Deep Reinforcement Learning for Communication Flow Control in Wireless Mesh Networks

Qingzhi Liu, Long Cheng, Adele Lu Jia, and Cong Liu

Abstract

Wireless mesh network (WMN) is one of the most promising technologies for Internet of Things (IoT) applications because of its self-adaptive and self-organization nature. To meet different performance requirements on communications in WMNs, traditional approaches always have to program flow control strategies in an explicit way. In this case, the performance of WMNs will be significantly affected by the dynamic properties of underlying networks in real applications. With providing a more flexible solution in mind, in this article, for the first time, we present how we can apply emerging Deep Reinforcement Learning (DRL) on communication flow control in WMNs. Moreover, different from a general DRL based networking solution, in which the network properties are pre-defined, we leverage the adaptive nature of WMNs and propose a self-adaptive DRL approach. Specifically, our method can reconstruct a WMN during the training of a DRL model. In this way, the trained DRL model can capture more properties of WMNs and achieve better performance. As a proof of concept, we have implemented our method with a self-adaptive Deep Q-learning Network (DQN) model. The evaluation results show that the presented solution can significantly improve the communication performance of data flows in WMNs, compared to a static benchmark solution.

Introduction

A wireless mesh network (WMN) is a multi-hop wireless network made up of communication nodes following a mesh topology. Because of some specific characteristics, such as cost-effectiveness, self-configuration, and self-organization, WMN has been considered as one of the encouraging technologies for IoT and has been widely used for data communications in various scenarios (e.g., smart buildings). To optimize the communication performance (e.g., data throughput) of WMNs, traditional approaches mainly rely on programming explicitly to control data flow. However, as the number of communication nodes and the complexity of data applications continuously grow, those approaches start to meet performance challenges, because it is hard for networks to respond dynamically.

Deep learning (DL) represents an effective solution to tackle the problem of flow control in communication networks [1]. Compared to traditional techniques, DL can greatly enhance the flexibility of a network to meet performance requirements dynamically. Among all the emerging DL technologies, Deep Reinforcement Learning (DRL) has demonstrated its outstanding capability in handling complex communication flows in mobile and wireless networks scenarios [2]. Similar to existing approaches in communications and networking [3], we can also train a DRL model over a WMN, in which the network features are fixed (e.g., with a pre-defined clustering pattern), to optimize the flow control of the WMN. However, the performance of the finally trained DRL model will be limited. The main reason is that the pre-selected feature values of a WMN directly affect the training of its DRL model. In this condition, it will be hard for us to choose the suitable property values at the beginning of training to guarantee that the trained DRL model can always achieve the best possible performance, due to the dynamic properties of WMNs.

To leverage the emerging DRL technologies to WMNs, in this article, we introduce a self-adaptive DRL approach for communication flow control in WMNs. Different from a general DRL-based networking solution [3], we focus on exploring the self-adaptive nature of WMNs, and on that basis to improve the performance of DRL. Specifically, we use a DRL model to optimize the flow control among WMN clusters, and a self-adaptive model to dynamically change the network clustering patterns. The two models will affect each other during the training process. On the one hand, the flow routes generated from DRL will trigger the self-adaptive model to adjust the network clustering pattern (e.g., add and reduce clusters). On the other hand, the adjusted clustering pattern from the self-adaptive model will push the DRL model to find better flow routes. This forms a mechanism similar to positive feedback. In this case, the finally trained DRL model is able to take advantage of the dynamic properties of WMNs and thus achieve better communication performance.

In general, the main contributions of this work are summarized as follows:

- We have introduced how to formulate the communication flow control problem in WMNs by following a typical Deep Q-Learning Network (DQN) model.

Digital Object Identifier:
10.1109/MNET.011.2000303

Qingzhi Liu is with Wageningen University & Research; Long Cheng (corresponding author) is with North China Electric Power University; Adele Lu Jia is with China Agricultural University; Cong Liu is with Shandong University of Technology.

- Based on the presented DQN-based flow control solution in WMNs, to further optimize the communication performance, we have proposed a self-adaptive DRL approach by seamlessly combining the DQN model with a self-adaptive clustering model.
- We have implemented our method, and our experimental results show the proposed approach can achieve better communication performance compared to a benchmark solution.

The remainder of this article is organized as follows. The related works are discussed in the following section. A typical DQN solution for cluster-based flow control is then presented. Our framework design for combining a DRL model and a self-adaptive model is then presented. Following that the experimental results are presented. Future work is then discussed, and we conclude this article in the final section.

Related Work

DRL for Wireless Networks

As one of the most emerging technologies, DRL is capable of handling large and complex optimization problems with high-dimensional state spaces and low-dimensional action spaces. Its main idea is that a function $Q(s, a)$ is used to predict the expected long term reward when taking action a at state s. In this case, the DRL model knows the estimated reward value of each action in its current state, and then it can decide which actions should be taken to achieve the maximum cumulative rewards in the long run.

