Reinforcement Learning for value-based Placement of Fog Services

Filippo Poltronieri*, Mauro Tortonesi*, Cesare Stefanelli*, Niranjan Suri†‡
* Distributed Systems Research Group, University of Ferrara, Ferrara, Italy
{filippo.poltronieri,mauro.tortonesi,cesare.stefanelli}@unife.it
† Florida Institute for Human and Machine Cognition (IHMC), Pensacola, FL, USA
nsuri@ihmc.us
‡ US Army Research Laboratory (ARL), Adelphi, MD, USA
niranjan.suri.civ@mail.mil

***Abstract*—Optimal service and resource management in Fog Computing is an active research area in academia. In fact, to fulfill the promise to enable a new generation of immersive, adaptive, and context-aware services, Fog Computing requires novel solutions capable of better exploiting the available computational and network resources at the edge. Resource management in Fog Computing could particularly benefit from self-* approaches capable of learning the best resource allocation strategies to adapt to the ever changing conditions. In this context, Reinforcement Learning (RL), a technique that allows to train software agents to learn which actions maximize a reward, represents a compelling solution to investigate. In this paper, we explore RL as an optimization method for the value-based management of Fog services over a pool of Fog nodes. More specifically, we propose FogReinForce, a solution based on Deep Q-Network (DQN) algorithm that learns to select the allocation for service components that maximizes the value-based utility provided by those services.**

***Index Terms*—Fog Computing, Service Management, Reinforcement Learning.**

I. Introduction

Fog Computing applications require smart, adaptive, and robust resource management solutions, capable of dealing with the highly dynamic nature of the environment and the challenging demands of a new generation of immersive, context-aware, and latency-sensitive services [1]. Service management in Fog and Edge Computing has thus received an increasing attention in scientific literature, to solve several problems such as the optimal placement of Virtual Network Functions (VNFs) and services [2], [3].

Resource management solutions for Fog applications require frequent, and possibly continuous, re-configurations of network and services in order to guarantee high levels of Quality-of-Experience (QoE) and Quality-of-Service (QoS) [4]. In turn, this requires the adoption of relatively complicated continuous optimization solutions, whose proper setup often requires the tuning of several parameters, that further increase the problem complexity.

A different and potentially very promising approach lies in the adoption of self-* approaches, leveraging solutions that are capable of autonomously learning the best strategies to adapt to the current conditions. To this end, a recent trend is the adoption of Reinforcement Learning (RL): an evolving area of machine learning originally inspired by the psychology of animal learning [5]. RL consists of a goal-oriented training of an agent to learn the optimal actions (a policy) to interact with a specific environment [6]. The applicability of RL tools has been investigated in several fields, such as the optimal placement of Network Function Virtualization (NFV) [7], [8], energy-efficient resource allocation [9], and latency minimization [10].

RL can be a valuable building block for Fog Computing management solutions. Differently from supervised learning methods, that require human intervention for labeling data, RL allows to naturally train a software agent to learn an optimal policy by interacting directly with the environment. This provides a valuable tool to tame the dynamicity of Fog Computing environments, which require continuous interventions to manage the available resources and meet the current applications' requirements [3], [11]. Therefore, Fog Computing management solutions can leverage RL for automatically tuning the configuration parameters when the environment conditions change to guarantee the delivery of the expected QoS and QoE.

Motivated by the promising capabilities of such techniques, this work investigates the application of RL as a candidate for the online/continuous optimization of a Fog management framework. Towards that goal, we focus on Value-of-Information (VoI) optimization as resource management criterion for Fog Computing applications. VoI methodologies and tools aim to find an optimal configuration for the available computational and network resources that maximizes the end-users utility of Fog services, which prioritizes the most important data to be processed and disseminated, thus effectively addressing the data deluge of IoT applications. As a promising methodology for addressing information management and prioritization in constrained environments, VoI has been recently proposed for resource management in several works [12], [13], [14].

