DA-DRLS: Drift Adaptive Deep Reinforcement Learning based Scheduling for IoT Resource Management

Abishi Chowdhury[a,*], Shital A Raut[a], Husnu S. Narman[b]

[a] *Department of Computer Science and Engineering, Visvesvaraya National Institute of Technology, Nagpur, Maharashtra, India 440010.*
[b] *Computer Science Division, College of Information Technology and Engineering, Marshall University, Huntington, West Virginia, United States.*

Abstract

In order to fulfill the tremendous resource demand by diverse IoT applications, the large-scale resource-constrained IoT ecosystem requires a robust resource management technique. An optimum resource provisioning in IoT ecosystem deals with an efficient request-resource mapping which is difficult to achieve due to the heterogeneity and dynamicity of IoT resources and IoT requests. In this paper, we investigate the scheduling and resource allocation problem for dynamic user requests with varying resource requirements. Specifically, we formulate the complete problem as an optimization problem and try to generate an optimal policy with the objectives to minimize the overall energy consumption and to achieve a long-term user satisfaction through minimum response time. We introduce the paradigm of a deep reinforcement learning (DRL) mechanism to escalate the resource management efficiency in IoT ecosystem. To maximize the numerical performance of the entire resource management activities, our method learns to select the optimal resource allocation policy among a number of possible solutions. Moreover, the proposed approach can efficiently handle a sudden hike or fall in users' demand, which we call *demand drift*, through adaptive learning maintaining the optimum resource utilization. Finally, our simulation analysis illustrates the effectiveness of the proposed mechanism as it achieves substantial improvements in various factors, like reducing energy consumption and response time by at least 36.7% and 59.7% respectively and increasing average resource utilization by at least 10.4%. Our approach also attains a good convergence and a trade-off between the monitoring metrics.

Keywords: IoT resources, Deep reinforcement learning, Demand drift, Energy consumption, Response time, Resource utilization, Simulation

1. Introduction

The rapid advancement of miniature devices for IoT brings forth a proliferation of diverse IoT ecosystem applications under three prominent domains: environment, society, and industry (Pattar et al., 2018). Considering the dynamicity of IoT ecosystem and the heterogeneous resource requirements of concurrent IoT applications, it is evident that the resource management problem is a substantial issue in this ecosystem, that needs a prudent attention (Issarny et al., 2016). From service providers' viewpoint, it is desirable that the mapping of IoT application requests with the available IoT resources must be accomplished in such an efficient way that it could be beneficial for both service providers and clients (Wollschlaeger et al., 2017). Fig. 1 depicts a typical resource management scenario in an IoT ecosystem where, the users request for different IoT services, and the service providers allocate IoT resources to accomplish the requests maintaining a balance between quality and cost. In order to validate the effectiveness of resource mapping, several important monitoring metrics such as delay, average waiting time, response time, information accuracy, coverage, energy consumption, etc. can be considered (Li et al., 2014). The entire task of resource management, i.e. resource selection and resource allocation in an IoT ecosystem faces several challenges (Delicato et al., 2017; Gubbi et al., 2013):

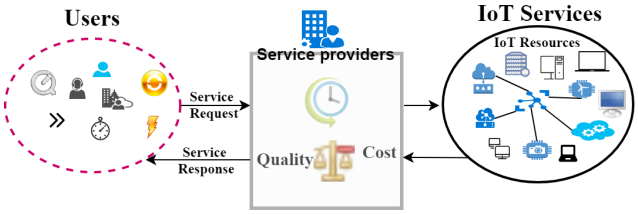


Fig. 1. A typical resource management scenario in context of users as well as service providers

*Corresponding author
Email addresses: abishi.chowdhury@gmail.com (Abishi Chowdhury), saraut@cse.vnit.ac.in (Shital A Raut), husnu@ou.edu (Husnu S. Narman)

Challenge 1. Complex environment: The wide range of software and hardware devices, a variety of connected things, and above all numerous end users collectively form the complex IoT ecosystem. To take the full leverage of IoT ecosystem, several practical applications are built on top of this ecosystem. Most of the real-time IoT applications require a huge amount of sensor data for complex processing. To extract intelligent information from this data, there is a great need for storage and processing resources which we actually lack in the current IoT scenario (Ge et al., 2017).

Challenge 2. Heterogeneity and dynamicity: The heterogeneous and dynamic nature of IoT applications as well as the contributor devices (from miniature sensors to compelling data center nodes), makes the resource management a challenging concern (Alam et al., 2017). Moreover, the highly dynamic execution environment increases the complexity of this process (Gazis, 2017).

Challenge 3. Resource constrained IoT nodes: Due to the small sizes and frequently changing location property, most of the IoT devices are not capable enough to access the power all the time. So, the low power consumption is a universal constraint in IoT (Muhammed et al., 2017; Alaa et al., 2017; Tsiftes & Voigt, 2018).

Challenge 4. Real-time processing: IoT ecosystem potentially deals with up to millions of parallel requests from a diverse range of smart applications as well as fast responses within a strict deadline. Real-time and online processing of IoT applications have different requirements in terms of resources, in comparison to applications running on traditional cloud platforms (Bedi et al., 2018). Thus, resource management in IoT ecosystem is a challenging problem of multiple orders that calls for innovative solutions.

Researchers have been made significant efforts towards finding the solutions for better resource management in an IoT environment. In our previous survey (Chowdhury & Raut, 2018), we summarized potential resource management schemes that effectively addressed various issues regarding IoT resource management. Although progressive work is going on in this context, some critical concerns as mentioned below still need to be addressed.

- The dynamic service quality for continuous arriving IoT application requests should be met.
- Resource management system needs to be more adaptive due to the changing IoT environment.
- More automation is required while managing heterogeneous resource requirements by IoT applications.
- Quick response with a better longevity of the IoT network is required.

Revisiting the above challenges, we propose an adaptive machine learning approach, i.e., drift adaptive deep reinforcement learning based scheduling (DA-DRLS) model that efficiently schedules the heterogeneous IoT application requests in a non-preemptive manner and allocate adequate resources for processing. Starting from an unknown task allocation policy, DA-DRLS eventually learns to carry out a proper action as time progresses in order to optimize two critical objectives: minimizing energy consumption as well as response time. We model the continuous request arrival pattern as the Poisson process (Adan & Resing, 2015) where the arrival rate can fluctuate in a peculiar manner as the demand changes suddenly. DA-DRLS can quickly adapt this sudden change in demand that we named as *demand drift* through adaptive learning while optimizing our primary scheduling objectives.

Contributions: The key contributions of this paper are summarized as follows:

- We design DA-DRLS, a deep reinforcement learning based scheduling model that aims to find an optimal policy for automatic resource allocation to the requests in an IoT ecosystem while minimizing the two important objectives: energy consumption and response time.
- DA-DRLS is adaptive to changes in the demand by continuously observing demand drift and taking countermeasures to update the allocation policy dynamically. To the best of our knowledge, this is the first attempt to tackle such circumstances during resource management in an IoT environment.
- To dynamically adjust the weights of the policy parameters, DA-DRLS provides a trade-off option between different policy parameters for service specific modifications.
- We evaluate the performance of DA-DRLS with varying arrival rates of the requests to ensure that our model can perform efficaciously under heavy traffic flow as well. It performs better than state-of-the-art approaches in context of energy consumption, response time, and resource utilization.

Roadmap: In the subsequent sections of the paper, Section 2 discusses about some relevant work done in this area. Section 3 demonstrates the problem formulation. In Section 4, we describe the modeling of IoT requests with the resources, whereas, Section 5 provides a detailed description of our complete resource scheduling method. A meticulous analysis of our proposed DA-DRLS method with respect to several objectives is presented in Section 6. Section 7 provides some notable observations and finally Section 8 concludes the paper with the final remarks.

2. Related Work

Sensor based IoT devices have limited energy, and processing capabilities, thus they are often not adept to execute sophisticated processing of huge data generated by vast IoT devices. This instigates one of the major challenges of IoT ecosystem, i.e., efficiently manage the scarce resources especially the energy resources. Aiming this challenge, abundant research has been carried out in recent years, including power saving (Taneja, 2014; Van et al., 2016; Kumar et al., 2018), energy efficient allocation (Zhai et al., 2018; Wang et al., 2016; Sarangi et al., 2018), energy

harvesting (Sanislav et al., 2018; Yang et al., 2017; Kamalinejad et al., 2015), etc. Also, potential algorithms have been evolved addressing several other important quality of service (QoS) parameters, such as response time, coverage, delay, throughput (Delicato et al., 2017),(Narman et al., 2017; Sharma et al., 2018; Sohn et al., 2018; Leu et al., 2014), and so forth. However, there is relatively little work on developing a general purpose resource management strategy based on machine learning as a sustainable preference than traditional human-generated heuristics. In this paper, our main intention is to provide a powerful machine learning approach for IoT resource management in order to handle dynamic IoT requests. Therefore, we concentrate more on machine learning based resource scheduling schemes available in IoT ecosystem and several related domains while omitting the details regarding other resource management schemes for IoT.

