Deep Reinforcement Learning for IoT Network
Dynamic Clustering in Edge Computing

Qingzhi Liu*, Long Cheng†, Tanir Ozcelebi∗, John Murphy†, Johan Lukkien∗
∗MCS, Eindhoven University of Technology, The Netherlands
†CS, University College Dublin, Ireland
Email: q.liu.1@tue.nl, long.cheng@ucd.ie, t.ozcelebi@tue.nl, j.murphy@ucd.ie, j.j.lukkien@tue.nl

Abstract—Processing big data generated in large Internet of Things (IoT) networks is challenging current techniques. To date, a lot of network clustering approaches have been proposed to improve the performance of data collection in IoT. However, most of them focus on partitioning networks with static topologies, and thus they are not optimal in handling the case with moving objects in the networks. Moreover, to the best of our knowledge, none of them has ever considered the performance of computing in edge servers. To solve these problems, we propose a highly efficient IoT network dynamic clustering solution in edge computing using deep reinforcement learning (DRL). Our approach can both fulfill the data communication requirements from IoT networks and load-balancing requirements from edge servers, and thus provide a great opportunity for future high performance IoT data analytics. We implement our approach using a Deep Q-learning Network (DQN) model, and our preliminary experimental results show that the DQN solution can achieve higher scores in cluster partitioning compared with the current static benchmark solution.

I. INTRODUCTION

Edge computing is an efficient solution to process large amounts of data collected from Internet of Things (IoT). Traditional collected data sets in IoT are of small size, such as sensing data. Therefore, a non-optimized data collection solution is not likely to affect the performance of the edge computing. Because the edge servers can re-allocate the data to multiple servers through wired networks, which can easily achieve very high communication speeds in the order of Gigabits per second. However, with the fast development of IoT in recent years, the property of data flow in IoT changes. Firstly, data in IoT becomes increasingly larger, such as video, images, etc. Secondly, data comes from not only homogeneous static IoT devices, but also from heterogeneous dynamic IoT devices. To process these data in edge servers in an efficient way, such as applying advanced parallel computing techniques, we need effective data collection in IoT, since a non-optimized data collection solution could result in performance bottlenecks for parallel computing. In this work, we focus on the load balancing issues in such scenarios.

To date, many solutions for data collection in IoT networks and load balancing for parallel computing have been proposed respectively. However, almost all the techniques in the two fields are totally independent from each other, and the research on optimizing the performance on both aspects is limited due to two main challenges.

• Different Requirements: The requirements on IoT networks and parallel computing are different. On the one hand, most solutions for data collection in IoT focus on decreasing the communication cost, such as the total number of hops in multi-hop communication. On the other hand, most solutions for parallel and distributed computing focus on balancing the data size collected in edge servers. For example, from the IoT viewpoint, the deployed edge servers should be close to most of the IoT devices, so that the communication latency can be minimized. However, from the computing viewpoint, balanced data size on the edge servers could greatly speed up the data processing [1].

• Dynamic Networks: The data collection solution should not only consider the topology of a network, but also the dynamic objects in the network. As a dynamic object moves inside the detection range of IoT devices, such as proximity sensing, an IoT device would produce data triggered by the object. In this condition, the data collection scheduling should be adaptive to the dynamic networks. However, most existing solutions just focus on a static data flow pattern.

In this paper, we propose a deep reinforcement learning (DRL) based solution for IoT network clustering. In each cluster, the data produced from cluster member IoT devices is collected to the edge server located in the cluster. In general, our solution has two main properties.

• We assume each IoT device produces regular data of a fixed size periodically. Our solution partitions the IoT network into clusters to maximize the communication performance of IoT networks and computing performance of edge servers.

• We assume objects move in the area of IoT network, which trigger the neighbor IoT devices to produce data that has larger size than regular data. Our solution adapts the pattern of network clustering to these changes in the collected data.