Built upon our previous work [14], in which we formalized an optimization framework for the value-based management of Fog services, this paper investigates the applicability of RL and Deep RL (DRL) techniques as an optimization tool

978-3-903176-32-4

for learning the optimal VoI allocation of Fog services [15]. In this paper, we first introduce RL and we report the VoI optimization model for Fog Service management defined in [14]. Then, we define a Markov Decision Process (MDP) to model the system and the reward signal. Finally, we present FogReinForce, a learning algorithm leveraging a Deep Q-Network (DQN) that given a service component allocation is capable of migrating service components to achieve increased value-based utility for the end-users of Fog services.

The remainder of the paper is organized as follows: Section II discusses RL and gives an overview of its application in related fields. Section III illustrates the value-based optimal placement of Fog services described in [14]. Then, Section IV formalizes a Markov Decision Process (MDP) to solve the given problem problem using the DRL algorithm described in Section V-A. Section V-B discusses the integration of FogReinForce into a management framework. Finally, Section VI presents an evaluation of the proposed algorithm and Section VII concludes this manuscript.

II. Related Work

RL is an evolving field of machine learning consisting in a goal-oriented training in which an agent interacts with an environment to learn the best possible actions to reach a specific goal [5], [6]. At each action is assigned an immediate reward and the goal of the agent is to learn a policy that would maximize the sum of those rewards. In order to learn a good rewarding policy, the agent needs to solve the same task multiple times in which it will be capable of exploring a consistent number of states.

RL has been applied to different sort of problems from learning how to play classic games to network and service management. Dab et al. formalize an RL problem to learn the best offloading decisions in order to minimize energy consumption on the devices-side under latency constraints for 5G applications in [16]. The proposed strategy is evaluated with extensive simulations in NS3 and proved to be successful in reducing computation time. Long-Term latency minimization for Fog Computing is also discussed in [10]. This paper proposes an RL approach combined with evolution strategies for dealing with real-time task assignment and reducing computation latency in the long-term period.

In [17], Nakanoya et al. propose an interesting and cost-effective technique for applying RL to online optimization of Virtualized Network Functions (VNF) sizing and placement. In particular, the authors propose a two-step RL that divides the learning process into two phases with the aim of reducing the learning exploration steps. In [3] Chen et al. discuss two Double DQN learning algorithm for task offloading decision in MEC for mobile ultra-dense in sliced RAN. The authors in [18] discuss a Q-learning based load-balancing algorithm for Fog Networks that enables to reduce the processing time and overload probability of networks.

Li et al. investigate RL for Network Slicing in [19] by illustrating general concepts and applications for resource management by comparing different scheduling algorithms. An interesting formulation leveraging DQN is the one discussed in [20]. In this work, the authors propose an algorithm called JTOBA that iteratively solves the problem of best joint task offloading and bandwidth allocation in MEC. A different RL application is described in [21], in which a fast task allocation (FTA) algorithm leveraging DRL is proposed to allocate tasks among heterogeneous UAVs. The authors claim that FTA is highly adaptive even in case of different set of tasks and environment variability.

III. System Model and Notation

In general, the idea of Fog/Edge computing paradigms is to allocate information processing close to both users and devices in order to provide reduced latency and increased QoE and QoS. To this end, intelligent allocation of service and accurate management of resources are required to best exploit the few resources available at the edge.

In this work, we will model resource management according to the *Adaptive, Information-centric, and Value-based (AIV)* service and information maturity model defined in [22], [14]. The AIV notation allows us to model Fog services as a coordinated composition of re-usable and independent service components that can be combined in a workflow fashion.

A. VoI Optimal Resource Allocation Problem

Let us report the VoI Optimal resource allocation problem that we defined in [14]. First, the building blocks to model service component allocations are:

- a set of Fog Service $\mathbb{S}$;
- a set $\mathbb{SC}$ of service component sc_k;
- a set $\mathbb{U}$ of user groups u_h.

Furthermore, it is worth specify that, a Fog service s is a composition of one or more service components sc. These building blocks allow us to model a service component allocation among a pool of fog devices and to calculate the VoI value of delivered messages m.

