SCALABLE REINFORCEMENT LEARNING FOR ROUTING IN AD-HOC NETWORKS BASED ON PHYSICAL-LAYER ATTRIBUTES

Wei Cui and Wei Yu

Electrical and Computer Engineering Department
University of Toronto, Canada
e-mails: {cuiwei2, weiyu}@ece.utoronto.ca

ABSTRACT

This work proposes a novel and scalable reinforcement learning approach for routing in ad-hoc wireless networks. In most previous reinforcement learning based routing methods, the links in the network are assumed to be fixed, and a different agent is trained for each transmission node — this limits scalability and generalizability. In this paper, we account for the inherent signal-to-interference-plus-noise ratio (SINR) in the physical layer and propose a more scalable approach in which a single agent is associated with each flow and is trained using a novel reward definition and according to the physical-layer characteristics of the environment. This allows a highly effective routing strategy based on the geographic locations of the nodes in the ad-hoc network. The proposed deep reinforcement learning strategy is capable of accounting for the mutual interference between the links and is capable of producing highly effective routing solutions over the entire network in a scalable manner.

Index Terms— Routing, reinforcement learning, ad-hoc wireless network, physical layer, distributed optimization

1. INTRODUCTION

Routing in wireless ad-hoc networks is a complex problem involving sequential decisions in each hop in order to build up a route that optimizes certain network utilities. Due to the lack of centralized control, routing in wireless ad-hoc networks needs to be performed in a distributed manner. Starting from the source node, the selection of next node in each hop requires considerations of multiple often-conflicting factors. Most existing routing algorithms assume that the network has a fixed set of links and aim at network-layer objectives such as congestion control, minimizing transmission delay, or maximizing successful transmission probability. However, these works often overlook physical-layer considerations such as the flexibility for each node to connect to different neighboring nodes in a wireless medium, the mutual interference between the transmission links, and the abilities for each link to do rate adaptation. Recognizing this gap, this paper advocates a routing protocol at the *physical layer*, taking into account the channel strengths and the interference levels. Assuming a path-loss model for the wireless channels, these factors are largely determined by the geographical locations of the mobile nodes. The motivates us to consider a new approach to routing based on the locations of each node and its neighbours and the physical-layer characteristics of the surrounding wireless channels.

As routing amounts to successive selections of nodes in each hop, the number of all possible routes is combinatorial in nature, therefore finding the globally optimal route by exhaustive search is computationally prohibitive in large networks. Decentralized discrete optimization algorithms previously proposed for routing [1, 2, 3, 4, 5, 6] are mostly heuristic in nature. From a different perspective, routing involves sequential decision making, in which each decision is a function of the current state of the network. Thus, routing can be readily modeled as a Markov decision process [7] and naturally fits into the realm of reinforcement learning. In this direction, many previous works [8, 9, 10, 11, 12, 13, 14, 15, 16] have employed the classical *Q-learning* [17] algorithm to train agents to find the optimal route. In these works, a distinct agent is associated with each transmission node. Each agent maintains a list of reachable neighbors and a Q-value table of “fitness scores” for these neighbors as the potential next hop (e.g. time or number of hops to reach the destination). After training, the route is determined by the agents who select the neighbor with the highest fitness scores as the next hop. Further, [18] proposes the use of a specific Q-function estimator [19] for the agent, allowing the same agent to be applied to different nodes. Moreover, [20] explores the use of deep reinforcement learning [21] as a complement to a proposed heuristic routing approach. Finally, [22] focuses on *stochastic routing*, where each node is characterized by a forwarding probability depending on its selfishness and remaining energy. Reinforcement learning is used on each node to come up with the forwarding probability. Note that none of these works perform routing based on physical-layer attributes.

Collectively, the approaches in those works in using reinforcement learning on routing for ad-hoc wireless networks are restricted by all or at least some of the following modeling and design choices:

- **C.1** The ad-hoc network model assumes a fixed set of connections where two nodes are either reachable from each other or not.
- **C.2** A distinct agent is associated with every node in the network and is trained specifically for this node.
- **C.3** The same network must be used for both training and testing.