To achieve the goal, we propose a Deep Q-Learning Network (DQN) model. The preliminary experiments prove the feasibility of our solution. The experimental results show that the DQN model improves the system performance significantly in the IoT network with dynamic objects compared with the solution using statically partitioned clusters.
Fig. 1. The system model of IoT network and parallel edge servers. The data collection is based on the partitioned clusters.

The paper is organized as follows. Related work is discussed in Section II. The system model is present in Section III. The preliminary experimental results are explained in Section IV. The DQN based solution is explained in Section V. The future work is discussed in Section VI and the paper is concluded in Section VII.

II. RELATED WORK

Cluster based data collection is widely used in wireless networks [2] [3]. Typically, the network nodes are partitioned into several groups, and the member nodes in a cluster send the data to the cluster head. Network clustering is used to optimize the performance of the network, such as energy consumption, communication latency, etc. Most of the network clustering solutions are based on the global information of the network. To cope with the situation without global information, some researches focus on self-adaptive self-organization systems [4]. For example, [5] makes the network converge to the optimized network clustering by the local decision of each node using evolutionary theory. To cope with more complicated resource management problems in networks, some researches exploit reinforcement learning (RL). For example, [6] presents a solution that translates the tasks with multiple resource demands into a RL problem. Its experimental results show that the RL model has comparable performance to the state-of-the-art heuristic solutions. [7] presents a resource allocation mechanism based on DRL in vehicle-to-vehicle communications. The solution makes each agent optimize sub-band and transmission power for satisfying the latency constraints using distributed local information.

III. SYSTEM MODEL

The system model includes an IoT network and edge computing servers as shown in Figure 1. The IoT network consists of homogeneous devices. We assume that the IoT devices are randomly deployed in the area, and each IoT device has multi-hop communication routes to the edge servers. Each IoT device produces data periodically, and transfers messages to the edge servers by multi-hop communication. Several edge servers are deployed to make parallel computing for the collected data from the IoT network. For collecting data from IoT devices to multiple edge servers, the network is partitioned into clusters, and each edge server resides in a cluster. Each edge server is responsible for collecting the data from the IoT devices of a cluster.

We suppose each IoT device sends regular report data of a fixed size periodically to the edge server of its cluster. An object moves in the area of IoT network, and triggers the neighbor IoT devices to produce data larger size. For example, the object could be a person moving around. The sensor attached to each IoT device detects the proximity of the person. Once a person appears in the detection range of an IoT device, the IoT device captures the image of the person and sends image data to the edge server of its cluster.

The network clustering depends on the position of the moving object in the IoT network and the data collection requirements. Figure 2 presents an example. To guarantee the collected data size in both edge servers are balanced, the network clustering pattern must change to adapt to the movement of object in the area. In this paper, we utilize a DRL approach to find the optimized network clustering in the IoT network with dynamic objects.

IV. DEEP REINFORCEMENT LEARNING SOLUTION

We use Deep Q-Learning Network (DQN) as the DRL model for dynamic network clustering. We assume that an agent performing the DQN model resides in an edge server, and all the required information for calculating the DQN model is collected to the agent.

A. Actions

Suppose a set of edge servers \( P = \{p_1, \ldots, p_n\} \) reside in the network with integer \( i \in [1, n] \). A cluster-core \( \alpha_i \) is set at the position of edge server \( p_i \). To partition the IoT network into clusters in the initialization, we first select the IoT network node that is closest to the cluster-core \( \alpha_i \) as the cluster head \( h_i \). After that, we partition the network into clusters \( C = \{c_1, \ldots, c_n\} \) with \( i \in [1, n] \) based on the cluster heads. To simplify the implementation, we partition
the network to Voronoi clusters. After the initialization, each edge server resides in a cluster.

To control the partition of clusters, we move the cluster-cores of clusters. The action state space \( A \) is the movements of all the cluster-cores. In our implementation, each cluster-core has five movement actions: Up, Down, Left, Right, Stay. At each time step, the DQN model selects and performs a movement action on a cluster-core. After the movement, the network re-selects the IoT network node that is closest to the cluster-core \( o_i \) as the cluster head \( h_i \), and re-partitions the network into clusters based on the new cluster head. An example of moving cluster-cores and partitioning clusters is shown in Figure 3.

