A Reinforcement Learning Algorithm for Resource Provisioning in Mobile Edge Computing Network

Huynh Thi Thanh Binh†, Nguyen Phi Le†, Nguyen Binh Minh†, Trinh Thu Hai†, Ngo Quang Minh†, Do Bao Son*,
†School of Information and Communication Technology, Hanoi University of Science and Technology, Vietnam
*Faculty of Information Technology, University of Transport Technology, Vietnam
Email: {binhht, lenp, minhnb}@soict.hust.edu.vn
Email: {hai.tt184255, minh.nq173554}@sis.hust.edu.vn
Email: sondb@utt.edu.vn

***Abstract*—Mobile edge computing (MEC) is a model that allows integration of computing power into telecommunications networks, to improve communication and data processing efficiency. In general, providing power to ensure the computing power of edge servers in the MEC network is very important. In many cases, ensuring continuous power supply to the system is not possible because servers are deployed in hard-to-reach areas such as outlying areas, forests, islands, etc. This is when renewable energy prevails as a viable source of power for ensuring stable operation. This paper addresses resource provisioning in the MEC network using renewable energy. We formulate the problem as a Markov Decision Problem and introduce a new approach to optimize this problem in terms of energy and time costs by using a reinforcement learning technique. Our simulation validates the efficacy of our algorithm, which results in a cost three times better than the other methods.**

***Index Terms*—Mobile Edge Computing, Fog Computing, Resource Provisioning, Markov Decision Process, Energy Harvesting, Proximal Policy Optimization.**

I. Introduction

The idea of Mobile Edge Computing (MEC) (also known as Fog Computing) was proposed in 2014 by the European Telecommunications Standards Institute (ETSI) as a way to extend the ability of cloud computing to the edge of the network to improve data processing and storage. MEC provides information Technology and cloud-computing capabilities within the Radio Access Network (RAN) in close proximity to mobile subscribers [1]. MEC servers are deployed along with base stations to offload workloads from mobile users. The concept of fog computing [2] is similar to MEC, in which jobs and applications will be partially processed or processed and then transferred to the cloud server for completion [3]. Compared to mobile cloud computing (MCC), MEC has many advantages such as low latency, speed up computation, reduce network traffic and ensure quality of service (QoS). An effective MEC platform should provide an offload policy reasonable. Depending on the attributes of the job, this policy helps determine whether the job will be handled at the edge or in the cloud [4].

MEC servers use less power than large data centers in cloud systems. However, with the popular implementation of MEC, energy has become a major concern. For devices that do not have a stable power supply or battery-powered devices with a limited power source, the ability to perform calculations may be affected by insufficient power, services will be interrupted or out of stock. We can use resource management techniques to optimize execution time and operating costs [5], [6], so we can save energy. Also we can increase the capacity of the battery or charge the battery more often. However, large battery capacity will increase hardware costs, and charging in some cases is not possible if the nodes are deployed in hard-to-reach locations [7]. This challenge can be overcome with the development of energy harvesting technology, which allows the use of renewable energy such as wind-solar energy, which will contribute to powering the system. Off-grid renewable energy, such as solar radiation and wind energy, has recently emerged as a viable and promising power source for various IT systems thanks to the recent advancement of energy harvesting techniques [8], [9]. Compared with traditional grid energy which is normally generated by coal-fired power plants, employing renewable energy significantly reduces the amount of carbon emission. Moreover, the use of renewable energy sources reduces the need for human intervention, which is difficult if not impossible for certain types of application scenarios where the devices are hard and dangerous to reach. However, designing green MEC powered by renewable energy is much more challenging compared to green communication. The harvested energy is integral to the operation of both the base stations and the edge servers, so it will determine the power consumption policy of the entire system.

The main contributions of this paper are as follows:

- Propose an energy-wise optimal resource provisioning model for the MEC network as a Markov Decision Problem (MDP).
- Propose an approach to solve mentioned MDP, namely Proximal Policy Optimization (PPO) algorithm.
- Analyze and assess the experimental results to show the efficiency of the proposed algorithm.

The rest of this paper is organized as follows. Section II introduces related works. Section III presents the definitions used for formulating the problem. Section IV describes the MDP, while the solution proposed to solve MDP is elaborated in section V. Section VI explains the setup of our experiments

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

and reports the computed results. The paper concludes in section VII with discussions on the future extension of this research.

