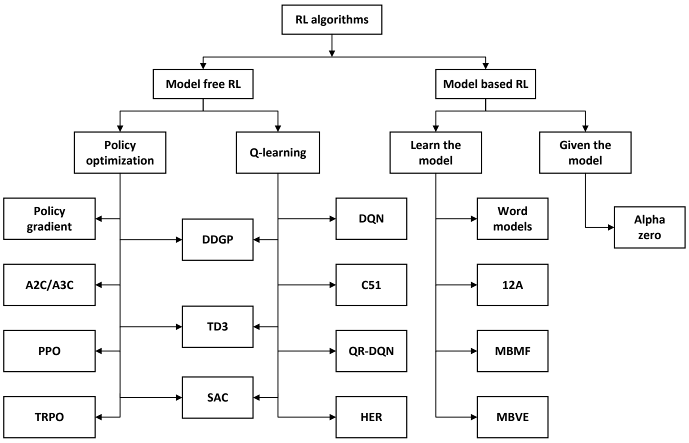

Fig. 6. Taxonomy of RL algorithms.

In order to achieve the maximal return, the RL algorithms have to find the optimal policy, that has an associated optimal state value function or state-action value function. Mathematically, the optimal policy π is found as:

$$V_*(s,a) = \underset{\pi}{\max} V_{\pi}(s), \tag{9}$$

or

$$Q_*(s,a) = \underset{\pi}{\max} Q_{\pi}(s,a). \tag{10}$$

The next sub-section is to review the key algorithms and taxonomies used in the literature to find the optimal policies for different scenarios.

B. Taxonomy of RL Algorithms

Broadly, RL can be classified into two categories including model-free and model-based methods [50]. A non-exhaustive list of RL based algorithms in these two classes is presented in Fig. 6.

In the model-based model, the agent is supported to make plan as it can see a range of future possibilities of choices and thus deciding between its option well ahead. Thus the agent can filter out the results into a learned policy. For example, the authors of [51] used this approach and called it as AlphaZero (AZ), where in the sample efficiency improved significantly over other methods which were also not having models. However, this approach exposes shortcomings, as the learning by the agent of the model is based only on the experience which itself creates many challenges. The biggest one is that of bias which can be exploited by the agent thereby forcing the agent to perform below par in real environment, secondly it is very computation intensive, which can ultimately results in failure to pay sometime.

In contrary, the model-free algorithms do not need "model" as a result the sample efficiency is lower but they are easier in implementation and tuning, which makes them quite popular than its counter-part. The algorithms in this type can be further divided into based on the learning to be carried out. Accordingly, there are two types of learning carried out. The first one is policy optimization in which the parameter θ is optimized for a policy $\pi_\theta\left(\cdot|s_t\right)$ based on the recent collection of the data. Some of key examples of this optimization are A2C/A3C, [52], where performance is maximized by suing gradient ascent and another is PPO [53], where a surrogate objective function is used as a indirect measure of the performance. The second learning method is Q-Learning in which learning of an approximator $Q_\theta(s,a)$ gives an optimal function $Q^*(s,a)$. Objective function for Q-Learning approach is based on the Bellman equation [54]. Recently this approach is used by in [55] called as DQN which is milestone for deep RL and [56] called as C51, where the returns are learned leading to the policy expectation as Q^*. In particular, there are some certain algorithms used simultaneously policy optimization and Q-Learning to compromise the strengths and weakness of either sides. For example, the authors of [57] proposed DDPG which simultaneously learns policy as well Q-function. In addition, the proposition in [58] proposed a combined approach of SAC and Q-learning with the help of stochastic policies and entropy regularization, thereby giving higher scores.

The model based RL are not well defined methods as models can be used in many orthogonal ways. Broadly they can be classified, based on whether model is given or the model is learned. The learning of the model based approach is used by [59] and called it as MBMF where pure planning technique, model predictive control is used in the selection of the action on some standard benchmark tasks for deep RL. The given model approach is used by [51], called its as AZ in which explicit representation of the policy is learned with the pure planning which produces an action that have strong influence as compared to when policy alone would have produced.

IV. RL-based Algorithms for Resource Allocation in the Fog Computing System

This section summarizes key RL-based algorithms in the literature to address the resource allocation problems in the fog computing environment, which are discussed according to key types, namely Rl-based and DRL based methods. In particular, the review analysis is conducted to emphasize on describing how the components of RL-based solutions in the fog computing such as state space, action space, MDP, and reward are formulated.

A. Resource Sharing and Management

Considering the fog computing systems as a resource pool with multiple kinds of resources (i.e., CPU, GPU, storage, and memory), the resource sharing is an important mechanism to allocate the resource efficiently. In principle, the resource sharing algorithms requires the collaboration of fog entities for exchanging their available resource states, thus facilitating the resource allocation.

The work [60] studies the resource management for conserving the energy consumption in the Fog Radio Access Networks (F-RAN), which can provide two main types of services: communication and computation. Considering the dynamic of edge and fog resource states, the network controller is able to make the fast and intelligent decisions on the user equipment (UE) communication modes (i.e., C-RAN (Cloud-RAN) mode, D2D mode, and FAP (fog access point) mode) and the on-off states of processors to minimize the long-term power consumption of systems. The well-trained DRL model is built on the system architecture to derive this optimal decision. In this model, the state space is defined as $S=\{s=(s^{processor},s^{mode},s^{cache})\}$, where $s^{processor}$ is a vector representing the current on–off states of all the processors, s^{mode} is a vector representing the current communication modes of all the UEs, and s^{cache} is a vector consisting of the cache state at each D2D transmitter. The network controller is able to control the on-off state of a processor and communication mode of a UE each time step. Thus, to reduce the number of action, the action space is defined as $A=\{a=(a_{processor},a_{mode})\}$, where $a_{processor}$ indicates "turn on" or "turn off" action for a processor, and a_{mode} represents to change the communication mode for a UE (i.e., C-RAN, FAP, or D2D). To achieve the green F-RAN, the reward function is defined as the negative of system energy consumption is incurred by operation of processor in the cloud, fronthaul transmission, and wireless transmission in the fog tier. To enhance the performance of proposed algorithm, multiple factors are developed and integrated in the DRL model. Firstly, the prioritized replay is proposed to reuse the experienced transition more effectively. Secondly, double DRL is used to overcome the optimistic Q-value estimation as well as improve the learning process in cases of environment change. In particular, the transfer learning is integrated to accelerate the learning process, thus allowing the quick convergence of learning. These key factors result in the superiority of proposed algorithms compared to the related and baseline works.

Internet of vehicles (IoV) where all vehicles are connected has emerged as a fundamental element of smart cities using real-time data to react instantly to user requests. The vehicular fog computing (VFC) has appeared as an emerging paradigm that facilitate the dynamic problems of networking, caching, and computing resources to enhance the efficacy of next-generation vehicular networks [61]. In these networks, the vehicles both in movements or parked status equally sever as fog nodes, which have limited resources for offering the services such as communication, computation, and storage. Considering the immense dynamic and highly complicated nature of VFC environment, to enhance the QoS such as real-time response is of the challenging job. This sort of problem in the vehicular applications is investigated in [62], which aims at seeking efficient resource allocation strategies to minimize the service latency minimization. The Rl algorithm accordingly is developed to achieve the target that employed an LSTM-based DNN to predict the movement and parking status of vehicles, thus facilitating the resource assignment. In addition, the proposed RL technique uses the latest techniques i.e., proximal policy optimization technique, which has the ability to learn continuously the dynamic environment, and therefore can adjust to decide the resource allocation correspondingly.