B. Reward

The reward of an action is the satisfactory to the requirements on both IoT network and edge computing aspects. We define the two requirements that the network clustering should fulfill as follows.

- **Edge Servers**: Name the size of data collected to each edge server \( p_i \) as \( d_i \), and \( D = \{d_1, ..., d_i, ..., d_n\} \) with \( i \in [1, n] \). To maximize the parallel computing speed by edge servers, the collected data size should be balanced in the edge servers. We use the average absolute deviation value of \( D \) that is normalized in \([0, 1]\) to quantify this load balancing requirement. We have \( W = \frac{1}{n} \sum_{i=1}^{n} \left[ 1 - \frac{|d_i - \text{mean}(D)|}{\text{mean}(D)} \right] \).

- **IoT Network**: Name the total number of communication hops to transfer all the data in cluster \( c_o \) to edge server \( p_i \) as \( b_i \). \( B = \{b_1, ..., b_i, ..., b_n\} \) with \( i \in [1, n] \). As we assume the IoT network uses multi-hop communication, more communication hops means higher communication time for collecting the same amount of data. To keep the speed for collecting data of each cluster on the same level, the number of communication hops for transferring all the data in each cluster should be balanced. We use the average absolute deviation value of \( B \) that is normalized in \([0, 1]\) to quantify the requirement of IoT network. We have \( H = \frac{1}{n} \sum_{i=1}^{n} \left[ 1 - \frac{|b_i - \text{mean}(B)|}{\text{mean}(B)} \right] \).

We combine the two indexes together as the reward of DQN model. The reward function at time \( t \) is expressed as \( r_t = W \times H \). The objective of DQN model is to find a policy to maximize the expected cumulative discounted rewards \( R_t \) as follows, where \( \beta \in [0, 1] \) is the discount factor for reward \( r_t \) and \( k \) is the number of time steps from \( t \).

\[
R_t = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \beta^k r_{t+k} \right]
\]

C. Q-Value

Name \( s_t \) as the input state data of the DQN model at time \( t \) in state space \( S \), which is an array value concatenated by: (i) the adjacency matrix of the network; (ii) the cluster ID of each node; (iii) the size of data that is produced in each node. A policy \( \pi \) is a mapping from the state space \( S \) to the action space \( A \). DQN aims to find the optimal policy to maximize \( R_t \). The Q-value \( Q(s_t, a_t) \) is defined as the reward \( R_t \) when taking action \( a_t \) in state \( s_t \) using policy \( \pi \). We use Q-value to measure the quality of a certain action in a given state. Our DQN model uses a fully connected DNN with weight set \( \theta \) to approximate the Q-function. After taking the action, the agent receives the reward \( r_t \) and the IoT network moves to a new state \( s_{t+1} \). We select the action that maximizes the Q-value as the one to be taken in the state \( s_t \).

\[
a_t = \arg \max_{a \in A} Q(s_t, a)
\]

The improved policy with Q-values is based on the following update equation, where \( \tilde{Q} \) is the improved policy and \( \alpha \in [0, 1] \) is the learning rate.

\[
\tilde{Q} (s_t, a_t) = Q(s_t, a_t) + \alpha [r_t + \beta \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)]
\]

The DQN model updates its DNN at each iteration to minimize the loss function \( L \).

\[
L = |r_t + \beta \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)|^2
\]

V. Preliminary Experimental Results

We evaluate the performance of the DQN model in a simulation IoT network. The network is deployed in an square area of 150m \( \times \) 150m. The nodes are randomly scattered in the area. As a preliminary experiment, we test the model in 40 nodes with 2 clusters, 60 nodes with 3 clusters, and 80 nodes with 4 clusters respectively. The data transmission range of nodes is 40m, 35m, 30m for 2, 3 and 4 clusters respectively. The detection range of object is 2 times larger.
than the transmission range. We set one dynamic object in the IoT area. The speed of object is 1m/s, and it is initialized at the position of a random cluster head. We randomly select another cluster head as the moving direction of the object. The object moves to the direction until it is outside the deployment area. After that, the object moves to the opposite direction. We assume each node produces one message of 1Kb every 10 seconds. If the object is in the detection range of a node, the node produces a message of 10Kb. The communication speed between IoT nodes is 1Kb/s. To simplify the simulation, we ignore the interference in wireless communication. The DQN model is processed every 10 seconds. Each round of training lasts for 500 seconds. The parameter values of the DQN model are shown in Table I.