II. Related works

There have been several research on renewable energy sources to power mobile devices or MEC servers. Trinh et al. [10] study MEC's issues related to energy management on end-user limited energy devices. They tested and analyzed the effectiveness of policies that reduce the processing data load and the impact of these policies on power consumption according to different data requirements. But they are only considering optimizing the power consumption on the user side. Yang et al. [11] also only consider minimizing the power consumption on end users' devices by modeling the convex problem and solving with a low-complexity iterative algorithm in which closed-form solutions are obtained in each step. Mao et al. [12] investigated MEC systems with energy harvesting mobile devices. The execution cost, which addresses the execution delay and task failure, was adopted as the performance metric. A dynamic computation offloading policy, namely, the Lyapunov optimization-based algorithm fog offloading computation was developed. Alternatively, some existing solutions on wireless powered MEC systems have also been proposed to exploit the ambient RF signals to supply the mobile devices. For instance, in [13], a wireless powered single-user MEC system was considered where the user harvested RF energy from a dedicated access point (AP) for computation offloading, and the CPU frequency for each required CPU cycle was optimized. In [20], the authors considered a multi-antenna AP delivering RF energy to multiple users, where the computing tasks were jointly executed by the AP and users via optimizing transmit energy beamforming, offloading decision and resource allocation with the minimum of the AP's energy consumption. The authors of [14] designed a new time frame in a binary computation offloading, in which an AP first broadcasted the RF energy in the downlink and then the energy-constrained mobile devices offloaded their tasks to the AP at their allocated time slots. With energy powered by the wireless RF signal component, the fair energy efficiency framework in a multi-user MEC system was proposed in [15] where full-duplex was employed at AP to support energy delivery and computation offloading simultaneously. The work in [16] focused on a wireless powered multi-user MEC system where a multi-antenna AP and an MEC server were separately placed. The aim was to minimize the total energy cost of all mobile devices with joint optimization of the offloading decisions, time switch, local computation/offloading powers. In order to eliminate the double-near-far effect in a two-user wireless powered MEC system, the end-user device closer to AP was selected as a relay to help offload the far-away end-user device's computation tasks to the edge cloud. In this circumstance, [17] paid more attention to minimize the total transmit energy of the AP and [7] concentrated on the maximization of energy efficiency under the constraints of the computational tasks. Considering a hybrid MEC system consisting of a multiple-antenna cellular base station and a WiFi access point, the authors of [18] studied the problem of maximizing the total energy saving of all mobile devices by jointly optimizing the computation and radio resources along with dynamic interface selection. Recently, the study on RF-based energy harvesting was applied to an unmanned aerial vehicle-enabled multi-user MEC system [19], in which both binary and partial computation offloading modes were respectively taken into account for the computation rate maximization problems. The authors of [20] investigated the optimal resource provisioning policy of an energy harvesting MEC, in which the state space is formulated as discrete and finite. But in the reality of communication network, the application can be limited since the state space of real systems is continuous and hence, more complex. Our study improves the model to allow a continuous-value state space, which means a better representation and adopts a robust algorithm to solve the resulting problem.

III. System Description

We study one version of the Mobile Edge Computing Network [1], which includes a base station and a set of edge servers, located in the same position and share the same power source in cell cite. We also have a green energy source to provide power to the system and batteries to store excess energy. Our model is based on [20] but we have some modifications on calculating the electricity usage on edge servers.

A. Workload

We consider a discrete-time model by isolating the working time frame into schedule vacancies of equivalent length filed by $t = 0, 1, ...$, every one of which has a span that matches the timescale at which the edge gadget can change its processing limit (for example number of dynamic servers). We use $x \in \mathcal{L}$ to represent a position coordinate in the service zone $\mathcal{L}$. Let $\lambda(x,t)$ show the expected workload arrival rate in position *x*, and $\theta(x,t)$ be the wireless transmission rate between the base station and position *x*, which relies on the present uplink wireless channel conditions. So $\lambda(t) = \sum_{x\in\mathcal{L}} \lambda(x,t) \in [0, \lambda_{max}]$ is The total workload is taken to the edge system, where λ_{max} is the maximum workload possible at the edge system. The system determines the number of workload $\mu(t) \leq \lambda(t)$ that will be handled locally. The remaining workload $\nu(t) \triangleq \lambda(t) - \mu(t)$ will be transferred to the cloud for processing. The edge system also determines at the beginning of the time slot the number of servers used, represent by $m(t) \in [0, M]$, where M is the maximum number of edge servers. These servers are used to handle the local workload $\mu(t)$. Because changing the quantity of servers during work execution are troublesome and much of the time inconceivable, we just permit deciding the quantity of servers used at the beginning of each time slot but not within the slot.