Up to now, DRL has been used for complex tasks in wireless networks, including the cases with little or no prior knowledge of user workloads or underlying networks. For example, the work in [4] proposes a decentralized resource allocation mechanism to optimize sub-band and power level for transmission. The work in [5] dynamically controls the clustering pattern in IoT for balancing the communication load in the network and the data size in edge servers. The work in [6] attaches the DRL scheduler in an SDN controller for controlling network flows. The work in [7] uses DRL to maximize the long-term expected number of successful transmissions in a dynamic multi-channel access problem. The work in [8] proposes a MAC protocol by using a DRL model to learn an optimal channel access strategy in heterogeneous wireless networks. Although all these DRL solutions are able to manage networks effectively, they are designed based on static network environment models. In comparison, our framework has explored the dynamic properties of WMN. It can adaptively reconstruct underlying WMNs to improve the searching space of a DRL model and consequently to further improve the communication performance.

Clustering for Wireless Networks

Cluster-based communication in wireless networks has been widely studied in recent years [9]. For example, the work in [10] focuses on reducing network congestion, and the work in [11] aims to improve data throughput. Although these clustering solutions are effective on network partitioning, their target is to construct a static clustering pattern so that explicit flow control rules can be deployed. Compared to these approaches, our method can adjust the network clustering pattern dynamically during the training of a DRL model. In this way, the communication performance will gradually increase with the dynamical change of clustering patterns.

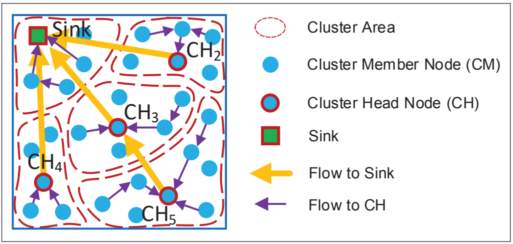

FIGURE 1. An example of cluster-based WMNs. The WMN is partitioned into clusters. The CMs forward data to the CHs, and the CHs forward data to the sink node.

Cluster-Based Flow Control by DRL in WMN

In this section, we first introduce a typical scenario of cluster-based flow control in WMN. Then, we present a DRL based framework for flow control optimization.

Cluster-Based WMN

A WMN is a wireless communication network organized in a mesh network topology. In a WMN, devices communicate with each other by multi-hop routes. The key property of WMN is that it can easily adapt the communication route to the changing of network topology. For example, when a new device joins in the network (e.g., turning on a cell phone) or an existing device leaves the network (e.g., hardware failure), the rest of the network devices can update their neighbor connectivity to formulate a new network topology. In this way, the devices in a WMN can always set up routes by intermediate nodes to communicate with each other.

Network clustering is a common way for flow control in WMNs. As illustrated in Fig. 1, the communications nodes in a WMN have been partitioned into a set of clusters, which can be built by using various partitioning approaches. Without loss of generality, in this work, we have adopted the commonly used Voronoi clustering approach [12]. In each cluster, a node is assigned as the cluster head node (CH), and the other nodes are named as cluster member nodes (CM). CH is responsible for collecting data from the CMs in a cluster. In each cluster, CMs can send data to the CH by multi-hop communications. In comparison, CHs have a larger transmission range than CMs, so that they can directly communicate with their neighbor CHs. Namely, CHs actually play the role of a backbone network for WMNs to provide communication connections between clusters.

In a typical application scenario like data aggregation, all the communication nodes will send data to a specified CH. In this case, to distinguish CHs, the CH specified to aggregate all the data is named as the sink. In the aggregation process, each CM first leverages multi-hop com-

In our solution, we assume that an agent resides in a server to observe network states, and takes actions calculated by a DQN model to control communication flows of a WMN. Moreover, we suppose that the WMN has been partitioned into clusters, and a sink node is required to collect data from the other nodes in the network. Then, the aim of the DQN model is to optimize the flow routes from CHs to the sink.

munication to forward data to its CH. Then, the CH sends data to the sink by multi-hop communication in the backbone network.

Formally, we can model a WMN as an undirected graph G. In detail, $V = \{v_1, ...v_i, ...v_n\}$ represents the nodes with $n = |V|$ and $i \in [1, n]$. $H = \{h_1, ...h_j, ...h_m\}$ represents the CHs with $m = |H|$, $j \in [1, m]$ and $H \subset V$. The network G is partitioned into clusters $C = \{c_1,... c_j, ...c_m\}$ with $m = |C|$, $j \in [1, m]$ and $|C| = |H|$. Each CH h_j resides inside a cluster c_j. Here, we name c_i and c_j as neighbor clusters if there is a pair of neighbor nodes residing in the two clusters, respectively. Based on existing solutions for routing data among CHs [13], we introduce a DRL-based approach for flow control in the following section.

DRL Solution for Flow Control

DQN [14] is a typical model used in DRL. Generally, a DQN model observes state s_t at time t, and it takes action a_t based on policy π to receive reward r_t. To evaluate the effectiveness of action a_t taken for state s_t, a Q-value $Q(s_t, a_t)$ is used in DQN models. The Q-value is defined as the expected accumulated discounted rewards when taking action a_t in state s_t. The main objective of a DQN model is to find a policy π to maximize the Q-value. On this basis, for a given state s_t, the DQN model will be able to select an action a_t to maximize the Q-value.

In our solution, we assume that an agent resides in a server to observe network states, and takes actions calculated by a DQN model to control communication flows of a WMN. Moreover, we suppose that the WMN has been partitioned into clusters, and a sink node is required to collect data from the other nodes in the network. Then, the aim of the DQN model is to optimize the flow routes from CHs to the sink.

