



# A self-learning approach for proactive resource and service provisioning in fog environment

Mohammad Faraji-Mehmandar<sup>1</sup> · Sam Jabbehdari<sup>2</sup> ·  
Hamid Haj Seyyed Javadi<sup>3</sup>

Accepted: 9 April 2022 / Published online: 12 May 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

## Abstract

With increasing growth in IoT, the number of devices connected to the Internet is constantly growing. Moreover, the increase in the volume of data and their transmission through the Internet of Things, as well as the existence of inadequate bandwidth, limits cloud-based storage and data processing. Both fog and cloud computing provide the storage space, application, and data for users; however, fog is more proximate to the end user with wider geographical distribution. When bringing the computing resources closer to the required location in the fog environment, the efficiency of the system increases, and the distance at which data must be transmitted decreases. On the other hand, implementing IoT applications and satisfying the requests of end users in fog computing will create new challenges in resource allocation and dynamic resource provisioning. The flexible and usually automatic mechanisms require the determination of required virtual resources to minimize the resource consumption and service level agreement (SLA). In this paper, we introduce a framework for increasing resource management efficiency in the IoT ecosystem based on deep reinforcement learning (DRL). The proposed deep neural network (DNN) method for estimating value functions improves adaptability to different oscillating conditions, learns past sensible strategies, and as a self-learning adaptive system by replicating interactions with the fog environment. The DRL algorithm finds the best destination for implementing IoT services to compromise between minimizing average power consumption, minimizing average service latency, reducing costs, and balancing resource allocation. Finally, through simulations, we show that under different loading rates, the policy used compared to other comparable solutions is to increase utilization and reduce the rate of delay, while ensuring an acceptable level of service quality.

---

✉ Mohammad Faraji-Mehmandar  
m.Faraji@piau.ac.ir

Extended author information available on the last page of the article

**Keywords** Internet of Things (IoT) · Resource provisioning · Auto-scaling · Deep reinforcement learning · Fog computing · Application-based services

## 1 Introduction

The application of connected smart devices such as the Internet of Things (IoT) has witnessed exponential growth. The development of IoT enables individuals and objects to enter any information space at every time and place and communicate with each other. An increase in the number of IoT devices has inevitably led to the production of a mass volume of data that needs to be processed, stored, and properly made available to end users. The big volume of data makes the current processing and storage capacity fail in responding to the demands and working with traditional computing models, such as distributive computing and cloud computing, more complicated [1–4]. Cloud computing is taken as an effective method in data processing because of its high computational and storage capacity. As the cloud computing pattern is a focused model and most computations are done in the cloud, the data processing speed has considerably increased; however, unfortunately, the bandwidth has not increased to the same extent. Therefore, the network bandwidth is one of the problems of cloud computing that could result in long delays. In some IoT applications, the system might require a very short response time and displacements. These include traffic light systems in smart traffic, smart networks, intelligent health care, emergency reactions, and other delay-sensitive applications. The delay due to data transmission is unacceptable. Sometimes, it is required to make decisions and take actions regarding the event, in which delay will lead to bad consequences. In addition, some decisions can be made locally, without any compulsion in transmission to the cloud. Even if decisions need to be made in the cloud, it is not necessary to transmit all data to the cloud and classify because all data are not beneficial for analysis and decision making. In other words, these challenges that are created due to the rapid growth of IoT, and associated with the bandwidth of network, reliability, and security could be tackled based on cloud model and independently. The use of computing resources close to the users is proposed as the solution to these issues to achieve local processing and storage and reduce the transmission size and data latency [5, 6].

Internet of Things (IoT) is one of the technologies that have made a huge leap in the world. IoT deals with large amounts of data, including video, audio, and text, that are not easy to be processed and store. These smart objects that are located everywhere generate and collect large amounts of raw data. Cloud computing plays an important role in the development of IoT, while providing processing and storage services for a large amount of data. However, many IoT applications suffer from cloud computing challenges such as latency and lack of relocation support and location awareness. Transferring these data to cloud data centers for processing or storage will definitely create bandwidth occupation, and in addition, increase the response time of IoT applications. On the other hand, transferring data outside the organization's borders is considered in some

industries as a security breach. Under such circumstances, transferring data to cloud data centers is not very practical and cost-effective, and another computing model should be used for this purpose. Fog computing, which is almost known as the evolution of cloud computing, helps to provide solutions to these challenges. This model facilitates the deployment of IoT services and applications, reduces the time latency, and consequently, provides the possibility of real-time processing and analysis, reducing the loss of communication bandwidth and costs [7, 8].

## 1.1 Motivation

IoT is made up of heterogeneous environments with unpredictable traffic. Heterogeneous environments in IoT are reactive, which makes it difficult to predict traffic in a variety of situations. Failure to predict traffic in heterogeneous environments and increased connection of multiple devices with diverse traffics will cause congestion and queuing in the network. One of the essential requirements of computing systems, which improves network performance, is resource management and dynamic resource provisioning effectively. When IoT services communicate with fog resources, the resources must be provisioned in such a way as not to be less or more than the service requirement. The hardware resources available in the fog node are limited compared to the cloud server. Moreover, the pattern of using IoT resource usage is also dynamic and time varying. As a result, the resources allocated to requests must be dynamically scaled to enable the best utilization of available resources, and static deployment will not be able to adapt to such changes. In other words, the traffic generated by users and sensors is irregular and oscillatory. If the used resources are fixed, there will be times when requests inbound from devices require more or fewer resources to perform transactions [9–11]. These conditions can sometimes lead to over-provisioning or under-provisioning. If the resources are less than the required IoT service, then the service cannot be performed at all or will be done incompletely. If the resources are more than the IoT service requirement and the workload requires fewer resources, the resources and costs will be wasted, and the efficiency will be reduced. Therefore, fog services should be dynamically placed on the resources to respond to the requests at the earliest time with the best response through optimal and dynamic resource management. In addition, it should be ensured that no node is overloaded with power or less than the required capacity. The basis for the dynamism of fog services is the deployment and release of IoT services on fog computing devices or cloud servers to reduce costs, resource power consumption and achieve less latency and therefore, optimal quality of service (QoS) for IoT applications. To this end, resource provisioning in fog computing is one of the challenging issues, as the important issue here is optimal fog resource provisioning so that all users can utilize each resource based on their needs. If these resources are provisioned and managed automatically, and without the users' intervention, there will be a significant reduction in criteria such as response time, cost and power consumption, and increase in productivity [12, 13].

## 1.2 Our approach and contributions

In static problems, the main goal is to find or better estimate the optimal points. However, in dynamic problems, not only the main goal of the static state should be satisfied, the optimal point/s should also be pursued at the earliest possible time. The reason is that in dynamic environments, due to environmental variability, it is possible to change the optimal point to another area of the search space. Consequently, such problems face more challenges than static ones, and these types of problems are considered hard problems (NP-hard class). Thus, the proposed algorithm should be able to show good performance in situations where the environment is facing uncertainty and track the optimal variable. Therefore, those algorithms should be utilized that can adapt themselves to the varying environmental conditions.

The main purpose of this study is to reduce the average latency and cost of service provisioning of IoT applications and increase the utilization in the fog environment. To this end, we intend to create a new method for resource management related to IoT applications using neural network and DRL approaches. This method can automatically, relying on previous experiences, achieve an effective strategy for scheduling overtime. The issue of resource provisioning is in the realm of decision-making issues, and due to the dynamic conditions of IoT networks and the difficulty of their modeling, it is required to use an online and adaptable method to solve them. Therefore, it is not practical to use exploratory and law-based methods to this end. Using machine learning, it is possible to create estimative models with acceptable accuracy from systems that have complex behavior through direct use of data without applying predetermined rules.

Reinforcement learning is one of the branches of machine learning in which an entity called an agent learns how to select the best possible actions with the aim of maximizing the cumulative reward through continuous interaction with its surroundings. Deep learning is also one of the sub-areas of machine learning, which is a good choice for online learning due to its ability to represent the intrinsic relationships between system inputs and outputs. Therefore, it can be concluded that DRL is considered as an appropriate and good choice for automatic and online decision making in resource provisioning of IoT application problems in fog computing. In addition, the innovations of this article include:

- Selecting the best server with the minimum response time along with optimal use of bandwidth to increase network efficiency and resource utilization, as well as load balance in heterogeneous IoT network to increase customer satisfaction and QoS.
- Presenting a framework for dynamic resource provisioning using the deep reinforcement learning technique for effective use of processing node resources.
- The results of simulation tests show that the proposed method in this paper has been better than the three basic algorithms in terms of the average waiting and response time and better completion of the tasks, and making effective use of resources through appropriate distribution of tasks.

The remainder of this paper is organized as follows: In Sect. 2, we review the related research about resource provisioning in fog computing. In Sect. 3, we explain the proposed solution in more detail. In Sect. 4, the simulation results finally the conclusions and future works are presented in Sect. 5.