It is a significant challenge to present high quality, low bit-rate variation, and live streaming assistance for vehicles because of the dynamic characteristics of wireless resources and channels of IoV. A unique transcoding and streaming system for the maximization of video bit-rate and reduces bit-rate variance and time-delays in VFC powered IoV is presented. The scheme jointly optimizes the scheduling of vehicles, selection of bit-rate, and spectrum/computational resource allocation as an MDP problem. A deep RL algorithm i.e., soft actor-critic based on the highest entropy frame is employed for the solution of MDP [63]. Moreover, an asynchronous advantage actor-critic (A3C) RL-based cooperative resource allocation and computation offloading frame for vehicular networks is presented [64].

In another VFC application, a resource sharing scheme for supporting task offloading in terms of VFC is presented in [65]. In this model, the incentivization of vehicles is performed upon sharing the resource of idle computing over dynamic pricing. In the particular case, task priority, availability of service, and mobility of vehicles are comprehensively acknowledged. A MDP diagram is formulated for task offloading due to the dynamic vehicular environment that aims to maximize the average latency-aware use of tasks in a time. Based on the DRL method, a soft actor-critic is developed for the maximization of the policy of entropy and anticipated reward. Moreover, a mobile network operator (MNO) preference and switching problem is formed by simultaneously analyzing switching cost, various prices that can be charged by diverse MNOs and fog servers, and quality-of-service alterations within MNOs. A switching policy that is based on a double deep Q network (DQN) is presented proving to reduce each vehicle's long-term mean cost with promising reliability and latency performance [66]. Similarly, modeling of optimal computational offloading policy as MDP problem while considering ultra dense system for mobile edge computing (MEC) is performed. A deep Q-network based on the RL algorithm as a computation offloading method is presented to overcome the large dimensionality that will determine the optimum policy for dynamic statistics and no prior knowledge [67]. A semi-MDP is formulated for the optimum and agile framework of resource slicing that simultaneously allocates the storage, radio resources, and computing of the network provider to various slice requests. Dual NN of Deep Q-learning method is implemented that improves the performance by outperforming other RL-based approaches for network slicing management [68].

B. Task Scheduling

Overall, in the IoT-fog-cloud systems, task scheduling is referred to as making decisions on which tasks are processed by the IoT layer, the fog layer, or the cloud layer to achieve the target design objectives [12], [69]. In the most of applications, the main objective of scheduling algorithms is to minimize the task execution delay. However, an efficient scheduling design may improve other system performance indicators simultaneously such as reduced energy consumption, and balanced workload distribution.

To minimize the long-term service delay and computation cost for the fog computing systems under task deadline and resource constraints, a double deep Q-learning (DDQL)-based scheduling algorithm is introduced in the work [70]. Considering a fog-based IoT system with hierarchical architecture, the work aims at developing schedulers (also known as agents in the RL algorithm) each of which is embedded in a corresponding gateway (GW) to allocate resources (i.e., virtual machines (VM) embedded in the fog nodes, and the cloud servers) for executing tasks. To reduce the dimension of RL-based algorithm, paths with the updated resource states are modeled as state space of system as $S = \{s(t)^i_j = (uCPU^i_j, uMemory^i_j, uStorage^i_j)\}$. In this formula, three elements represents the resource utilization of $path_i$ in term of CPU, memory, and storage, respectively at the moment t that $task_j$ arrives at. The agent (i.e., scheduler) is responsible for assigning a certain resource (i.e., VM in fog or cloud) to process the task through the action. Thus, the action space is formulated as $A = \{a^i_j | 1 \leq a^i_j \leq vmn_i\}$, where a^i_j is the action that is taken by the $agent_i$ and for a $task_j$, and vmn_i is the total number of VMs in $path_i$. To obtain the end-to-end (E2E) service delay, the immediate reward function is defined as $IR^i_j(a) = 1/nSD^i_j$, where $IR^i_j(a)$ is the immediate reward after taking action a for the $task_j$ in $path_i$, and nSD^i_j is the normalized service delay of $task_j$ in $path_i$. nSD^i_j accounts for waiting time delay in $path_i$ to get VM_a, execution delay by VM_a, transmission, and propagation time of $task_j$. In this model, the objective to achieve the minimized E2E delay is enabled since the agent tries to maximize the cumulative reward through efficient action selection. In particular, to select the optimal action the double DQL concept is introduced in the algorithm in which each agent is supported by two separate models as Q_1 and Q_2 for action selection, and Q-value calculation, respectively. With two Q values, the agents are able to reduce the probability of taking valid and inefficient action, thus accelerating the learning process. For evaluating the performance of framework, the work firstly creates a simulation environment with multiple agents, state space, action space, and reward functions. Then, the proposed DDQL-based scheduling algorithm is applied in this environment to assign appropriately which fog nodes will process which tasks in order to achieve the objectives. In particular, the target network and experience replay mechanisms are integrated into the DDQL-based scheduling policy to cease the fluctuation of results.

In the data-driven IoT-based systems, the end devices or IoT sensors constantly generate online tasks, which requires the upper layer such as fog computing or cloud to process within the deadlines. The nature of online tasks exhibits critical challenges for the system to conduct the task scheduling since there exists an inherent lack of prior information relating to the task arrivals. The issues stress the need for adaptive scheduling solutions, which have been investigated and developed in the literature using the RL principle. In particular, the DRL-based approaches exhibit many effectiveness to deal with the situation of online task scheduling. For example, the work [22] designed an efficient scheduler based on a forward neural network (FNN), which is able to schedule n online tasks at a time to reduce the overall task slowdown. Although the algorithm is well performed in case of predetermined n, it exposes the limitation in terms of flexibility since adjusting n may lead to adverse performance of system. In the same method, the RL-based model in [71] is designed to make the scheduling decision for each arrival task. However, such method is well applied in the cloud environment in stead of fog computing since it can induce considerable overheads. The work presented in [72] reveals these aforementioned limitations, and proposes a neural task scheduling (NTS) to release them. In principle, NTS adopts the model of recurrent neural network (RNN) based on the pointer network [73] to obtain more flexible output sequences. In addition, the network is integrated by the long short-term memory (LSTM) techniques, and attention mechanism [74] to enhance the flexibility and learning efficiency when handling long sequences of tasks. From the RL design perspective, the space state S is modeled as the system state in time slot t represented by all n_t pending tasks with their characteristics (i.e., resource demands of tasks, and execution durations), and amount of remaining resources in future M time slots. In mathematical form, $S = \{s_t\}$, where s_t is a matrix of size $n_t \times (m + 1 + m \cdot M)$, and m is the number of resource types (e.g., CPU, storage, and memory). Regarding the action space, the action at time slot t is defined as $a_t = \{j_1, j_2, \cdots, j_{n_t}\}$, which determines the order of resource allocation for the n_t pending tasks. The reward function is defined as $r_t = \sum_{j \in n_t} 1/l_j$, where l_j is the duration of task j execution. Thus, the average slowdown of task is minimized as the agent is aimed at maximizing the cumulative reward. Through the extensive simulation, the algorithm is able to efficiently reduce the slowdown of an average task slowdown while ensuring the best QoS.