Reinforcement learning (RL), an active research area of machine learning, enables wireless networks to observe their specific environment, and eventually learns to take optimal or near optimal scheduling decisions in spite of different levels of dynamicity in the corresponding operating environment (Yau et al., 2013). Xue, Yuan, et al. (Xue et al., 2008) proposed a distributed RL approach to provide QoS differentiation among different prioritized requests as well as to ensure a better network performance, such as delay, throughput, and spectrum utilization. Another RL based spectrum selection mechanism was developed by Di Felice, Marco, et al. (Di Felice et al., 2011) that effectively learns to make a perfect balance between sensing, transmitting and switching of channels; thus, achieved a significant performance gain in terms of throughput. To reduce the end-to-end delay, Bhorkar, Abhijeet A., et al. (Bhorkar et al., 2012) suggested an RL framework which can find the nearest route through opportunistic routing. In a wireless sensor network (WSN), cooperative learning allows neighboring nodes to share information among themselves in order to accelerate the learning process and to gain prolong network life (Khan & Rinner, 2012). Wei, Zhenchun, et al. (Wei et al., 2017) proposed a job scheduling method based on Q-learning (a popular RL) with shared value function. The method significantly reduces the switches between nodes; hence, achieves a better performance with less energy consumption than cooperative Q-learning. To reduce the overall energy consumption in an IoT sensory environment, an RL based power saving mechanism was proposed by TS Pradeep, et al. (Kumar & Krishna, 2018) at different layers in IoT. Based on the eminence of different types of tasks, the system learns to trigger an action on high power mode or low power mode which in turn results in low energy consumption. Despite that, RL suffers due to the complexity of RL-based scheduler, the complexity increases as the number of state-action pairs increases. To overcome this condition, deep RL (DRL) schemes that use deep neural network (DNN), are being studied nowadays and are attaining much attention in several application domains, such as in VANET for vehicle image classification (Zhao et al., 2017), medical domain to learn a policy for optimal dose to a person (Nemati et al., 2016), and traffic signal monitoring system (Li et al., 2016). DRL is also being adopted notably for resource management in different operating environments. Mao, Hongzi, et al. (Mao et al., 2016) devised a method where DRL efficiently schedules jobs with multiple resource demands while minimizing the average slowdown. In this approach, the reward function has been defined based on the reciprocal duration of the task in order to direct the agent towards the target. Different classes of prioritized traffic in IoT can be scheduled optimally by effective use of network resources to avoid channel congestion. Chinchali, Sandeep, et al. (Chinchali et al., 2018) presented a DRL based scheduler that can dynamically adapt to traffic differentiation, and to the respective reward functions set by network administrators, to optimally schedule IoT traffic. So far, the literature discussed regarding resource management with RL, one important limitation is that the dynamicity, and uncertainty of modern operating environment like IoT ecosystem are not addressed properly. As the operating environment does not remain consistent for a longer period of time, the learning agent must unlearn the previous knowledge as it may become outdated, and quickly learn about the new atmosphere by increasing the learning rate. Once learned, it should maintain the previous learning rate. Thus, while managing the IoT resources and scheduling user requests through RL, a balanced trade-off should be maintained by carefully adjusting the learning rate parameter, which motivates the work of this paper.

3. Problem Formulation

The problem addressed in this paper is the allocation of IoT resources to service requests generated by the users within an IoT ecosystem in an energy efficient and highly responsive manner. A typical IoT ecosystem contains a collection of heterogeneous resources, and each service request (SR) is provided with different instances of these resources as per the necessity. This allocation of resources requires effective decision making so that overall quality of service (QoS) can be maintained. Each resource has a particular energy requirement based on its capacity for its working and hence, the allocation should include a trade-off between the QoS and overall energy consumption. The trade-off is application specific. Let us consider the following three cases:

Case 1: High QoS is required, for example, in case of smart health monitoring.

Case 2: Efficient energy consumption is required due to long time functionality of the devices for several IoT applications, for example, in case of weather monitoring.

Case 3: A balance between two parameters is required, like, in case of smart homes.

So, formally, our objective is to devise an optimal policy for a request to resource mapping that will maximize

the QoS and minimize the overall energy consumption (EC) in a resource constraint IoT ecosystem. Let the IoT resource pool $\{RS\}$ contain a set of n heterogeneous types of resources as $\{\underbrace{RS_1}_{rs_1}, \underbrace{RS_2}_{rs_2}, ..., \underbrace{RS_n}_{rs_n}\} \in \{RS\}$. Each service request has a specific set of resource demand ($SR_i = \underbrace{rs_{i1}, rs_{i2}, ..., rs_{ij}}_{rs^i}$) for j ($j <= n$) different types of resources. The resource allocation process provides a mapping between $\{RS\} \rightarrow \{rs^i\}$. Now the objective is to maximize the QoS and minimize energy consumption for each service request i.e. $max(QoS(SR_i))$ and $min(EC(SR_i))$ so that a cumulative maximization of QoS and cumulative minimization of energy consumption can be achieved. Precisely, the problem is formulated as the following optimization problem:

$$\begin{aligned} &Resource(SR_i) = A(SR_i, \pi(SR_i)) \\ &\pi(SR_i) = Max\Big(\sum_{\forall i} QoS(SR_i) + \sum_{\forall i} \frac{1}{EC(SR_i)}\Big) \end{aligned} \tag{1}$$

where, $Resource(SR_i)$ is the resultant resource mapping for a service request SR_i, $A()$ is the allocation function for service request SR_i using the required policy $\pi(SR_i)$. To have a trade-off between QoS provided to the users and the overall energy consumption, the following trade-off equation is used, where, the trade-off parameter η can be adjusted according to the specific demands of the users.

$$\pi(SR_i) = Max\Big(\eta \sum_{\forall i} QoS(SR_i) + (1-\eta) \sum_{\forall i} \frac{1}{EC(SR_i)}\Big) \tag{2}$$

An important non-functional requirement that relates to QoS is Service Level Agreement (SLA) (Alodib, 2016) which becomes increasingly important in the IoT ecosystem. Due to the dynamic nature of IoT ecosystem and its complex processing, it is challenging to achieve the targeted QoS. Therefore, in order to avoid customer dissatisfaction, QoS requirements are declared by imposing binding SLAs. SLAs are the contracts perceived to exist between the service provider and users. Thus, to provide guaranteed system performance, we formulate our objective function as:

$$\begin{aligned} &\pi(SR_i) = Max\Big(\sum_{\forall i} \frac{1}{RT(SR_i)} + \sum_{\forall i} \frac{1}{EC(SR_i)}\Big) \\ &subject\ to\ RT(SR_i) \leq SLR \end{aligned} \tag{3}$$

where, $RT(SR_i)$ is the *response time* which ensures the $QoS(SR_i)$, and $EC(SR_i)$ is the overall *energy* consumption for processing the i^{th} service request.

This simple generalized objective function given in Equation 3 (detailed in Section 5 as the reward function) has a multifarious perspective:

- The allocation of resources to each service request can be done in many possible ways; our target is to select the **optimum suitable** policy for allocation while maximizing the objective function.
- The policy of allocation should provide a **long-term benefit** instead of a short-term profit.
- Arrival of the service requests is unpredictable as it depends on the user requirements which can **change over time**. So, here can be a change in the service arrival scenario (*a demand drift*) which affects the existing policy. Therefore, policy update needs to be done whenever a **demand drift** occurs.

A *demand drift* is the statistical change in service demand over a period of time. The detection of demand drift must be done by monitoring the service arrival patterns. Once it is detected, the allocator must update the policy for such dynamic nature of the operating environment.

Therefore, the overall scheduling of the IoT resources is an optimization problem, and we try to solve it using a first-order iterative optimization algorithm, i.e., gradient descent. To address this entire problem, first, we need to model this problem.

4. System Model: IoT Request-Resource Mapping

We adopt Markov Decision Process (MDP) to model decision making and optimization of resource allocation process to answer the service requests in an IoT ecosystem. Modeling the complete resource management scenario requires an efficient modeling of service arrival process and then forming the desired scheduling policy for proficient mapping of system resources with the incoming requests.

4.1. Modeling Requests Arrival Process

We consider the collection of different heterogeneous resources as one resource pool which can be synthesized as a single server with multiple resources. A service request can demand a random amount of subset of resources from the resource pool. Arrival of these incoming requests in an IoT ecosystem can be modeled using an M/M/1 queue model where heterogeneous user requests follow the Poisson process with arrival rate λ and the service times follow an exponential distribution with mean $1/\mu$. The Poisson process is one of the most commonly used models for different stochastically independent requests (Adan & Resing, 2015; Abdullah & Yang, 2014). Each request arrives with a request vector which specifies the amount of resources required over its indices. To reduce the complexity of policy generation (though M/M/1 queue has an infinity buffer capacity), we consider a batch of requests with length m for scheduling purpose and policy generation while considering the remaining requests that are going to be processed in the next slot residing in a buffer of length M for ready reference.

4.1.1. Request Queue

As soon as any request arrives, it is added to the request queue. The resource allocation process always keeps

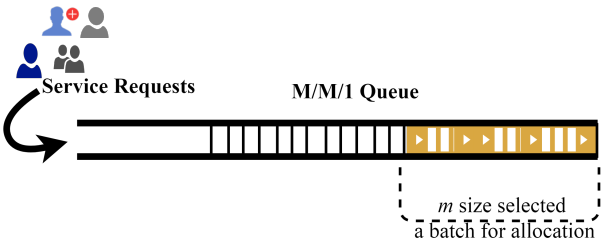


Fig. 2. Queue management for request processing

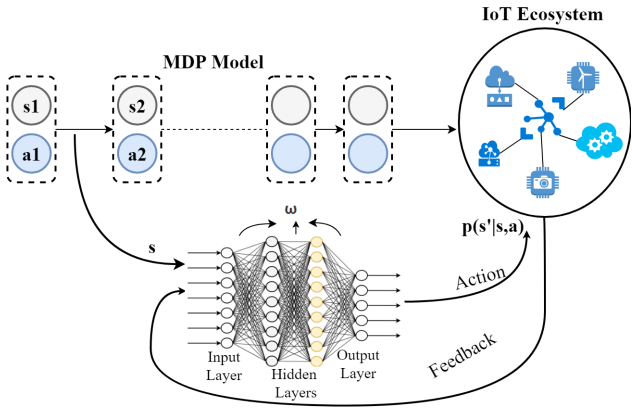


Fig. 3. The policy generation model in context of an IoT ecosystem

track of the request queue and generates the policy to allocate resources to the SR in the queue. Whenever the required resources are available for the requests in the queue, the service requests are removed from the queue and resources are allocated. The length of the queue is a crucial factor and affects the performance of the overall system. Fig. 2 illustrates that we consider a batch of size m requests from M/M/1 queue for resource scheduling purpose.