## 2 Related work

Deployment of cloud and edge applications, such as IoT services, must be made compatible in run time with minimum human intervention to maintain performance. Software containers can be used to simplify the deployment and management of IoT services. To identify the key features of deployment solutions available in IoT services and to summarize and organize the related approaches, a classification of different provisioning approaches in the edge environment has been presented, which is based on seven questions: why, who, when, where, which, what, and how so that the commonalities and differences of the works done in this area can be easily identified. Figure 1 shows a classification of different provisioning approaches in fog/edge environment.

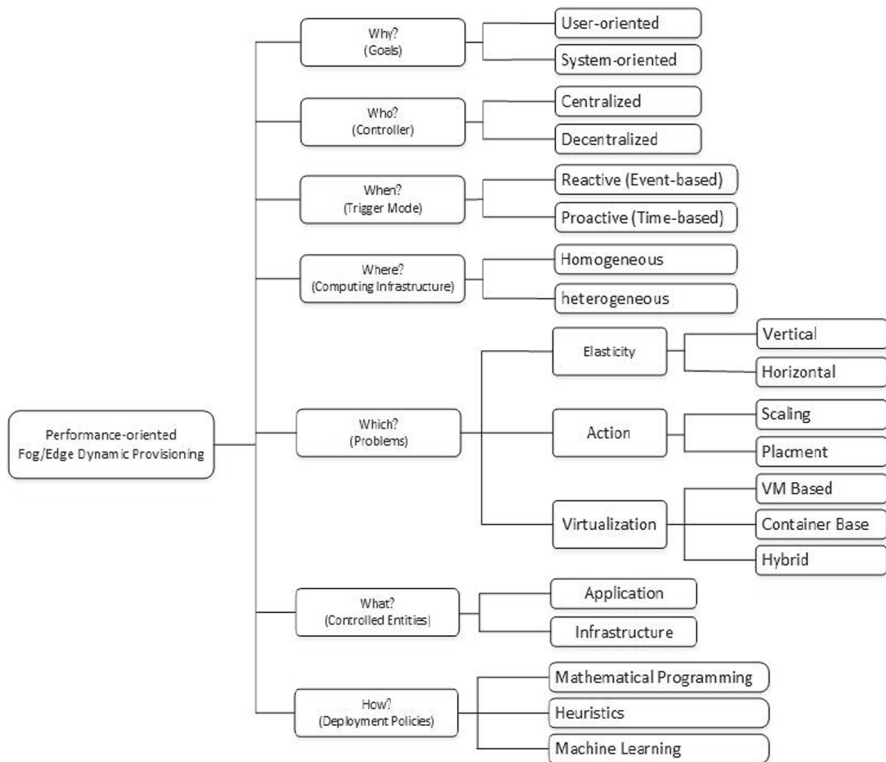


Fig. 1 Classification of different provisioning approaches in fog/edge environment

## 2.1 Why

This question is related to the estimation of quantitative criteria of optimization and target function during provisioning the required IoT services. To determine the goals of resource provisioning, several criteria have been improved in the studied researches, which are broadly classified into user-oriented (consumer-desire) and system-oriented (provider-desire) criteria. Consumers are concerned about the performance of their applications, such as response time, cost, and energy consumption, while providers are more interested efficient use of resources such as throughput and resource efficiency metrics. If a user wants to get things done faster, he has to spend more as the faster resources are usually costly. Therefore, there is always a trade-off between costs and run-time optimization. Most researches consider user-oriented criteria [14–21], while fewer researches consider a combination of consumer and provider-desire criteria [22–25].

## 2.2 Who

In most studies, resource management and deployment of IoT services are the responsibility of an external controller or synchronization tool to improve modularity and flexibility. The deployment controller is responsible for managing and deploying IoT services based on self-compatible mechanisms. This entity decides on adding or removing resources. A coordinator and centralized controller [16, 17, 19, 23] would be considered as a bottleneck for scalability and would not have the required efficiency due to latency between the components of the controller system and managed system in case IoT services are distributed in a non-centralized manner. Therefore, most studies utilize the decentralized controller [14, 22, 25–28] for the analysis and deployment of services.

## 2.3 When

This question deals with the temporal aspects of dynamic resource provisioning of fog services that are decreased or increased in proportionate to the users' demand rate. Concerning the varying running conditions of the programs and their long run time, it is necessary to implement the scalability operation at the proper time. Early resource provisioning leads to resource loss and increased cost. On the other hand, late resource provisioning will lead to SLA violations and user dissatisfaction. Provisioning deficiency will be created due to failure in provisioning required resources in relation to the average input workload, and the provisioning surplus will be created by providing more resources than the average input workload, which are both unfavorable situations. The resource provisioning time could be reactive [14–18, 22, 23, 26] and periodically or proactive [19, 20, 25, 28] or event based. In time-based reactive resource provisioning, the resource provisioning decisions are periodically made after workload behavior variation and are widely used in studies due

to simplicity. In proactive resource provisioning, complicated techniques are used for the prediction of future demands in advance to adjust resources to the predicted amount. Therefore, the reactive approaches are not able to make provisioning decisions in sudden workload traffics, and usually, proactive strategies are used to confront the workload oscillations.

## 2.4 Where

The aim of this question is to specify the modeled computing infrastructure by the resource provisioning policy. The decisions of cloud infrastructure suppliers' aim at the optimization of resources in data centers, while the supplier of IoT applications makes local decisions for each applied service to run on the virtual machines based on received workload and the cost of using services. In some studies, a homogeneous [14, 18, 23] computing infrastructure is used where the resource management problem is implemented locally and distributed in the fog/edge layer. Some other solutions in resource management consider resource heterogeneity [15–17, 19, 22, 24–27], where the specifications of geographically computing environments (such as processing capacity, resource storage, existing resources, or network latency) are better described.

## 2.5 Which

The aim of this question is to answer which types of scalability, visualization, and supplying issues persist?

### 2.5.1 (A) Elasticity

An elastic IoT application dynamically manages computational resources based on its workload, improves the availability of the programs, and limits the implementation costs. The scalability problem determines how new containers are added to the computing system or platform in the face of workload variations, and the load balancer distributes the workload among all available virtual machines (i.e., horizontal scaling), and, or more resources are added to the same running virtual machine to perform application demand (i.e., vertical scaling). The rapid response to small workload variations is usually implemented through vertical scaling and a reaction to sudden peak workload through horizontal scaling. Most studies consider only horizontal scaling operations [14, 15, 17, 18, 21–23, 25–27], and a limited number of them use a combination of horizontal and vertical scaling operations [19].

### 2.5.2 (B) Provisioning problem

Concerning the dynamic resource provisioning, the problems are divided into two scaling groups (application and virtual machine scaling) and placement (f applications and virtual machine scaling). Application scaling (AppScale) refers to the

determination of the increase and decrease in the number of applications, or the constituting units of applications (service), and virtual machine scaling (VMScale) refers to determination of increase and decrease in the number of virtual machines and their related resource such as processor and memory. Application placement (AppPlace) refers to determination of the place of the applications on a virtual or physical machines, and virtual machine placement (VMPlace) refers to the determination of the place of virtual machines on physical machines. Some studies consider the placement strategy for deployment of services [14, 15, 17, 18, 21–23, 25–27], and some of them have used scaling operations [16, 19, 20, 24, 28].

### 2.5.3 (C) Virtualization

There are two virtualization techniques: hypervisor and container virtualization. Hypervisor virtualization involves the use of the special operating system for virtualization in a physical server and capturing the main resources such as RAM, CPU, and disk in the host system, and assigning and managing resources between the operating systems that are installed as guests. A container includes all things that are required for run time, including an application and all its dependencies, libraries and configuration systems, etc. The said items are packaged as a package. Containers are lighter and use fewer resources. Most studies consider container virtualization in the fog/edge layer [15, 16, 19, 22, 28].

## 2.6 What

This question identifies the resources that should be supplied in the deployment of IoT services such as application-level (IoT services, container virtual machines) and the infrastructure-level entities (processor, memory, and network). Therefore, some of the provisioning policies act based on the manipulation of IoT applications without changing their computational resources [17, 21], and others dynamically change the computational resources at the infrastructure level [20].

## 2.7 How

This question intends to answer this question as to how be it possible to solve the provisioning problems using algorithms, techniques, and technologies. This phase is the main core of each automatic scaling method for which various soft computational techniques (such as genetic algorithm, neural networks, fuzzy logic, and queue theory), greedy algorithm, reinforcement learning, etc., can be utilized. The techniques and main technologies applied in the algorithms are VM migration, VM resizing, workload prediction, etc. In this study, the existing approaches are classified based on methods that use them, including mathematical programs [14–16, 22, 23], heuristics [17, 18, 24, 26, 27], and machine learning solutions [19–21, 25, 28].