B. Delay cost

We model the base station as a queueing system [21], the average usage of the base station is $\rho(t) = \sum_x \lambda(x,t)/\theta(x,t)$,

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

and a total wireless access and transmission tardiness is $c_{wi}(t) = \sum_x \lambda(x,t)/[\theta(x,t)(1-\rho(t))]$. Next, we model the latency of handling the workload generated at the edge servers. For the workload is processed local, the tardiness cost $c_{dl}(t)$ is mainly handle the latency due to limited computing power at the local edge servers. The transmission tardiness from end-user devices to local servers is negligible due to the same location. We represent the tardiness performance in time slot t by $c_{dl}(m(t),\mu(t))$. The service process at a server is modeled as an M/G/1 queue and the tardiness cost of system is $c_{dl}(m(t),\mu(t)) = \frac{\mu(t)}{m(t)S-\mu(t)}$, where S is the service rate of each server in local.

For the workload is offloaded, the tardiness cost $c_{off}(t)$ is primarily the transmission tardiness due to network round trip time (RTT), which varies according to the network congestion status. We model the network congestion status, represent by k(t), including RTT and tardiness from the cloud for simple. RTT can determine with basic RTT measurement tools (like using the ping command) and the cloud latency can be obtained from cloud service providers based on the resources they provide. Thus, the offloaded tardiness cost is $c_{off}(k(t),\lambda(t),\mu(t)) = k(t)(\lambda(t)-\mu(t))$. The total tardiness cost of system is

$$c_{delay}(k(t),\lambda(t),m(t),\mu(t)) = c_{dl}(m(t),\mu(t)) + c_{off}(k(t),\lambda(t),\mu(t)) + c_{wi}(\lambda(t)) \quad (1)$$

C. Power

In a time slot, the edge system needs an amount of power to meet the power requirements of the base station and the edge server. First, depend on the offloading and the autoscaling plan, we represent the power consumption of base station by $p_{base}(\lambda(t)) = p_{sta} + p_{dyn}(\lambda(t))$ where p_{sta} is the static power for the basic operation of base station and $p_{dyn}(\lambda(t)) = \alpha.\lambda(t)$ is the dynamic power consumption for transmitting workload to system, where α is dynamic power consumption for transmitting one unit of worload. The power supplied to edge servers depends on the number of active servers to handle local workload. We denote $p_{edge}(\lambda(t),m(t),\mu(t))$ as the computing power of all edge servers, which linear with $\lambda(t)$, $m(t)$ and $\mu(t)$, to represent the computing power demand. The total power needs in time slot t is

$$p(\lambda(t),m(t),\mu(t)) = p_{base}(\lambda(t)) + p_{edge}(\lambda(t),m(t),\mu(t)) \quad (2)$$

We use $g(t)$ as the green power budget. Because the lack of certainty of the renewable energy supply, $g(t)$ is performed after the system makes a decision about the offloaded workload and the number of active servers. We rely on the state of the environment $e(t)$, so that the system can estimate its energy budget in the current time slot. For example, the day in good weather conditions will get much more solar energy. we model $g(t)$ and $e(t)$ as an independent and identically distributed with $e(t)$ given, which satisfies a conditional probability distribution $P_g(g(t)|e(t))$.

D. Battery

When the energy generated from renewable energy sources is not enough to meet the power requirements, the battery will be used ensure the stable operation of the system. We denote B is the battery capacity. We represent the battery state at starting point of time slot t by $b(t) \in [0,B]$. To guarantee basic operation of the system, We represent $b(t) = 0$ to this limit voltage.

At the start of each time slot t, the amount of renewable energy gained is unforeseeable. the edge system uses a policy which meets $p_{edge}(\lambda(t),m(t),\mu(t)) \le max\{b(t) - p_{base}(\lambda(t)),0\}$ to avoid activating redundant power supplies by making offloading and autoscaling expansion decisions.