4.1.2. Response Time

Considering the system is in a stable state, we can obtain mean *response time* using the Little's law (Adan & Resing, 2015) as: Mean number in the system = Arrival rate (λ) * Mean response time (RT).

$$RT = \left(\frac{\rho}{1-\rho}\right)\frac{1}{\lambda};\ where, \rho = \frac{\lambda}{\mu} = effective\ utilization$$
$$thus, RT = \frac{\frac{1}{\mu}}{1-\frac{\lambda}{\mu}};\ subject\ to\ 0 < \lambda < \mu \tag{4}$$

4.2. Deep Reinforcement Learning Model

The absolute IoT scheduling process is synthesized using deep reinforcement learning which is based on an MDP model as the quality of resource mapping process is enhanced with repeated steps. The main constituents of a basic MDP model are *state, action,* and *reward*. A basic policy generation model is shown in Fig. 3.

- A *state* s_t is a fully encapsulated information about an IoT ecosystem at time t which represents the decision-making factors.
- An *action* a_t is the allocation of resources to the incoming IoT service requests. After performing each action, there is a transition from current state s_t at time t to a new state s_{t+1} at time $t+1$.
- *Reward* $R_t = R(s_t, a_t, s_{t+1})$ is the feedback from the system after a change in the current state of the system. Based on the effects on network performance, a positive or negative reward is received every time when an agent performs an action.

In context of an IoT ecosystem, each state provides abstract information about some parameters, and values of these parameters are changed when the system passes to the next state after performing an action. Further, these parameters or we can say the *monitoring metrics* are used to monitor the states of the system. The values of these metrics provide an inference that, whether the system is in a good state or not? The results of monitoring metrics can be understood as a feedback of the action resulting in the current state. For each resource allocation action, a feedback can be recorded which is the ultimate reward for that particular action. The trivial task is to decide this feedback and generate a policy $\pi(s) : s_t \rightarrow a_t$ (i.e., $\pi(s) = a_t$) based on this feedback that maximizes the cumulative rewards: $R_t = \sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})$, where, γ is a discount factor or delayed reward parameter, *s.t.* $0 \leq \gamma < 1$. Considering the monitoring metrics as *response time* (RT) and *energy consumption* (EC), the reward should be provided to the action based on the results of these parameters. The scheduling approach should predict the consequences of the actions and decide whether to execute those actions or not. It is better to predict the effect of an allocation on the IoT ecosystem prior to the actual allocation of resources.

4.2.1. Deep Neural Network

A deep neural network (DNN) model can effectively be applied in those applications where the number of possible states are significantly large. Recently, it is being widely used to function approximation, classification, and prediction purpose for large-scale RL tasks. Motivated by the high accuracy of DNN (Mnih et al., 2013; Alam et al., 2016), we take the advantage to use the combination of DNN with RL, jointly known as Deep Reinforcement Learning (DRL), for decision making during resource scheduling through an ideal learning.

4.2.2. Demand Drift

The current service arrival scenario gets changed when there is a change in the demand. To reflect the dynamicity of IoT ecosystem, we consider sudden fluctuations in the request arrival pattern. It is a challenge to make the scheduling process adaptive to these changes, as it requires

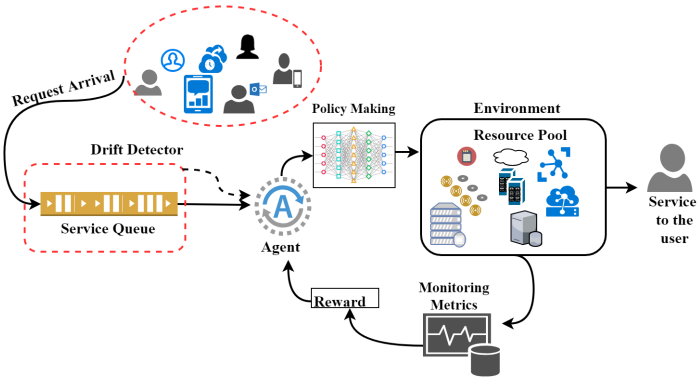


Fig. 4. The proposed model with continuous arrival of user requests. The agent allocates resources based on the policy, and monitoring metrics are used to update the policy based on rewards, drift detector detects and notifies the agent.

a re-optimization of the objective function. To overcome this challenge, we propose a dynamic adaptive technique which can allocate adequate resources to the incoming requests even when there is a demand drift.

5. DA-DRLS: Drift Adaptive Deep Reinforcement Learning based Scheduling

To schedule the incoming service requests reside in the request queue, the proposed approach DA-DRLS, as depicted in Fig. 4, tries to find the appropriate resources for allocation by selecting an optimal policy in context of *RT* and *EC*. DA-DRLS includes the following five essential steps:

- Exploration of the complete environment and generation of different possible policies (trajectories).
- Estimation of the expected reward for each trajectory which is inversely proportional to response time and energy consumption.
- Maximization of the expected cumulative rewards.
- Exploitation of the policy with the maximum reward for the resource allocation.
- Monitor and adaptation according to the environmental changes.

Some commonly used notations in the paper are described in Table 1.

5.1. Probabilistic formulation of resource allocation sequence

A resource allocation sequence is a possible allocation scheme for allocating resources to the current IoT service requests. It consists of a request to resource mapping. A number of possible mappings are possible which can be estimated using combinatorics. MDP which is basically an extended Markov Chain with augmented state space, is used to generate all these sequences. A possible resource allocation sequence is a sequence of states and actions of MDP model. To generate the best trajectories, we consider the following:

- Probability of a possible allocation is synthesized as a change in a state.

Table 1: Common notations with descriptions

Notation	Description
S	Set of states
A	Set of possible actions
$r()$	Reward function
s	$s \in S$
a	$a \in A$
T	Total number of states
τ	A trajectory of length T which is a sequence of states and actions
ω	Policy parameter
χ	Policy for scheduling
N	Number of episodes
R_s	Cumulative reward at state s
α	Learning rate for policy update
∇_ω	Gradient for policy update
γ	Delayed reward parameter

- Formulation of the distribution of resource allocation policy parameterized by a policy parameter.

To perform an efficient scheduling, we require an optimal policy χ_ω parameterized by a policy parameter ω. The probability of changing a state from state s_t to s_{t+1} is given by MDP transition probability (Bellman & Dreyfus, 2015) as shown in equation 5.

$$p\Big((s_{t+1}, a_{t+1})\big|(s_t, a_t)\Big) = p(s_t|s_t, a_t)\chi_\omega(a_{t+1}|s_{t+1}) \quad (5)$$

Exploration process over states and actions gives rise to a distribution over state and action sequences parameterized by ω which is exactly needed to approximate. Therefore, if we have a time horizon of length T (length of sequence $= T$), then the policy or distribution is given by:

$$p_\omega(s_1, a_1, ..., s_T, a_T) = p(s_1)\prod_{t=1}^{T}\chi_\omega(a_t|s_t)p(s_{t+1}|s_t, a_t) \quad (6)$$

5.2. Feedback of allocation and its maximization

Each state of the IoT ecosystem state model provides an abstract information about the monitoring metrics which can be a combination of several other parameters. For the proposed work, we consider the response time and energy consumption as the prominent parameters. The abstract information for state s and action a is given by the following reward function:

$$r(s, a) = \frac{1}{RT(s, a)} + \frac{1}{log(EC(s, a))} \quad (7)$$

where, $RT()$ and $EC()$ are the resultant response time and energy consumption of the system at state s.

Response Time: Response time is the difference between the arrival time and the time at which the resources are allocated to a service request. $RT(s, a)$ is the average response time for the service requests allocated at state s by taking action a. Average response time can be calculated using Equation 4.

Energy Consumption: The energy consumption after state s is the aggregated sum of energy consumption of the L allocated service requests at state s.

$$EC(s,a) = \sum_{l=1}^{L} \big(EC(SR_l^s)\big) \tag{8}$$

The overall objective is to minimize both of these parameters which results in maximization of Equation 7. This objective can be achieved by maximizing the objective function over all the states in the consecutive explorations. Indeed, the maximization results in a distribution over states and actions which can be used as a policy estimator to schedule the set of requests. The policy which maximizes the expected value $E_{\tau\sim p_\omega(\tau)}$ of reward (Mnih et al., 2013) can be estimated as a chain of states, actions, and a trajectory τ sampled as $p_\omega(\tau)$, is given by:

$$\omega^* = argmax_\omega E_{\tau\sim p_\omega(\tau)}\left[\sum_{t=1}^{T} r\big(s_t, a_t\big)\right] \tag{9}$$

Where, ω^* is the estimated policy for a trajectory τ of length T. This can also be written as:

$$\omega^* = argmax_\omega \sum_{t=1}^{T} E_{(s_t,a_t)\sim p_\omega(s_t,a_t)}\left[r\big(s_t, a_t\big)\right] \tag{10}$$

Equation 10 is basically an optimization function that needs to be maximized by altering ω which defines the policy of resource allocation. To estimate an optimal value ω^*, a gradient is required to modify its value for different scenarios.