Furthermore, a task scheduling issue is investigated in the edge computing situation and various tasks are scheduled to virtual machines for the maximization of the long-term task satisfaction degree. The problem is expressed as MDP for which the state, action, state transition, and reward are created. For time scheduling and resource allocation, DRL is implemented, recognizing the heterogeneity of the tasks and the diversity of possible resources [75]. For the fairness of multi-resource considering diverse tasks, an online task scheduling system i.e., *FairTS* based on DRL techniques is proposed. The systems learn undeviatingly from experience to efficiently reduce the mean task slowdown while guaranteeing multi-resource fairness among the tasks [76]. Moreover in industrial applications, network traffic and computational offloading are explored using RL techniques for investigating the tradeoff within service delay and energy consumption. A cost minimization problem by employing the frame of MDP is formulated followed by the proposal of dynamic RL and scheduling algorithm algorithms to resolve the offloading determination problem [77].

Even though Fog networking is an encouraging technology to handle the limitations of the cloud and the current networks, there are yet challenges that persist to be evaluated in the future. Most significantly, there is a necessity for a distributed intelligent platform at the edge that controls distributed computing, networking, and storage resources. Optimal distribution decision in Fog networks faces several challenges because of contingencies linked with task requirements and available resources at the Fog nodes and the extensive range of computing power capabilities of nodes [78]. Moreover, delay within nodes must be considered for the distribution decision that can result in increased processing time. Hence, the difficulties being encountered by the Fog computing model are diverse and numerous; they include significant decisions about i) whether offloading at Fog nodes should be done or not, ii) offloading of the optimal number of tasks, and iii) given the corresponding resource limits, mapping of incoming tasks to possible Fog nodes [79]. Considering the above challenges, the proposed algorithm expresses the offloading problem as an MDP subject to the Fog node's behavior and dynamics of the system. MDP enables Fog nodes to offload their computation-intensive tasks by choosing the most proper adjacent Fog node in the presence of ambiguities on the task requirements and availability of resources at the Fog nodes. Nevertheless, the system is unable to accurately predict the transition possibilities and rewards due to dynamically varying incoming task requirements and resource states. To resolve this dilemma, RL can be used to solve MDPs with unknown reward and transition functions by making observations from experience.

C. Task Offloading and Redistribution

The imbalance of workload among the fog resources mainly caused by the heterogeneity of fog computing environment can degrade the performance of computing systems in the long-term operation. This urges the need to develop efficient mechanisms to address the situation through offloading and redistributing the load.

The task offloading problem considering the uncertainties of mobility of end user (EU) devices, mobility of cloudlets, and the resource availability of cloudlets is studied in [80]. A deep Q-network (DQN) [81] is formulated to learn an efficient solution and then deriving the optimal actions on how many tasks will be processed locally by end-user devices and how many task will be offloaded by the cloudlets. In this proposed model, the state space is defined as a $S = \{s = (Q^u, Q^C, D)\}$, where Q^u, Q^c, and D denote the queue state of EU device, the queue state of cloudlets, and the distance state of cloudlets, respectively. The mobility of devices and cloudlets affect the performance of their communication, thus the distance state is used to capture the change of computing system. To determine the optimal offloading decision, the action space is defined as $A = \{a = (a_0, \cdots, a_i, \cdots, a_N)\}$, where a_0 and a_i is are the number of tasks to be processed locally or by cloudlet i, respectively. The immediate reward function is defined as $R(s, a) = U(s, a) - C(s, a)$, where $U(s, a)$, and $C(s, a)$ are immediate utility and cost function, which are calculated as following equations.

$$U(s, a) = \rho \sum_{i=0}^{N} \log(1 + a_i) \qquad (11)$$

$$C(s,a) = \omega_1 I(s,a) + \omega_2 E(s,a) + \omega_3 D(s,a) + \omega_4 \Gamma(s,a) \tag{12}$$

Recall that ρ is a utility constant and N is number of cloudlets deployed in the systems for assisting the offloading processes. In addition, $I(s,a)$, $E(s,a)$, $D(s,a)$, and $\Gamma(s,a)$ are immediate required payment, energy consumption, delay and task loss probability, respectively. Therefore, by maximizing the cumulative reward the algorithm is enable to obtain the maximized utility as well as minimized operation cost. The Q-network can be considered as a neural network approximator with an approximate action-value function $Q(s,a;\theta)$ with weights θ. At each decision period, the user first takes the state vector $s = (Q^u, Q^c, D)$ as the input of the Q-network and obtains the Q-values $Q(s,.)$ for all possible action a as outputs. Then, the user selects the action according to the ϵ-greedy method. Furthermore, the Q-network is trained by iteratively adjusting the weights θ to minimize a sequence of the loss functions, where the loss function at time-step t is defined as

$$L_t(\theta_t) = \mathbb{E}[(r_t + \gamma \max_{a\prime} Q(s_{t+1}, a\prime; \theta_{t-1}) - Q(s_t, a_t; \theta_t))^2]. \tag{13}$$

Although the DQN-based algorithms are able to achieve the excellent performance in the high-dimensional decision-making problems [81] the proposed algorithm is evaluated in the simulation environment with only 4 cloudlets. The task arrival rate is varied to analyze the impact on the queue states of EU devices and cloudlets. In addition, no comparative analysis is performed in the work to compare with baseline or related works, thus the feasibility of performance improvement is unexplored.

Balancing the workload among nodes and simultaneously minimizing the task processing delay are studied in [26]. In a SDN-based fog computing system, a SDN fog controller is able to know the global information relating to the system states (e.g., task profiles, and queue states of fog devices), thus deriving the optimal task offloading. Using the DRL-based approach, the fog controller serves as the agent to make the decision. In this model, the state space is defined as $S = \{s = (n^l, w, Q)\}$, where n^l is the fog node, w is number of tasks to be allocated per unit time, and $Q = \{(Q_1, \cdots, Q_N)\}$ is a vector indicating the number of tasks currently remaining in the queues of N fog nodes. The action space is in form $A = \{a = (n^0, w^0)\}$, in which n^0 is a neighbor node of n^l, and w_0 is number of tasks to be offloaded by n^0. Aiming at maximizing the utility and simultaneously minimizing the task processing delay and overload probability, the immediate reward function is modeled as $R(s,a) = U(s,a) - (D(s,a) + O(s,a))$, where $U(s,a)$ is the immediate utility, $D(s,a)$ is immediate delay accounting for waiting delay in queues, communication delay for offloading, and execution delay at local device and offloading neighbor node, and $O(s,a)$ is overloaded probability averaging for n^0, and n^l. Q-learning with ϵ-greedy algorithm is applied in the Rl-based model to derive the optimal action selection.