We use a static cluster partition algorithm without DQN model for comparison. In the comparison solution, each edge server is set as a cluster head, and the network is partitioned to Voronoi clusters. We compare the average reward score between our DQN solution and the Voronoi clustering solution.

The experimental results of 2, 3 and 4 clusters are shown in Figure 4(a), 4(b) and 4(c) respectively. The results of DQN model are shown by the moving average with window size 11, and the result trend is illustrated by a 4 round polynomial curve fitting. In all the three experiments, our DQN approach has significant improvement compared with the benchmark results. After convergence, the improvements are around 56%, 47%, and 32% in 2, 3 and 4 clusters respectively. The improvement decreases as the number of clusters increases. The main reason is that we only select one cluster-core to move in a time step. If all the cluster-cores perform actions in a time step, the chance to find the cluster partition with higher reward would increase.

VI. FUTURE WORK

There are multiple points in the DRL model that we will improve in the future work.

- We use only one object with direct moving pattern in the implementation. This moving pattern cannot represent the real dynamic objects in IoT systems. We will evaluate whether the reinforcement learning model could cope with multiple objects moving in a more realistic pattern, such as random walk.
- To simplify the DQN model, we set one cluster-core in each cluster. In our next step work, we will set multiple cluster-cores in a cluster. This will provide smaller granularity for controlling the cluster shape.
- This paper assumes that the IoT network is static, and only the object moves around. It is unknown whether the reinforcement learning solution can cope with an IoT network in which every node moves randomly.

VII. CONCLUSIONS

This paper presents a DRL solution for dynamic network clustering in IoT networks with edge servers. The aim is to fulfill the requirements from both the IoT network and edge computing by optimizing the clustering of data collection. Our preliminary experiments show that the proposed DQN model can achieve better results compared to the static benchmark solution. In such scenarios, our work has showed that it is feasible to apply DRL techniques to IoT networks and edge computing. Moreover, we also believe that our design and implementation have offered an alternative to the current network clustering solutions.

ACKNOWLEDGMENTS

Long Cheng is supported by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 799066.

REFERENCES

[1] L. Cheng, S. Kotoulas, T. E. Ward, and G. Theodoropoulos, “Improving the robustness and performance of parallel joins over distributed systems,” Journal of Parallel and Distributed Computing, vol. 109, pp. 310–323, 2017.
[2] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on., IEEE, 2000, pp. 10–pp.
[3] G. Chen, C. Li, M. Ye, and J. Wu, “An unequal cluster-based routing protocol in wireless sensor networks,” Wireless Networks, vol. 15, no. 2, pp. 193–207, 2009.
[4] Q. Liu, A. Pruteanu, and S. Dulan, “Gradient-based distance estimation for spatial computers,” The Computer Journal, vol. 56, no. 12, pp. 1469–1499, 2013.
[5] Q. Liu, S. Dulan, and M. Warnier, “Area: an automatic runtime evolutionary adaptation mechanism for creating self-adaptation algorithms in wireless networks,” Proceedings of the Spatial Computing Workshop in AAMAS, pp. 23–28, 2013.
[6] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, “Resource management with deep reinforcement learning,” in Proceedings of the 15th ACM Workshop on Hot Topics in Networks. ACM, 2016, pp. 50–56.
[7] H. Ye and G. Y. Li, “Deep reinforcement learning for resource allocation in v2v communications,” in 2018 IEEE International Conference on Communications (ICC). IEEE, 2018, pp. 1–6.