5.3. Policy generation process

A policy should be selected after a certain number of explorations, and the exploration parameter decides the number of explorations required before finalizing the policy. Each exploration generates a particular trajectory, the optimum trajectory from already generated trajectories is used for further exploration. Thus, after the formation of all possible trajectories, the best suitable trajectory is selected. This process includes mimicking all the possible trajectories by adopting a DNN which is used to increase the probability of selecting a suitable trajectory with the help of a gradient. The resultant maximum probable trajectory is used for resource allocation or further exploration. We consider the standard reinforcement learning framework in which a learning agent interacts with IoT ecosystem that is modeled as MDP. The state, action, and reward at each time instance $t \in \{1, 2, ..., T\}$ are denoted as ($s_t \in S$, $a_t \in A$, and $r(s_t, a_t)$), respectively. We focus on a course of policy gradient based RL algorithm that learns by performing gradient-descent on the policy parameters. The policy gradient is used to update the policy due to its adaptability with the DNN. The objective is to maximize the anticipated aggregated reward (Mnih et al., 2013; Bellman & Dreyfus, 2015). The slope of the objective function, i.e. Equation 10 is given by:

$$\nabla_\omega = E_{\tau\sim\chi_\omega(\tau)}\left[\sum_{t=1}^{T} \nabla_\omega log\chi_\omega(a_t|s_t) \sum_{t=1}^{T} r(s_t, a_t)\right] \tag{11}$$

Here, $\sum_{t=1}^{T} r(s_t, a_t)$ is the anticipated total reward (deterministic) for choosing action a in state s, and in this way it proceeds following χ_ω. The key thought in policy gradient strategy is to assess the angle by observing the direction of execution that is obtained by pursuing χ_ω. The magnitude of action in the direction of the gradient is parameterized by the second term which is aggregated reward in Equation 11. For our aimed approach, we use a delayed reward strategy using γ parameter. The reward at a state s_t is the cumulative sum of rewards of the states $\{s_{t+1}, s_{t+2}, ..., s_T\}$ which can be reached from current state discounted by γ. This cumulative reward at a state $s^{'} \in \{1, 2, ..., T\}$, given by Equation 12, is considered as the magnitude of gradient for the proposed approach.

$$R_{s'} = \sum_{t=s'}^{T} \gamma^{t-s^{'}} r_t \tag{12}$$

By using Monte Carlo strategy (Schulman et al., 2015), the agent generates sample trajectories and applies them to compute the cumulative discounted reward as an unbiased estimation of the feedback from IoT ecosystem. Based on this feedback, the agent updates the policy by updating ω using Equation 11 and 12 as given in Equation 13 while using a learning parameter α to regulate the step size of the gradient.

$$\omega \leftarrow \omega + \alpha\nabla_\omega log\chi_\omega(a_t|s_t)R_{s'} \tag{13}$$

5.3.1. Scheduling agent

The decision maker for electing appropriate actions is the agent, and the complete IoT ecosystem is the agent's environment. The agent picks actions based on a policy, characterized as a probability distribution over actions. χ : χ(s, a) $\rightarrow$ [0, 1]; χ(s, a) is the probability that action a is taken in a state s. The agent observes the environment through perceptions and is influenced by taking actions. The agent completes the following tasks before performing the allocation.

- It generates trajectories using Equation 5,6.
- It estimates the expected reward using Equation 7.
- It performs the optimization through Equation 10.
- It monitors changes in the environment and ensures its adaptability using drift adaptive learning in Section 5.4.

Fig. 4 portrays the complete proposed approach where, the agent generates the policy based on the DNN network labeled as policy making in the figure. Algorithm 1 shows the complete process of policy generation and its updating

Algorithm 1: DRL Scheduling

Result: Policy generation

1 Input: Set of Service Requests (SR)
2 **for** *each timestep* **do**
3 $\quad \nabla_\omega = 0$
4 $\quad$ **for** *each incoming set of request* **do**
5 $\quad\quad$ **while** *episode* $i \leq N$ **do**
6 $\quad\quad\quad \alpha = Drift - monitoring()$
7 $\quad\quad\quad$ explore $p_\omega\big(s_1^i, a_1^i, ..., s_T^i, a_T^i\big) = p\big(s_1\big)\prod_{t=1}^{T}\chi_\omega\big(a_t|s_t\big)p\big(s_{t+1}|s_t, a_t\big) = \boldsymbol{\chi_\omega}$
8 $\quad\quad\quad$ **for** *j=1 to* T^i **do**
9 $\quad\quad\quad\quad$ **calculate** $r_j^i(s_j, a_j)$
10 $\quad\quad\quad$ **end**
11 $\quad\quad\quad$ **for** $j = 1$ *to* T^i **do**
12 $\quad\quad\quad\quad R_j^i = \sum_{t=j}^{T^i}\gamma^{t-j}r_t^i$
13 $\quad\quad\quad$ **end**
14 $\quad\quad\quad$ **for** $t = 1$ *to* T^i **do**
15 $\quad\quad\quad\quad \nabla_\omega = \nabla_\omega + \alpha\nabla_\omega \log\chi_\omega(a_t^i|s_t^i)R_t^i$
16 $\quad\quad\quad$ **end**
17 $\quad\quad$ **end**
18 $\quad$ **end**
19 $\quad \omega = \omega + \nabla_\omega$
20 **end**

Algorithm 2: Drift-monitoring()

Result: Learning parameter

1 α= Learning parameter
2 Service arrival distribution B
3 **for** *each monitoring_timestep* **do**
4 $\quad$ get the current service distribution $B^{'}$
5 $\quad$ Measure the KL-Divergence for service arrival rate
6 $\quad KL = D(B||B^{'})$
7 $\quad$ **if** $KL \geq \delta$ **then**
8 $\quad\quad \alpha = \alpha * 1.5$
9 $\quad\quad$ **return** α
10 $\quad$ **end**
11 $\quad$ **else**
12 $\quad\quad$ **return** α
13 $\quad$ **end**
14 **end**

process after several explorations. The complete model is explored, and N trajectories of length T are generated as shown in line 7. The reward for each state and action of each explored trajectory is calculated as shown in lines 8 to 10. A cumulative discounted reward is calculated (lines 11 to 13) using discount parameter γ. The gradient and its magnitude are calculated and the policy is updated using ∇_ω. The slope ∇_ω of the objective function in Equation 10 is actually a matrix of order $T \times N$ which is then added to the existing policy parameter matrix ω. Our proposed approach also monitors the incoming services and detects any change in their patterns and thus updates the policy. The agent gets notified by the drift monitoring process whenever a drift occurs.

5.4. Drift adaptive learning

Algorithm 1 uses *drift-monitoring()* function to get the value for learning parameter. The objective of this function is to monitor the changes and perform the desire actions when the amount of change is greater than a threshold value δ. Kullback-Leibler (KL) Divergence (Bigi, 2003) is used to measure the dissimilarity between the arrival rates at two different time instances of the incoming service requests. This dissimilarity can be considered as *demand drift* and the drift management requires its proper identification and quantification from continuously arriving service requests. Consider that the service arrival follows a true probability distribution $G(X)$ over a global service instance X and in a limited time horizon, an arbitrary probability distribution $B(x)$ is observed (where $x \in X$). If there is a change in the observed service arrival distribution, it increases or decreases the number of waiting requests in the queue over time. Considering two service arrival distributions B and $B^{'}$, B in the previous instance and $B^{'}$ in current instance, the distance between them is calculated using KL divergence or relative entropy as:

$$\boldsymbol{D}(B||B^{'}) = \sum_{x\in X} B(x) log\frac{B(x)}{B^{'}(x)} \tag{14}$$

So, if there is a sudden change in the arrival pattern of the services, then this sudden change will result in staleness in the learned network. There are two ways to adapt these changes; one is to naively relearn the complete network and the second is to perform the learning with a higher learning rate as used in the dynamically updating surrogate (Schulman et al., 2015) systems. To update the system according to the changing environment, an adaptive learning approach is proposed in which learning rate is updated as described by $Drift - moniroing()$ function in Algorithm 2. The threshold value δ (Kifer et al., 2004) is used to regulate that when the learning rate should be updated. It has been empirically observed that with $\delta = 0.05$, less number of policy updates are required to achieve the desired objectives.

6. Evaluation

We have performed a rigorous simulation analysis to evaluate the performance of DA-DRLS. The ultimate aim of our evaluation is to answer the following obvious queries regarding resource management in IoT ecosystem at present scenario.

- How to optimize the constraints of IoT resource management?
- How much improvement in the performance as compared to other benchmark algorithms in terms of response time and energy consumption?
- How efficiently it utilizes the system resources?
- How does the proposed approach achieve the final solution while there is a demand drift?
- How much re-optimization is required whenever a demand drift occurs? How quickly it converges?
- How does the trade-off affect the output?
- How much execution time it requires to perform scheduling?

Simulation Setup: We have simulated our proposed approach using Keras (Chollet et al., 2015) with Theano (Bastien et al., 2012) python3 library which helps to efficiently optimize mathematical expressions. The system configuration is: 8 GB RAM, Intel(R) Core (TM) i5-6200U CPU @ 2.30Ghz, 2.40 GHz with 64-bit operating system. We have used synthetic dataset (Akdere et al., 2008; Ghosh & Simmhan, 2018) for this analysis in which service requests are generated using Poisson process. To emulate the real time IoT scenario, a range of arrival rates from 0.2 to 0.9 under the same service time constraint is considered for the service requests. Each service request, generated by this process, is a combination of different amount of resources required to satisfy that request. Each service request demands resources in form of a request set $\{r_1, r_2, ..., r_n\}$ for a time t; $\{1t\ to\ 5t\}$. Each r_i entry in the request set is the amount of i^{th} resource required by a service. The resource pool contains n different types of resources $R_1, ..., R_n$, we have considered $n = 5$, and there are 5 instances of each resource type. Each resource instance is available with a random capacity of rc_{ij} $(j = 1\ to\ 5)$ unit between $10 - 100$. For simulation purpose, we have assumed each resource instance is provided with an energy consumption requisition for its functioning which is $e_{ij} = ((rc_{ij}/10) * t)$ *Joule*. The resource capacity is divided by 10 to get a normalized energy consumption requirement proportional to the capacity of that particular resource. An instance of a resource is allocated to the service request at a time. We have assumed that all the resources are active and a request will be processed only when its entire resource demand gets fulfilled. No fraction allocation is possible for a particular resource demand.