The benefit of task offloading becomes prominent in the case of fog nodes by the selection of the appropriate nodes and suitable resource management while assuring the QoS requirements of the users [82]. An attempt in the case of heterogeneous service tasks within the multi-fog nodes where both joint tasks offloading and resource allocation management is considered. The problem formulation is based on a partly visible stochastic game where cooperation of each node is performed resulting in the maximization of combined local rewards. A deep recurrent Q-network (DRQN) method is applied to cope with partial visibility and to guarantee the accuracy and convergence of NN, adaptive exploration-exploitation approach is utilized [82]. Furthermore, for IoT applications, sequential allocation of the fog nodes restricted resources in the case of heterogeneous latency needs is considered. The problem formulation of the Fog radio access network is made as MDP followed by different RL approaches such as Q-learning, Monte Carlo, SARSA, and Expected SARSA to make optimal decision-making policies [83].

In many of fog-enabled networks the task nodes which are fog nodes having tasks in queues to be processed are unknown about the resource information of their neighbors. Therefore, offloading decisions require a trade-off between exploiting the empirically best nodes and exploring other nodes to find more beneficial actions, which is simply addressed by ϵ-greedy algorithm [84], [85]. However, this low-complexity solution is time-consuming for approaching the convergence and non-optimal [86]. In this context, multi-armed bandit (MAB)-based solutions are developed to address these kinds of shortcomings [87]. In particular, the upper-confidence bound (UCB) mechanism is integrated for obtaining the guaranteed performance and low complexity [88], [89]. Accordingly, the work [88] introduced BLOT (bandit learning-based offloading of tasks) algorithm to offloading non-splitable tasks. Meanwhile, D2CIT- a decentralized computation offloading is proposed in [89] to offloading the subtasks, which are constitutes of a high-complexity tasks.

For the sake of clarity, Table III summarizes the notable applications of RL in the fog computing resource allocations proposed in the literature.

V. Challenges and Open Issues of RL-based Algorithms in Fog Resource Allocation

Although RL is a powerful approach to introduce intelligence to fog computing-based systems, however, there are still many challenges and open issues that need to be addressed and overcome to fully exploit the potential of RL in assisting the fog paradigm. This section enumerates key challenges and correspondingly explores the open issues regarding the utilization of RL-based approaches for solving the resource allocation problems in the fog computing environment. We identify and discuss them according to three key classes relating to the RL, the fog computing environment, and the computing tasks.

A. RL-related Challenges

1) Nature of RL-based Algorithms: Naturally, the RL-based algorithms are time and resource consuming since they require

TABLE III
THE SUMMARY OF KEY RL-BASED ALGORITHMS IN THE RESOURCE ALLOCATION PROBLEMS.

Resource allocation problem	Reference	RL algorithms and features	Objective
Resource sharing	[60]	DRL Integrating prioritized replay Using transfer learning	Minimizing the energy consumption
	[62]	DRL Integrating LSTm-based DNN to predict vehicle movements Using procimal policy optimization	Minimizing the service latency
	[63], [64]	DRL Soft actor-critic technique Using the highest entropy frame	Maximizing the video bit rate Reducing the bit-rate variance Reducing the time delays of streaming services
	[65]	DRL Soft actor-critic technique Using dynamic pricing	Maximizing the resource utilization Reducing the time delays of VFC services
	[66]	DQN Adaptive switching policies	Minimizing the long-term cost Ensuring the guaranteed latency Guaranteed operation reliability
Task scheduling	[70]	Double deep Q-Learning (DDQL) Using multi-agent for multi-scheduler Using double Q for action selection and evaluation Integrating the target network & experience replay technique	Reducing the average delay Minimizing the computation cost
	[34]	DRL Modeling RNN with pointer network Integrating LSTM & attention mechanism	Reducing average task slowdown Ensuring the best QoS
	[90]	DRL	Reduced average task slowdown Multi-resource fairness
	[75]	DRL Integrating policy-based REINFORCE algorithm Using fully-connected NN (FCN) to extract the features	Average task satisfaction degree Task success ratio
	[77]	Dynamic RL scheduling (DRLS) Deep dynamic scheduling (DDS)	Saving energy consumption Reducing task execution delay
Task offloading	[80]	DQN	Save energy consumption Reduce task execution delay Minimized task loss probability
	[26]	DRL Q-learing ϵ-greedy	Load balancing Minimized computation cost
	[82]	Q-learning SARSA and expected SARSA Monte Carlo	Load balancing Minimized computation cost
	[83]	DRQN Q-learing ϵ-greedy	Maximizing the total served utility Minimizing the fog resource idle time
	[88], [89]	MBA Using the bandit learning technique Integrating upper confidence bound (UCB)	Reducing the task execution delay Minimizing energy and computation cost

a large volume of data collected through exploration and exploitation processes to derive the effectiveness of learning model. Meanwhile, the fog computing resources are heterogeneous and limited in terms of computation, storage, and energy compared to the cloud computing servers. Therefore, running the RL-based algorithms on the fog devices in the long term operation is an challenging job that calls for the appropriate and lightweight algorithm designs to tackle this challenge.

2) Load Balancing in RL-enabled Fog Computing: The RL approaches can be helpful for nodes to find the optimal policies (i.e., the number of tasks and size of tasks, offloadees in both the fog stratum and cloud stratum) to offload their requested tasks. If the overload probability and processing time in minimal the actions selection is considered optimal [91]. However, the action selection in the most of reviewed works is mainly dependent on the exploration policy. This situation probably leads to the greedy and optimistic decisions, which choose the more powerful fog resources to offload the tasks, thus resulting in the imbalance of workload consequently. To strike the imbalance situation from RL perspective, merging model-free RL learning with a model-based learning method can provide bias-free results having less dependency on exploration policy.

3) Task Scheduling in RL-enabled Fog Computing: Even though the fog node is equipped with better storage and computing power, however, it still possesses resources much lesser than cloud servers. Due to network heterogeneity, complexity, and uncertainty of the wireless environment, task scheduling in the fog computing environment has been categorized as a complex problem [70]. The RL algorithm can model complex problems, however by increasing the state and action space dimensions the hardware requirements of the scheduler will also need to increase. If we consider deep RL solutions for multi-dimensional problems, we would require multiple deep learning solutions that will add to computational power, storage, and memory requirements. The RL should provide a lightweight but efficient solution to a complex task scheduling problem.

4) Energy Efficiency Trade-off in RL-enabled Fog Computing: In time-critical IoT applications, RL-assisted fog computing systems could bring intelligence features to utilize readily available data and computing resources. Minimizing delay and energy consumption simultaneously has been a key challenge for RL algorithms [92]. First, the training of learning model requires high-energy consumption and similarly, fast learning on time-critical systems given limited samples is a complex problem for RL-enabled fog systems. Thus, there are many open challenges in deploying large-scale state-action space RL models for resource-constrained fog systems.

5) Real-time Responsiveness in RL-enabled Fog Computing: Ultra-reliable and low latency communication (URLLC) and real-time interaction are one of the main enablers of IoT and 5G communication [93]. Fog computing can support the computation tasks with low latency, thus enabling to provide some kinds of real-time applications. In the case of a heterogeneous IoT network, some applications are delay tolerant while others are delay sensitives. The RL algorithm provides intelligent utilization of resources capability to fog nodes, however, RL algorithms also consume time to process large-scale state-action-reward tuple [94]. In the case of multidimensional states and action space, the processing time further increases. Therefore, one of the critical challenges of the RL-enabled fog network is to intelligently satisfy time-critical IoT applications. The deep RL system can learn more quickly and efficiently through episodic memory and meta-learning technique, which have been not explored in the literature [94].