Each of these modeling and design choices imposes limitations and has drawbacks. **C.1** abstracts away pivotal physical-layer characteristics of the network, and cannot account for objectives such as SINR based QoS. **C.2** requires a number of agents to match to the network size, therefore limiting scalability and generalization capability. **C.3** further limits agents' abilities to adapt to network changes. Attempting to address the limitation from **C.3**, [10] let agents *explore* all new neighbors opportunistically, which is essentially retraining. As the layout of an ad-hoc network can frequently change, any solution designed under **C.3** is inherently insufficient.

In this paper, we introduce a novel and scalable deep reinforcement learning approach to routing for ad-hoc wireless networks based on physical-layer inputs, constraints and objectives. As a main novelty, we associate one agent to each *data flow* from the

This work is supported by NSERC and Huawei Technologies Canada.

source node to the destination node, thus allowing significant reduction in the number of agents needed. More importantly, the agent executes sequential node selection tasks along the route, thus following a Markov decision process unlike the previous methods. The fact that the agent takes physical-layer information as input is crucial — this allows the same agent to be used for all nodes along the route. Furthermore, we show that the same model parameters of the agent can also be shared across all distinct data flows, or even generalized across different ad-hoc networks, while achieving some high-quality global objective. Finally, the agent can also be designed to adapt to network characteristics such as the density of neighbors.

2. ROUTING FOR AD-HOC WIRELESS NETWORK WITH PHYSICAL-LAYER MODEL

Consider a wireless ad-hoc network with N mobile nodes, and M data flows, each with a source and a destination at fixed locations. Each node in the network is equipped with a single antenna for transmitting and receiving purposes. For simplicity, we neglect the self-interference and adopt an idealized full-duplex assumption so that simultaneous transmitting and receiving is possible in each node[1]. Each data flow consists (potentially) multiple hops from the source to the destination, with a number of intermediate mobile nodes acting as relays. Each mobile node can relay only for a maximum of one data flow. We use $\mathcal{F}$ to denote the set of data flows; $\mathcal{S}$ and $\mathcal{T}$ to denote the set of sources and destination terminals; and $\mathcal{N}$ to denote the set of mobile nodes. The task of routing is to select the ordered list of mobile nodes as relay stations for each $f \in \mathcal{F}$.

To utilize reinforcement learning methods, the environment around each node, comprised of the neighboring mobile nodes, needs to be assumed to follow a stationary distribution. To this end, we propose to characterize the ad-hoc network layouts through a node density distribution function. Specifically, we assume a density profile across the network and assume that the mobile nodes are distributed locally uniformly within each sub-region.

At the physical layer, the maximum transmission rate between a pair of nodes is characterized by the capacity of the underlying wireless channel, which is a function of the signal-to-noise-and-interference (SINR) at the receiver. In details, consider a link with the transmitting node $i \in \mathcal{S} \cup \mathcal{N}$ and the receiving node $j \in \mathcal{T} \cup \mathcal{N}$, with p_i as the transmit power of node i, σ^2 as the background noise power, and $h_{ij} \in \mathcal{C}$ as the channel strength, under full frequency reuse over bandwidth W, the maximum transmission rate of this link is characterized as:

$$R_{(i,j)} = W \log \left(1 + \frac{|h_{ij}|^2 p_i x_i}{\sum_{\substack{k \neq i,j \\ k \in \mathcal{T} \cup \mathcal{N}}} |h_{kj}|^2 p_k x_k + \sigma^2} \right) \quad (1)$$

The binary variable x_i ($i \in \mathcal{S} \cup \mathcal{N}$), indicating whether the node n_i is transmitting or idle, is determined by the routing solution (i.e. if node n_i is used by any route or not). Throughout this paper, the transmit power p_i is assumed to be fixed. Further, we assume that the entire frequency band is reused by all the nodes, hence the spectral efficiency is low due to the interference from nearby links.

A data flow consists of a sequence of links, starting at the source, ending at the destination, in which the receiving node of the previous link is the transmitting node of the next link. Since the data is continuously transmitted along the route, the overall data rate for each data flow is determined by its *bottleneck* link capacity (i.e. the minimum data rate among all its links). Specifically, consider a data flow $f \in \mathcal{F}$ taking the route $n_1 \rightarrow \ldots n_i \rightarrow \ldots n_r$, where $n_1 \in \mathcal{S}$ and $n_r \in \mathcal{T}$ are the source and the destination terminals. The overall data rate of the flow f is

$$R_f = \min_{i=1,2,\ldots,r-1} R_{(n_i,n_{i+1})} \quad (2)$$

The objective of the routing algorithm is to select the intermediate nodes for each data flow to optimize some global objective.