The agent considers a set of service requests of length 10 for allocation of the resources and service requests beyond 10 as pending requests. It uses count of these pending requests (we set maximum pending requests = 20) while exploring for the optimal policy.

The amount of different resources required by each service is randomized using a random function. The complete approach is analyzed by running 20 Monte Carlo simulations (Schulman et al., 2015) while having exploration parameter ranging from 50 to 10000. For the ease of understanding, we have divided this range into 10 parts named simulation episode as shown in Table 2.

Table 2: Analysis of reward while increasing the number of explorations

Simulation Episodes	No.of Explorations	Maximum Reward
1	50	-26.17
2	100	-24.233
3	150	-23.132
4	200	-23.0313
5	250	-23.302
6	300	-23.0291277
7	400	-23.0290063
8	500	-23.02897605
9	2000	-23.023376
10	10000	-23.023365

An allocation policy is generated by mimicking all possible allocations in form of an MDP model. The state transition probabilities of the MDP model are considered to form different trajectories of a DNN. We have employed a fully connected DNN with one hidden layer and 20 neurons (Mao et al., 2016). The policy is generated by optimizing the weights using gradient from Equation 11. The learning rate used for the DNN is 0.001 (Xu et al., 2017) and in case of demand drift, Algorithm 2 is used to update the learning rate for instant adaptability. Once the simulation is started, no preemption will occur as we have not considered any priority level of the requests.

In order to validate the efficacy of our proposed approach, we have compared DA-DRLS with three standard approaches; First Come First Serve (FCFS), basic Q-Learning approach (Huang et al., 2011; Peng et al., 2015) while considering only response time (QLR) as a monitoring parameter, and Q-Learning approach while considering both the response time and energy consumption (QLRE) as the monitoring metrics. The next consecutive subsections demonstrate a meticulous analysis considering some eminent aspects in this regard.

6.1. Performance Analysis with varying Arrival Rates

Fig. 5 and 6 respectively plot the optimal energy consumption and response time (as per their specific objectives) of DA-DRLS, FCFS, QLR, and QLRE over different arrival rates after performing different number of explorations. As the arrival rate increases, the number of requests increases and the number of pending requests also increases which in turn result in higher energy consumption and response time. Clearly, our proposed approach achieves a significant amount of improvements for both the two monitoring metrics at each arrival rate than other three approaches. From Fig. 5, we can observe that FCFS uses maximum energy for every arrival rate as it is the basic scheduling approach and used in simple small systems. It cannot perform intelligent operations, so is not efficient for the complex ecosystem like IoT where its heterogeneous requests need different length of resources. QLRE shows better performance than QLR because QLRE uses energy consumption as a parameter in its optimization function. However, QLRE has to explore all combinations of states

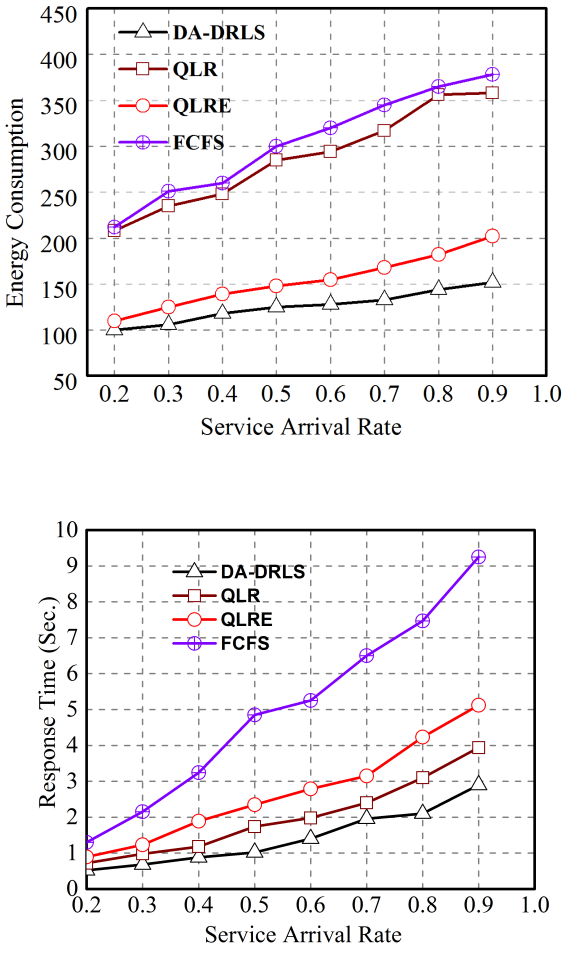


Fig. 6. Response time analysis while increasing the service arrival rate

and actions, resulting in a large number of possible states which in turn requires higher computation to generate the required policy. Our method, DA-DRLS on contrary, does not calculate the exact value of the optimization function and estimates its value using policy gradient. Thus, DA-DRLS achieves an early convergence, thereby reducing overall energy consumption than QLRE. On an average DA-DRLS consumes a less energy of 58.02% than FCFS, 56.27% than QLR, and 36.70% than QLRE.

Fig. 6 shows the response time analysis. For the same reason as energy consumption analysis, DA-DRLS outperforms all the three techniques by a reduction of 90.2%, 59.7%, and 66.2% than FCFS, QLR, and QLRE respectively. Here, one interesting point should be noticed that QLR has lower response time than QLRE. This is because QLR only considers response time in its Q function where QLRE needs to consider both response time and energy consumption.

In the rest of the analysis, we have fixed arrival rate parameter as 0.7 to ensure a balanced and reasonable amount of load in the system and investigated the outcomes with different exploration counts.

Table 3: Optimal energy consumption and response time at 10000 exploration count for DA-DRLS, QLR, QLRE, and FCFS

	DA-DRLS	QLR	QLRE	FCFS
Iterations	10000	10000	10000	NA
Max Rewards	-23	-44	-38	NA
Energy Consumption	133	280	168	345
Response Time	1.96	2.56	3.15	6.5

Table 4: Analysis of energy consumption (*Joule*) while increasing the number of episodes and optimizing energy consumption

Simulation Episodes	No.of Explorations	DA-DRLS	QLR	QLRE	FCFS
1	50	254	344	264	345
2	100	205	317	235	345
3	150	151	277	220	345
4	200	141	300	205	345
5	250	138	340	196	345
6	300	137	345	196	345
7	400	139	299	178	345
8	500	134	280	175	345
9	2000	134	341	172	345
10	10000	133	317	168	345

6.2. Reward

As the exploration proceeds, the reward gets updated as described in Algorithm 1. The average reward value is observed while varying the number of explorations (N in Algorithm 1) from 50 to 10000, Table 2 illustrates these observations. The reward value is considered as optimized when the amount of changes in the objective parameters is notably less. Early smoothness means that the maximum reward is achieved with less number of explorations. The approximate highest reward can be achieved with only 150 explorations, although for a robust scenario the value of this parameter is kept large. The overhead of large value of this parameter (exploration) is not directly proportional to its quantity because as the system achieves the maximum reward, it requires less computation to get that reward again. So, in the initial iterations, the agent takes longer time to achieve the reward as compared to the subsequent iterations. Table 3 shows the best result obtained after running 20 simulations with 10000 explorations each and a comparison with other approaches for reward based on energy consumption, and response time.

6.3. Energy Consumption

The overall energy consumption for completion of a service request SR_l, $l \in L$, is given by $\sum_{\forall rc_{ij} \to SR_l} e_{ij}$, where, $\forall rc_{ij} \to SR_l$ are the resources allocated to SR_l request. To reduce the complexity of the energy consumption analysis, the energy required for allocation process is ignored. With the increasing number of explorations from 50 to 10000, the proposed approach moves towards the optimal solution and the energy consumption tends to be lower. Less energy consumption implies an efficient mapping of requests with the resources. The analysis of overall energy consumption while adjusting the exploration parameter is given in Table 4. The agent achieves an effective energy

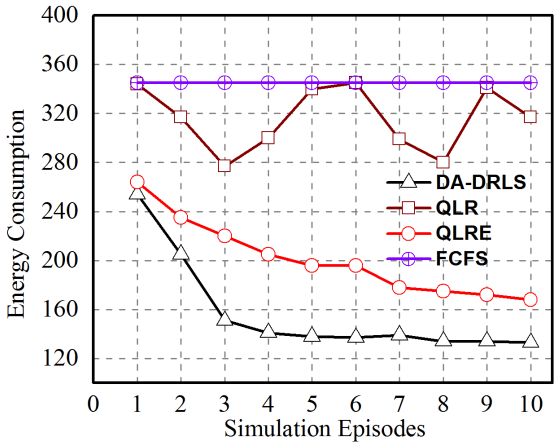


Fig. 7. Energy consumption analysis of the proposed approach for different episodes and its comparison with the other approaches

consumption when exploration parameter reaches 150. After this much explorations, the overall energy consumption does not experience any significant decrement because exploration count 150 onwards there is no substantial change observed in the policy. Fig. 7 demonstrates the comparison of the proposed approach with the three mentioned approaches as before. FCFS results in a constant energy consumption irrespective of its explorations from 50 to 10000 as it simply schedules the requests as soon as they come to the available resources. The energy consumption does not get affected no matter how many times we explore in case of FCFS because with the different simulation episodes there will be no change in the policy, that is why the graph shows a linear plot with respect to increasing number of episodes. The conventional QLR does not use energy consumption as a parameter for policy improvement, so, the energy consumption fluctuates while exploring the policy. For example, it shows low consumption at 150 and high at 250. On the other hand, QLRE shows the result of the policy where the energy consumption is used as a parameter for policy update. Hence, it consumes less energy than QLR. Remarkably, DA-DRLS quickly estimates the optimal policy through gradient, thus obtains the approximate optimum request-resource mapping which in turn comes out as the lowest energy consumption. While exploring, there is a rapid decay till 150 and it approaches a continuous decay till 200 as it learns to find better policies.