The mathematical programming approaches utilize the software-based approaches to manage the frameworks. The mathematical solutions include integer linear programming [16], queue theory [15], and Markov-based processes



[23]. Nguyen et al. [22] presented a framework with the aim of provisioning elastic resources in container-based fog computation for resource provisioning based on the elastic fog method on the Kubernetes platform. Their focus is on solving two problems of network latency by collecting network traffic and allocating computational resources (elastic resources) in accordance with changes in demand in applications. Pereira et al. [23] programmed system use optimization and its preparation for input workload to determine the capacity in fog computing through minimization of run time. In addition, they evaluated the proposed method for running web server performance through a stochastic-based method using Markov chain. Santos et al. [14] have considered a content-based mechanism as a solution for the deployment of services in fog nodes as an ILP problem. The intended mechanism has considered the energy and latency-based limitations of video-based applications as well as QoE to download the video content in the edge and evaluate the resource supply. Stavrinides et al. [15] studied the performance of a heterogeneous fog environment and effective factors on it by effective balancing of IoT workloads and consideration of deadline limitations and scheduling challenges. The presented dynamic programming algorithm improves the system performance based on the probabilities of input data location within two stages of task selection and proper allocation of virtual machines. Dinh et al. [16] developed a computation system model where it is possible to lease multiple fogs and cloud nodes, which takes into account the decisions on resource allocation, edge processing costs, and cloud leasing options. They proposed an offline algorithm using future request information and an online decision algorithm for each loading without any future relevant information.

Heuristics as a trial and error solution is the most popular way to improve the NP-hard problems at run time, which utilizes the approximate solutions instead of exact solutions such as greedy approaches [17, 24], threshold-based exploration [27], and special designed solutions [18, 26]. Yousefpour et al. [17] designed a QoS-aware Dynamic Fog Service Provisioning (QDFSP) framework with consideration of cost, latency, and QoS needs and discussed how it could be improved for fog service providers and their customers in terms of improved QoS and cost savings. The QDFSP problem, an optimization problem with INLP formulation, and two greedy algorithms are introduced that should be periodically implemented. Porkodi et al. [18] invented a fuzzy clustering with Flower Pollination meta-heuristic algorithm as a resource provisioning technique for fog computing. The proposed model includes a three-step procedure: The first step is standardization and normalization of the resource features. The next step is fuzzy clustering for partitioning the developed resources and scalability of resource search for better exploitation of resources. Finally, the invented resource provisioning model is developed relying on the optimal fuzzy clustering. Madan et al. [26] presented a computational framework for smart cities with the aim of resource provisioning based on demand for the mobile networks using Flying Fog which is handled through introducing the allocation and resource provisioning concepts in the form of fog units. The lease period for the allocated resources is defined based on the preventive resource provisioning model. Bahreini et al. [24] pursued the aim of minimizing the energy consumption in edge computational systems and resource allocation so that the net profit of the service provider will be maximized. The problem is formulated as MILP, and the evaluation

is done using a meta-heuristic algorithm through extensive experimental analysis on the samples of the problem. Siasi et al. [27] proposed an SFC-based provisioning method in fog-cloud hybrid architecture by studying the concept of service performance chains in fog and cloud computing nodes. To meet the different latency requirements, they used a heuristic search method to balance the network latency, resource consumption, energy consumption, and cost in high traffic volumes.

In machine learning, scaling decisions are based on learning techniques such as reinforcement learning [19, 20], neural networks [21], and fuzzy logic, which is a combination of artificial intelligence, statistics, and mathematics that through an agent learn to make appropriate decisions through interactions with the environment. Faraji et al. [19] provided a distributed computational framework for autonomous resource management for time-varying work flows in which the resource provisioning self-management system operates through a reinforcement learning approach and uses a backup vector regression technique to predict future resource requirements in the fog computing environment. Abdollah et al. [28] proposed an automated predictive scaling method for microservices that are running on Fog MDC to meet the response time of the SLO program. The proposed approach uses an automated scalability method based on reactive rules to collect training data sets to construct a predictive automated scale model through preprocessing and post-processing steps. In fact, this approach learns the predictive automated scalability model through tree regression by increasing the artificial workload. Liu et al. [25] carried out a study with the aim of managing the elastic resources based on error-correction workload prediction in edge/cloud environment in response to workload variations due to users' requests. In fact, they acted based on workload prediction (combination of ARMA and ENN) and used an error-correction model to improve the prediction precision and workload migration model to reduce the migration time. Kim et al. [20] performed a study intending to optimize the energy output using reinforcement learning in the edge. This paper proposes an adaptive and lightweight scaling engine based on a custom-designed reinforcement learning algorithm. Al-Makhadmeh et al. [21] designed an efficient scalability resource allocation structure based on deep learning and an effective framework for balancing the user density and their requests based on available resources and thereby maximizing the reliability of IoT services and user satisfaction.

Table 1 presents the comparison of the research conducted on resource provisioning in fog computing based on the proposed classification.

### 3 Proposed approach

Resource provisioning and allocation to provide in fog computing, users want their requests not to go to the cloud as much as possible and to be answered close to the edge, or fog node with a minimum guarantee of service quality and minimum cost and the service providers seek for return of the maximum amount of profit and capital. As the availability of resources and workloads change dynamically, this dynamic behavior poses challenges for service providers and users. The two main issues of service quality requirements (such as reliability, latency, cost, supply, and proper

**Table 1** Comparison of the related works deployment solutions

References	Goals	Deployment policies (utilized technique)	Trigger mode (method)	Problems (controlled entities)	Advantages	Weaknesses
Nguyen et al. [22]	Latency, throughput	Mathematical (container-based)	Reactive (time-based)	Placement (container)	Detailed analysis	Algorithm 1 quickly crashes in a cluster with many running applications Weak organization, Weak presentation
Pereira et al. [23]	Throughput, response time	Mathematical (Markov chain)	Reactive (event-based)	Placement (container)	Appropriate and simple proposed closed-form	Ignoring some important network factors such as jitter and packet loss for the sake of comprehensiveness
Santos et al. [14]	Cost, QoE	Mathematical (AHP)	Reactive (time-based)	Placement (application)	Comprehensive system model	Incomprehensive structure due to the lack of predefined platforms such as cloud to deal with resource-intensive tasks that fog system cannot fulfill
Stavrinides et al. [15]	Response time	Mathematical (queuing model)	Reactive (event-based)	Placement (container)	Priority policy for high priority tasks	The high complexity of proposed online and offline allocation algorithms in overloaded situations
Dinh et al. [16]	Cost	Mathematical (polynomial approximation)	Reactive (time-based)	Scaling (container)	Excellent mathematical presentation for the allocation algorithms	

Table 1 (continued)

References	Goals	Deployment policies (utilized technique)	Trigger mode (method)	Problems (controlled entities)	Advantages	Weaknesses
Yousefpoor et al. [17]	Delay, cost, delay violation	Heuristics (greedy)	Reactive (time-based)	Placement (application)	Appropriate proposed architecture	The high complexity of the proposed cost algorithm when the delivered services' rate is high
Porkodi et al. [18]	QoE	Heuristics (optimal Flower)	Reactive (time-based)	Placement (application)	Simple system model	High probability of local minimal in the proposed fuzzy clustering method, Sensitive to noisy data and outliers
Madan et al. [26]	Energy, latency	Heuristics (distance-based)	Reactive (event-based)	Placement (application)	Appropriate system model	High UAV costs in overload situations, weak presentation, weak analysis
Bahreini et al. [24]	Energy, profit	Heuristics (iterative-based)	Reactive (event-based)	Scaling (container)	Simple system model	The high complexity of algorithms, not realistic in the presence of the enormous problem's size
Siasi et al. [27]	Energy, delay, cost	Heuristics (graph-based)	Reactive (time-based)	Placement (container)	Effective organization, In-depth analysis	The high complexity of provisioning scheme in peak hours
Faraji et al. [19]	Cost, delay violation	Machine learning (RL, SVR)	Proactive (time-based)	Scaling (container)	A comprehensive presentation of the MAPE-K loop model	The high complexity of the planning phase when the workload is increased

**Table 1** (continued)

References	Goals	Deployment policies (utilized technique)	Trigger mode (method)	Problems (controlled entities)	Advantages	Weaknesses
Abdullah et al. [28]	Response time	Machine learning (DTR)	Proactive (event-based)	Scaling (container)	Comprehensive analysis	Incomprehensive system model, prone to forecast negative values for near-zero data requests because of the applied regression model
Liu et al. [25]		Machine learning (ARMA, ENN)	Proactive (time-based)	Placement (container)	Appropriate proposed architecture, good presentation	The accuracy of the proposed model is exceptionally susceptible to outliers of historical data
Kim et al. [20]	QoS, Delay	Machine learning (DRL)	Proactive (event-based)	Scaling (infrastructure)	The DRL approach need not any prior knowledge to learn	As a type of bandit method, the epsilon-greedy algorithm does not measure the variance to avoid local optima
Makhadmehet al. [21]	Delay, response time	Machine learning (deep learning)	Reactive (time-based)	Placement (application)	Simple proposed mode	In the case when there are several simultaneous incoming requests, there is not included any priority queue for high priority requests

AHP Analytic hierarchy process, RL, reinforcement learning, SVR, support vector regression, DTR, decision tree regression, ARMA, auto-regressive moving average, ENN, Elman neural network

resources allocation) and the acceptable level of efficiency and productivity of the system are among the factors that must be considered to maintain the satisfaction of parties. The mathematical models confront an increase in the variables and constraints with an increase in the search space of the problem, and solving the problem of service deployment using traditional methods requires a lot of time and money, which contradicts the goals of fog computing in the face of real-time applications.