- If $p_{base}(\lambda(t)) \ge b(t)$, $p_{edge}(\lambda(t),m(t),\mu(t))$ will be zero, that means the edge system transmits all workloads to the cloud when the current battery power can't even support basic operation and transmission in the current slot. In this case, the backup power source will be used to maintain basic operation for the current slot. To activate the backup power needed a cost is $c_{bak}(t) = \psi p_{base}(\lambda(t))$, where $\psi > 0$ is a constant representing the cost of backup power usage. The battery state then changes to $b(t+1) = b(t) + g(t)$ in next time slot.
- If $p_{base}(\lambda(t)) \le b(t)$, the workload $\mu(t) \le \lambda(t)$ be processed at the edge servers, and the power demand must satisfy $p_{edge}(\lambda(t),m(t),\mu(t)) \le b(t) - p_{base}(\lambda(t))$. The battery will be charged or discharged depending on the renewable energy collected $g(t)$ and the power of the edge servers $p_{edge}(\lambda(t),m(t),\mu(t))$, accordingly:

- If $g(t) \ge p(\lambda(t),m(t),\mu(t))$, then the excess $g(t) - d(\lambda(t),m(t),\mu(t))$ is stored in the battery until the battery is full capacity:

$$b(t+1) = \max\{b(t)+g(t)-p(\lambda(t),m(t),\mu(t)),B\} \quad (3)$$

- If $g(t) < p(\lambda(t),m(t),\mu(t))$, then the battery has to be discharged to deal with the energy shortage $p(\lambda(t),m(t),\mu(t)) - g(t)$.

$$b(t+1) = b(t) + g(t) - p(\lambda(t),m(t),\mu(t)) \quad (4)$$

We represent $c_{battery}(t)$ as the battery devaluation cost in a time slot. We ignore the case of system failure during charging or discharging the battery for simplicity. Specifically, $c_{battery}(t) = \omega \cdot max\{p(\lambda(t),m(t),\mu(t)) - g(t),0\}$ where $\omega > 0$ is the normalized unit devaluation cost.

IV. Problem Statement: Decision Process

We model this resource provisioning problem as a MDP. Next we introduce elements of this model.

A. Definitions

- **State space**: Each state s is a four-dimensional vector consisting of the following parameters: λ (the workload arrival rate), b (the battery level), k (the congestion rate in the network), and e (the indicator of green-power harvesting potential). These values are from a continuous range, which means the state space is infinite.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

- **Action space**: The action a is a scale factor in the range $[0, 1]$. This value will determine the computing demand p_{edge} from the continuous range [0, $\max p_{edge}$], where $\max p_{edge} = \min(b - p_{base}, p_{edge}(m_{max}, \lambda))$. Based on p_{base}, the final autoscaling and offloading actions (number of active servers m and amount of offload workload μ, respectively) is obtained by minimizing the cost.
- **Transition function**: The transition functions of λ, k, e is independent of the action taken, and are modeled by an implicit probability distribution.
  The transition function of b, however, is deterministic, specifically:

$$b(t+1) = \begin{cases} \max(\min(b(t) + g(t), B), 0), & \text{if } p_{base} > b(t) \\ \max(\min(b(t) - p_{base}(\lambda(t)) - a(t) + \\ g(t), B), 0), & \text{otherwise} \end{cases}$$

- **Cost**: The cost function is the summation of delay cost and energy cost (includes back-up power and battery power):

$$\begin{aligned} c(t) = (1-\beta)(c_{delay}(k(t), \lambda(t), m(t), \mu(t))) \\ + \beta(c_{bak}(\lambda(t)) + c_{battery}(p_{edge}(t), g(t))) \end{aligned}$$

  where $\beta(\beta \in [0, 1])$ is the balance coefficient between delay cost and energy cost. $\beta = 0.5$ means that delay cost and energy cost have same priority in optimizing; if $\beta > 0.5$, our mechanism focuses on minimizing delay cost with higher priority than energy cost; inversely, $\beta < 0.5$, energy cost is more prioritized than delay cost.
  Since the state transition does not depend on $m(t)$ or $\mu(t)$, they can be optimized given $s(t)$ and $a(t)$ by solving the following optimization problem:

$$\min_{m,\mu: d(m,\mu) = p_{edge}} c_{delay}(k(t), \lambda(t), m(t), \mu(t))$$

- **Policy**: is a function that maps a state from the state space to a probability distribution over the action space, that is, it represents the likelihood that each action is taken given the state.