To facilitate that, we define a connectivity matrix X for the CHs, and an action a_t is used to adjust the routing connectivity of each CH. The number of rows and columns in X equals the number of CHs, and the value X_{ij} in the row i and column j represents the route from h_i and h_j. If h_i needs to forward data to h_j, then two conditions will have to be met:

- Its local routing table has a route from h_i to h_j.
- The value of X_{ij} is 1.

Therefore, to control the routing table of each CH, we define two actions for a_t:

- Set X_{ij} to 1: flows are allowed to pass from h_i to h_j.
- Set X_{ij} to 0: flow are prohibited to pass from h_i to h_j.

Moreover, the observed state s_t is modeled as a concatenated array value of multiple network states, including:

- Cluster ID of CHs.
- Cluster ID of data source nodes.
- Connectivity matrix of CHs.
- Data arriving rate between CHs.
- Data arriving rate from CHs to sink.

To capture the communication performance in flow controls, three parameters have been used to describe the reward function of our DQN model:

- Data throughput Θ_t: the normalized amount of data received at the sink node per time unit.
- Data arriving rate Γ_t: the rate between the amount of data sent by the CHs and the amount of data received by the sink node.
- Connectivity rate Λ_t: the percent that CHs can build routes to the sink node.

We choose the above three parameters because a high data throughput Θ_t is important for WMNs. Moreover, Γ_t can be used to adjust and avoid unbalanced routes, such as that only part of data sources have connections to the sink. In the meantime, Λ_t can be used to control the percent of data source nodes that have connections with the sink. For a general case, we define the reward r_t of action a_t as $r_t = \Theta_t \cdot \Gamma_t \cdot \Lambda_t$.

For the model above, the probability that a state transition from s_t to s_{t+1} with action a_t and reward r_t can be described by a transition probability $p(s_{t+1}, r_t | s_t, a_t)$. We define all the state transition probabilities in terms of a state transition matrix, which is determined by the network environment, such as the communication speed, the size of a message, and the number of CHs. In our model, the state transition matrix is unknown to the agent. In this case, we can model the states, actions, rewards, and state transitions of the system as a Markov Decision Process (MDP), which is used to calculate the optimal policy by the DQN model.

Limitations of DRL Solution

The DRL solution described above can be used to optimize the routes of communication flows for a WMN. However, the approach still follows a conventional way of applying DRL, that is, the DRL model operates in an environment with pre-defined states and actions. Specifically, the DRL model optimizes the flows in static clusters, in which the number and position of clusters are fixed. In this case, the dynamic properties of WMNs, such as self-adaptive and self-organization, have not been explored, and thus the networking performance of the presented solution will be still sub-optimal. To support our argument, we list two limitations of the solution as follows.

- In the case that the number of CHs is small, it will be possible that a large number of flows pass over the same CHs, which will cause network congestion. To describe this problem, we have used a concept called *Congested-CH* in this work. Specifically, for a CH h_i, the throughput of its incoming flows and outgoing flows are called I_i and O_i, respectively. If the rate O_i/I_i is lower than a pre-defined threshold value K_i, then we claim that there is congestion in h_i. It should be noted that the value of each K_i can be customized based on the performance and service requirements of a WMN. Then, we can use K_i to detect multiple Congested-CHs in each round of training. For simplicity, we just use Γ_t to represent all the K_i in our implementation. Moreover, without loss of generality, we only detect a single Congested-CH in each

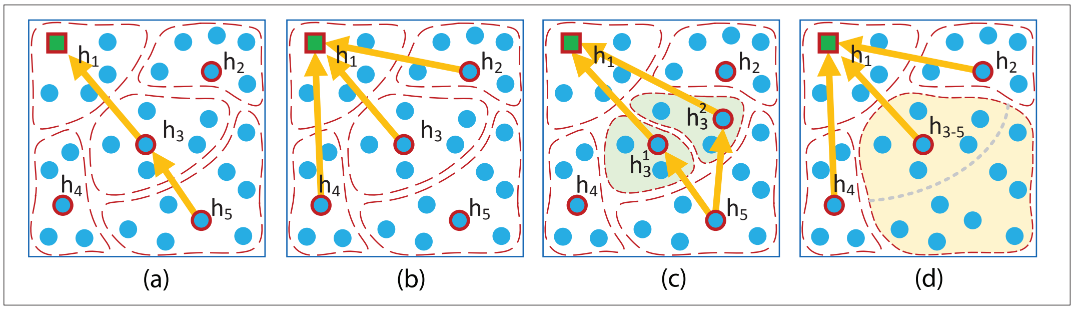

FIGURE 2. Examples of using self-adaptive DRL to control cluster-based WMNs. Reduce Congested-CHs by changing the network from a) to c). Reduce Idel-CHs by changing the network from b) to d): a) the flows are from h_3 and h_5 to h_1; b) the flows are from h_2, h_3, and h_4 to h_1; c) reduce the Congested-CH h_3 by separating c_3 to c_3^1 and c_3^2; d) reduce the Idle-CH h_5 by merging c_3 and c_5 to c_{3-5}.

training round, by selecting the CH with the maximum number of incoming flows in the case that Γ_t is smaller than a given value.

