2019 IEEE International Conference on Web Services (ICWS)

Data-intensive Application Deployment at Edge: A Deep Reinforcement Learning Approach

Yishan Chen*, Shuiguang Deng*§, Hailiang Zhao*†, Qiang He‡, Yin Li* and Honghao Gao¶
*College of Computer Science and Technology, Zhejiang Univeristy, Hangzhou, China
†School of Computer Science and Technology, Wuhan Univeristy of Technology, Wuhan, China
‡School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia
¶Computing Center, Shanghai University, China
Email: {cysaloft, dengsg, cnliying}@zju.edu.cn, hliangzhao97@gmail.com, qhe@swin.edu.au, gaohonghao@shu.edu.cn

***Abstract*—Mobile Edge Computing (MEC) has already developed into a key component of the future mobile broadband network due to its low latency. In MEC, mobile devices can access data-intensive applications deployed at edge, which are facilitated by service and computing resources available on edge servers. However, it is difficult to handle such issues while data transmission, user mobility and load balancing conditions change constantly among mobile devices, edge servers and the cloud. In this paper, we propose an approach for formulating Data-intensive Application Edge Deployment Policy (DAEDP) that maximizes the latency reduction for mobile devices while minimizing the monetary cost for Application Service Providers (ASPs). The deployment problem is modelled as a Markov decision process, and a deep reinforcement learning strategy is proposed to formulate the optimal policy with maximization of the long-term discount reward. Extensive experiments are conducted to evaluate DAEDP. The results show that DAEDP outperforms four baseline approaches.**

***Keywords*-Application deployment, deep reinforcement learning, data-intensive, energy harvesting.**

I. Introduction

Mobile Edge Computing (MEC) was proposed as a new computing paradigm to enable mobile MEC, such as service deployment [1], computation offloading [2], communication management, service composition [3]–[5], security threats and edge caching, etc. [6]. Data-intensive application deployment is a fundamental issue in the MEC environment. As the amount of mobile traffic increases exponentially, it is challenging to deploy applications under the circumstances of unpredictable transmission rate and load balance because edge service deployment is always interdependent with the data. Many problems have yet to be solved, e.g., how to choose edge servers for deploying services or caching data, how to obtain an optimal deployment, how to ensure mobile users' quality experience, etc., [7], [8].

The work presented in this paper is motivated by component service and data deployment via deep reinforcement learning at the edge. The approach proposed is referred to as Data-intensive Application Edge Deployment Policy (DAEDP). Most of the previous researches on this issue adopted heuristic and approximation algorithms. Over the past two years, the reinforcement learning technique is beginning to be appreciated due to its high effectiveness and feasibility. In this paper, we consider the deployment problem as a Markov decision process (MDP) [9], [10]. In order to lift the curse of high-dimensionality in the state space, we propose a deep reinforcement learning algorithm to formulate the optimal data-intensive application deployment policy. Briefly, the main contributions of this paper can be summarized as follows.

1) An approach is proposed for formulating data-intensive application edge deployment policy that optimizes the quality of data-intensive services experienced by mobile users while minimizing the monetary cost for ASPs.

2) We consider an MEC environment, where mobile devices are equipped with energy harvesting components. We then perform a detailed analysis of mobile devices' battery energy levels.

3) We model the DAEDP as a Markov decision process and propose a deep reinforcement learning method for formulating the optimal policy. The simulation results show that DAEDP outperforms four baseline approaches.

II. Related Works

Most of the existing studies on the deployment of data-intensive services in the MEC environment can be classified into three categories: approximation algorithms, heuristic algorithms and reinforcement learning.

Approximation algorithms can be used to approximate both the NP-complete and P problem with higher complexity. Tao Ouyang et al. [11] designed an approximation algorithm based on Markov approximation to seek a near-optimal solution to mobility-aware dynamic service placement in the MEC environment. Shiqiang Wang et al. [12] proposed an approximation algorithm for online placement of multi-component applications in the MEC environment.