The VoI Optimal resource allocation problem aims at maximizing the total VoI delivered to the end-users of a set of Fog services during a fixed time-window. More specifically, we have:

$$f_n = \textsc{Total}_{VoI}(t_n, t_{n+1}) = \sum_{m \in \mathbb{M}(t_n, t_{n+1})} VoI_{\Theta}(m). \quad (1)$$

To measure the total VoI delivered within a time window, we also consider a Proximity Relevance Decay (PRD) and a Time Relevance Decay (TRD) functions to take into account the decay a message is subjected to from its origination (location/time) to its delivery (location/time). In particular, both PRD and TRD functions assumes values in $[0, 1] \in \mathbb{R}$ and they act as decay multiplier. Moreover, because modeling such functions it is a per se challenging task, it is convenient to specify an average decay for each class of messages:

$$VoI_{\Theta}(m, \mathbb{I}_m, s, t) = \quad VoI(m_t, s) \quad \times \sum_{m_t \in M} [\overline{TRD(m_t)} \times \overline{PRD(m_t)} \times \sum_{u_t \in U} \overline{U(u_t, m_t)}] \quad (2)$$

where $\mathbb{I}_m$ is the set of input messages used for the generation of message m, s a Fog service, and m_t represents a class of messages.

In a previous work [14], we modeled eq. (2) within the Phileas simulator [23] and we used Meta-heuristics for its optimization. Instead, in this paper we present a model to enable the optimization of (2) using DRL techniques.

IV. A Markov Decision Process for VoI allocation

To exploit RL techniques in our Fog Service Management Framework, we need to define a Markov Decision Process (MDP) for the system model [24]. An MDP is a general framework (particularly suited for RL) for defining decision making problems [5]. Let us note, that using proper modeling of states and actions is essential for RL tasks. In fact, RL in general requires a good knowledge of the whole problem and a proper reward definition in order to allow the training process to converge.

The MDP for VoI allocation defines a set $\mathcal{S}$ of states s, a set $\mathcal{A}$ of actions that allow an agent to move from a state s to another state s', and a Reward policy R the defines the reward given by an action that moves the environment into a different state. In particular, $R_a(s, s')$ is the immediate reward for performing action $a \in \mathcal{A}$ under state $s \in \mathcal{S}$. Finally, the goal of the MDP is to find the optimal policy $a = \pi(s)$ that gives the best action $a \in \mathcal{A}$ under state $\in \mathcal{S}$ that allows to maximize the Q-function $Q(s, a)$ for each state action pair.

With regards to a state $s \in \mathcal{S}$, we define it as an array-like service component allocation $s = \{sc_1, sc_2, ..., sc_n\}$ in which the value of the i-th element represents the fog device d_j where the service component sc_i is allocated. For instance, {3, 2, 4, ..., k, 3 } is an example of a state $s \in \mathcal{S}$, where the service component sc_1 (the first element of the state array) is allocated on fog device d_3 and k is $|\mathbb{D}|$, the number of fog devices.

On the other hand, we define an action $a \in \mathcal{A}$ as the selection of a fog device $d \in \mathbb{D}$ where to allocate a service component sc_i on. For example, an action $a = 2$ indicates to allocate the service component on fog device d_2. Furthermore, let us note that, because the state and action spaces are finite, the formalized MDP is finite MDP.

More specifically, the VoI allocation MDP consists of a sequence of n discrete time steps $t = 0, 1, 2, ..., i, ..., n$ in which an agent analyzes each service component sc_i (where the i index corresponds to the i time step) and decides whatever or not allocates it on a different fog device d_j. When an action $a \in \mathcal{A}$ under state $s \in \mathcal{S}$ is performed, a new state s' is reached, and the agent gets an immediate reward $R_a(s, s')$. Finally, for modeling rewards, we decide to adopt the simplistic but effective strategy of assigning 1 to those actions that define an allocation capable of improving the value of (2) with respect to previous state, and 0 to the others.

V. The FogReinForce DRL Solution

Having defined the MDP formulation for the VoI allocation of service components allows us to adopt a suitable RL algorithm for optimizing the problem. Given the dimension of the state space, which grows exponentially with the number of service components to allocate, we investigate DRL solutions, and we propose a Deep Q-Network (DQN) based learning algorithm FogReinForce. Finally, we discuss the adoption of FogReinForce within a management framework for enabling the continuous optimization of Fog Computing resource management.