- In the condition that the number of CHs is large, it will be possible that no flow passes over some CHs. This means that these CHs will be in an idle state, so that collecting their states and controlling their routing tables will consume unnecessary communication resources. To represent this issue, we define an *Idle-CH* as: if both the throughput of incoming flows I_i and outgoing flows O_i of CH h_i are zero during communications, then h_i is an Idle-CH.

It is clear that the limitations above will negatively affect the performance of DRL. In the following section, we will propose a self-adaptive DRL approach to tackle this problem. Our method can optimize the number and position of clusters with the changing of communication workloads, and reduce the number of Congested-CHs and Idle-CHs for a WMN.

A Self-Adaptive DRL Framework

In this section, we first introduce a self-adaptive clustering approach to reduce Congested-CHs and Idle-CHs. Then, we present how to combine the method with the proposed DQN model to form a self-adaptive DRL solution for flow control in WMNs.

Self-Adaptive Clustering

When using the DRL model as presented earlier, the occurrence of Congested-CHs will imply that the optimized routing in the existing state-space is sub-optimal. To achieve the best possible performance, we can add new clusters in the network to diverge the congested flows. Suppose CH h_i is a Congested-CH, and the flow comes from its neighbor CH h_j. Then, we select a CM from c_i as a new CH, and name the original CH h_i and the new CH as h_i^1 and h_i^2, respectively. In this way, the cluster c_i will be partitioned into two sub-clusters c_i^1 and c_i^2, and the flow from h_j can be routed to h_i^1 and h_i^2. For example, as illustrated in Fig. 2a, some data flows are from h_3 and h_5 to h_1, and we assume that h_3 is a Congested-CH. To diverge the congested flows, we partition c_3 to clusters c_3^1 and c_3^2. Then, part of the flows originated from h_5 will be routed to h_3^1 and part to h_3^2 as shown in Fig. 2c. In this case, the congestion in h_3 can be reduced. Many existing solutions can be used to select CH and partition clusters in

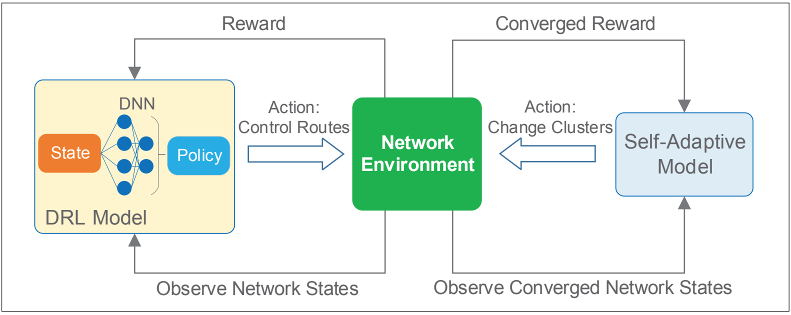

FIGURE 3. The proposed self-adaptive DRL framework for communication flow control in WMNs. The DRL model and self-adaptive model affect each other by changing the network environment.

our approach (e.g., [12]). Regardless, their details are not the focus of this article, since we aim to handle the case with adaptive changes of CHs. In this condition, to simplify our implementation, we randomly add a new CH in the cluster with the Congested-CH, and assume the new CH has the same neighbor connectivity as the Congested-CH. Then, the cluster can be partitioned into two sub-clusters based on the same partitioning solution as introduced earlier.

For the case of Idle-CHs, it is unnecessary to control their routing tables, and we can merge them with their neighbor clusters. Specifically, suppose h_i is an Idle-CH and h_j is one of its neighbor CHs. Then, we merge the two clusters into a single cluster c_{i-j}, and re-name h_j to h_{i-j} as the CH of c_{i-j}. For example, as demonstrated in Fig. 2b, there are flows from h_2, h_3 and h_4 to h_1, but h_5 is an Idle-CH. Therefore, the flow control of h_5 becomes unnecessary. Then, we merge c_3 and c_5 to c_{3-5}, as depicted in Fig. 2d. In this condition, the new CH of the cluster c_{3-5} is h_{3-5}. Similar to the case of reducing congested-CHs, in our implementation, we directly remove the Idle-CH and merge its CMs to the neighbor clusters.

Operation Procedure of Self-Adaptive DRL

Based on the presented self-adaptive clustering method and the DQN model, the operation procedure of our self-adaptive DRL framework is illustrated in Fig. 3. It has two main components: a DRL model and a self-adaptive model. During the procedure, the two models will affect each other by changing the properties of the network environment (e.g., number and position of CHs). The main steps of the procedure are summarized as follows:

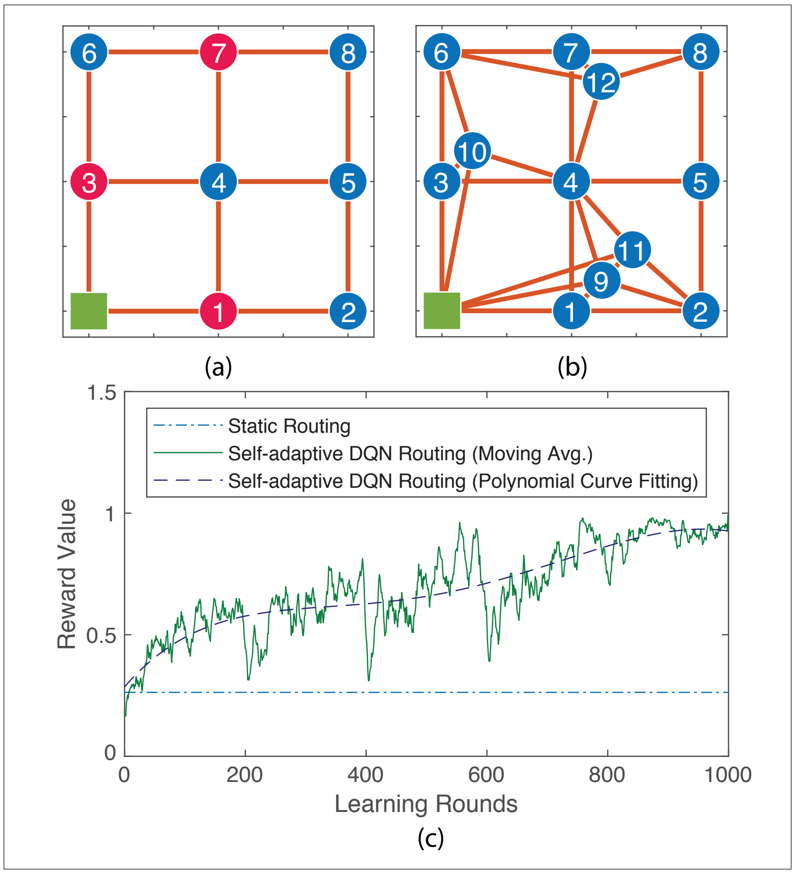

FIGURE 4. The self-adaptive DRL model optimizes the flow control by reducing the Congested-CHs in the grid WMN: a) The initial grid WMN; b) The WMN is changed by the self-adaptive DRL model; c) The rewards of the self-adaptive DRL model and the benchmark solution.

1. In the initialization, the system sets CHs at random positions in the WMN, and partitions the WMN into clusters.
2. The DRL model optimizes the routes from the CHs to the sink node as presented above.
3. DRL is put on standby when its reward value starts to converge. In the meantime, the self-adaptive model starts to reduce Congested-CHs and Idle-CHs to form a new CH network using the method introduced above.
4. The new CH network built by the self-adaptive model is used as a new network environment for the DRL model.
5. The procedure from 2 to 4 repeats until the system finds a flow control solution that fulfills a required reward value.

Performance Evaluation

As proof of concept, we have evaluated our framework in a network simulation environment using Python. The code we have used in this section is available at https://github.com/qingzhi-liu/self-adaptive-DRL-in-mesh-networks.

Experimental Setup

For simplicity, we have skipped the step of building clusters from the WMN in our experiments. Instead, we directly build a network of CHs, and specify some CHs as the data source to send messages periodically to a sink. Similar to many applications, such as home automation and precision agriculture, we set the transmission range of each CH to 11m, and use the shortest hopping path as the route from the CHs to the sink. To simulate dynamic communication flows, we randomly select a CH from a pre-defined set of data source CHs every two learning rounds, and let the selected CH transmit a 640 KB message to the sink. We assume that the maximum communication flow speed from a CH to the sink is 10 KB per second. If there are multiple flows passing through the same CH, the communication speed of a flow on this CH equals the maximum speed divided by the number of flows. In this way, we can calculate the flow speed from CHs to the sink and also the transmission time of each message. At the same time, we set a time-out value of 128 seconds for each message. If a message cannot arrive at the sink within the time-out, we count the communication of the message as a failure. These configurations follow the typical parameters of the IEEE 802.15.4-based Zigbee protocol, which is widely used in low-power short-range Internet of Things applications.

We have implemented the DQN model by Keras. The learning rate is 0.0001. The discount value is 0.5. The minimum epsilon value is 0.01. We consider that the routing has been optimized by the DQN model, if the converged reward value r_t is higher than 0.9. The Deep Neural Network (DNN) model of DQN has two hidden layers with 200 and 100 neurons. The neuron number of the DNN input layer equals the size of the network state s_t. The neurons of the DNN output layer maps to the elements of the connectivity matrix X.

To demonstrate the performance advantages of our framework, we have compared our approach with a benchmark solution, which is a typical flow control structure based on static network clustering [15]. Specifically, the solution routes data from the CHs to the sink by the shortest hopping path, and the number and position of clusters are kept the same as the initial state in the whole experiment.

In the figures of our experimental results, we have used the following symbols to describe the network topology. The green square 🟩 represents the sink node. The blue disc ● represents the CHs. The red disc ● and the blue circle ○ represent the Congested-CH and Idle-CH respectively.

Reduce Congested-CHs in Grid Networks

As the first experiment, we evaluate how the self-adaptive DRL approach improves communication performance by reducing Congested-CHs in a grid WMN. The grid WMN of 9 CHs is deployed in a 20m × 20m area as shown in Fig. 4a. The nodes 5, 7, 8 are the data source. The red nodes represent the Congested-CHs without self-adaptive DRL. After 1000 rounds of learning, the network evolves to Fig. 4b. Some new CHs are added beside the Congested-CHs to diverge data flow. The experimental result is shown in Fig. 4c. The reward value using self-adaptive DRL is much higher than the benchmark result. In the training procedure, we found that the reward value started to converge after 200 rounds of training. For the simplicity of our tests, we set the self-adaptive model to change the network topology every 200 rounds of training. This is also the reason why the reward value has a large amplitude variation at 200, 400, 600, 800 rounds. In more detail, the self-adaptive model has enlarged the state space of the optimal route for the DRL model by changing the network topology every 200 rounds.