6) Security and Privacy in RL-enabled Fog Computing: The RL algorithm in fog nodes collects and processes a large amount of critical data from network devices. Fog devices are distributed and contains limited resources. It is challenging for RL-enabled fog devices to execute proper security solutions in parallel to other learning mechanisms due to limited computing power. Similarly, there is an absence of trust between IoT devices and edge computing nodes [95], [96]. The RL combined with Blockchain can solve the trust issue. In such a solution, optimizing various Blockchain metrics using RL technique is a critical decision that needs to be explored.

7) Advance of Optimization Algorithms: In fact, the reinforcement learning algorithms are kinds of time-consuming works since it requires an extensive time dedicated for learning process. Therefore, the RL-based algorithms should be advanced to reduce the convergence time, thus accelerating the decision makings. In addition, the performance of RL-based algorithms are dependent on the complexity of fog networks, arrival rate of tasks. Thus for achieving a good trade-off of the training time cost and the performance, it is strongly recommended to prepare the sample data set to a reasonable scale with sampling technology to reduce the complexity of scheduling model. Advancing algorithms is required to improve the speed and effectiveness of learning process. How to reduce the dimension of problems (i.e., state space and action space) to accelerate the learning process is open issue.

B. Fog Computing Environment Related Challenges

The presence of fog computing tier is increasingly essential in the IoT-enabled systems to meet any application requirement. However, the various applications require different designs of fog computing architecture (i.e., either specific or agnostic), which totally can contribute as a challenging factor to use RL in this context. Because there is no common Rl-based solutions which can be used for different fog computing architectures and applications.

1) RL-based Resource Allocation in F-RAN: For densely deployed IoT devices, cloud radio access network (C-RAN) architecture is proposed. C-RAN improves spectral efficiency and reduces energy consumption. However, the demand for IoT devices and applications is increasing placing a high burden on centralized cloud computing. Busy cloud servers, limited fronthaul capacity causing large computation and transmission delays. Some IoT applications are delay-sensitive and cannot tolerate such delays. To handle this problem, F-RAN is a critical solution for the Fifth-generation (5G) communication systems to support the URLLC requirement for IoT devices and applications [97]. The fog nodes are capable of performing signal processing, computation, and RF functionalities. IoT environment is heterogeneous in nature with various traffic transmission rates and latency needs. The fog nodes are expected to allocate resources in Fog-RAN efficiently. RL method provides a solution for efficient resource utilization along with satisfying low-latency requirements for various IoT applications. Multi-agent RL is utilized to enhance network resource utilization in heterogeneous environments. Similarly, the model-free RL technique is used for user scheduling in heterogeneous networks for network energy efficiency [98]. Various RL methods such as Q-learning, SARSA, Expected SARSA (E-SARSA), and Monte Carlo (MC) are used for resource allocation problems. However, there are still open issues such as providing dynamic resource allocation framework with heterogeneous service time. Similarly, collaborative resource allocation mechanism with multiple fog nodes are one of the future directions.

2) RL-based Power Consumption Optimization for F-RAN: F-RAN with 5G support is well suited to provide various IoT services including healthcare, IIoT, and autonomous vehicles. In F-RAN, each device can operate in different communication modes including D2D, C-RAN, or FAP. In resource management mechanism, communication mode selection problem is considered as NP-hard due to network dynamics and continuously changing environment. Nevertheless, applying deep reinforcement learning (DRL) has shown considerable benefits for complex environment with high dimensional raw input data. The devices can obtain the desired information locally without accessing base station due to the caching capabilities of D2D transmitters. In this way, the burden on fronthaul can be reduced by traffic offloading and turning off some processors in the cloud for energy optimization. The dynamics of cache state of D2D transmitter can be modeled as MDP [99]. In such MDP problem, the network controller learns to minimize the system power consumption by adjusting devices communication modes and processors on-off states at each learning step. However, in the DRL-based resource allocation mechanism, further research is required to utilize power control of D2D communicating devices, sub-channel allocation and fronthaul resource allocation for improving the F-RAN performance.

3) RL for Ultra-dense Fog Network: An ultra-dense network with an assistant of fog computing is a promising solution to sustain the increasing traffic demand in wireless networks. Network densification brings a number of challenges including resource management [100]. Machine learning particularly RL has been proven to solve resource management challenges effectively. In an ultra-dense fog network scenario, RL algorithms need to enable the parallelism and partition of large-scale networks to manage the computational load of fog devices. Similarly, in the case of wired and wireless backhaul networks, bandwidth allocation must be considered since it is an important factor affecting on the performance. The powerful capability of deep learning and neural networks can enhance the performance of resulting DRL-based methods to solve high dimension problems. Many proposed models have reduced the dimension by efficiently configuring the state and action spaces. However, none of existing works has been investigated and developed the RL-based algorithms for the large scale fog computing systems, which basically contain a large number of fog nodes. Practically, this is open issue.

4) Reliability of Fog Networks: The complex nature of fog computing environment may cause the reliability problem for the fog network. For example, the dynamic mobility of fog nodes which strongly impacts on the fog resource status must be taken into account in designing the algorithms. In addition, the outdated channel probably blocks the communication between the fog nodes [101]. Moreover, VMs in the both fog stratum and cloud may fail and lead to QoS degradation [102]. Hence, reliability of VMs should also be considered when addressing the fog resource provisioning problem. None of reviewed algorithms covers and consider the reliability of fog network, thus opening the issue for future studies.

5) Security and Trust in Fog Network: A fog node is responsible to ensure the security and trust for other devices. Fog node must ensure the global concealing process on the released data. Fog nodes must ensure a mechanism for all nodes to have a certain level of trust in each other. The fog node handles the workload for other nodes in real-time. Protecting the integrity in case of a malicious attack is one of the challenges of fog computing. Similarly, authentication is essential to provide a secure connection between the fog node and other devices. Authentication is needed to provide in real-time data communication particularly in a scenario where nodes are moving from one coverage area to another. The user must experience a minimum delay in real-time services while traveling. The latency is caused by the authentication process performed in the fog node. During the authentication process, there is a possibility of user identity exposure to attackers. Authorization and authentication in fog networks are one of the major concerns. Providing real-time services in a fog computing environment with secure authentication is one of the priority research areas.

C. Computing Task Related Challenges

In the IoT-based context, requesting computing tasks are varied in wide range in terms of profiles, complexity levels, and resource requirements for computation. These property variations serve as a challenging factor to apply RL-based algorithms.

1) Big Data Analytics: The IoT and end user devices increasingly produce a huge amount of high-dimensional big data to be processed at the fog nodes [103]. A selection of appropriate prediction model and RL parameters, e.g., learning rate and discount factor are needed to considered carefully to obtain an optimized model for big data analytics. A proper analytic model can produce accurate results and can learn from heterogeneous data sources. In big data analytics, one of the challenges a learning algorithm faces is how to distribute big data among resources constrained fog devices fairly.