Heuristic algorithms are proposed to produce a sub-optimal solution within a reasonable time frame. To name a few, from the aspect of micro clouds or geo-distributed clouds, Mennan Selimi et al. [13] proposed to leverage state

§Corresponding author.

information about the network to inform service placement decisions. Bart Spinnewyna el al. [14] completed both a near-optimal distributed genetic meta-heuristic and a scalable centralized heuristic based on subgraph isomorphism detection.

In the past few years, reinforcement learning is gradually applied in the MEC environment. Shiqiang Wang et al. [15] formulated a sequential decision making problem for service migration using the framework of MDP. Xianfu Chen el al. [16] proposed a deep Q-network-based strategic computation offloading algorithm to formulate the optimal policy without having priori knowledge of the dynamic statistics. Lixing Chen et al. [17] investigated the edge service placement problem of an ASP under a limited budget, where the ASP dynamically rents computing/storage resources on edge sites to host its applications in close proximity to mobile users.

III. System Model

A. System Description

We consider an MEC system that consists of N mobile devices equipped with energy harvesting (EH) components, denoted by $\mathcal{N}$, M small-cell base stations (SBSs), denoted by $\mathcal{M}$, and one micro base station (MBS). Each SBS is bound with a small cell (SC) equipped with a shared edge computing platform. Each SBS is interconnected via X2 Link [18] for data transmission. Application service providers (ASPs) can rent SBSs from communication facility providers (CFPs) to deploy data-intensive applications. The MBS provides ubiquitous radio coverage and direct access to the cloud center. The whole time horizon is scattered into time slots, denoted by $\mathcal{T} \triangleq \{1, 2, ...\}$.

From the ASP's perspective, we focus on S data-driven applications and D access data for deployment, which are denoted by $\mathcal{S}$ and $\mathcal{D}$, respectively. Each access data size is represented by $\mu_d, d \in \mathcal{D}$. Let us denote the access data set required by the sth app as $\mathcal{D}_s \in \mathcal{D}$. In each time slot $t \in \mathcal{T}$, the ASP chooses a set of SBSs from $\mathcal{M}$ for apps and access data deployment. Let us denote the set of SBSs covering the ith mobile device in time slot t as $\mathcal{M}_i^t$, which can vary across time slots because of the user mobility.

B. Latency Reduction Evaluation

Our objective is to rent SBSs with the minimum monetary cost to maximize the latency reduction for all mobile devices with an edge latency deadline. Let us denote the input data size (in bits) of the sth app as μ_s, which requires η_s CPU cycles. We model the mobile user's demand as an i.i.d. Bernoulli distribution. Based on this model, the sth application service is requested by mobile device i with probability ρ_i^s. We set $\mathbf{A}_i^t \triangleq \{\times_{s\in\mathcal{S}} A_{i,s}^t\} \in \{0,1\}^S$ as the request vector for the ith mobile device, i.e., $\Pr\{A_{i,s}^t = 1\} = \rho_i^s$.

As mentioned before, the service request of mobile devices can only be responded to by edge sites or the cloud center, and transmission latency will be greatly reduced if the required apps are available on the chosen edge site. For each app $s \in \mathcal{S}$ and access data $d \in \mathcal{D}$, we set $I_s^t, I_d^t \in \mathcal{M}$ as the deployment decision. For simplicity, we assume that all SBSs charge the *same* price for a unit time.