A. FogReinForce Learning Algorithm

To solve the VoI allocation MDP problem, we propose the FogReinForce learning algorithm, which exploits DQN for implementing the learning process. FogReinForce aims at finding the VoI optimal allocation for service components by learning the optimal policy $a = \pi(s)$ that selects the best actions to take under a particular state $s \in \mathcal{S}$ to obtain a performing service components allocation in a finite number of steps N.

Alg. 1 gives a simplified illustration of the FogReinForce algorithm, which implements a DQN algorithm with experience replay. Alg. 1 takes as input the values for ϵ, γ, and the update_step used during the training of the neural network. The replay memory RM is used to store the transitions and for sampling a mini-batch during the training with experience replay [25].

As depicted in Alg. 1, we define the allocation problem as an episodic task with a maximum number of episodes $max\_episode$. Let us specify that, FogReinForce defines an episode as a finite number of steps N, after which the episode is considered finished. With regard to the number of steps, we decided to use $n = 2 \times k$, where $k = |SC|$ is the number of service components to allocate. This would allow the agent to perform two actions on the same service component.

At the beginning of a new episode, the initial state s_1 is reset to the random state s_r generated in the initialization phase of the algorithm. This is a common practice when defining a DRL training process, however, different policies can be chosen. During each step, the agent interacts with the environment by selecting at step t a fog device for the service component sc_t. The new state is evaluated using (2) to calculate the total VoI that new state s_{t+1} generates. During these steps, the goal of the agent is to maximize the cumulative reward over the episode given a finite amount of actions.

B. Continuous Optimization for Fog Computing

To illustrate readers how we intend to realize continuous optimization of Fog Computing resource management we present a proof-of-concept framework that incorporates FogReinForce into a learning component that continuously checks for improvements. To this end, Fig. 1 depicts the

Algorithm 1: FogReinForce pseudo-algorithm

Data: ϵ, γ,update_step
Result: π a policy for mapping states into actions
$Q_0(s,a) = 0$ initialize action-value function to 0;
initialize target action-value Q-network for Q estimation;
initialize replay memory RM;
initialize a random allocation s_r to be used as initialization state;
for i *in* $1, max\_episodes$ **do**
  initialize state $s_1 = s_r$;
  initialize $er = 0$ episode reward ;
  for t *in* $1, N$ **do**
    if $rand() \geq \epsilon$ **then**
      select action $a_t = argmax\{Q(s_t, a)\}$;
    else
      select a random action a_t;
    take action a_t and generate the next state s_{t+1};
    evaluate s_{t+1} using (2) and calculate the reward r_t;
    set s_t as the next state s_{t+1};
    store transition in replay memory RM;
    update er;
    if t *is an update step and there enough transitions in memory* **then**
      sample a random subset of transitions from RM and learn;
    else
      calculate discounted reward using γ;
  decrease ϵ and make it decay;

envisioned architecture for allowing continuous optimization of Fog services running in a Fog Computing site. More specifically, the Fog management framework includes three main components: a learning component, a Fog controller, and a Fog monitor.

Firstly, the Fog monitor is responsible for collecting information regarding a Fog Computing site including users' interests, available fog devices, running services, and network conditions. In addition, the Fog monitor updates the collected information into the Learning Component to create an input configuration for the Phileas simulator.

The Learning component is to find the optimal VoI allocation of service components on the resources available at the Fog site. To this end, the Phileas simulator is configured with the data collected by the Fog monitor to be a realistic representation of the environment at the Fog Computing site. In order to achieve the optimal allocation of service components on Fog devices, the Learning Component runs FogReinForce for a finite number of episodes. In this way, FogReinForce interacts with the Phileas simulator to find the service components configuration that maximizes the feedback representing the total VoI delivered during the simulation.