2) Data Fragmentation for Parallel Computing Exploitation: Besides the Big data analytics related tasks, many of tasks in the IoT-enabled systems can be complex in terms of size, data structure. For example, the input data of a ML computing task can contain four types of data: text, image, video, and audio, which requires many kinds of resources to process. However, the limitation of fog computing resource causes imbalance of workload among the fog nodes since many of them with insufficient available resources is unable to process a single task. The data division is a key approach to solve this issues in the complicated heterogeneous fog environment [27]. However, the diversity of input data structure in practical application requires alternative division solutions for improving or optimizing the system performance. For instance, as the data dependency constraints among the substasks are taken into account in the associated workflow model and the collaborative task offloading mechanism must be adapted to such a change accordingly. In addition, the data can be divided according to different features such as by size explicitly. In this way, an optimization can be formulated to find the optimal number of data subsets and associated sizes of data subset for optimizing the system performance. Although the input data of tasks can be divided to take the benefit of parallel computing it may raise the large space as applying RL models. Therefore, to achieve the efficient trade-off between the performance and time-consuming training it requires to search for an optimal number of data subsets divided from the input data.

VI. Conclusions

The fog computing has been integrated in a wide range of IoT-enabled systems as a support computing resources to cure the pressure of cloud computing resources, thus improving the operation performance of systems. However, the fog computing environment is a complex resource pool in terms of heterogeneity, mobility, and dynamic change, which serve as critical barriers for achieving efficient and effective resource allocation strategy. In addition, the computing tasks are varied with respective to task characteristics and resource demands. Moreover, the most of efficient heuristics algorithms in the literature lack the adaptivity and flexibility to respond to the uncertainties of fog computing environment. These aforementioned challenges stress a need to develop an alternative for resource allocation solutions to flexibly deal with the complexity of fog computing environment.

This paper surveys the literature on the applications of RL in the fog computing environment to allocate the computing resources for task computation and execution. The concept of RL is briefly introduced to highlight accordingly the role and algorithmic model to support deriving the optimal decision makings in many practical applications (e.g., game, robotics, and finance). The start-of-the art literature review is conducted to describe intensively the key RL-based solutions for the resource allocation problems in the fog computing environment. We identify and analyze these algorithms according to three major problems, namely, resource sharing, task scheduling, and task offloading. Finally, the work also explored and discussed the key challenges faced by the nature of RL-based algorithms, the fog computing environment, and the computing tasks in the variety of practical applications. The corresponding open issues are also presented for further studies.

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[2] Y. Saleem, N. Crespi, M. H. Rehmani, and R. Copeland, "Internet of things-aided smart grid: Technologies, architectures, applications, prototypes, and future research directions," *IEEE Access*, vol. 7, pp. 62 962–63 003, 2019.

[3] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial iot data scheduling based on hierarchical fog computing: A key for enabling smart factory," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4590–4602, Oct. 2018.

[4] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through internet of things," *IEEE Internet Things J.*, vol. 1, no. 2, pp. 112–121, Apr. 2014.

[5] H. Tran-Dang and D. Kim, "An information framework for internet of things services in physical internet," *IEEE Access*, vol. 6, pp. 43 967–43 977, 2018.

[6] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future Generation Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016.

[7] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[8] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 46–59, Jan. 2018.

[9] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Generation Comput. Syst.*, vol. 87, pp. 278–289, Oct. 2018.

[10] S. Patil-Karpe, S. H. Brahmananda, and S. Karpe, "Review of resource allocation in fog computing," in *Smart Intelligent Comput. Applicat.*. Springer Singapore, pp. 327–334. [Online]. Available: https://doi.org/10.1007/978-981-13-9282-5_30.

[11] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4712–4721, Oct, 2018.

[12] C. Mouradian, *et al.*, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.* vol. 20, no. 1, pp. 416–464, 2018.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press.

[14] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures artificial intelligence machine learning*, vol. 4, no. 1, pp. 1–103, 2010.

[15] J. Gedeon, F. Brandherm, R. Egert, T. Grube, and M. Mühlhäuser, "What the fog? edge computing revisited: Promises, applications and future challenges," *IEEE ACCESS*, vol. 7, pp. 152 847–152 878, 2019.

[16] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in iot networks via reinforcement learning," in *Proc. IEEE ICC*, 2019.

[17] X. Dutreilh, *et al.*, "Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow," in *Proc. ICAS*, 2011.

[18] X. Lin, Y. Wang, and M. Pedram, "A reinforcement learning-based power management framework for green computing data centers," in *Proc. IEEE IC2E*, 2016.

[19] J. Yuan, X. Jiang, L. Zhong, and H. Yu, "Energy aware resource scheduling algorithm for data center using reinforcement learning," in *Proc. ICICTA*, 2012.

[20] Y. Li, Y. Wen, D. Tao, and K. Guan, "Transforming cooling optimization for green data center via deep reinforcement learning," *IEEE Trans. Cybern.*, vol. 50, no. 5, pp. 2002–2013, 2020.

[21] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.* vol. 34, no. 6, pp. 26–38, Nov. 2017.

[22] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. ACM HotNets*, 2016.

[23] H. Che, Z. Bai, R. Zuo, and H. Li, "A deep reinforcement learning approach to the optimization of data center task scheduling," vol. 2020, pp. 1–12. [Online]. Available: https://doi.org/10.1155/2020/3046769.

[24] Y. Wei, L. Pan, S. Liu, L. Wu, and X. Meng, "Drl-scheduling: An intelligent qos-aware job scheduling framework for applications in clouds," *IEEE ACCESS*, Sep. 2018.

[25] C. Shyalika, T. Silva, and A. Karunananda, "Reinforcement learning in dynamic task scheduling: A review," *SN COMPUT. SCI.*, vol. 1, no. 6, Sep. 2020.

[26] J.-y. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," in *Proc. IEEE WCNC*, 2019.

[27] H. Tran-Dang and D.-S. Kim, "Task priority-based resource allocation algorithm for task offloading in fog-enabled IoT systems," in *Proc. IEEE ICOIN*, 2021.

[28] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing iot service delay via fog offloading," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, Apr. 2018.

[29] A. Ahmed, *et al.*, "Fog computing applications: Taxonomy and requirements.", *arXiv preprint arXiv:1907.11621*, 2019.

[30] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "Dats: Dispersive stable task scheduling in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3423–3436, Apr. 2019.

[31] Z. Liu, Y. Yang, K. Wang, Z. Shao, and J. Zhang, "Post: Parallel offloading of splittable tasks in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3170–3183, 2020.

[32] H. Tran-Dang and D.-S. Kim, "Frato: Fog resource based adaptive task offloading for delay-minimizing iot service provisioning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, pp. 2491–2508, 2021.

[33] K. Guo, M. Sheng, T. Q. Quek, and Z. Qiu, "Task offloading and scheduling in fog ran: A parallel communication and computation perspective," *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 215–218, 2019.

[34] S. Bian, X. Huang, Z. Shao, and Y. Yang, "Neural task scheduling with reinforcement learning for fog computing systems," in *Proc. IEEE GLOBECOM*, 2019.