1) Latency by edge computing. Similar to [17], the latency comes from the wireless uplink transmission and edge computation. Let us denote the small-scale fading channel power gains from the ith mobile device to the jth SBS by $\zeta_{i,j}^t$, which is assumed to be exponentially distributed with a given unit mean. The corresponding channel power gain can be obtained by $h_{i,j}^t \triangleq \zeta_{i,j}^t g_0 (d_0/d_{i,j}^t)^\theta$, where d_0 denotes the reference distance between i and j, $d_{i,j}^t$ denotes the real distance between i and j in the tth time slot, θ denotes the pass-loss exponent and g_0 denotes the pass-loss constant. As a result, we can obtain the achievable rate of the sth app $R_{i,j}^t(s)$ with

$$R_{i,j}^t(s) \triangleq \omega \log_2\left(1 + I^{-1}\frac{h_{i,j}^t E_i^t(s)}{b_{i,\mathrm{tx}}^t(s)}\right), j \in \mathcal{M}_i^t, \quad (1)$$

where ω represents the allocated bandwidth, I is the received average power of interference, and $E_i^t(s)$ and $b_{i,\mathrm{tx}}^t(s)$ represent the allocated energy (in energy units) and the transmission latency for the sth app, respectively. Thus

$$R_{i,j}^t(s) b_{i,\mathrm{tx}}^t(s) = \mu_s. \quad (2)$$

Besides, the latency of computing $b_{i,\mathrm{ex}}^t(s)$ is

$$b_{i,\mathrm{ex}}^t(s) = \max_{d\in\mathcal{D}_s} \frac{\mu_d}{\lambda_{j,I_d^t}} + \frac{\eta_s}{f_j}, j \in \mathcal{M}_i^t, \quad (3)$$

where f_j represents the maximum CPU cycle frequency of edge site j, λ_{j,I_d^t} is the transmission rate from SBS j to SBS I_d^t through the X2 Link. As a result, the total latency experienced by the sth app for mobile device i is

$$b_{i,e}^t(s, j^\star) = b_{i,\mathrm{tx}}^t(s) + b_{i,\mathrm{ex}}^t(s), \quad (4)$$

where $j^\star = \mathrm{argmax}_{j\in\mathcal{M}_i^t} h_{i,j}^t$.

2) Latency by cloud computing. The latency comes from the wireless transmission, the backbone Internet transmission and the cloud computation. The achievable rate $R_{i,0}^t(s)$ for the sth app from mobile device i to MBS can be obtained in the same way as (1), with every j replaced by 0. Besides, we can obtain the transmission latency $b_{i,\mathrm{tx}'}^t(s)$ in the same way as (2). Compared with edge computing, an additional transmission delay caused by the travel across the backbone Internet is included:

$$b_{i,\mathrm{Int}}^t(s) = \frac{\mu_s}{\lambda} + \tau^t, \quad (5)$$

where λ is the backbone transmission rate and τ^t is the round trip time. Besides, the computation latency at the cloud center $b_{i,\mathrm{ex}'}^t(s)$ is calculated by

$$b_{i,\mathrm{ex}'}^t(s) = \frac{\eta_s}{f_0}, \quad (6)$$

where f_0 represents the maximum CPU cycle frequency of the edge site which belongs to MBS. Thus, the total latency caused by cloud computing to the sth app on mobile device i is

$$b_{i,c}^t(s) = b_{i,\mathrm{tx}'}^t(s) + b_{i,\mathrm{Int}}^t(s) + b_{i,\mathrm{ex}'}^t(s). \tag{7}$$

Combined with $b_{i,e}^t(s, j^\star)$ and $b_{i,c}^t(s)$, the total latency reduction is:

$$\Delta_i^t = \sum_{s\in\{s'\in\mathcal{S}|A_{i,s'}^t=1\}} \left(\mathbb{1}\{I_s^t = j^\star\}\cdot\left(b_{i,c}^t(s) - b_{i,e}^t(s,j^\star)\right)\right). \tag{8}$$

C. Energy Harvest and Consumption

With EH component equipped in each mobile device i, E_i^t units of energy arrive at the mobile device at the beginning of the tth time slot. We set Q_i^t as the battery energy level of the ith mobile device at the beginning of time slot t. It evolves following the following equation:

$$Q_i^{t+1} = \min\{Q_i^t - \sum_{s\in\{s'\in\mathcal{S}|A_{i,s'}^t=1\}} E_i^t(s) + E_h^t, Q^{\max}\}, \tag{9}$$

where $Q^{\max}$ is the maximum number of energy units that can be stored and E_h^t is the number of units harvested by EH components which collect energy from the wireless environment. Note that the mobile device i drops the requests, if the battery energy is insufficient, i.e., $Q_i^t \le 0$. In this situation, $E_i^t(s) = 0$ for every s in $\{s \in \mathcal{S}|A_{i,s}^t = 1\}$. $E_i^t(s)$ will be an action variable which influences the entire network state in the following problem formulation.