Reduce Congested-CHs in Random Networks

In the second experiment, we evaluate how the communication performance is improved by our method through the reduction of Congested-CHs in a random WMN. The random WMN of 16 CHs is deployed in a 30m × 30m area, as demonstrated in Fig. 5a. Nodes 1, 2, 3, and 4 are the data source. After 400 rounds of learning, the network evolves to Fig. 5b, and the experimental result is demonstrated in Fig. 5c. The same as the result presented in Fig. 4c, the performance of the self-adaptive DRL is much better than the benchmark solution. Specifically, at the beginning of the learning, the reward value of the self-adaptive DRL is lower than the benchmark solution. After about 200 rounds of learning, its reward value is at the same level as the benchmark solution, but still lower than 0.6. To cope with this condition, the self-adaptive model checks the data flow of each node, and finds that there is congestion on node 2. Therefore, at about the 200th round, the self-adaptive model adds an additional CH beside node 2. Then, from round 200 to 400, the DRL model optimizes the route in the new topology. Finally, the flow congestion is reduced, and consequently the converged result of the self-adaptive DRL is much higher than the benchmark solution.

Reduce Idle-CHs in Random Networks

For the final evaluation, we test whether the self-adaptive DRL can reduce Idle-CHs without impacting communication performance. The random WMN with 20 CHs is deployed in a 30m × 30m area as depicted in Fig. 6a, and nodes 1, 2, and 3 are the data source. Compared to the experiments presented above, we have increased the number of CHs, and decreased the number of data source nodes. In this case, there are more Idle-CHs, and the target of the self-adaptive DRL is to reduce the Idle-CHs. In the experiment, after 400 rounds of learning, the network evolves to the case demonstrated in Fig. 6b, and the number of CHs decreases from 20 to 15. In the meantime, both the self-adaptive DRL model and the benchmark solution have reward values close to 1.0, as shown in Fig. 6c. In detail, it can be observed that the reward value of the self-adaptive DRL has a significant decrease near the 200th learning round. The main reason is that the self-adaptive model merges some Idle-CHs, and the network topology is changed near the 200th learning round. Thereafter, from the learning round 200 to 400, the DRL model optimizes the route in the new topology again, and finally the reward value converges to the optimal value. Compared to the static routing solution, the converged reward of the self-adaptive DRL model is on the same level, in which case the optimal reward value is 1. This means that the self-adaptive DRL model can successfully reduce the number of Idle-CHs while keeping the reward value at an optimal level.

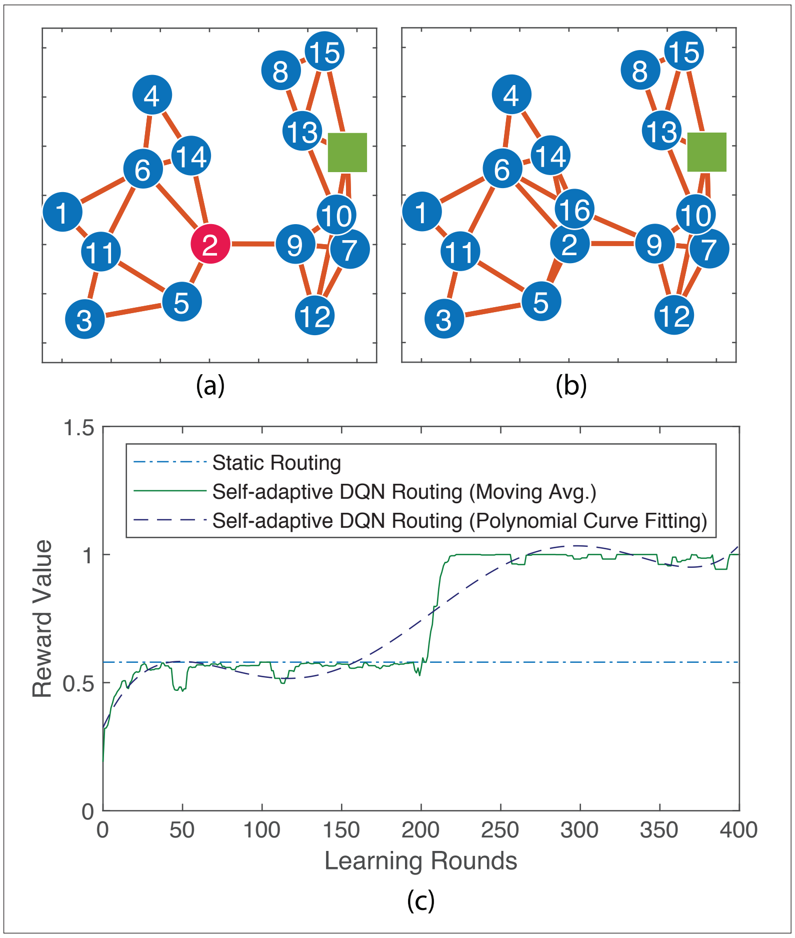

FIGURE 5. The self-adaptive DRL model optimizes the flow control by reducing the Congested-CHs in the random WMN: a) The initial random WMN; b) he WMN is changed by the self-adaptive DRL model; c) The rewards of the self-adaptive DRL model and the benchmark solution.