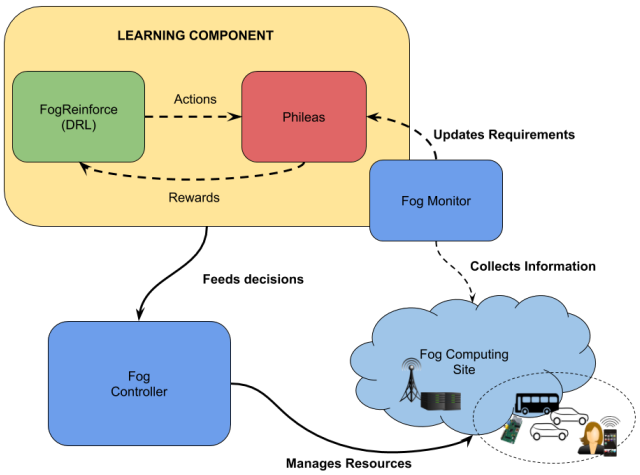

Fig. 1. An overview of the management framework for enabling the continuous optimization of Fog Computing environments.

When the Learning component finds a suitable allocation, it instructs the Fog Controller where to allocate the service components running at the Fog Site. Finally, for enabling online optimization, this loop runs continuously, thus ensuring to tackle the dynamicity of Fog Computing environments and deal with users' mobility, system's load, and other conditions.

VI. Evaluation

To verify the capabilities of FogReinForce, we use the realistic Fog Computing scenario defined for the evaluation of VoI management framework in [14]. This scenario is a representation of a Fog Computing use-case in Smart City, in which smart citizens exploit the functionality of different Fog services provided by the municipality (traffic monitoring, air quality info, etc.). More specifically, the testbed contains the description of 7 devices, 9 data sources, 4 user groups, and 8 service components. Finally, each fog service is defined as the composition of 2 service components for a total of 4 Fog services.

To reenact realistic conditions, we set the locations of devices, data sources, and user groups defined in the Smart City scenario with a latitude and longitude position in accordance with the GPS position system. This allows setting fixed communication ranges for users and devices reenacting wireless communications. We believe this to be essential in our experiments, because it drives FogReinForce to allocate service components according to a minimum distance policy, e.g. privileging those allocations that process data messages close to where they will be consumed.

The aim of these experiments is to verify the capabilities of FogReinForce in optimizing the total VoI delivered to the end-users of Fog services during a fixed time-window. More specifically, FogReinForce will learn how to configure service components on the available fog devices using the total VoI delivered as feedback.

TABLE I
CONFIGURATION PARAMETER FOR THE Q-NETWORK.

number of layers	2
γ	0.95
lr	0.0005
$start$	1
ϵ_{end}	0.01
mini batch size	64

A. Reinforcement Learning Configuration

FogReinForce leverages on a Deep Q-Learning Network (DQN) implemented in Python, which exploits the pytorch library https://pytorch.org for implementing the training of the Deep Q-network (neural-network) responsible for mapping states in action values. We chose Python as programming language because it is a valuable tool for experiencing with machine learning and data analysis. In addition, the pytorch library provides a user-friedly API for implementing the state-of-the-art machine learning models and optimization algorithms.

The Q-Network is implemented as a two-layers neural network with 64 nodes for each hidden layer, a $\gamma = 0.95$ the discount rate to determine the present value of future rewards, a learning rate of $lr = 0.0005$, and ϵ value starting from 1 and annealing to 0.01. Finally, for experience replay we set the mini-batch size to 64. To summarize the configuration, Table I shows the value of each configuration parameter.

During the training phase, FogReinForce interacts with the Phileas simulator using a HTTP REST interface, thus allowing the two different software modules to communicate. More specifically, given a state s representing a service components allocation, FogReinForce makes a request (via HTTP) to Phileas to calculate the value of (2) for the given configuration. Even if performance-wise is not great, this is a common practice to integrate software components written in different programming languages. Future versions of this optimization framework will provide a better integration of FogReinForce's features within the VoI management framework to reduce the required training time and speed-up the entire process.

Given the time-complexity of calculating the value of (2), which evaluation requires a simulation run, we choose to configure training phase with a number of episodes of 1000 to give the agent a fair amount of iterations to learn an optimal policy, i.e. the optimal value for $Q(s, a)$. In addition, each episode is defined as a sequence of 16 steps after which the episode is considered done. As described in the previous Section, if the action of allocating a service component to another fog device (different configuration) brings to a more performing allocation (in terms of total generated VoI) the agent gets a reward of 1, otherwise it gets 0.