[35] N. Fernando, *et al.*, "Opportunistic fog for iot: Challenges and opportunities," *IEEE Internet Things J.,* vol. 6, no. 5, pp. 8897–8910, 2019.

[36] Y. Xiao and C. Zhu, "Vehicular fog computing: Vision and challenges," in *Proc. IEEE PerCom Workshops*, 2017.

[37] H. A. Khattak, S. U. Islam, I. U. Din, and M. Guizani, "Integrating fog computing with vanets: A consumer perspective," *IEEE Commun. Standards Mag.* vol. 3, no. 1, pp. 19–25, 2019.

[38] T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam, “Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud,” in *Proc. ACM MobiHoc*, 2013.

[39] J. Oueis, E. C. Strinati, S. Sardellitti, and S. Barbarossa, “Small cell clustering for efficient distributed fog computing: A multi-user case,” in *Proc. IEEE VTC-Fall*, 2015.

[40] W. Masri, I. A. Ridhawi, N. Mostafa, and P. Pourghomi, “Minimizing delay in iot systems through collaborative fog-to-fog (f2f) communication,” in *Proc. ICUFN*, 2017.

[41] R. Lindelauf, “Nuclear Deterrence in the Algorithmic Age: Game Theory Revisited,” *NL ARMS*, p. 421, 2021.

[42] C. Kim, “Deep reinforcement learning by balancing offline Monte Carlo and online temporal difference use based on environment experiences,” *Symmetry*, vol. 12, no. 10, p. 1685, 2020.

[43] B. Kővári, F. Hegedüs, and T. Bécsi, “Design of a reinforcement learning-based lane keeping planning agent for automated vehicles,” *Applied Sciences*, vol. 10, no. 20, p. 7171, 2020.

[44] S. S. Mousavi, M. Schukat, and E. Howley, “Deep reinforcement learning: An overview,” in *Proc. IntelliSys*, pp. 426–440, 2016.

[45] O. L. V. Costa, E. Assumpção Filho, E. Boukas, and R. Marques, “Constrained quadratic state feedback control of discrete-time markovian jump linear systems,” *Automatica*, vol. 35, no. 4, pp. 617–626, 1999.

[46] S. Mahadevan, “Average reward reinforcement learning: Foundations, algorithms, and empirical results,” *Machine learning*, vol. 22, no. 1, pp. 159–195, 1996.

[47] Y. Chandak, G. Theocharous, J. Kostas, S. Jordan, and P. Thomas, “Learning action representations for reinforcement learning,” in *Proc. ICML*, 2019.

[48] A. Kanervisto, C. Scheller, and V. Hautamäki, “Action space shaping in deep reinforcement learning,” in *Proc. IEEE CoG*, 2020.

[49] A. Kumar, T. Buckley, J. B. Lanier, Q. Wang, A. Kavelaars, and I. Kuzovkin, “OffWorld gym: Open-access physical robotics environment for real-world reinforcement learning benchmark and research.” *arXiv preprint arXiv:1910.08639*, 2019.

[50] T. M. Moerland, J. Broekens, and C. M. Jonker, “Model-based reinforcement learning: A survey.” *arXiv preprint arXiv:2006.16712*, 2020.

[51] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, “Mastering chess and shogi by self-play with a general reinforcement learning algorithm,” *arXiv preprint arXiv:1712.01815*, 2017.

[52] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *Proc. ICML*, 2016.

[53] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms.” *arXiv preprint arXiv:1707.06347*, 2017.

[54] J. B. Serrano, S. Curi, A. Krause, and G. Neu, “Logistic Q-learning,” in *Proc. AISTATS*, 2021.

[55] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[56] M. G. Bellemare, W. Dabney, and R. Munos, “A distributional perspective on reinforcement learning,” in *Proc. ICML*, 2017.

[57] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[58] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proc. ICML*, 2018.

[59] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, “Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning,” in *Proc, IEEE ICRA*, 2018.

[60] Y. Sun, M. Peng, and S. Mao, “Deep reinforcement learning-based mode selection and resource management for green fog radio access networks,” *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1960–1971, 2019.

[61] Y. He, N. Zhao, and H. Yin, “Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach,” *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, 2017.

[62] S.-s. Lee and S. Lee, “Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information,” *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10 450–10 464, 2020.

[63] F. Fu, Y. Kang, Z. Zhang, F. R. Yu, and T. Wu, “Soft actor-critic drl for live transcoding and streaming in vehicular fog computing-enabled iov,” *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1308–1321, 2020.

[64] J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, “Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach,” *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6214–6228, 2019.

[65] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, “Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning.” *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, 2020.

[66] X. Zhang, Y. Xiao, Q. Li, and W. Saad, “Deep reinforcement learning for fog computing-based vehicular system with multi-operator support,” in *Proc. IEEE ICC*, 2020, pp. 1–6.

[67] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, “Performance optimization in mobile-edge computing via deep reinforcement learning,” in *Proc. IEEE VTC-Fall*, 2018, pp. 1–6.

[68] N. Van Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, “Optimal and fast real-time resource slicing with deep dueling neural networks,” *IEEE J. Selected Areas Commun.*, vol. 37, no. 6, pp. 1455–1470, 2019.

[69] J. C. Guevara and N. L. S. da Fonseca, “Task scheduling in cloud-fog computing systems,” *Peer-to-Peer Netw. Applicat.*, vol. 14, no. 2, pp. 962–977, Jan. 2021. [Online]. Available: https://doi.org/10.1007/s12083-020-01051-9

[70] P. Gazori, D. Rahbari, and M. Nickray, “Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach,” *Future Generation Comput. Syst.*, vol. 110, pp. 1098–1115, 2020.

[71] Y. Wei, L. Pan, S. Liu, L. Wu, and X. Meng, “Drl-scheduling: An intelligent qos-aware job scheduling framework for applications in clouds,” *IEEE ACCESS*, vol. 6, pp. 55 112–55 125, 2018.

[72] S. Bian, X. Huang, Z. Shao, and Y. Yang, “Neural task scheduling with reinforcement learning for fog computing systems,” in *Proc. IEEE GLOBECOM*, 2019.

[73] O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer networks,” in *Proc. NIPS*, 2015, pp. 2692–2700. [Online]. Available: https://arxiv.org/pdf/1506.03134.pdf

[74] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Proc. NIPS*, 2014.

[75] S. Sheng, P. Chen, Z. Chen, L. Wu, and Y. Yao, “Deep reinforcement learning-based task scheduling in iot edge computing,” *Sensors*, vol. 21, no. 5, p. 1666, 2021.

[76] S. Bian, X. Huang, and Z. Shao, “Online task scheduling for fog computing with multi-resource fairness,” in *Proc. IEEE VTC-Fall*, 2019.

[77] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, “Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications,” *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 976–986, 2018.

[78] L. Yu, L. Chen, Z. Cai, H. Shen, Y. Liang, and Y. Pan, “Stochastic load balancing for virtual resource management in datacenters,” *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 459–472, 2016.

[79] G. Lee, W. Saad, and M. Bennis, “An online optimization framework for distributed fog network formation with minimal latency,” *IEEE Trans Wireless Commun.*, vol. 18, no. 4, pp. 2244–2258, 2019.