IV. Problem Formulation

As mentioned before, our goal is to rent SBSs with minimum monetary cost to maximize the latency reduction for all mobile devices with an execution deadline denoted by $\hbar$. We denote $\mathbf{x}_i^t = (\mathbf{A}_i^t, Q_i^t)$ as the state of the ith mobile device in time slot t. This way, the entire network state can be described by $\mathbf{x}^t = (\mathbf{A}^t, \mathbf{\Theta}^t) \in \mathcal{X} = \{0,1\}^{N\times S} \times \{-Q^{\max}, ..., -1, 0, 1, ..., Q^{\max}\}^N$, where $\mathbf{A}^t \triangleq \{\times_{i\in\mathcal{N}} A_i^t\}$, $\mathbf{\Theta}^t \triangleq \{\times_{i\in\mathcal{N}} Q_i^t\}$. Denote $\mathbf{E}^t = \{\times_{i\in\mathcal{N}} \mathbf{E}_i^t\}$, where $\mathbf{E}_i^t \triangleq \{\times_{s\in\mathcal{S}_i} E_i^t(s)\}$ and $\mathcal{S}_i^t \triangleq \{s' \in \mathcal{S}|A_{i,s'}^t = 1\}$. With observation $\mathbf{x}^t$ at the beginning of the tth time slot, the edge computing system decides an action $\mathbf{y}^t = (\mathbf{I}_\mathcal{S}^t, \mathbf{I}_\mathcal{D}^t, \mathbf{E}^t) \in \mathcal{Y} = \mathcal{M}^S \times \mathcal{M}^D \times \{0, 1, ..., Q^{\max}\}^{S\times N}$, following a stationary control policy Φ, i.e., $\mathbf{y}^t = \Phi(\mathbf{x}^t)$, where $\mathbf{I}_\mathcal{S}^t \triangleq \{\times_{s\in\mathcal{S}} I_s^t\}$, $\mathbf{I}_\mathcal{D}^t \triangleq \{\times_{d\in\mathcal{D}} I_d^t\}$.

In every time slot t, the cost of deploying apps and access data is $\phi\cdot\sum_{j\in\mathcal{M}}(\mathbb{1}\{I_s^t = j\} + \mathbb{1}\{I_d^t = j\})$, where ϕ is the rent for every time slot. The cost of dropping requests can be calculated by $\varsigma\cdot\sum_{i\in\mathcal{N}}\mathbb{1}\{Q_i^t \le 0\}$, where ς is the weighted parameter. The cost of violating the edge latency deadline can be calculated by $\kappa\cdot\sum_{i\in\mathcal{N}}\mathbb{1}\{b_{i,e}^t \ge \hbar\}$. Besides, for those $s \notin \mathcal{S}_i^t$, penalty will be applied for the energy waste caused: $\sum_{i\in\mathcal{N}} E_i^t(s)\cdot\left(\mathbb{1}\{s \notin \mathcal{S}_i^t \wedge E_i^t(s) \ne 0\}\right)$. Therefore, the reward for taking the action $\mathbf{y}^t$ under Φ in time slot t is