### 3.1 The proposed framework

In this section, a resource provisioning framework will be presented for the implementation of the proposed approach. As seen in Fig. 2, the proposed approach is taken from three-layer fog computing architecture. The first layer includes IoT devices and sensors. All smart devices, including mobile phones, sensors, and tablets, are supported in this layer. Not only does fog provide services based on Ad Hoc, but also estimates resource consumption and can allocate the estimated resources. In this layer, each device produces layers that might need some processing, storage, and computational operation. The users' requests are entered into the access points and then into the admission control component. This deadline component compares each

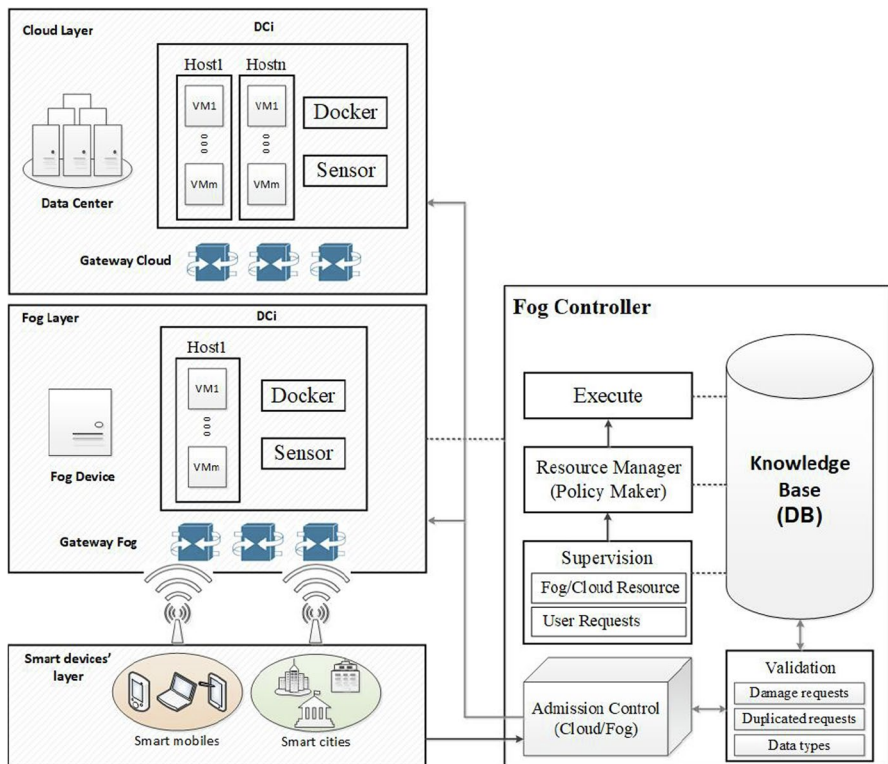


Fig. 2 The proposed approach

request to the threshold value and sends it to the gateway for processing in the cloud if above the threshold value (the processing is not real-time and latency-sensitive). Otherwise, the processing is real-time, and it is required to be sent to the fog layer for quick processing. The fog layer is responsible for resource provisioning and fog node allocation for operating on the data produced by IoT devices. The cloud layer consists of virtual machines of cloud data centers that big data or non-time-sensitive applications are transferred to for processing purposes. The main component of the proposed resource provisioning framework is responsible for controlling all three layers in the proposed framework.

The proposed controlling component automatically and efficiently manages the fog layers and is consisted of the following components:

**Supervision:** This component is responsible for collecting the criteria that are related to IoT devices and fog and cloud layers. The supervision components can get the required data (such as CPU, storage, and required RAM for data processing) through communicating with the IoT layer. It can also collect the resource data (such as CPU efficiency, use of storage, network traffic, and active/inactive resources) through communicating with fog devices or data centers.

**Validation:** This component has the responsibility of investigating the data taken from the supervision component. This component includes a data type checker module, duplicate, and damaged requests.

**Knowledge Database:** DB module is responsible for maintaining input data, including IoT device-level requests, fog device, and cloud data center data. The validation component updates this data after review.

**Admission control:** The admission manager module is responsible for verifying the location of processing the requests received from IoT devices using data stored on the DB. If the request deadline exceeds a certain threshold, the request should be sent to the cloud layer. Taking such a decision, the data resources management module sends the request to execute module to send it to cloud data center for processing and storage. Otherwise, the request shall be processed in the fog layer by sending it to the Policy Maker component.

**Resource manager:** This component is responsible for managing and decision making about resource provisioning in the fog layer and processing the received requests through Admission Manager Module using the DRL technique. Then, it decides how to provide the resources concerning the received request.

**Execute component:** It is responsible for the execution of decisions taken the Policy Maker module. This module specifies that the request shall be processed in the fog or cloud layer through the admission manager. In addition, the resource manager determines the number of required resources for the requests sent to the fog layer. Finally, the requests are sent to the destination to be executed.

**Table 2** Notations and definitions

Parameter	Description
$\tau$	The time interval between two rounds (in milliseconds)
$A$	Set of application servers hosted on containers in fog nodes. $a_i \in A, i = 1, \dots, n$
$Thr_i$	Maximum time value need to process on cloud servers or fog environment for $a_i$
$QoS_i$	Desired quality of service for $a_i$
$U_{a_i}$	CPU demand of the $a_i$ (in million instructions per request)
$M_{a_i}$	Memory demand of the $a_i$ (in byte)
$S_{a_i}$	Storage demand of the $a_i$ (in byte)
$U_{f_j}$	The processing power of the fog node $j$ , in MIPS
$M_{f_j}$	Memory capacity of the fog node $j$ , in bytes
$S_{f_j}$	Storage capacity of the fog node $j$ , in bytes
$C_{FN}^U$	Unit cost of process at fog node $j$ (per million instructions)
$C_{FN}^S$	Unit cost of storage at fog node $j$ (per byte per second)
$C_{FN}^M$	Unit cost of main memory at fog node $j$ (per byte per second)
$C_{CN}^U$	Unit cost of process at cloud node $k$ (per million instructions)
$C_{CN}^S$	Unit cost of storage at cloud node $k$ (per byte per second)
$C_{CN}^M$	Unit cost of main memory at cloud node $k$ (per byte per second)
$\lambda_{ik}^{in}$	Incoming traffic rate to cloud server $k$ from IoT nodes for service $i$ (request/second)
$\lambda_{ij}^{in}$	Incoming traffic rate to fog node $j$ from IoT nodes for service $i$ (request/second)
$\lambda_{ijk}^{out}$	Rate of outcome traffic from fog node $j$ to cloud server $k$ for service $i$ (request/second)
$\lambda_{ijj'}^{out}$	Outcoming traffic rate for service $i$ from fog node $j$ to fog node $j'$ (request/second)
$cc_{(s,d)}$	Cost of communication link (s,d) per unit bandwidth/second
$rt_{(s,d)}$	Rate of transmission (bandwidth) of link (s,d)
$pd_{(s,d)}$	Propagation delay of link (s,d)
$Size_i^{req}$	Average request length of service $i$ , in byte
$Size_i^{res}$	Average response length of service $i$ , in byte
$\beta_a^k$	BinaryvariableshowingifServiceahostedoncloudserverk
$\alpha_a^j$	BinaryvariableshowingifServiceahostedonfognodej
$\gamma_a^j$	BinaryvariableshowingifServiceaDeployedonFognodej
$E_{FN}^U$	Unit communication energy consumption of a fog node
$MDE_{f_j}$	The maximum dynamic energy consumption of the fog node $j$

### 3.2 Problem formulation

To model the fog infrastructure network, we use an undirected graph  $G = (N, A)$  for representing the communication network between resources in the fog ecosystem, so that the node set  $N$  includes a set of resources and the edge set  $A$  includes the communication links between the resources, such that the communication link delay (in ms) depends on the propagation delay, bandwidth, queuing delay, and the network traffic condition. This section provides the notations and performance metrics used in the proposed method, as shown in Table 2. According to the QoS requirements of IoT



services and the computation capabilities of fog nodes, the IoT services can be placed on a series of fog nodes. The main decision variables of the optimization problem are the placement binary variables, defined below:

$$\beta_a^k = \begin{cases} 1 & \text{if Service } a \text{ hosted on cloud server } k \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\alpha_a^j = \begin{cases} 1 & \text{if Service } a \text{ hosted on fog node } j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Note that the fog service manager periodically calculates the IoT service placement plan according to objective functions in certain  $\tau$  time intervals.

### 3.2.1 Performance metrics

In this paper, we consider three kinds of objective functions, namely throughput, cost, and energy consumption.