[80] D. Van Le and C.-K. Tham, “A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds,” in *Proc. IEEE INFOCOM WKSHPS*, 2018.

[81] V. Mnih, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[82] J. Baek and G. Kaddoum, “Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks,” *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1041–1056, 2020.

[83] A. Nassar and Y. Yilmaz, “Reinforcement learning for adaptive resource allocation in fog ran for IoT with heterogeneous latency requirements,” *IEEE Access*, vol. 7, pp. 128 014–128 025, 2019.

[84] M. Min, *et al.*, “Learning-based privacy-aware offloading for healthcare iot with energy harvesting,” *IEEE Access*, vol. 6, no. 3, pp. 4307–4316, 2018.

[85] M. Min, *et al.*, “Learning-based computation offloading for iot devices with energy harvesting,” *IEEE Trans Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, 2019.

[86] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Machine learning*, vol. 47, no. 2, pp. 235–256, 2020.

[87] D. A. Berry and B. Fristedt, “Bandit problems: sequential allocation of experiments (monographs on statistics and applied probability),” *London: Chapman and Hall*, vol. 5, no. 71–87, p. 7, 1985.

[88] Z. Zhu, T. Liu, Y. Yang, and X. Luo, “Blot: Bandit learning-based offloading of tasks in fog-enabled networks,” *IEEE Trans Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2636–2649, 2019.

[89] S. Misra, S. P. Rachuri, P. K. Deb, and A. Mukherjee, "Multi-armed bandit-based decentralized computation offloading in fog-enabled IoT," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 10010–10017, 2021.

[90] S. Bian, X. Huang, and Z. Shao, "Online task scheduling for fog computing with multi-resource fairness," in *Proc. IEEE VTC-Fall*, 2019, pp. 1–5.

[91] F. M. Talaat, M. S. Saraya, A. I. Saleh, H. A. Ali, and S. H. Ali, "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment," *J. Ambient Intell. Humanized Comput.*, pp. 4951–4966, 2020.

[92] Q. D. La, M. V. Ngo, T. Q. Dinh, T. Q. Quek, and H. Shin, "Enabling intelligence in fog computing to achieve energy and latency reduction," *Digital Commun. Netw.*, vol. 5, no. 1, pp. 3–9, 2019.

[93] R. K. Naha, *et al.*, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE ACCESS*, vol. 6, pp. 47 980–48 009, 2018.

[94] M. Botvinick, *et al.*, "Reinforcement learning, fast and slow," *Trends Cognitive Sci.*, vol. 23, no. 5, pp. 408–422, 2019.

[95] P. Illy, G. Kaddoum, C. M. Moreira, K. Kaur, and S. Garg, "Securing fog-to-things environment using intrusion detection system based on ensemble learning," in *Proc. IEEE WCNC*, 2019.

[96] A. Abeshu and N. Chilamkurti, "Deep learning: The frontier for distributed attack detection in fog-to-things computing," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 169–175, 2018.

[97] N. N. Khumalo, O. O. Oyerinde, and L. Mfupe, "Reinforcement learning-based resource management model for fog radio access network architectures in 5g," *IEEE ACCESS*, vol. 9, pp. 12 706–12 716, 2021.

[98] A. Nassar and Y. Yilmaz, "Resource allocation in fog ran for heterogeneous iot environments based on reinforcement learning," in *Proc. IEEE ICC*, 2019, pp. 1–6.

[99] Y. Sun, M. Peng, and S. Mao, "Deep reinforcement learning-based mode selection and resource management for green fog radio access networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1960–1971, 2018.

[100] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Commun. Surveys Tuts*, vol. 20, no. 3, pp. 1826–1857, 2018.

[101] T. H. L. Dinh, M. Kaneko, E. H. Fukuda, and L. Boukhatem, "Energy efficient resource allocation optimization in fog radio access networks with outdated channel knowledge," *IEEE Trans Green Commun. Netw.*, vol. 5, no. 1, pp. 146–159, 2021.

[102] J. Yao and N. Ansari, "Fog resource provisioning in reliability-aware iot networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8262–8269, 2019.

[103] C. Prabhu, *Fog Computing, Deep Learning and Big Data Analytics-Research Directions*. Springer, 2019.

Hoa Tran-Dang (M'17) received the B.E. degree in Electrical and Electronics Engineering from Hanoi University of Science and Technology (HUST), Vietnam and the M.S. degree in Electronics Engineering from Kumoh National Institute of Technology (KIT), South of Korea in 2010 and 2012, respectively. He pursued the Ph.D. degree with University of Lorraine, France during 2013-2017. He currently works in NSL laboratory, department of ICT convergence engineering at Kumoh National Institute of Technology, South of Korea as a research professor. His research interests include wireless sensor networks, Internet of things (IoT), physical Internet, and radio resource management in wireless industrial networks.

Sanjay Bhardwaj (M'17) received his Ph.D. degree from the Department of IT convergence Engineering, Kumoh National Institute of Technology, Gumi, South Korea in 2020. From 2012 to 2018 he worked as Assistant Professor in the Department of Electronics and Communication at Shoolini University, India. He is currently Postdoctoral Researcher at ICT Convergence Research Center, Kumoh National Institute of Technology, Gumi, South Korea. His research areas of interest are bio-inspired CRNs, IoT, IIoT and URLLC in the industrial wireless network.

Tariq Rahim (M'21) has completed his Ph.D. degree in IT Convergence Engineering from the Wireless and Emerging Network System Laboratory (WENS Lab), at Kumoh National Institute of Technology (KIT), South Korea. Currently, he is working as a Postdoctoral Fellow and Researcher at the ICT-CRC, KIT, South Korea. Earlier, he received his MS. and B.Sc. degree from Beijing Institute of Technology, China and COMSAT Institute of Information and Technology, Pakistan, respectively. His research interests mainly include signal processing, image processing, medical image analysis, deep learning, video processing, and quality of services of high frame rate videos.

Arslan Musaddiq (M'21) received his Ph.D. degree from the Department of Information and Communication Engineering, College of Engineering, Yeungnam University, Gyeongsan-Si, South Korea. He is currently associated with the ICT Convergence Research Center, Kumoh National Institute of Technology, South Korea as a Postdoctoral fellow. His research interests are primarily in the areas of wireless networking, Internet of Things, wireless resource management, routing protocols, and ad hoc networks.

Dong-Seong Kim (S'98-M'03-SM'14) received his Ph.D. degree in Electrical and Computer Engineering from the Seoul National University, Seoul, Korea, in 2003. From 1994 to 2003, he worked as a full-time researcher in ERC-ACI at Seoul National University, Seoul, Korea. From March 2003 to February 2005, he worked as a Postdoctoral Researcher at the Wireless Network Laboratory in the School of Electrical and Computer Engineering at Cornell University, NY. From 2007 to 2009, he was a Vistiting Professor with Department of Computer Science, University of California, Davis, CA. He is currently a Director of KIT Convergence Research Institute and ICT Convergence Research Center (ITRC program) supported by Korean government at Kumoh National Institute of Technology. He is IEEE and ACM Senior Member. His current main research interests are real-time IoT, industrial wireless control network, networked embedded system and fieldbus.