$$\begin{aligned} & r(\mathbf{x}^t, \Phi) \\ = \ & \sum_{i\in\mathcal{N}}\Delta_i^t - \varrho\cdot\Bigg(\phi\cdot\sum_{j\in\mathcal{M}}\left(\mathbb{1}\{I_s^t = j\} + \mathbb{1}\{I_d^t = i\}\right) \\ + \ & \varsigma\cdot\sum_{i\in\mathcal{N}}\mathbb{1}\{Q_i^t \le 0\} + \kappa\cdot\sum_{i\in\mathcal{N}}\mathbb{1}\{b_{i,e}^t \ge \hbar\} \\ + \ & \xi\cdot\sum_{i\in\mathcal{N}} E_i^t(s)\cdot\left(\mathbb{1}\{s\notin\mathcal{S}_i^t \wedge E_i^t(s) \ne 0\}\right)\Bigg), \end{aligned} \tag{10}$$

where $\varrho, \phi, \varsigma, \kappa, \xi$ are weighted parameters. Taking the expectation over the network state $\mathbf{x}^t$ and the control action induced by a given policy Φ, the expected long-term reward on an initial network state $\mathbf{x}^1$ can be expressed by

$$V(\mathbf{x}, \Phi) = \mathbb{E}_\Phi\left[(1-\gamma)\sum_{t=1}^{\infty}\gamma^{t-1} r(\mathbf{x}^t, \Phi)|\mathbf{x}^1 = \mathbf{x}\right], \tag{11}$$

where γ is the discount factor. Therefore, the problem can be formulated as finding the optimal policy $\Phi^\star$ to obtain the maximum long-term discount reward:

$$\Phi^\star = \underset{\Phi}{\operatorname{argmax}}\, V(\mathbf{x}, \Phi), \forall\mathbf{x}\in\mathcal{X}. \tag{12}$$

V. DQN-based Application Deployment

In this section, we first include the concept of status into the model, and then propose an algorithm called DAEDP for learning the optimal solution.

MDP is a basic theoretical model for reinforcement learning. Given $\Phi(\mathbf{x}^t)$, the $\{\mathbf{x}^t : t\in\mathbb{N}_+\}$ is a controlled Markov chain with the state transition probability below:

$$\Pr\{x^{t+1}|x^t, \Phi(\mathbf{x}^t)\} = \Pr\{A_i^{t+1}\}\cdot\Pr\{Q_i^{t+1}|Q^t, \Phi(\mathbf{x}^t)\}. \tag{13}$$

The optimal state-value function $\{V(\mathbf{x}), \forall\mathbf{x}\in\mathcal{X}\}$ which satisfies the Bellman's optimality equation can be represented as:

$$V(\mathbf{x}) = \left\{(1-\gamma)r(\mathbf{x}) + \gamma\sum_{\mathbf{x}'\in\mathcal{X}}\Pr\{\mathbf{x}'|\mathbf{x}\}V(\mathbf{x}')\right\}, \tag{14}$$

where $r(\mathbf{x})$ is the reward for taking the action $\mathbf{y}$ under the state $\mathbf{x}$ and the subsequent state $\mathbf{x}'$. The size X of the state space $\mathcal{X}$ can be calculated as $X = 2^{N\times S}\times(1+2Q^{max})$, where X grows exponentially as the number $N\times S$ changes. In such an extremely huge network state space, learning the optimal solution is exponentially complex.

Given the optimal action-value function $Q, \forall\mathbf{x}\in\mathcal{X}$, we have $V(\mathbf{x}) = max\ Q(\mathbf{x})$, which can be represented as:

$$Q(\mathbf{x}) = (1-\gamma)r(\mathbf{x}) + \gamma\sum_{\mathbf{x}'\in\mathcal{X}} Pr\{\mathbf{x}'|\mathbf{x}\}V(\mathbf{x}'). \tag{15}$$