**3.2.1.1 (A) Throughput** The first objective is throughput metric or service acceptance ratio, that means the maximization of deploying the number of IoT services on the fog nodes (instead of cloud servers) while meeting the QoS requirements of each IoT service, as expressed by Eq. (3):

$$T = \sum_{j \in FN} \sum_{a_i \in A} \alpha_{a_i}^j \tag{3}$$

**3.2.1.2 (B) Addressing cost model** The second objective is the cost, which means the minimization of the total cost for executing IoT service requests. The cost optimization problem can be formulated as problem C

$$\begin{aligned} C = & \text{Cloud Computational Cost(C1)} \\ & + \text{Fog Computational Cost(C2)} \\ & + \text{Communication Cost(C3)} \\ & + \text{Deployment Cost(C4)} + \text{Penalty Cost(C5)} \end{aligned}$$

The cost components details are defined below.

$$C1 = \sum_{k \in CN} \sum_{a_i \in A} (C_{CN}^U \times U_{a_i} \times \lambda_{ik}^{in} + C_{CN}^S \times S_{a_i} + C_{CN}^M \times M_{a_i}) \times \beta_{a_i}^k \times \tau \tag{4}$$

$$C2 = \sum_{j \in FN} \sum_{a_i \in A} (C_{FN}^U \times U_{a_i} \times \lambda_{ij}^{in} + C_{FN}^S \times S_{a_i} + C_{FN}^M \times M_{a_i}) \times \alpha_{a_i}^j \times \tau \tag{5}$$

$$C3 = \left( \sum_{j \in FN} \sum_{j' \in FN} \sum_{a_i \in A} cc_{(j,j')} \times \lambda_{ijj'}^{out} + \sum_{j \in FN} \sum_{k \in CN} \sum_{a_i \in A} uc_{(j,k)} \times \lambda_{ijk}^{out} \right) \times (Size_i^{req} + Size_i^{res}) \times \tau \tag{6}$$

$$C4 = \sum_{j \in FN} \sum_{a_i \in A} uc_{(FRP,j)} \times S_{a_i} \times \gamma_{a_i}^j \times \tau \tag{7}$$

$$C5 = \sum_{j \in FN} \sum_{a_i \in A} \max(0, Violation_i^{\%} - (1 - QoS_i)) \times \lambda_{ij}^{in} \times Penalty_i \times \tau \tag{8}$$

where cloud resources cost and fog resource cost are the total cost for all cloud servers and fog nodes for serving IoT requests into cloud and fog tier, respectively. The communication cost is determined according to the cost of communication links between fog nodes and communication links between fog nodes and cloud nodes when service offloading to the cloud. Deployment cost is the summation of service deployment from fog service manager module in the cloud to fog. The penalty cost for each service is the penalty that the fog service manager should pay per request of service if the delay requirements are violated.

**3.2.1.3 (C) Energy consumption model** The third objective is energy consumption, and it depends on the computation and communication energy of the fog nodes.

The energy consumption optimization problem can be formulated as problem E:

$$E = \text{Communication Energy Consumption(E1)} + \text{Computational Energy Consumption(E2)}$$

The energy consumption components details are defined below.

$$E1 = \sum_{j \in FN} \sum_{j' \in FN} \sum_{a_i \in A} E_{FN}^U \times \lambda_{ijj'}^{out} \times \alpha_{a_i}^j \times \tau \tag{9}$$

$$E2 = \sum_{j \in FN} EC_{comp}^{Static} + \sum_{j \in FN} EC_{comp}^{Dynamic} \tag{10}$$

$$E3 = EC_{comp}^{Static} = \begin{cases} \delta iff_j \text{ is active} \\ 0 \text{ otherwise} \end{cases} \tag{11}$$

$$E4 = EC_{comp}^{Dynamic} = U_{f_j} \times MDE_{f_j} \times \tau \tag{12}$$

The communication energy consumption is specified by the service delay and the geographical locations between two fog nodes. Besides, the dynamic energy consumption of a fog node is determined as a linear equation based on the CPU utilization of the fog node and the maximum dynamic energy consumption. If the CPU utilization of the fog node is 100%, the amount of the dynamic energy consumption will be maximum; otherwise, it will be calculated based on the CPU utilization in the current time interval.

### 3.2.2 Constraints

Given the resource capacity of cloud servers and fog nodes, the use of fog node resources and cloud servers should not exceed their capacity, as formulated by the following equations:

$$\sum_{a_i \in A} \alpha_{a_i}^j \times S_{a_i} < S_{f_j}, \forall j \in FN \quad (13)$$

$$\sum_{a_i \in A} \alpha_{a_i}^j \times M_{a_i} < M_{f_j}, \forall j \in FN \quad (14)$$

$$\sum_{a_i \in A} \beta_{a_i}^k \times S_{a_i} < S_{c_k}, \forall k \in CN \quad (15)$$

$$\sum_{a_i \in A} \beta_{a_i}^k \times M_{a_i} < M_{c_k}, \forall k \in CN \quad (16)$$

$$\sum_{a_i \in A} \alpha_{a_i}^j \times U_{a_i} < U_{f_j}, \forall j \in FN \quad (17)$$

$$\sum_{a_i \in A} \beta_{a_i}^k \times U_{a_i} < U_{c_k}, \forall k \in CN \quad (18)$$

$$\beta_{a_i}^k \in \{0, 1\}, \forall a_i \in A, i = 1, \dots, n \text{ and } k \in CN \quad (19)$$

$$\alpha_{a_i}^j \in \{0, 1\}, \forall a_i \in A, i = 1, \dots, n \text{ and } j \in FN \quad (20)$$

### 3.3 Deep reinforcement learning-based resource provisioning policy

Reinforcement learning is one of the machine learning techniques where the learning agent is rewarded (along with delays) after evaluating each action. The reinforcement learning mechanism in the dynamic resource provisioning problem consists of two decision-making elements (reinforcement learning algorithm) and environment (IoT services ecosystem and fog computing resources). Initially, the environment creates a situation (determining the initial amount of resources for each service) for the agent to react to it based on its knowledge (increase of resources, decrease of resources, appropriate resources). Then, the environment simultaneously sends the next status and reward of the previous action to the agent. The decision maker updates its knowledge based on the reward that it has got for its previous action. This cycle continues until the environment sends the

final status to the agent and ends the cycle. In most reinforcement learning methods, we analyze the changes in the results to see how the results will change with our decisions. This can be done with a better understanding of the system dynamics or through smart trial and error to determine which decisions work best. In recent years, reinforcement learning algorithms have evolved as a result of merging with neural networks, so that the speed of the learning process has improved in order to solve complex problems. In deep reinforcement learning, the ability to understand deep learning is well combined with the ability to make reinforcement learning decisions. Reinforcement learning uses a table to store action values and concerning the complexity and diversity of the environment status, the use of a table for storing all values of action is impractical, and the continuous search of the related status in a big table is time-consuming. The proposed deep reinforcement learning directly utilizes the neural network with parameter  $\omega$  to approximate Q function and produce action values. The main components of the DRL model are completely shown as follow:

#### (A) Reinforcement learning

*State (S)*  $s_t$  Status includes full information about the ecosystem of IoT services in t time stage and shows the decision-making agents. An example of a system includes the current allocation of resources and IoT services requests such that a sequence of resource allocation to IoT services leads to the system status. We need to provide a suitable status to act as the neural network input, and because of several service requests at a time, we select the first few requests as a small batch for resource provisioning in order to maintain a consistent state display. In addition, the presence of large possible combinations of allocation sequences leads to an MDP model with a different number of states. Based on the predefined objective, status refers to the configuration of various resources such as CPU values and memories.

*Action (A)* An action leads to the allocation of the required resources (allocation of the selected container) to the request of incoming IoT services. Every IoT service has requirements, and if the selected container has resources greater than or equal to these requirements, that container is considered a valid choice otherwise, it is considered invalid. After each action, a transition from the current state in t time step to a new state  $s_{t+1}$  occurs in the time step t + 1. While provisioning the incoming IoT services, a large number of actions are possible in the operating space, which in turn complicates the entire work of provisioning. During training, DRL-based resource provisioning tries to learn the distribution of actions along with weights and biases known as rewards. An action refers to setting up a unit of a single resource, such as increasing a CPU unit.

*Reward (R)*  $r_t = (s_t, a_t, s_{t+1})$  Reward is the immediate feedback of the system after evaluating each action of the decision maker and changing the current state of the system by the environment. For each action of the agent and the extent of its effects on the performance of the network, a positive or negative reward is given to the agent, which indicates the extent to which an agent performs well in stage t. Our goal is to minimize cost, average response time, and overall energy consumption. The combination of IoT ecosystem performance metrics

(monitoring metrics) indicates the state of the system. In the proposed method, the reward function is based on a combination of cost, efficiency, and energy consumption parameters, the ultimate goal of which is to maximize future cumulative rewards following the selected policy.