Limitations and Further Research Directions

Based on the above experimental results, we believe we have opened up a new framework solution for communication flow control in WMNs. Namely, we can strengthen the capability of DRL by employing a self-adaptive clustering approach. At the same time, we are aware of some key points for the further improvement of our design as follows:

- It will be challenging for our framework to handle a highly dynamic WMN, in which the network topology changes frequently. The main reason is that it is difficult to train a DRL model in real-time, especially when the size of a network is large. If we apply our framework directly to that case, then we will have to train a DRL model every time when the topology changes. A possible solution is to build a static CH network over a dynamic WMN, and then use our framework on the static CH network to optimize the communication flow control.
- It will be worth exploring more comprehensive network clustering approaches for our self-adaptive DRL model. In this article, we have only considered two factors (i.e., position and number of clusters) in our self-adaptive model, and we can integrate more advanced clustering factors into the model to further strengthen our method.

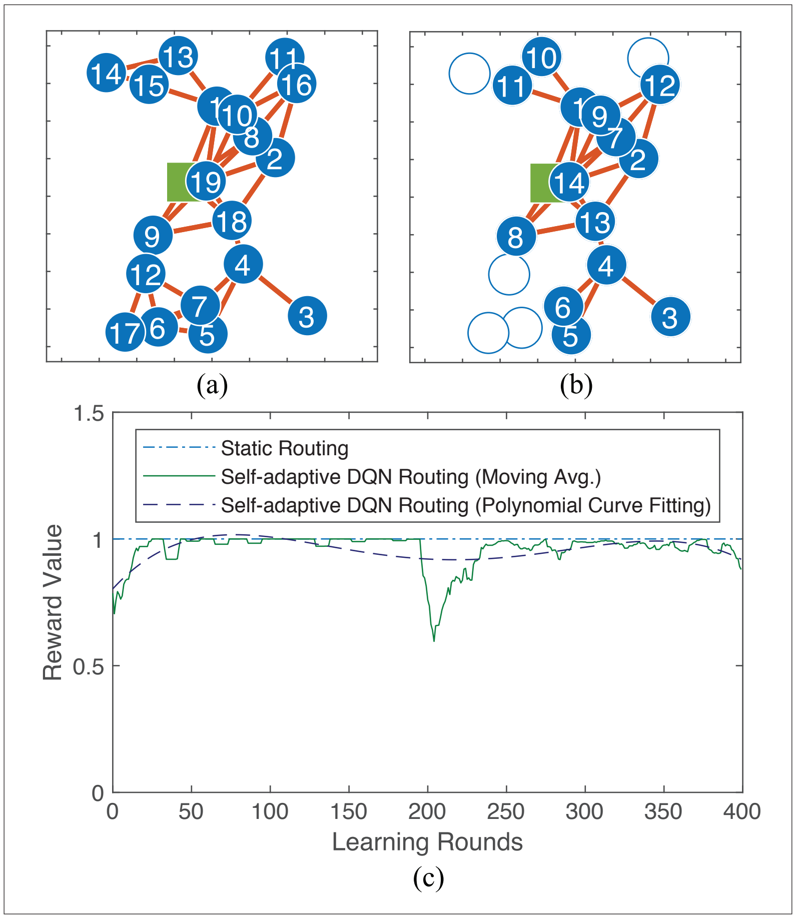

FIGURE 6. The self-adaptive DRL model optimizes the flow control by reducing the Idle-CHs in the random WMN: a) The initial random WMN; b) The WMN is changed by the self-adaptive DRL model; c) The rewards of the self-adaptive DRL model and the benchmark solution.

- It will be interesting to explore the load-balancing of network clustering in our framework. In this work, if there is flow congestion, a cluster will be split by adding CHs. However, this will make the size of the split cluster much smaller than the initial state. In the meantime, the more the congestions are, the smaller the size of the split cluster will be. This will make the cluster size unbalanced, and further decrease the efficiency of the model. Therefore, a partitioning method that can add new CHs while keeping cluster size balanced will be desirable.
- We have used a connectivity matrix to represent the action space in our DQN model. With the number of CHs increasing, the size of the action space will grow exponentially. In this case, our approach will be inefficient and non-scalable. To remedy this problem, a possible solution is to pick out only the matrix elements that represent neighbor nodes rather than all the nodes, and then operate the DQN action on the selected elements.

Conclusions

In this article, we have introduced how we can use the emerging DRL approach to optimize cluster-based flow control in WMNs. Different from a general DRL method that highly relies on a predefined network environment, we have explored the dynamic properties of WMNs and proposed a novel self-adaptive DRL solution, which can reconstruct underlying networks during the DRL model training process. We have presented the detailed design of our approach, and our experimental results have shown that our solution can achieve significant performance improvement compared to a benchmark solution.

It is expected that the introduced method is of value to the applications that are sensitive to networking performance in WMN and IoT scenarios. Our future work mainly lies in extending the proposed approach with more advanced network clustering algorithms, and our long term goal is to develop a highly efficient and intelligent system to automate the management of complex WMNs.

Acknowledgment

Part of this work was supported by the Beijing Municipal Science & Technology Commission (Z181100005118016); the National Natural Science Foundation of China (61902222); and the Taishan Scholars Program of Shandong Province (tsqn201909109).