As first state we select a random allocation of service components on fog devices. Other options such as a greedy to calculate a feasible starting state are possible. However, opting for a random state is a good assumption to verify if given a fair amount of training iterations, the agent can learn a sequence of actions leading to a high-value VoI state. In fact, the agent should be capable of learning from whatever starting point given a finite number of steps and episodes [5].

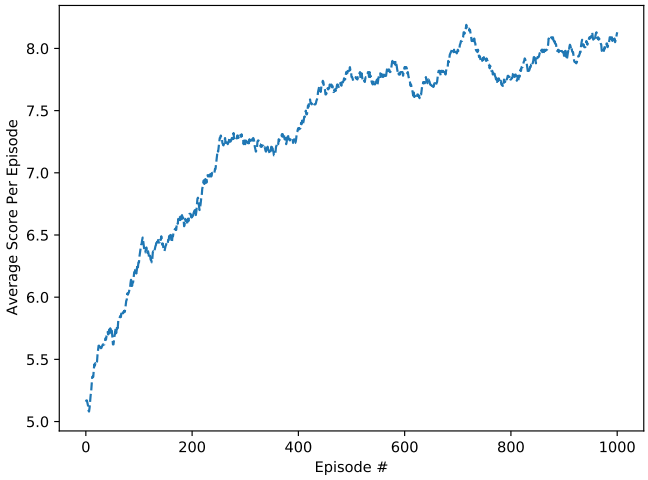

Fig. 2. The training of the DRL agent over the VoI model for 1000 episodes.

B. Results

We configure FogReinForce using the parameters discussed in Section VI-A to learn the optimal policy $a = \pi(s)$ that would maximize the total VoI (2) delivered within a specific time window. To this purpose, we configure the Phileas simulator to reenact the fictional Smart City scenario for a limited time window of six minutes. This is to reduce the Q-network training time and to allow the evaluation of different service component configurations. As illustrated in Alg. 1, at the beginning of each episode, FogReinForce initializes the initial state using a randomly generated state. Then, at each step, it learns the actions that maximize the cumulative reward of the episode. During the evaluation, for each episode, we collect the score, i.e. the sum of rewards, and the state $s_{optimal}$ which leads to the best value of (2).

Fig. 2 reports the training phase of FogReinForce to optimize the VoI model defined in (2). More specifically, Fig. 2 depicts the time-series of the average scores that FogReinForce achieved during the 1000 episodes of the learning process. The reported trend is increasing, thus indicating that the algorithm is capable of learning a good rewarding policy. Given a finite number of 16 steps, FogReinForce can improve the current state up to 12 times for an episode. Moreover, it achieves a cumulative reward value of 8 after only 500 episodes. We believe this to be an encouraging result, which demonstrates the viability of DRL methodologies for our particular VoI management framework.

On the other hand, Fig. 3 illustrates the best VoI values for (2) achieved during the training of FogReinForce. It is worth to note, how the best value is achieved around the 600-th episode, thus indicating that FogReinForce is capable of improving the value of (2) in a relatively limited number of iterations. This also shows that FogReinForce can find a good

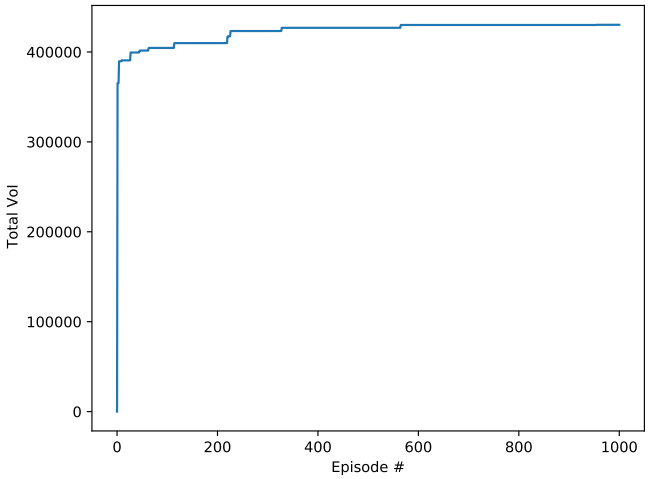

Fig. 3. The best values for the VoI model during the training phase.