B. Problem formulation

The resource provisioning problem in MEC network can be formulated as follows:

Input:

State space $\lambda(t) \times k(t) \times b(t) \times e(t)$ in time slot t with:

- workload at a time slot $\lambda(t)$,
- congestion state of network $k(t)$,
- battery level $b(t)$,
- environment state $e(t)$

Output:

Optimal policy $\pi : \lambda(t) \times k(t) \times b(t) \times e(t) \rightarrow a(t)$

Objective:

Minimize the cost function:

$$c_{delay}(k(t), \lambda(t), m(t), \mu(t)) \rightarrow Min$$

In the next section, we solve this problem using a reinforcement learning method.

V. Proposed method

In [20], the solution used a tabular learning method which approximates the value function c, the expected discounted total cost for a action state pair, (similar to Q). The policy is determined by minimizing the value function c over the set of actions for every state. This method is feasible for a discrete action set, but impractical for a continuous one since the minimizing operation can be tricky with non-linear approximation function. After this resource provisioning problem is formulated as a continuous MDP, including state space, action space, reward function and transition function, deep reinforcement learning methods, such as PPO can be used. Here we adopt an easy-to-implement framework of PPO2 [22]. Below, we will give a description of the algorithm 1.

1) Initialize a policy θ_0 of the policy network $\pi(a|s, \theta)$. Our goal is to optimize this policy network.
2) Initialize the parameter w_0 to optimize the value function network $v(s, w)$. Since the value function has not been implicitly defined in our problem, our agent have to learn it during the optimizing process.
3) For each iteration, we have the following steps:
   - Run our current policy over T time steps.
   - Calculate the advantage estimator. The estimator we use is:

$$A_t = r_t + \gamma r_{t+1} + \ldots + \gamma^{T-t-1} r_{T-1} - \hat{v}(s_t, w_i). \quad (5)$$

   - Optimize the surrogate objective function

$$L = L^{CLIP}(\theta) + c_1 L^{VF}(w) + E_t[c_2 S[\pi_\theta](s_t)] \quad (6)$$

     with respect to θ and w using gradient descent. We want to maximize this surrogate objective function. The new θ and w is updated.

Algorithm 1 PPO Algorithm

Initialize these parameter:
    θ_0 of the policy network $\pi(a|s, \theta)$
    w_0 of the value function network $\hat{v}(s, w)$
for $i = 0, 1, 2, \ldots, N$ **do**
    Run policy π_{θ_i} over T time steps
    Calculate advantage (5)
    Optimize surrogate (6) w.r.t θ, w
    Update θ_{i+1}, w_{i+1}
end for

VI. Performance evaluation

A. Experimental Settings

In our experimental study, we consider a system has 15 edge servers. The system data are updated at the beginning of every timeslot, each of which is 15 minutes. The workload to the system per second is randomly taken from 10 units to 100 units. The network congestion per unit is is randomly taken from 0.02 second to 0.06 second. The service rate is 20 units/second for each edge server. The battery volume is

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE I: Enviroment parameters

Parameter	Value	Unit
Number of servers	15	nodes
Length each time slot	15	minutes
Battery capacity	2000	watt hour (Wh)
Server service rate	20	units/second
Workload (λ)	[10,100]	units/second
Network congestion (k)	[0.02,0.06]	second/unit
Environment state (e)	{0, 1, 2}	
Back up power coefficient ϕ	0.15	
Battery depreciation coefficient ω	0.01	
Base station static power	300	Watt
Server power consumption	150	Watt
time steps per episode	96	

2000 Wh. We use traces of harvesting green energy in the real world from March 2016 to August 2016 in Belgium [23]. The environmental state is obtained by mapping from the current time in the day to the range [0,2] in the following manner: the time period of the day from 6 pm to 6 am corresponds to the range [0,0.5) - indicating low potential of energy harvesting, the time period from 6 am to 9 am and 3 pm to 6 pm corresponds to the range [0.5,1.5) - indicating medium potential, and the time period from 9 am to 3 pm corresponds to the range [1.5,2] - indicating high potential. Other parameters are shown in Table I. Training setup. For the training process, during each iteration the current policy is run for 96 time steps - the equivalent of 1 day of data. The model is trained for 96000 time slots which means 1000 iterations.

The settings of the experimental environment are shown in Table II. The simulation and training were developed in Python because it is a good support for running reinforcement learning techniques.

TABLE II: Simulation Setup

System	Intel® Core^{TM} i5, CPU 2.30GHz
Memory	8 GB
Simulator	python
Operating system	macOS High Sierra

Based on the above metrics, we compare the results obtained by the proposed algorithm with two approaches: Random and Fixed power.