*Policy* ( $\pi$ ) policy is a strategy that represents a state-of-action mapping that shows the probability distribution over actions in each state. Each mode in the IoT ecosystem provides information about system monitoring metrics (cost, response time, and power consumption), and the values of these metrics change as the system moves to the next mode after performing an action. The result of criteria monitoring feedback is an action that leads to the current status of the system. For each action of resource allocation, final reward feedback is assigned to that particular action. The goal is to maximize future cumulative rewards based on feedback from consecutive actions:

$$r_t = \sum_{i=0}^{\infty} \gamma^i r_{a_i}(s_t, s_{t+1}) \tag{21}$$

where  $0 \leq \gamma \leq 1$  is a discount factor or delayed reward parameter ( $\gamma = 0.99$  default). Different actions in each situation (mapping the resources required for IoT services) lead to a large number of possible policies. In this paper, we use a deep neural network (DNN) as a function approximation to approximate the probabilities of possible actions and create an approximate probability distribution of a state-action pair.

(B) Deep Neural Network (DNN)

The resource allocation policy, which changes at different times, can be approximated using an approximate trained network (DNN) that has its own parameters. The approximate DNN network takes the status of complete information about the IoT service ecosystem as the network input and generates the probability of selecting the necessary actions using  $\theta$  policy parameters as the weight and bias of the output network. A basic policy production model is shown in Fig. 3.

Our proposed algorithm creates an initial policy or action-practical relationship for each set of requests. By interacting with the system environment, the

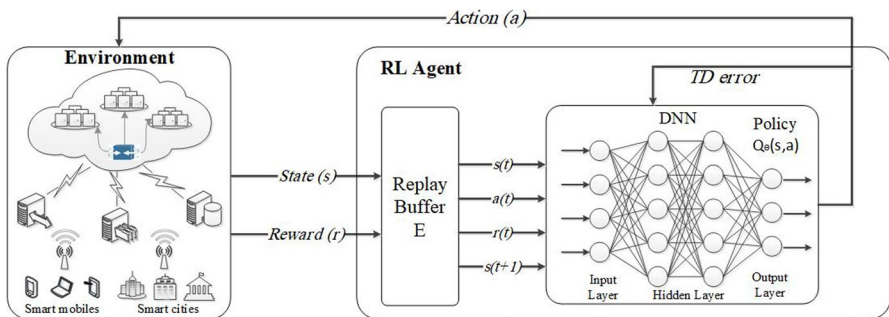


Fig. 3 Policy production models

DNN senses the current state of the system and examines possible future situations by setting parameters following a process of rewards maximization. Therefore, it learns the optimal policy through multiple iterations so that better measures can be taken for resource provisioning. As resource provisioning progresses, a database is created used to further improve the parameter  $\theta$  policy and supply new requests based on this updated policy parameter. The descending gradient algorithm is used to update the weights to select beneficial actions. In this way, after several training repetitions, the optimal weights and biases are obtained, which maximizes the cumulative reward generated during resource provisioning.

---

**Algorithm 1:** Deep RL – based resource provisioning
 

---

```

1: Begin
2: Stage 1: /* initialize parameters */
3: initialize experience replay memory E to capacity M
4: initialize Q – network with random weights  $\theta$ 
5: set target DNN network weights  $\hat{\theta}$  equal to  $\theta$ 
6: set values for parameters: discount factor( $\gamma$ ), batch size, update step (U), exploration rate ( $\epsilon$ )
7: Stage 2: /* initialize parameters */
8: for episode = 1 to end;
9: observe current state  $s_j$ 
10: Generate a random number p between 0 and 1
11: if  $\epsilon > p$  :
12: The controller randomly selects an action
13: else :
14: The controller selects the action :  $a_t = \arg \max(Q(s_j, a, \theta))$ 
15: end if
16: execute the action  $a_t$ 
17: transfer to the new state  $s_{j+1}$ 
18: save the reward r
19: store the tuple (sj, action, sj + 1, r) in E
20: draw a batch from E
21:  $y = Q(s_j, \theta)$ 
22:  $y_{target} = \hat{Q}(s_{j+1}, \hat{\theta})$ 
23:  $y_{action} = r + \gamma \cdot \max(y_{target})$ 
24: execute one training step on main DNN network
25: every U steps set  $\hat{\theta} = \theta$ 
26: End
  
```

---

The proposed deep RL approach is shown in Algorithm 1. The first part of the algorithm sets the initial values of the parameters. Line 3 is allocated to setting up playback memory E, which is a data structure containing the experiences created by the agent and stores all data needed for DNN training. First, two neural network weights are set to the same values (lines 4–5); then, the other parameters are set to line 6. In each iteration of for loop, the agent observes the current state (line 8) and selects an action to perform that depends on the exploration rate (–greedy), which leads to a balance between exploration strategies and exploitation of previous experiences (line 10). Exploration identifies a randomly selected action (line 11). In action, the most appropriate action in the current state is returned by the DNN prediction network (line 14). One of the challenges in this problem is the proper distribution of services among virtual machines in a way that the proper

and effective use of these resources is done. Exploration strategy in (*-greedy*) politics will overcome this challenge by choosing random actions. After performing the action, the agent again observes the situation obtained by the environment and the corresponding reward. (Lines 16–18) After storing the experience created by the agent in the replay memory E (line 19), this is the point where deep learning is entered. First, a nonlinear approximation (main Q network) predicts Q values for the given state  $s_j$  (line 21). Then, the target Q values are evaluated via the target DNN network (line 22) and used in the Belman Update formula (line 23). After upgrade, the main DNN network is trained by performing a training step on the cost function (line 24), and after U step of execution, the weight of the target DNN network is adjusted to the weight of the main DNN network (line 25).

### 4 Performance evaluation

In this section, the results of implementing the FRP\_DRL method for resource provisioning based on deep reinforcement learning in a fog environment will be investigated. iFogSim is used for accurate and extensive simulation of the fog, which provides the possibility of modeling and simulating the fog computing environment. Using this tool, it is possible to evaluate the resource management and resource provisioning policies in edge and cloud computing under different scenarios. This simulator can evaluate the resource management policies by focusing on their effect on latency, energy consumption, operational cost, and utilization. This tool supports the devices in edge, computing cloud, and network links for evaluation of productivity. The main applied model which is supported in iFogSim is Sense-Process-Actuate. In this model, the sensors monitor data produced in IoT, follow sand processes the programs executed in fog tools, and at the end, the final decisions will be transferred to data transfer space [29, 30].

#### 4.1 Simulation setting

The simulation setting used in the simulation is presented in Table 3. The average size of the request is set on U (10: 26) KB, the application response on U (10:20) KB, and the

**Table 3** The specifications of each fog domain

Parameter	Description	Parameter	Description
$Thr_i$	10ms	$C_{FN}^U, C_{CN}^U$	0.002 perMI
$QoS_i$	U(90, 99.99)%	$C_{FN}^S, C_{CN}^S$	0.004 perGbps
$U_{a_i}$	U(50, 200)Mlperreq	$C_{FN}^M, C_{CN}^M$	0.004 perGbps
$M_{a_i}$	U(2, 400)MB	Core, Edgelinek	10 Gbps, 1 Gbps
$S_{a_i}$	U(50, 500)MB	$pd_{(j,k)}$	U(15, 35) ms
$U_{f_j}$	U(800, 1300)MIPS	$pd_{(IoT,k)}$	U(1, 2) ms
$M_{f_j}$	U(2, 400)MB	$Size_i^{req}$	U(10, 26) KB
$S_{f_j}$	$\geq 25MB$	$Size_i^{res}$	U(10, 20) KB

required value for processing services is set on U (50: 200) per request. The QoS level is assumed different for applications, which is a random number between U(90, 99.99)%. As far as fog services are latency-sensitive, the penalty for violating the latency threshold is a random number in U (10: 20). The processing capacity of each fog node is assumed as equal to U (800; 1300) MIPS, the processing capacity of each cloud server to 20 times the MIPS fog nodes U (16–26 K). The storage capacity of fog nodes is above 25 GB, and the storage capacity of cloud services is 10 times the fog nodes. The memory capacity of fog nodes and cloud servers is 8 GB and 32 GB, respectively. The propagation delay between IoT nodes and fog nodes is U (1; 2) microseconds, and between fog nodes and cloud servers are U (15; 35) microseconds. The communication path between IoT nodes and fog nodes is 1 Gbps links and between fog nodes and cloud servers is 10 Gbps. The cost of communication between fog nodes is 0.2 per Gb, and between cloud servers and fog nodes are 0.5 per Gb. The processing cost in fog nodes and cloud servers is 0.002 per MI. The storage cost in fog nodes and cloud servers is 0.004 per GB per second [17].

## 4.2 Simulation metrics

The main evaluation metrics are: mean utilization, average response time, average cost, and mean energy consumption.

- **Utilization:** The amount of MIPS allocated to services in the fog node divided by MIPS available in each node is called utilization.

$$\text{Utilization} = \frac{\text{Allocated\_MIPS}}{\text{Available\_MIPS}} \quad (22)$$

- **Response time:** The total deployment time (propagation and transmission delay) and mean execution time (processing delay) determine the service time or response time.

$$W = \text{UploadDelay}(W1) + \text{Transferringdelay}(W2) + \text{Expectedprocessingdelay}(W3) \quad (23)$$

where  $w1 + w2$  is deployment time and  $W3$  is execution time of service in fog node.