During the process of Q-learning, when updating every step, we take only one step further, estimate the target of a value function and make an improvement based on the current value function. Given the observations of the network state $\mathbf{x}^t$, the action $\mathbf{y}^t$, the reward $r(\mathbf{x}^t, \Phi)$, the request vector for the ith mobile device $\mathbf{A}_i^{t+1}$, the number of energy units harvested by the EH component, the next time slot t+1, the next network state $\mathbf{x}^{t+1}$ can be obtained using incremental summation:

$$Q(\mathbf{x})+ = \alpha^t \left((1-\gamma) r(\mathbf{x}) + \gamma \sum_{\mathbf{x}' \in \mathcal{X}} Pr\{\mathbf{x}'|\mathbf{x}\} V(\mathbf{x}') \right), \tag{16}$$

where $\alpha^t \in [0, 1)$ is a time-varying leaning rate. In fact, the true value function of the strategy is unknown. Due to its complex state and action space, we replace the real value function with the estimated value function using Neural Network. Algorithm (1) presents the pseudo code that summarizes DAEDP. The reward r_i^t can be obtained from equation (10). This algorithm does not need the knowledge about the entire system state. Our reinforcement learning method is model-independent and is capable of formulating the optimal control policy without any statistical information about the dynamic network.

Algorithm 1 DAEDP ALGORITHM

```
Require:
    Environment $E$;
    Initial state $\mathbf{x}_i^0$;
    Reward discount $\gamma$;
    Learning rate $\alpha$;
    Replay memory $\eth$ with a size of P;
    Action-value function $Q$ with random weights $\theta$;
    Target action-value function $\widetilde{Q}$ with weights $\theta^-$=$\theta$.
Ensure:
    The deployment strategy $\Phi$.
 1: $\theta$ = 0;
 2: for $i = 1$ to $n$ do
 3:     $\mathbf{x}_i^t = \mathbf{x}_i^0, \mathbf{y}_i^t = \Phi(\mathbf{x}_i) = \arg\max_{\mathbf{y}} \theta(\mathbf{x}_i^t, \mathbf{y}; \theta)$;
 4:     for $t$=1,2,... do
 5:         $r_i^t$,$\mathbf{x}_i^{t+1}$ is the reward and transfer status generated
            by action $\Phi^{\varepsilon}(\mathbf{x}_i^t)$ in $E$, respectively;
 6:         $\mathbf{y}_i^{t+1} = \Phi(\mathbf{x}_i^t)$;
 7:         Store transition $(\mathbf{x}_i^t, \mathbf{y}_i^t, r_i^t, \mathbf{x}_i^{t+1})$ in $\eth$;
 8:         Sample a random mini-batch of transitions repre-
            sented as $(\mathbf{x}_z^t, \mathbf{y}_z^t, r_z^t, \mathbf{x}_z^{t+1})$ from $\eth$;
 9:         $\Phi(\mathbf{x}_z^t) = \arg\max_{\mathbf{y}} \theta^-(\mathbf{x}_z^t, \mathbf{y}, \theta^-)$;
10:         $\theta^- = \theta + \alpha \left( r_z^t + \gamma \theta^-(\mathbf{x}_z^{t+1}, \mathbf{y}_z^{t+1}; \theta^-) - \theta(\mathbf{x}_z^t, \mathbf{y}_z^t; \theta) \right)$
            $(\mathbf{x}_z^t, \mathbf{y}_z^t; \theta)$;
11:         $\mathbf{x}_i^t = \mathbf{x}_i^{t+1}, \mathbf{y}_i^t = \mathbf{y}_i^{t+1}$
12:     end for
13: end for
```

VI. SIMULATION RESULTS

A. Experimental Setup

Table (I) presents the simulated parameter settings for the experiments. To simplify the calculation, the selected value

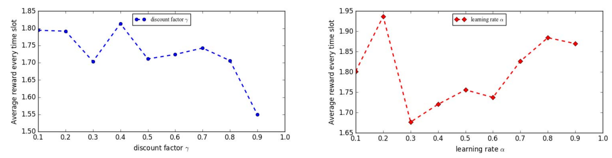

Figure 1. Average reward every time slot versus discount factor and learning rate

intervals are obtained by increasing or reducing the real value intervals as a whole. This operation will not impact the accuracy of the final results.