References

[1] Z. M. Fadlullah *et al.*, "State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems," *IEEE Commun. Surveys & Tutorials*, vol. 19, no. 4, 2017, pp. 2432–55.
[2] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 21, no. 3, 2019, pp. 2224–87.
[3] N. C. Luong *et al.*, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 21, no. 4, 2019, pp. 3133–74.
[4] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep Reinforcement Learning Based Resource Allocation for v2v Communications," *IEEE Trans. Vehicular Technology*, vol. 68, no. 4, 2019, pp. 3163–73.
[5] Q. Liu *et al.*, "Deep Reinforcement Learning for IoT Network Dynamic Clustering in Edge Computing," *IEEE/ACM Int'l. Symposium on Cluster, Cloud and Grid Computing*, 2019, pp. 600–03.
[6] X. Huang *et al.*, "Deep Reinforcement Learning for Multimedia Traffic Control in Software Defined Networking," *IEEE Network*, vol. 32, no. 6, 2018, pp. 35–41.
[7] S. Wang *et al.*, "Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks," *IEEE Trans. Cognitive Commun. Networking*, vol. 4, no. 2, 2018, pp. 257–65.
[8] Y. Yu, T. Wang, and S. C. Liew, "Deep-Reinforcement Learning Multiple Access for Heterogeneous Wireless Networks," *IEEE JSAC*, vol. 37, no. 6, 2019, pp. 1277–90.
[9] A. S. Rostami *et al.*, "Survey on Clustering in Heterogeneous and Homogeneous Wireless Sensor Networks," *The J. Supercomputing*, vol. 74, no. 1, 2018, pp. 277–323.
[10] S. R. U. Jan *et al.*, "An Energy-Efficient and Congestion Control Data-Driven Approach for Cluster-Based Sensor Network," *Mobile Networks and Applications*, vol. 24, no. 4, 2019, pp. 1295–1305.
[11] E. Balevi and R. D. Gitlin, "A Clustering Algorithm That Maximizes Throughput in 5G Heterogeneous F-RAN Networks," *Proc. IEEE Int'l. Conf. Commun.*, 2018, pp. 1–6.
[12] B. Kao *et al.*, "Clustering Uncertain Data Using Voronoi Diagrams and R-Tree Index," *IEEE Trans. Knowledge and Data Engineering*, vol. 22, no. 9, 2010, pp. 1219–1233.
[13] L. Xu, R. Collier, and G. M. O'Hare, "A Survey of Clustering Techniques in WSNs and Consideration of the Challenges of Applying Such to 5G IoT Scenarios," *IEEE Internet of Things J.*, vol. 4, no. 5, 2017, pp. 1229–49.
[14] M. Hessel *et al.*, "Rainbow: Combining Improvements in Deep Reinforcement Learning," *Proc. AAAI Conf. Artificial Intelligence*, 2018.
[15] S. Tyagi and N. Kumar, "A Systematic Review on Clustering and Routing Techniques Based Upon Leach Protocol for Wireless Sensor Networks," *J. Network and Computer Applications*, vol. 36, no. 2, 2013, pp. 623–45.

Biographies

Qingzhi Liu is a lecturer with the Information Technology Group, Wageningen University & Research, The Netherlands. He received a B.E. and a M.Eng. from Xidian University, China in 2005 and 2008, respectively. He received a M.Sc. (cum laude) and a Ph.D. from Delft University of Technology, The Netherlands in 2011 and 2016, respectively. He was a postdoctoral researcher at the System Architecture and Networking Group, Eindhoven University of Technology, The Netherlands from 2016 to 2019. His research interests include Internet of Things and machine learning.

Long Cheng is a professor with the School of Control and Computer Engineering at North China Electric Power University in Beijing. He received the B.E. from Harbin Institute of Technology, China in 2007, M.Sc from the University of Duisburg-Essen, Germany in 2010, and Ph.D from the National University of Ireland Maynooth in 2014. He was an assistant professor at Dublin City University, and a Marie Curie Fellow at University College Dublin. He also has worked at organizations such as Huawei Technologies Germany, IBM Research Dublin, TU Dresden and TU Eindhoven. His research focuses on high-performance data analytics, distributed systems, cloud computing, and process mining.

Adele Lu Jia received the B.S. degree from the Harbin Institute of Technology, China, in 2007, the M.Phil. degree from The Chinese University of Hong Kong in 2009, and the Ph.D. degree from Delft University of Technology, The Netherlands, in 2013. She is currently an associate professor with the Computer Science Department, China Agricultural University. Her research interests include complex network analysis and data mining.

Cong Liu received the B.S. and M.S. degrees in computer software and theory from Shandong University of Science and Technology, Qingdao, China, in 2013 and 2015, respectively, and the Ph.D. degree in computer science and information systems from Eindhoven University of Technology, Eindhoven, the Netherlands, in 2019. He is currently a full professor in the School of Computer Science and Technology at Shandong University of Technology. He has more than 60 publications in journals and conferences including TSC, TSMC, TASE, TBD, TCC, TITS, IoTJ, FGCS, DSS, ICWS, ICPC, and EuroVIS, among others. His current research interests include business process management, process mining, Petri nets, and emergency management.