- **Cost:** Cost is allocated to the node in respect to duration of running a service in the node as shown in Eq. (24).

$$C = \text{Computational Cost}(C1) + \text{Communication Cost}(C2) + \text{DeploymentCost}(C3) + \text{Penalty Cost}(C4) \quad (24)$$

where  $c1$  is computational cost,  $c2$  is communication cost,  $c2$  is deployment cost, and  $c4$  is penalty cost.

- **Energy Consuming:** The communication energy consumption is determined based on the service latency and the geographical location between two fog nodes. In addition, the computational energy consumption of a fog node is determined in the form of a linear equation based on using CPU of fog node and maximum dynamic energy consumption.



$$E = \text{Communication EnergyConsumption}(E1) + \text{ComputationalEnergyConsumption}(E2) \quad (25)$$

### 4.3 Workload

The type of artificial workload used in stimulation is a fast variable. The varied fast workload is constituted from 250 requests to 1350 services. The way this load is formed begins with defining several random peaks. The peak is then considered as the threshold, and the number of services is randomly increased to the peak and is randomly decreased to a certain amount in a similar way. The average number of services in this workload is 720.

### 4.4 The base methods for comparison

The base algorithms used for the evaluation of the performance of the proposed method are FRP\_LA (Fog Resource Provisioning Learning Automata) and FRP\_RL (Fog Resource Provisioning Reinforcement Learning).

**FRP\_LA[31]:** The approach is a combination of the Learning Automata and resource provisioning framework proposed in this study. The approach first uses time series prediction methods in the analysis phase and then the learning automation method in the planning phase. The reason for choosing this approach is that it is one of the reinforcement algorithms, and is used in environments with variable requests and has a good convergence time.

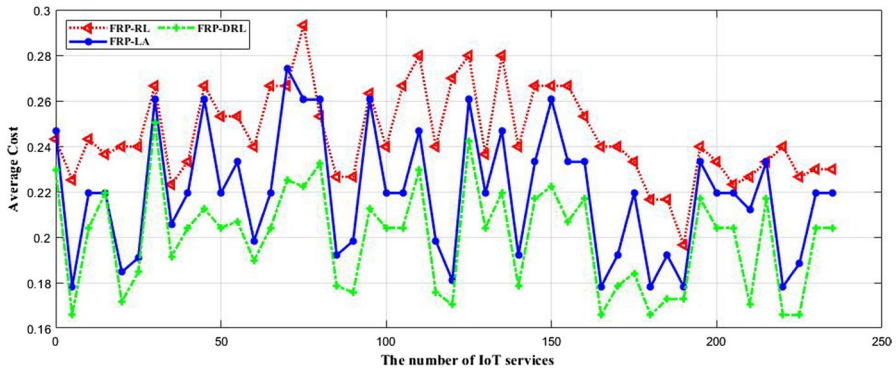
**FRP\_RL[32]:** In this algorithm, the service allocation in the fog environment is first random, and then in the planning section, the reinforcement learning method is used. As far as in the policy exploration section, the choices are just randomly made. It is necessary to compare the performance of the proposed method with this algorithm to determine the intelligence of the agent and the effect of learning from previous experiences in the performance of the agent.

### 4.5 Evaluation and results

Four scenarios have been considered for evaluation of the proposed method. In the stimulation of each scenario, one main evaluation criterion in the proposed algorithm will be investigated with FRP\_LA and FRP\_RL algorithms.

#### 4.5.1 First scenario: cost comparison

In this experiment, the cost in the proposed algorithm will be investigated with FRP\_LA and FRP\_RL algorithms. The main metric in comparison with resource provisioning algorithm performance is its cost. Figure 4 shows the cost of the proposed method in variable workload in comparison with FRP\_LA and FRP\_RL algorithms. The results indicate that using decision-making process of deep

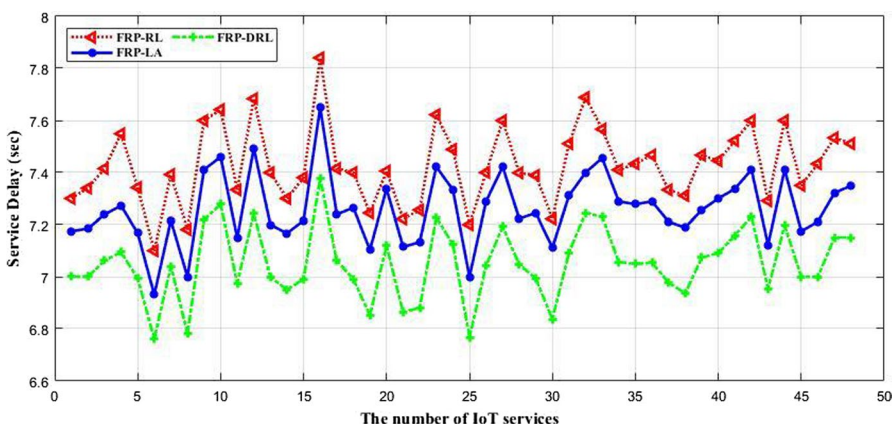


**Fig. 4** The cost of variable workload

reinforcement learning and creating appropriate limits for resource provisioning in the node will increase the precision. However, the main reason for cost reduction is the use of an appropriate immediate monitoring systems for getting the nodes' information, which increases the precision in resource provisioning in fog.

#### 4.5.2 Second scenario: comparison of response time

Response time, as one of the most effective objectives of service, has the main role in resource provisioning. If the desired response time of service (allowed response time) is not achieved, the resource provisioning must be performed once again. Figure 5 shows the average response time of the variable workload method compared to FRP\_LA and FRP\_RL algorithms. The results show that the proposed method has the desired performance in terms of response time. The reason is that the response time is equal to the time interval between the moments of arrival to exit of the request. This interval



**Fig. 5** Average response time of variable workload

includes waiting time as well as the execution time of the request, and as far as the execution and waiting times are part of the target function of the proposed algorithm in this paper, the lowest response time is achieved. In the proposed resource provisioning method, the provisioning will be improved due to multi-objective deep reinforcement learning process and the use of two important input variables, which are load and cost, since if the resource provisioning is not appropriately done, there will be a service without access to the node and therefore the response time to that service will increase. On the other hand, when choosing a physical machine whose virtual machines have less memory, in case it is overloaded, its virtual machines will migrate faster, which in turn, will reduce the violation of SLA.

#### 4.5.3 Third scenario: comparison of the utilization

Figure 6 shows the average utilization of the proposed method in variable workload structure in comparison with FRP\_LA and FRP\_RL algorithms. The results show that the proposed method has the desired performance in terms of productivity. Precise monitoring of the performance of the fog nodes and continuous detection of workload is very effective factor in controlling the status of the fog cells. Due to monitoring of the status of the nodes and the use of decision makers based on deep reinforcement learning, provisioning is properly done. The results show that in the proposed method, the number of migrations has been less than two others at all times by proper identification of the overloaded physical machines and also selecting the appropriate servers with the possibility of higher free processor capacity to locating the migrating virtual machines.

#### 4.5.4 Fourth scenario: comparison of energy consumption

Figure 7 shows the average energy consumption of the proposed method in variable workload structure in comparison with FRP\_LA and FRP\_RL algorithms. Upon receipt of the request, the provider decides on how to supply the resources based on

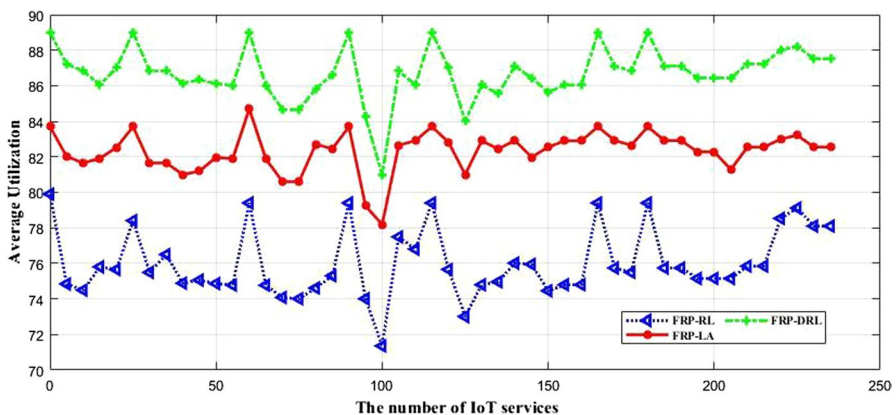
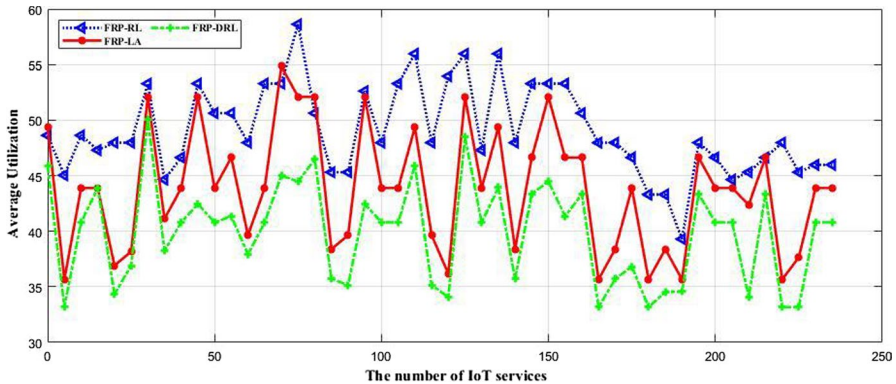