- Q-learning [24]: finds an optimal policy in the sense that it minimizes the the total cost over any and all successive steps from the current state.
- Myopic: this approach bypasses the time correlation between states and decisions of the system and minimizes the cost function given in the current time state by using all available battery power.
- Random power: this approach uses random computing power for edge calculations in each time slot.
- Fixed power: this approach uses fixed computing power (when the available power is insufficient, use maximum power) for edge calculations in each time slot.

B. Experimental Results

Figure 1 compares the average total cost between the proposed algorithm and the five others. The results were obtained by running simulation for 1000 times slots with coefficient balance $\beta = 0.5$.. It was evident that our proposed algorithm, PPO, results in the lowest cost, only approximately 54.17% of that of random power and fixed 1kW, respectively; $58,18\%$ of that of fixed 0.4kW; 67.37%. We observed that the solution that PPO found has a small average cost at 6.5 compared with 9.4, 9.5, 11, 12, 12.1 that Myopic, Q-learning, fixed 0.4kW, random, fixed 1.0kW achieves, respectively. That means PPO algorithm finds better solution. The best solution of PPO is found earlier in 600^{th} time slot, afterward, the result is stable.

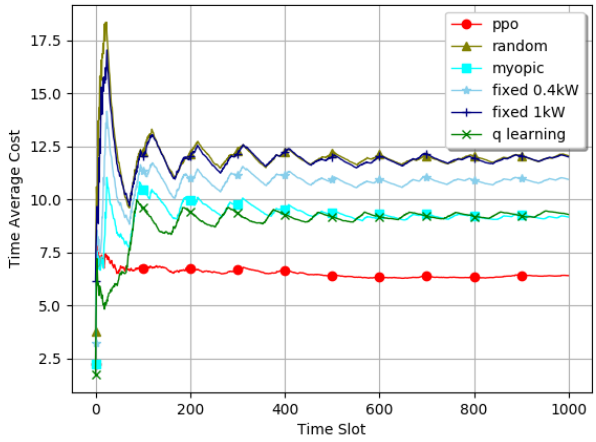


Fig. 1: Comparing average total cost of PPO, random power, Myopic, fixed power (0.4 and 1.0) and Q-learning

Figure 2 shows the average cost of energy for different algorithms. By considering the average cost of energy, we further demonstrate the efficiency of PPO over Myopic, Q-learning, random, fixed 0.4kW and fixed 1.0kW algorithms. The best performing methods are PPO and Deep Q Network (DQN), understandably, since they are able to take into account the implicit effects of the daily cycle of the environment states, ie. the energy harvesting potential on the long term reward. This means that they have the ability to make conservative actions even though the current battery resource is abundant, if there is an indication of future shortages in green power collection. Indeed, the average cost of energy of the solution found by PPO is just 0.99, while the solutions found by others is 4 to 7 times greater than PPO.

Considering the average cost of time, Myopic and fixed power 1.0kW perform slightly better than PPO as shown in Figure 3 when saves about 10% comparing with PPO algorithm. The Myopic strategy, unsurprisingly, achieves the lowest delay cost among the 6 methods since it only considers the immediate outcome. While this method is the best at

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

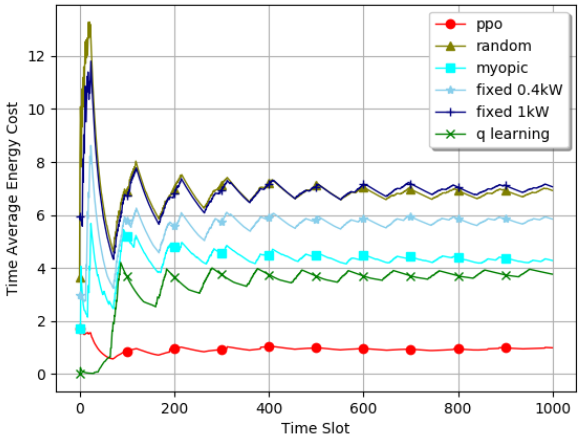


Fig. 2: Comparing average energy cost (back-up power and battery) of PPO, random power, Myopic, fixed power (0.4 and 1.0) and Q-learning