Table I
VARIABLE ASSIGNMENT INTERVAL

Variable	*Assignment Interval/unit*
Service memory size	[500,1000]
Service instructions	[500,1000]
Input data/Output data size	[0,1]
Access data size	[0,300]
SC storage	[2000,3000]
SC computing ability	[1000,2000]
Battery energy level of the mobile devices	[0,1000]
Units harvested by EH component	[0,50]
Energy allocated for apps	[0,500]

Suppose that there are $M = 8$ SCs, $N = 7$ mobile devices at the edge with the cloud and MBS. The energy units harvested by EH components were generated randomly according to its corresponding assignment interval. We set $\omega = 10^6 Hz$, $I = 10^{-3} W$, $Q^{\max} = 1500$ units, $f_0 = 3.9$ GHz, $\hbar$ = 2. For comparison, we implemented four baseline approaches, including the Simulated Annealing Algorithm (SAA) [19], the Ant Colony Algorithm (ACO) [20], the Optimized Ant Colony Algorithm (ACO_v) and the Hill Climbing Algorithm [21].

B. Experimental Results

This section presents and discusses the experimental results to evaluate DAEDP against the baseline approaches.

1) Effectiveness: We evaluated the performance of DAEDP with different reward discount γ and learning rate α. The results are presented in Fig. 1. From the perspective of reward discount γ, we set the learning rate α = 0.2. The average reward in every time slot does not indicate any specific patterns. When γ = 0.4, a relatively high reward value can be obtained. From the perspective of reward discount α, we set the learning rate γ = 0.4, we can see that when α = 0.2, the average reward per time slot value increases. Thus, in the following experiments, we set the reward discount γ = 0.4 and the learning rate α = 0.2.

2) Comparisons: Besides DAEDP, we also ran the four baseline approaches to obtain their average total reward performance and total latency reduction per time slot versus the units harvesting parameter $\lambda_{(h)}$ in Fig. 2. The results show that DAEDP always outperforms the four baseline approaches significantly with the maximum reward value.

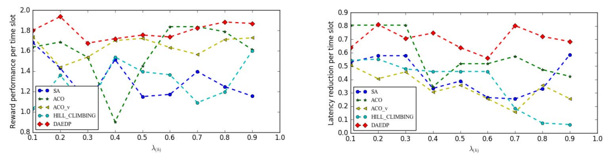

Figure 2. Performance of five approaches versus $\lambda_{(h)}$

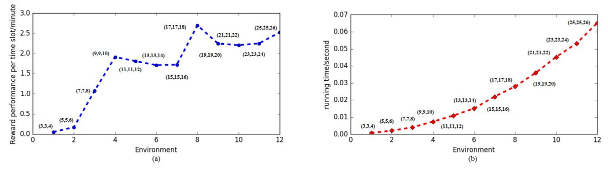

Figure 3. The effectiveness of DAEDP under different scales of systems

3) Scalability: We set the iteration times as 4,000, $\gamma = 0.4$ and $\alpha = 0.2$, then evaluated the effectiveness of DAEDP with data-intensive applications and SCs at different scales. We have tested 12 different edge environments, (3,3,4) means there are 3 services, 3 access data need to be deployed among 4 SCs. Fig. 3 (a) shows that as the system scales up in its size, the reward of large scale is always higher than the small scales in general. Fig. 3 (b) shows the running time of DAEDP. As the system scales up, the running time of DAEDP increases under the above given 12 edge environments.

VII. Conclusion

In this paper, we attempt to solve the problem of data-intensive application edge deployment with deep reinforcement learning approach. The results of extensive experimental results show that DAEDP can formulate the deployment policies with reward performance extremely close to the optimal ones. Moreover, the performance of DAEDP can always stabilize even when the size of network space grows exponentially. In the future, we will try to solve the data-intensive application edge deployment problem further sophisticated with the above issues. The problem will be more realistic and DAEDP will more generally practical.