Fig. 6 Average utilization of variable workload



**Fig. 7** Average energy consumption of variable workload

users' requests to maximize profit. The operational cost of a node is proportionate to its consumed energy, which is estimated through a linear function of CUP, memory, and disk. Therefore, energy consumption is calculated through operational cost as the target function. The proposed algorithm of this study distributes the requests in a more effective way and makes the service distribution among virtual machines in such a way that their idle time decreases and resource consumption increases and as well the service location with the least service delay. Furthermore, by choosing a physical machine with lower energy consumption, the proposed method has been able to consume less energy at all working times than the other two methods. On the other hand, due to timely identification of physical machines and the migration of virtual machines of these servers, the excessive overload of these servers and, as a result, their high energy consumption is prevented.

## 5 Conclusion and future work

In recent years, the production of IoT devices has grown significantly; it is so bulky, and IoT applications need to get very quick responses to their requests. Accordingly, the use of edge computing processing power to process this type of data has recently become common. The process of determining the load status of physical machines and automatically scaling resources appropriately for oscillating applications of IoT devices in fog environments can reduce energy consumption and prevent breaches of user service level agreement to avoid problems with over-provisioning and under-provisioning. Since the fog environment is dynamic and users' requests change over time, we need an optimal resource delivery method that can process more requests in less time. Reinforcement learning is adaptable to such an environment and can, as a result of learning, perform the best mapping of services to resources and manage IoT services efficiently. In this paper, an automated resource scaling framework to express interactions between modules designed for IoT devices, fog nodes, and cloud servers has been proposed, and deep reinforcement learning techniques have

been used. The proposed method was evaluated and compared with two other provision methods (FRP\_LA, FRP\_RL) in the iFogSim simulator environment. Minimize response time, cost, and energy consumption as well as increase efficiency rate. Activities that can be done in the future to continue the research are the use of prediction techniques such as neural networks to increase accuracy and their combination to estimate the input workload and identify optimal physical machines. Combining reinforcement learning with fuzzy logic, applying correlated learning, and integrating automatic resource scaling with resource placement and service migration between fog nodes for automatic scaling can significantly improve the proposed solution.

## References

1. Asghari P, Rahmani AM, Javadi HHS (2019) Internet of things applications: a systematic review. *Comput Netw* 148:241–261
2. Kalantary S, Akbari Torkestani J, Shahidinejad A (2021) Resource discovery in the Internet of things integrated with fog computing using markov learning model. *J Supercomput* 77(12):13806–13827
3. Gill SS et al (2019) Transformative effects of IoT, Blockchain and artificial intelligence on cloud computing: evolution, vision, trends and open challenges. *Int of Things* 8:100–118
4. Hashemi S, Zarei M (2021) Internet of things backdoors: resource management issues, security challenges, and detection methods. *Trans Emerg Telecommun Technol* 32(2):4142
5. Mouradian C et al (2018) A Comprehensive survey on fog computing: state-of-the-art and research challenges. *IEEE Commun Surv Tutor* 20(1):416–464
6. Hu P et al (2017) Survey on fog computing: architecture, key technologies, applications and open issues. *J Netw Comput Appl* 98:27–42
7. Gasmi K, Dilek S, Tosun S, Ozdemir S (2021) A survey on computation offloading and service placement in fog computing-based IoT. *J Supercomput* 78(2):1983–2014. <https://doi.org/10.1007/s11227-021-03941-y>
8. Taherizadeh S et al (2018) Monitoring self-adaptive applications within edge computing frameworks: a state-of-the-art review. *J Syst Softw* 136:19–38
9. Aslanpour MS et al (2018) Resource provisioning for cloud applications: a 3-d, provident and flexible approach. *J Supercomput* 74(12):6470–6501
10. Ghobaei-Arani M, Souri A, Rahmanian AA (2019) Resource management approaches in fog computing: a comprehensive review. *J Grid Comput* 18(1):1–42. <https://doi.org/10.1007/s10723-019-09491-1>
11. Singh S, Chana I (2016) Resource provisioning and scheduling in clouds: qoS perspective. *J Supercomput* 72(3):926–960
12. Li C, Bai J, Luo Y (2020) Efficient resource scaling based on load fluctuation in edge-cloud computing environment. *J Supercomput* 76(9):6994–7025. <https://doi.org/10.1007/s11227-019-03134-8>
13. Le T et al (2019) Machine learning methods for reliable resource provisioning in edge-cloud computing: a survey. *ACM Comput Surv* 52:1–39
14. Santos H et al (2020) A multi-tier fog content orchestrator mechanism with quality of experience support. *Comput Netw* 177:107288
15. Stavriniades GL, Karatzas HD (2002) Orchestration of real-time workflows with varying input data locality in a heterogeneous fog environment. In 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)
16. Dinh TQ et al (2020) Online resource procurement and allocation in a hybrid edge-cloud computing system. *IEEE Trans Wireless Commun* 19(3):2137–2149
17. Yousefpour A et al (2019) FOGPLAN: a lightweight qos-aware dynamic fog service provisioning framework. *IEEE Internet Things J* 6(3):5080–5096
18. Porkodi V et al (2020) Resource provisioning for cyber-physical-social system in cloud-fog-edge computing using optimal flower pollination algorithm. *ieee access* 8:105311–105319

19. Mehmandar MF, Jabbehdari S, Javadi HHS (2020) A dynamic fog service provisioning approach for IoT applications. *Int J Commun Syst* 33(14):e4541. <https://doi.org/10.1002/dac.4541>
20. Kim YG, Wu CJ (2020) AutoScale: energy efficiency optimization for stochastic edge inference using reinforcement learning. In: 2020 53rd annual IEEE/ACM I symp microarchitecture (MICRO).
21. Al-Makhadmeh Z, Tolba A (2021) SRAF: Scalable resource allocation framework using machine learning in user-centric internet of things. *Peer-to-Peer Netw Appl* 14(4):2340–2350
22. Nguyen ND et al (2020) Elasticfog: elastic resource provisioning in container-based fog computing. *IEEE Access* 8:183879–183890
23. Pereira P et al (2020) Stochastic performance model for web server capacity planning in fog computing. *J Supercomput* 76(12):9533–9557
24. Bahreini T, Badri H, Grosu D (2019) Energy-aware capacity provisioning and resource allocation in edge computing systems. Springer, Cham, pp 31–45
25. Liu B et al (2020) Workload forecasting based elastic resource management in edge cloud. *Comput Ind Eng* 139:106–136
26. Madan N et al (2020) On-demand resource provisioning for vehicular networks using flying fog. *Veh Commun* 25:100252
27. Siasi N et al (2020) Delay-aware sfc provisioning in hybrid fog-cloud computing architectures. *IEEE Access* 8:167383–167396
28. Abdullah M, Iqbal W, Mahmood A, Bukhari F, Erradi A (2021) Predictive autoscaling of microservices hosted in fog microdata center. *IEEE Syst J* 15(1):1275–1286. <https://doi.org/10.1109/JSYST.2020.2997518>
29. Gupta H et al. (2017) iFogSim a toolkit for modeling and simulation of resource management techniques in the Internet of things, edge and fog computing environments. *Softw Practice and Exp* 47(9): 1275–1296.
30. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw: Practice and Exp* 41(1), 23–50
31. Faraji-Mehmandar M, Jabbehdari S, Haj Seyyed Javadi H (2021) A proactive fog service provisioning framework for Internet of things applications: an autonomic approach. *Trans Emerg Telecommun Technol* 32(11): p. e4342
32. Deng X et al (2020) Task allocation algorithm and optimization model on edge collaboration. *J of syst archit* 110:101778

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Authors and Affiliations

Mohammad Faraji-Mehmandar<sup>1</sup>  · Sam Jabbehdari<sup>2</sup> ·  
Hamid Haj Seyyed Javadi<sup>3</sup>

Sam Jabbehdari  
s\_jabbehdari@iau-tnb.ac.ir

Hamid Haj Seyyed Javadi  
h.s.javadi@shahed.ac.ir

<sup>1</sup> Department of Computer Engineering, Parand Branch, Islamic Azad University, Tehran, Iran

<sup>2</sup> Department of Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran

<sup>3</sup> Department of Mathematics and Computer Science, Shahed University, Tehran, Iran