6.4. Response Time

The average response time is observed for different trajectories, and policies are updated according to its value. The gradient exploits the reward to update the policy for reducing the response time. The response time decreases as the policy approaches towards an optimal solution while passing through different episodes. Similar to energy consumption, after a sharp decrement till 150 explorations, the response time reaches to its near optimal reading as shown in Fig. 8. Here, a noteworthy observation is that as

Table 5: Analysis of response time (*sec*) while increasing the number of episodes for optimization

Simulation Episodes	No.of Explorations	DA-DRLS	QLR	QLRE	FCFS
1	50	5.25	5.1	5.55	6.5
2	100	3.85	4.5	4.98	6.5
3	150	2.19	3.05	3.9	6.5
4	200	2.18	2.9	3.85	6.5
5	250	2.1	2.88	3.78	6.5
6	300	2.08	2.7	3.79	6.5
7	400	2.05	2.55	3.4	6.5
8	500	1.96	2.56	3.35	6.5
9	2000	1.97	2.56	3.1	6.5
10	10000	1.96	2.4	3.15	6.5

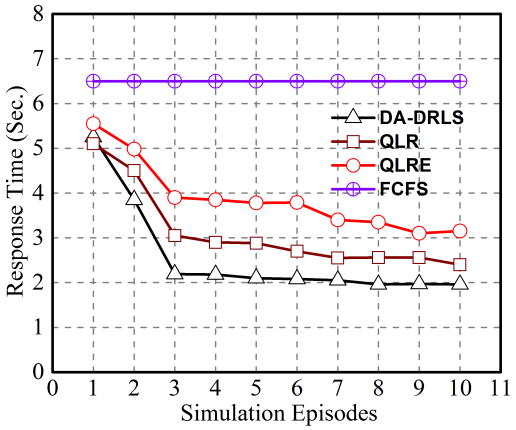


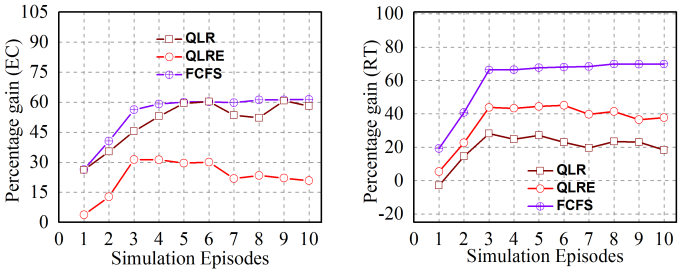
Fig. 8. Response time analysis of the proposed approach for different episodes and its comparison with other approaches

we got almost the best reward after 150 explorations, consequently, we have achieved near to ideal results for both the energy consumption and response time. As expected, in response time analysis also, DA-DRLS outstrips all the three approaches. The highest steady response time of 0.12 is observed for FCFS while QLR requires slightly less response time than QLRE as mentioned in Table 5.

In Fig. 9a and 9b, we have analyzed the amount of percentage gain by DA-DRLS with respect to different simulation episodes as compared to different approaches. The results reveal that DA-DRLS attains sufficient gains in context of both the monitoring metrics.

6.5. Resource Utilization

Resource Utilization (RU) is also an important performance metric which further ensures low load average in the system. Here, we formulate the RU factor and provide analysis in comparison with other above mentioned approaches. Suppose, there are total L number of service requests, denoted as $\{SR_1, SR_2, ..., SR_L\}$ after T time period that demand n different types of resources as a request set, $SR_l = \{r_1, r_2, ..., r_n\}$, where, $l = 1, ..., L$. If each resource has J number of instances and resource capacity of each instance is denoted by rc_{ij}, where, $i = 1, ..., n$, then



(a) Percentage gain for energy consumption

(b) Percentage gain for response time

Fig. 9. Percentage gain of the proposed approach as compared to other approaches for both monitoring metrics

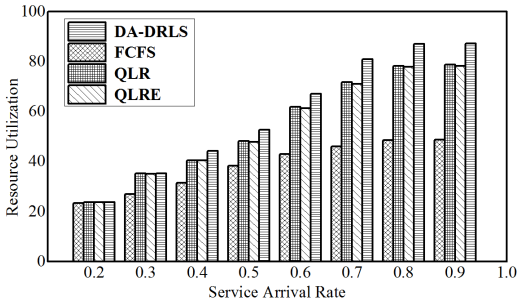


Fig. 10. Resource utilization analysis for different arrival rates

the total amount of resource (RC) the system has, can be calculated as:

$$RC = \sum_{i=1}^{n} \sum_{j=1}^{J} rc_{ij} \tag{15}$$

Thus, the resource utilization can be further calculated as:

$$RU = \frac{1}{L} \sum_{i=1}^{n} \left(\frac{\sum_{l=1}^{L} r_{il} \times tag_{il}}{\sum_{j=1}^{J} rc_{ij}} \right) \tag{16}$$

where, r_{il} is the requested amount of i^{th} resource type and tag_{il} is a binary valued variable that indicates whether any i^{th} type resource is allocated to the r_i resource demand by SR_l request. If this statement is true, then, $tag_{il} = 1$, otherwise, $tag_{il} = 0$.

Fig. 10 plots the average resource utilization of four approaches with respect to different arrival rates. For each arrival rate, we have estimated the utilization for 100 seconds (T) time span which is increasing for all the approaches with the increasing arrival rates. At the starting point, i.e. arrival rate up to 0.3, all the four approaches exhibit almost same utilization because there is a comparatively small amount of resource demand that the system can satisfy easily. In contrast, as time progresses, because of the straightforward allocation policy, the least number

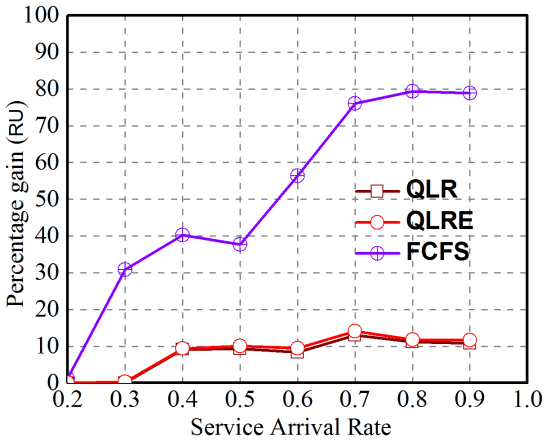


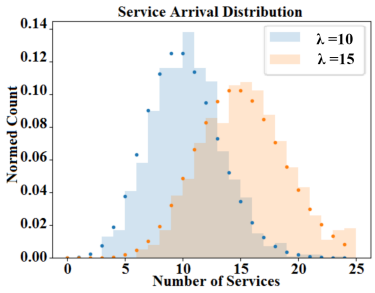
Fig. 11. Resource utilization percentage gain for different arrival rates

of requests of FCFS can be attained by the system. Thus, FCFS shows the least utilization throughout the different ranges of arrival rates. QLR and QLRE utilize almost same average resource, however, it should be noticed that utilization is slightly smaller in case of QLR. This happens because QLRE needs a little higher time to allocate proper resources for its requests. Above all, our approach, DA-DLRS has the highest utilization for each arrival rate because of its quick and efficient allocation policy. Fig. 11 shows the percentage gain of DA-DRLS in resource utilization with respect to other approaches.

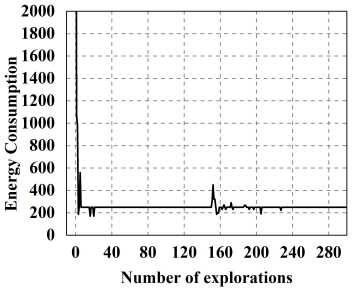
6.6. Convergence

This analysis is done to show the convergence of response time and energy consumption with drift in the final solution as there is a sudden change in demand. The resultant drift detection is shown in Fig. 12a. In a dynamic environment like IoT ecosystem, there can be certain cases which can cause drift. For example, if there is an immediate change in the demand that more service requests arrive or some resources are added to the resource pool which was previously not available. This affects or changes the result of the final solution. It is important to handle these changes because naively, it requires a complete re-optimization of the entire solution. Therefore, if the algorithm adapts itself according to these types of drifts then this re-optimization effort can be reduced.