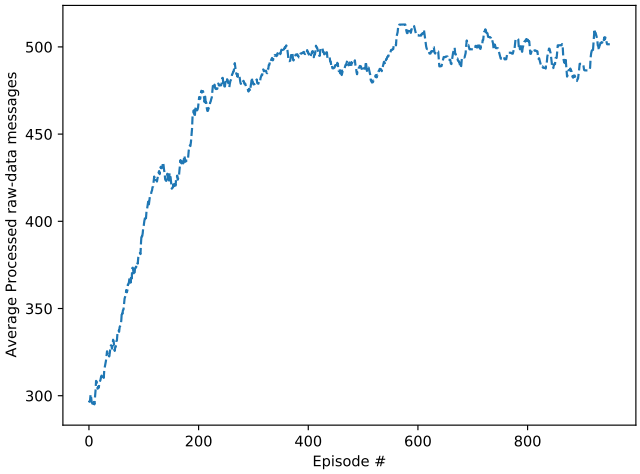

Fig. 4. The number of processed raw-data messages during the training phase.

rewarding policy to improve the allocation described by the random starting state s_r.

Another proof that FogReinForce is capable of improving the management of the processing resource at the edge comes from Fig. 4. More specifically, Fig. 4 depicts the number of raw-data messages that the fog devices can process during the time-window defined by the simulation time. It is worth noting how this number increases during the episodes and that this increasing trend is also related with service component allocations leading to higher VoI values. Finally, this demonstrated that VoI methodologies and tools are beneficial for addressing resource management.

Then, to compare the performance of FogReinForce with our previous work [14], Fig. 5 shows the best VoI values for FogReinForce and the preliminary distance heuristic presented in our previous work. More specifically, the distance heuristic aims to allocate services components on devices in the close proximity of both raw-data sources generating the data and

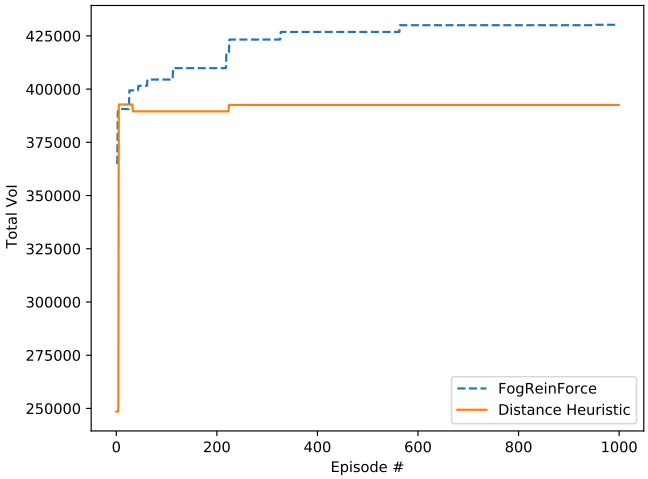

Fig. 5. The comparison between FogReinForce and the distance heuristic presented in our previous work.

users with the goal of reducing the overall distance generated by a service component allocation. The distance heuristic illustrated in Fig. 5 is optimized with the Quantum Particle Swarm Optimization (QPSO) meta-heuristics. Let us specify that for the distance heuristic, an episode corresponds to an iteration of the QPSO's algorithm. The results prove that for the given configuration FogReinForce is capable of outperforming the heuristic presented in our previous work, thus demonstrating the soundness of our approach.

VII. Conclusion and Future Works

This work presented our efforts for applying RL techniques into a VoI management framework for Fog service components. Starting from the Fog Service management framework defined in [14], we investigated DRL as another optimization tool for value-based service components placement. We defined an MDP for the VoI allocation problem to enable the application of RL algorithms.