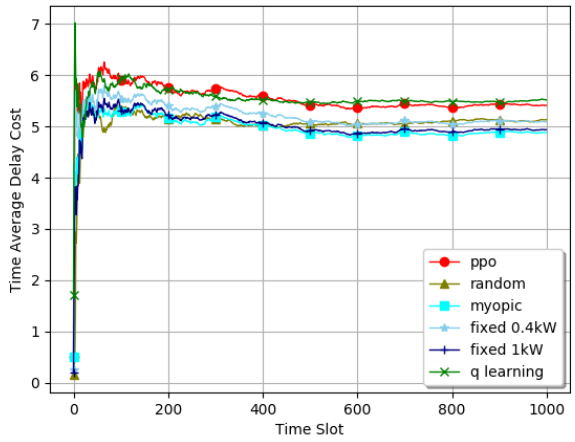
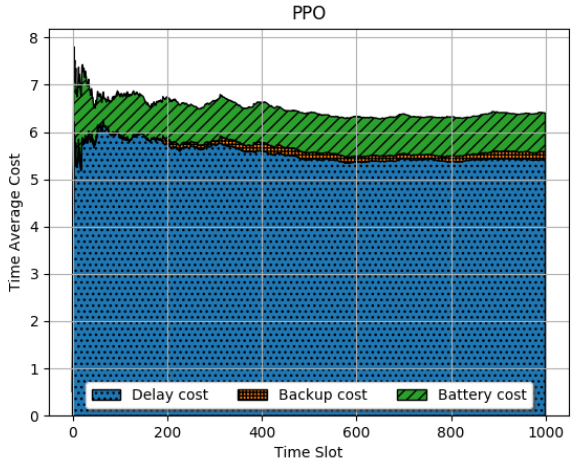


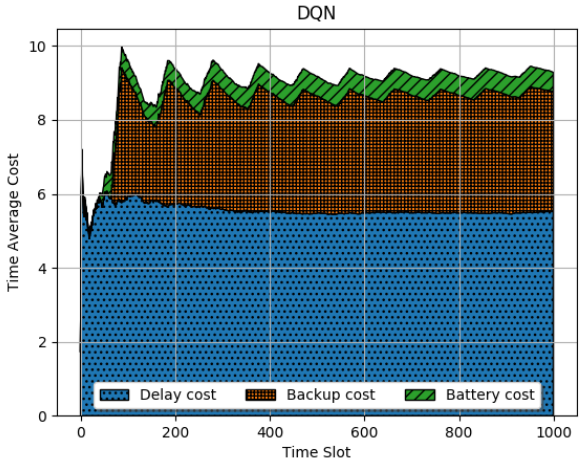
Fig. 3: Comparing average time cost of PPO, random power, Myopic, fixed power (0.4 and 1.0) and Q-learning

optimizing the proximate reward, the battery is drained quickly to serve the optimization purpose, leaving the future workload depending on back-up energy, which brings up the long-term reward significantly. On the contrary, using PPO or DQN, the system is willing to spend a little extra time to get energy efficiency and still ensure services without relying on back-up power sources. Overall, this trade-off is rather small (10%) compared with the entire delay cost.

The components that make up the total cost of the PPO and Q-learning algorithms in the 1,000 time slots are shown in Figure 4. The proposed PPO algorithm helps to significantly reduce standby electricity costs by avoiding standby energy. while Q-learning consumes 20 times more power. DQN performs worse out of the two because this requires an expensive dicretization process which is undesirable. However, since



(a) PPO



(b) Q-learning

Fig. 4: Components of average cost of PPO and Q-learning: energy (back-up power and battery) and time (delay)

DQN is more sample efficient than PPO, its training time is way shorter (only 20% that of PPO).

Figure 5 indicates the affect of balance coefficient β in energy cost and time cost achieved by PPO algorithm in 1000 time slots. In case $\beta = 0$, time cost is minimized, however, energy is too large at value of 16 because we only care about improving the time cost. When β to 0.5, energy cost decreases 8 times while time cost increases 1.085 times comparing with case $\beta = 0$.

The above results demonstrate that the PPO algorithm can reach better trade-off between energy cost and time cost than others schemes (Q-learning, Myopic, random power, fix power 0.4 kW, fixed power 1.0 kW).