Acknowledgment

This research was partially supported by the National Key Research and Development Program of China (No. 2017YFB1400601), Key Research and Development Project of Zhejiang Province (No. 2017C01015), National Science Foundation of China (No. 61772461), Natural Science Foundation of Zhejiang Province (No. LR18F020003 and No. LY17F020014).

References

[1] Y. Chen, S. Deng, H. Ma, and J. Yin, "Deploying data-intensive applications with multiple services components on edge," *Mobile Networks and Applications*, Apr 2019. [Online]. Available: https://doi.org/10.1007/s11036-019-01245-3

[2] C. Zhang, H. Zhao, and S. Deng, "A density-based offloading strategy for iot devices in edge computing systems," *IEEE Access*, vol. 6, pp. 73 520–73 530, 2018.

[3] S. Deng, H. Wu, W. Tan, Z. Xiang, and Z. Wu, "Mobile service selection for composition: An energy consumption perspective," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 3, pp. 1478–1490, July 2017.

[4] S. Deng, L. Huang, J. Taheri, J. Yin, M. Zhou, and A. Y. Zomaya, "Mobility-aware service composition in mobile communities," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 3, pp. 555–568, March 2017.

[5] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven iot service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54 258–54 269, 2018.

[6] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Feb 2018.

[7] S. Deng, L. Huang, Y. Li, H. Zhou, Z. Wu, X. Cao, M. Y. Kataev, and L. Li, "Toward risk reduction for mobile service composition," *IEEE Transactions on Cybernetics*, vol. 46, no. 8, pp. 1807–1816, Aug 2016.

[8] S. Deng, L. Huang, H. Wu, W. Tan, J. Taheri, A. Y. Zomaya, and Z. Wu, "Toward mobile service computing: Opportunities and challenges," *IEEE Cloud Computing*, vol. 3, no. 4, pp. 32–41, July 2016.

[9] C. C. White, "A survey of solution techniques for the partially observed markov decision process," *Annals of Operations Research*, vol. 32, no. 1, pp. 215–230, 1991.

[10] D. V. Le and C. Tham, "A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2018, pp. 760–765.

[11] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, Oct 2018.

[12] S. Wang, M. Zafer, and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.

[13] M. Selimi, L. Cerdà-Alabern, F. Freitag, L. Veiga, A. Sathiaseelan, and J. Crowcroft, "A lightweight service placement approach for community network micro-clouds," *Journal of Grid Computing*, pp. 1–21, 2018.

[14] B. Spinnewyn, R. Mennes, J. F. Botero, and S. Latre, "Resilient application placement for geo-distributed cloud networks," *Journal of Network and Computer Applications*, vol. 85, pp. 14–31, 2017.

[15] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *2015 IFIP Networking Conference (IFIP Networking)*, May 2015, pp. 1–9.

[16] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," *arXiv preprint arXiv:1804.00514*, 2018.

[17] L. Chen, J. Xu, S. Ren, and P. Zhou, "Spatio-temporal edge service placement: A bandit learning approach," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2018.

[18] T. Guo, A. ul Quddus, N. Wang, and R. Tafazolli, "Local mobility management for networked femtocells based on x2 traffic forwarding," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 326–340, Jan 2013.

[19] X. Zhihong, S. Bo, and G. Yanyan, "Using simulated annealing and ant colony hybrid algorithm to solve traveling salesman problem," in *2009 Second International Conference on Intelligent Networks and Intelligent Systems*, Nov 2009, pp. 507–510.

[20] Y. Dai, Y. Lou, and X. Lu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-qos constraints in cloud computing," in *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2, Aug 2015, pp. 428–431.

[21] P. ZhOU, Y. Tang, Q. Huang, and C. Ma, "An improved hill climbing search algorithm for rosa coupling," in *2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)*, May 2018, pp. 1513–1517.