Then, we present FogReinForce a DQN algorithm that allows an agent to find optimal solutions for the VoI service components allocation problem. More specifically, FogReinForce will learn an optimal policy to allocate a set of Fog services on a pool of Fog devices in a way that would maximize the total VoI delivered to the end-users in a given time-window. To evaluate the capabilities of FogReinForce, we devised a scenario on which FogReinForce proved to be efficient in finding a good-rewarding allocation policy and high VoI service components allocation in a relatively limited number of iterations of the training algorithm. Even if RL techniques suffer from the curse of high training time, we believe that there still room for improvement and that they provide interesting and novel ideas for continuous optimization. Finally, as future works we intend to investigate different DRL algorithms and to test the capabilities of FogReinForce in more dynamic scenarios.

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*. ACM Press, 2012. [Online]. Available: https://doi.org/10.1145/2342509.2342513

[2] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, Sep. 2019. [Online]. Available: https://doi.org/10.1016/j.sysarc.2019.02.009

[3] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2019, cited By 67.

[4] I. Comșa, R. Trestian, G. Muntean, and G. Ghinea, "5mart: A 5g smart scheduling framework for optimizing qos through reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1110–1124, June 2020.

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Bradford Books, 2018.

[6] A. Gosavi, "Reinforcement learning: A tutorial survey and recent advances," *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 178–192, 2009.

[7] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A deep reinforcement learning approach for vnf forwarding graph embedding," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1318–1331, Dec 2019.

[8] H. Yao, S. Ma, J. Wang, P. Zhang, C. Jiang, and S. Guo, "A continuous-decision virtual network embedding scheme relying on reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 864–875, June 2020.

[9] A. Kaur and K. Kumar, "Energy-efficient resource allocation in cognitive radio networks under cooperative multi-agent model-free reinforcement learning schemes," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1337–1348, Sep. 2020.

[10] L. Mai, N.-N. Dao, and M. Park, "Real-time task assignment approach leveraging reinforcement learning with evolution strategies for long-term latency minimization in fog computing," *Sensors*, vol. 18, p. 2830, 08 2018.

[11] F. Wei, G. Feng, Y. Sun, Y. Wang, and Y.-C. Liang, "Dynamic network slice reconfiguration by exploiting deep reinforcement learning," vol. 2020-June, 2020, cited By 0.

[12] N. Suri, G. Benincasa, R. Lenzi, M. Tortonesi, C. Stefanelli, and L. Sadler, "Exploring value-of-information-based approaches to support effective communications in tactical networks," *IEEE Communications Magazine*, vol. 53, no. 10, pp. 39–45, October 2015.

[13] S. Bharti, K. K. Pattanaik, and P. Bellavista, "Value of information based sensor ranking for efficient sensor service allocation in service oriented wireless sensor networks," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2019.

[14] F. Poltronieri, M. Tortonesi, A. Morelli, C. Stefanelli, and N. Suri, "Value of information based optimal service fabric management for fog computing," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–9.

[15] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[16] B. Dab, N. Aitsaadi, and R. Langar, "Q-learning algorithm for joint computation offloading and resource allocation in edge cloud," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 45–52.

[17] M. Nakanoya, Y. Sato, and H. Shimonishi, "Environment-adaptive sizing and placement of nfv service chains with accelerated reinforcement learning," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, April 2019, pp. 36–44.

[18] J. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2019, pp. 1–7.

[19] R. Li, Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018, cited By 37.

[20] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digital Communications and Networks*, vol. 5, no. 1, pp. 10 – 17, 2019, artificial Intelligence for Future Wireless Communications and Networking. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864818301469

[21] X. Zhao, Q. Zong, B. Tian, B. Zhang, and M. You, "Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning," *Aerospace Science and Technology*, vol. 92, pp. 588 – 594, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963818318704

[22] M. Tortonesi, M. Govoni, A. Morelli, G. Riberto, C. Stefanelli, and N. Suri, "Taming the IoT data deluge: An innovative information-centric service model for fog computing applications," *Future Generation Computer Systems*, vol. 93, pp. 888 – 902, 2019.

[23] F. Poltronieri, C. Stefanelli, N. Suri, and M. Tortonesi, "Phileas: A simulation-based approach for the evaluation of value-based fog services," in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Sep. 2018, pp. 1–6.

[24] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.