VII. Conclusion

In this work, we focus on resource provisioning problem in MEC network. A algorithm based on a reinforcement

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

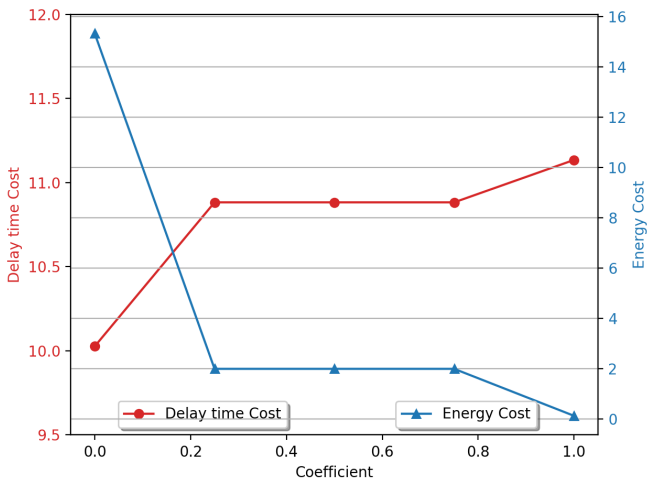


Fig. 5: Energy and time as the balance coefficient β changes

learning technique was proposed and evaluated performance. Comparing with other methods use random of fix power for edge calculations in each time slot, Our learning method gives three times better results. In the future, we will research, improve, apply more algorithms to solve the problem, especially reinforcement learning algorithms. In addition, we plan to expand problem by focusing on optimizing many other goals, such as execution time, transmission costs, computing resources, satisfy users. Constraints of resource limitation budget can be added for greater practicality.

ACKNOWLEDGMENT

This research is funded by Ministry of Education and Training of Vietnam under grant number B2020-BKA-13.

REFERENCES

[1] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal *et al.*, "Mobile-edge computing introductory technical white paper," *White paper, mobile-edge computing (MEC) industry initiative*, pp. 1089–7801, 2014.

[2] F. Computing, "Fog computing and the internet of things: Extend the cloud to where the things are," 2016.

[3] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*. ACM, 2015, pp. 37–42.

[4] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: Principles, architectures, and applications," in *Internet of Things*. Elsevier, 2016, pp. 61–75.

[5] H. T. T. Binh, T. T. Anh, D. B. Son, P. A. Duc, and B. M. Nguyen, "An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment," in *Proceedings of the Ninth International Symposium on Information and Communication Technology*, 2018, pp. 397–404.

[6] B. M. Nguyen, H. Thi Thanh Binh, B. Do Son *et al.*, "Evolutionary algorithms to optimize task scheduling problem for the iot based bag-of-tasks application in cloud–fog computing environment," *Applied Sciences*, vol. 9, no. 9, p. 1730, 2019.

[7] L. Ji and S. Guo, "Energy-efficient cooperative resource allocation in wireless powered mobile edge computing," *IEEE Internet of Things Journal*, 2018.

[8] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 443–461, 2010.

[9] S. Ulukus, A. Yener, E. Erkip, O. Simeone, M. Zorzi, P. Grover, and K. Huang, "Energy harvesting wireless communications: A review of recent advances," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 3, pp. 360–381, 2015.

[10] H. Trinh, D. Chemodanov, S. Yao, Q. Lei, B. Zhang, F. Gao, P. Calyam, and K. Palaniappan, "Energy-aware mobile edge computing for low-latency visual data processing," in *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2017, pp. 128–133.

[11] Z. Yang, J. Hou, and M. Shikh-Bahaei, "Energy efficient resource allocation for mobile-edge computation networks with noma," in *2018 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2018, pp. 1–7.

[12] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[13] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1757–1771, 2016.

[14] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.

[15] S. Mao, S. Leng, K. Yang, X. Huang, and Q. Zhao, "Fair energy-efficient scheduling in wireless powered full-duplex mobile-edge computing systems," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[16] N. Janatian, I. Stupia, and L. Vandendorpe, "Optimal offloading strategy and resource allocation in swipt-based mobile-edge computing networks," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2018, pp. 1–6.

[17] X. Hu, K.-K. Wong, and K. Yang, "Wireless powered cooperation-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2375–2388, 2018.

[18] B. Li, Z. Fei, J. Shen, X. Jiang, and X. Zhong, "Dynamic offloading for energy harvesting mobile edge computing: Architecture, case studies, and future directions," *IEEE Access*, vol. 7, pp. 79 877–79 886, 2019.

[19] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, 2018.

[20] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 3, pp. 361–373, 2017.

[21] E. Oh, K. Son, and B. Krishnamachari, "Dynamic base station switching-on/off strategies for green cellular networks," *IEEE transactions on wireless communications*, vol. 12, no. 5, pp. 2126–2136, 2013.

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

[23] "https://www.elia.be/en/grid-data/power-generation."

[24] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.