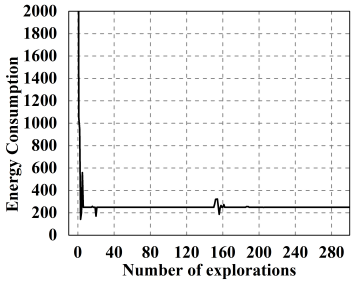
To emulate this scenario, we have induced the demand drift after the 150^{th} exploration, Fig. 12 shows the energy consumption and response time optimization analysis with respect to drift. Now, if a drift in the service request arrival pattern occurs, then the learned network got stale and the learning process needs to be initiated again. The new convergence of the reward requires a complete rerun of the entire process of policy generation. Several policy updates are required to achieve the final trajectory for the new



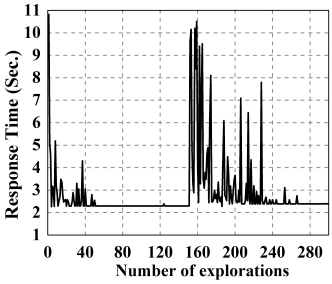
(a) Change detection in service arrival rate in two subsequent windows using KL-Divergence given in Equation 14



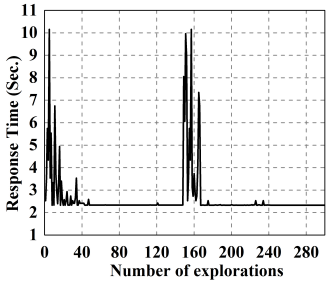
(b) Energy consumption updating process for the allocation of resources without drift adaptive learning



(c) Energy consumption updating process for the allocation of resources while considering drift adaptive learning



(d) Response time updating process for the allocation of resources without considering drift adaptive learning



(e) Response time updating process for the allocation of resources while considering drift adaptive learning

Fig. 12. Energy consumption and response time convergence analysis when there is a demand drift in service arrival pattern

set goal after drift. As shown in Fig. 12b, the energy consumption shows re-optimization form 151^{th} to 230^{th} exploration, and in Fig. 12d, response time from 151^{th} to 250^{th}. The proposed approach after using Algorithm 2 for adaptive learning, obtains an early convergence for both the metrics as less number of episodes are required to achieve less energy consumption and response time. From Fig. 12c, and 12e, we can observe that energy consumption achieves convergence between 151^{th} to 162^{th} and response time reaches to its convergence after 176^{th}.

6.7. Trade-off

The whole approach will be changed a bit if the objective function requires a trade-off between the monitoring parameters. The proposed approach is also tested for such trade-off situations. For the analysis of trade-off between response time and energy consumption, we have used a threshold parameter η. To achieve a trade-off reward using η, we have used the following Equation 17. The equation is similar to the Equation 7 except the difference that here a weight is provided to both the response time and the energy consumption.

$$r(s,a) = \eta \frac{1}{RT(s,a)} + (1-\eta)\frac{1}{log(EC(s,a))} \tag{17}$$

A high value of η gives more weight to response time and less to the energy consumption and vice-versa. For the

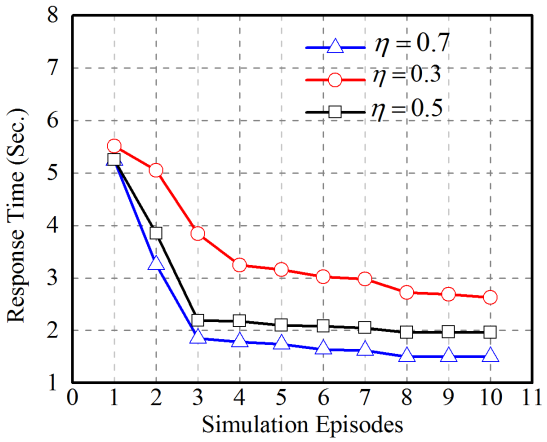


Fig. 13. Response time analysis providing different weights to response time parameter

analysis, we have used different values of η as $\eta = 0.3, 0.5,$ and 0.7 to develop a policy which minimizes the energy consumption and response time while providing different levels of importance to the response time and energy consumption parameter. Thus, a policy can be obtained that will result in a trade-off between the monitoring metrics while the value of the threshold trade-off parameter is an important factor to get such policies. Fig. 13 and Fig. 14 depict the results for response time and energy consump-

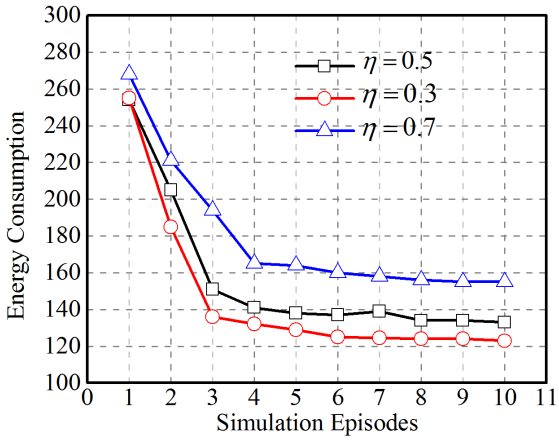


Fig. 14. Energy analysis with different weights to the energy parameter

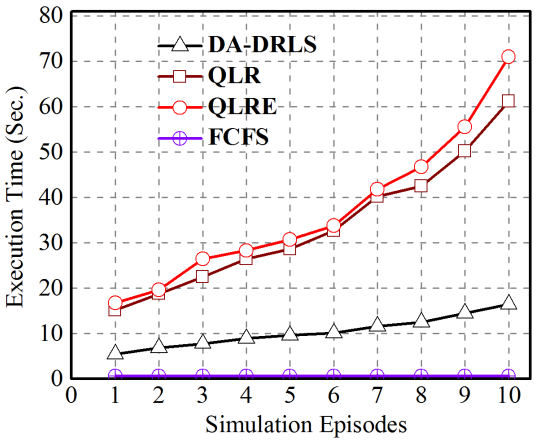


Fig. 15. Execution time analysis of the proposed DA-DRLS with FCFS, QLR, and QLRE approach

tion trade-off as the iteration of the episodes increases. The number of episodes iteration used in the analysis is same as in the previous analysis.

- Case 1: Interestingly, we can notice that when $\eta = 0.3$, the amount of reduction in the response time is less due to its less consideration in the reward maximization function given in Equation 17. Notwithstanding, energy consumption achieves the maximum reduction as its contribution weight is 0.7 in the resultant reward function.
- Case 2: When $\eta = 0.5$, both the two parameters gain average values as they contribute equal weights to the reward function.
- Case 3: For $\eta = 0.7$, the situation is just a reverse from Case 1.

7. Discussion

Our intended approach DA-DRLS, a probabilistic model of decision making, is elementarily a mixture of DNN and RL. Unlike supervised and unsupervised learning, DA-DRLS is appropriate for the agent-based modeling usually when the agents have the ability to attain a certain maximization of the objective function through interactions with the subject environment. In this section, we have enumerated the most significant observations of our research:

- It has been empirically observed that adding more number of neurons and hidden layers, does not substantially improve the energy consumption and response time for the considered scenario. On contrary, increase in the number of hidden layers and neurons, in turn, raises the overall complexity. Therefore, we have used one hidden layer for our analysis.
- If any unanticipated phenomenon occurs, the scheduler of DA-DRLS learns a fresh optimal policy using KL-divergence considering the recent circumstance and converges to a new maxima. Thus, it is well adaptive to the dynamically changing environment like IoT ecosystem.
- DA-DRLS deals satisfactorily with energy consumption and response time issues while allocating the incoming requests to the resources, ensuring a fair utilization. Despite this, one important research question that might be appeared here is that, DA-DRLS does not consider the priority levels of different incoming traffic. As different applications of IoT have different levels of urgency, therefore, priority can vary from application to application. In our approach, we have tried to contribute a generalized method to efficiently handle IoT application requests while allocating adequate resources.
- We have also attempted to present an analysis maintaining a trade-off between two important monitoring metrics: energy consumption and response time using a new parameter. However, deciding a trade-off is crucial depending upon different QoS requirements of diverse IoT applications, and it is the responsibility of the service providers to establish a trade-off between several QoS metrics.
- The conventional RL algorithms try to find the exact solution by exploring all the possible trajectories. On the other hand, the gradient based reinforcement learning combined with a DNN tries to approximate the solution which reduces the amount of computation. Fig. 15 demonstrates the execution time required for the proposed DA-DRLS with respect to other approaches. It is evident that DA-DRLS requires less execution time as compared to QLR and QLRE. However, in case of FCFS, there is no decision making computation required, so it has a considerably less execution time. On the other hand, it underperforms in context of energy consumption, response time, and resource utilization.

8. Conclusion and Future direction

IoT resources can be managed using a policy for allocation to the incoming user requests. This paper provides an efficient modeling of the IoT ecosystem where service requests arrive with heterogeneous resource demands. The proposed policy generation model fits well and provides an efficient request-resource mapping in an IoT ecosystem. It uses a combination of deep learning and reinforcement learning in order to explore the environment for different resource allocation possibilities and select the optimal solution for resource allocation. The simulation analysis shows that our proposed approach provides a significant amount of improvement in context of monitoring metrics as compared to FCFS, QLR, and QLRE. It has been observed that the proposed approach achieves the desired solution after several explorations and effectively penalize less prominent solutions using the proposed reward function. An extensive set of simulation results has revealed that:

- DA-DRLS is a robust mechanism that shows better results under heavy traffic flow situation also.
- It utilizes the system resources properly that ensures a balanced load in the system as well.
- The adaptability analysis shows that the approach is efficacious enough in case of changing environment. Overall, the paper provides a compelling solution to the resource allocation problem in IoT ecosystem while ensuring dynamic user requests and drifting environment.
- The proposed approach effectively provides trade-off between monitoring metrics: energy consumption and response time using different values of trade-off parameter.

As a future work, a priority based approach between same IoT applications and different IoT applications will be investigated using real experiment set up.

References

Abdullah, S., & Yang, K. (2014). An energy efficient message scheduling algorithm considering node failure in iot environment. *Springer Wireless personal communications*, *79*, 1815–1835.

Adan, I., & Resing, J. (2015). Queueing systems. *Eindhoven University of Technology, Department of Mathematics and Computing Science*, (pp. 1–182).

Akdere, M., Çetintemel, U., & Tatbul, N. (2008). Plan-based complex event detection across distributed sources. *Proceedings of the VLDB Endowment*, *1*, 66–77.

Alaa, M., Zaidan, A., Zaidan, B., Talal, M., & Kiah, M. L. M. (2017). A review of smart home applications based on internet of things. *Elsevier Journal of Network and Computer Applications*, *97*, 48–65.

Alam, F., Mehmood, R., Katib, I., & Albeshri, A. (2016). Analysis of eight data mining algorithms for smarter internet of things (iot). *Procedia Computer Science*, *98*, 437–442.

Alam, F., Mehmood, R., Katib, I., Albogami, N. N., & Albeshri, A. (2017). Data fusion and iot for smart ubiquitous environments: A survey. *IEEE Access*, *5*, 9533–9554.

Alodib, M. (2016). Qos-aware approach to monitor violations of slas in the iot. *Elsevier Journal of Innovation in Digital Ecosystems*, *3*, 197–207.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., & Bengio, Y. (2012). Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, .

Bedi, G., Venayagamoorthy, G. K., Singh, R., Brooks, R. R., & Wang, K.-C. (2018). Review of internet of things (iot) in electric power and energy systems. *IEEE Internet of Things Journal*, *5*, 847–870.

Bellman, R. E., & Dreyfus, S. E. (2015). *Applied dynamic programming* volume 2050. Princeton university press.

Bhorkar, A. A., Naghshvar, M., Javidi, T., & Rao, B. D. (2012). Adaptive opportunistic routing for wireless ad hoc networks. *IEEE/ACM Transactions On Networking*, *20*, 243–256.

Bigi, B. (2003). Using kullback-leibler distance for text categorization. In *Springer European Conference on Information Retrieval* (pp. 305–319).

Chinchali, S., Hu, P., Chu, T., Sharma, M., Bansal, M., Misra, R., Pavone, M., & Katti, S. (2018). Cellular network traffic scheduling with deep reinforcement learning. In *32 Association for the Advancement of Artificial Intelligence*.

Chollet, F. et al. (2015). Keras. URL: `https://github.com/fchollet/keras`.

Chowdhury, A., & Raut, S. A. (2018). A survey study on internet of things resource management. *Elsevier Journal of Network and Computer Applications*, *120*, 42–60.

Delicato, F. C., Pires, P. F., Batista, T. et al. (2017). Resource management for internet of things, . (pp. 1–116).

Di Felice, M., Chowdhury, K. R., Kassler, A., & Bononi, L. (2011). Adaptive sensing scheduling and spectrum selection in cognitive wireless mesh networks. In *Computer communications and networks (ICCCN), 2011 proceedings of 20th international conference on* (pp. 1–6).

Gazis, V. (2017). A survey of standards for machine-to-machine and the internet of things. *IEEE Communications Surveys & Tutorials*, *19*, 482–511.

Ge, Y., Zhang, X., & Han, B. (2017). Complex iot control system modeling from perspectives of environment perception and information security. *Springer Mobile Networks and Applications*, *22*, 683–691.

Ghosh, R., & Simmhan, Y. (2018). Distributed scheduling of event analytics across edge and cloud. *ACM Transactions on Cyber-Physical Systems*, *2*, 24.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Elsevier Future generation computer systems*, *29*, 1645–1660.

Huang, Z., van der Aalst, W. M., Lu, X., & Duan, H. (2011). Reinforcement learning based resource allocation in business process management. *Elsevier Data & Knowledge Engineering*, *70*, 127–145.

Issarny, V., Bouloukakis, G., Georgantas, N., & Billet, B. (2016). Revisiting service-oriented architecture for the iot: a middleware perspective. In *Springer International Conference on Service-Oriented Computing* (pp. 3–17).

Kamalinejad, P., Mahapatra, C., Sheng, Z., Mirabbasi, S., Leung, V. C., & Guan, Y. L. (2015). Wireless energy harvesting for the internet of things. *IEEE Communications Magazine*, *53*, 102–108.

Khan, M. I., & Rinner, B. (2012). Resource coordination in wireless sensor networks by cooperative reinforcement learning. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on* (pp. 895–900).

Kifer, D., Ben-David, S., & Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30* (pp. 180–191). VLDB Endowment.

Kumar, P., DSouza, M., & Das, D. (2018). Efficient anomaly detection methodology for power saving in massive iot architecture. In *Springer International Conference on Distributed Computing and Internet Technology* (pp. 256–262).

Kumar, T. P., & Krishna, P. V. (2018). Power modelling of sensors for iot using reinforcement learning. *International Journal of Advanced Intelligence Paradigms*, *10*, 3–22.

Leu, J.-S., Chen, C.-F., & Hsu, K.-C. (2014). Improving heterogeneous soa-based iot message stability by shortest processing time scheduling. *IEEE Transactions on Services Computing*, *7*, 575–585.

Li, L., Li, S., & Zhao, S. (2014). Qos-aware scheduling of services-oriented internet of things. *IEEE Trans. Industrial Informatics*, *10*, 1497–1505.

Li, L., Lv, Y., & Wang, F.-Y. (2016). Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, *3*, 247–254.

Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks* (pp. 50–56).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, .

Muhammed, T., Mehmood, R., & Albeshri, A. (2017). Enabling reliable and resilient iot based smart city applications. In *International Conference on Smart Cities, Infrastructure, Technologies and Applications* (pp. 169–184). Springer.

Narman, H. S., Hossain, M. S., Atiquzzaman, M., & Shen, H. (2017). Scheduling internet of things applications in cloud computing. *Springer Annals of Telecommunications*, *72*, 79–93.

Nemati, S., Ghassemi, M. M., & Clifford, G. D. (2016). Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach. In *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the* (pp. 2978–2981).

Pattar, S., Buyya, R., Venugopal, K., Iyengar, S., & Patnaik, L. (2018). Searching for the iot resources: Fundamentals, requirements, comprehensive review and future directions. *IEEE Communications Surveys & Tutorials*, *20*, 2101–2132.

Peng, Z., Cui, D., Zuo, J., Li, Q., Xu, B., & Lin, W. (2015). Random task scheduling scheme based on reinforcement learning in cloud computing. *Springer Cluster computing*, *18*, 1595–1607.

Sanislav, T., Zeadally, S., Mois, G. D., & Folea, S. C. (2018). Wireless energy harvesting: Empirical results and practical considerations for internet of things. *Elsevier Journal of Network and Computer Applications*, *121*, 149–158.

Sarangi, S. R., Goel, S., & Singh, B. (2018). Energy efficient scheduling in iot networks. In *SAC Proceedings of the 33rd Annual ACM Symposium on Applied Computing, 2018.* (pp. 733–740).

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning* (pp. 1889–1897).

Sharma, R., Kumar, N., Gowda, N. B., & Srinivas, T. (2018). Packet scheduling scheme to guarantee qos in internet of things. *Springer Wireless Personal Communications*, *100*, 557–569.

Sohn, I., Yoon, S. W., & Lee, S. H. (2018). Distributed scheduling using belief propagation for internet-of-things (iot) networks. *Springer Peer-to-Peer Networking and Applications*, *11*, 152–161.

Taneja, M. (2014). A framework for power saving in iot networks. In *IEEE Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on* (pp. 369–375).

Tsiftes, N., & Voigt, T. (2018). Velox vm: A safe execution environment for resource-constrained iot applications. *Elsevier Journal of Network and Computer Applications*, *118*, 61–73.

Van, D. P., Rimal, B. P., Chen, J., Monti, P., Wosinska, L., & Maier, M. (2016). Power-saving methods for internet of things over converged fiber-wireless access networks. *IEEE Communications Magazine*, *54*, 166–175.

Wang, K., Wang, Y., Sun, Y., Guo, S., & Wu, J. (2016). Green industrial internet of things architecture: An energy-efficient perspective. *IEEE Communications Magazine*, *54*, 48–54.

Wei, Z., Zhang, Y., Xu, X., Shi, L., & Feng, L. (2017). A task scheduling algorithm based on q-learning and shared value function for wsns. *Elsevier Computer Networks*, *126*, 141–149.

Wollschlaeger, M., Sauter, T., & Jasperneite, J. (2017). The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Industrial Electronics Magazine*, *11*, 17–27.

Xu, C., Qin, T., Wang, G., & Liu, T.-Y. (2017). Reinforcement learning for learning rate control. *arXiv preprint arXiv:1705.11159*, .

Xue, Y., Lin, Y., Feng, Z., Cai, H., & Chi, C. (2008). Autonomic joint session scheduling strategies for heterogeneous wireless networks. In *IEEE Wireless Communications and Networking Conference. WCNC 2008.* (pp. 2045–2050).

Yang, Z., Xu, W., Pan, Y., Pan, C., & Chen, M. (2017). Energy efficient resource allocation in machine-to-machine communications with multiple access and energy harvesting for iot. *IEEE Internet of Things Journal*, (pp. 1–1).

Yau, K.-L. A., Kwong, K. H., & Shen, C. (2013). Reinforcement learning models for scheduling in wireless networks. *Springer Frontiers of Computer Science*, *7*, 754–766.

Zhai, D., Zhang, R., Cai, L., Li, B., & Jiang, Y. (2018). Energy-efficient user scheduling and power allocation for noma based wireless networks with massive iot devices. *IEEE Internet of Things Journal*, *5*, 1857–1868.

Zhao, D., Chen, Y., & Lv, L. (2017). Deep reinforcement learning with visual attention for vehicle classification. *IEEE Transactions on Cognitive and Developmental Systems*, *9*, 356–367.