A network may consist of multiple flows. The global objective across the network is typically a function of the data rates of all the flows. For example, we may optimize the sum data rate, or the minimum data rate across all the flows in the network, i.e.,

$$\text{Sum-Rate} \quad \sum_{f \in \mathcal{F}} R_f \quad \text{or} \quad \text{Min-Rate} \quad \min_{f \in \mathcal{F}} R_f \quad (3)$$

Note that the data rates across the different flows have strong interdependencies. Specifically, since each intermediate node can only be used in one flow, the utilization of a node for one flow precludes its use for any other flow. Furthermore, the transmitted signal of one link would appear as interference for all other links. Thus, there is strong trade-off among the data rates of all flows in the network.

Optimizing just a single flow is already challenging: it is a discrete optimization problem and is non-convex due to the interference among the links within the route as in (1). Optimizing multiple flows is even more challenging due to the interactions between them. The approach taken in this paper is to let a single agent optimize a flow one hop at a time along a route using reinforcement learning, then doing so successively over all flows for optimizing their routes, in effect competing with each other. As shown later in the paper, this would already lead to highly effective solutions to problems such as (3).

3. FLOW BASED ROUTING WITH DEEP REINFORCEMENT LEARNING

The ad-hoc network routing problem can be modeled as a Markov decision process, but because the distributed agents can only observe local information, the multiple-flow routing problem belongs to the class of *multi-player partially observable Markov decision process* (MP-POMDP). To tackle this difficult problem, we propose an reinforcement learning approach with several novel design aspects.

3.1. Agent-to-Flow Association

A key innovation of this paper as compared to prior work is to associate an agent to each data flow. As the route is established hop-by-hop, the agent moves along with the frontier node of the partially established route and decides the best next hop for the frontier node. This *agent-to-flow* association significantly reduces the number of parameters involved as compared to most prior approaches that train a different agent for each node. It also significantly improves scalability and generalization ability of the overall approach. Furthermore, this approach can be generalized across multiple data flows and even across distinct ad-hoc networks, as shown in Section 4.

When optimizing multiple flows, we adopt a *sequential* ordering for establishing routes: i.e., we let one flow establish its entire route before the next flow, and do so over multiple rounds to allow the interference pattern to be fully established and observed by the agent. Note that for each agent, all the other agents' actions form a part of the environment. Thus, such sequential ordering allows each agent to observe the consequences of other agents' actions.

[1] To deal with the self-interference in more practical settings, it is possible to extend the current work to the setting with multiple frequency bands.

3.2. Actions, States, and Reward

The action space for the agent as it moves along the frontier node is the set of next-hop candidates. As this paper adopts a physical-layer network model in which the connections are not predetermined, we consider a fixed number c available neighbors with strongest channels (which have not been used by other routes) as candidates. In addition, to ensure sufficient exploration capability, we also add one *reprobe* action, which means that if the agent decides none of the c strongest neighbors is a suitable next hop, it will proceed to probe the next c strongest neighbors, until a suitable next hop is found. Thus, the agent's action space $\mathcal{A}$ consists of $c+1$ actions: one action for each of the c strongest neighbors, and one action for reprobing.

The agent's state space $\mathcal{S}$ is meant to summarize crucial factors about each of the c strongest neighbors. In our problem, routing is performed based on the distances to the neighbors and the physical orientation to the final destination. To this end, the agent gathers the following information from each neighbor:

S.1 The distance between the neighbor and the frontier node.

S.2 The angle difference between the directions from the frontier node to the neighbor and to the destination.

S.3 The total interference the neighbor is exposed to.

Here, **S.1** and **S.2** indicate the amount of progress toward the destination node that can be made if the neighbor is chosen as the next hop, while **S.1** and **S.3** together allow an estimation of the SINR to the neighbor (assuming a path-loss model for channel strength).

The agent's rewards should reflect the objective function. Therefore, the most intuitive reward definition is to set rewards to be all zero along the route until the last hop, which takes a reward value of (2), i.e., the minimum rate across the links in the route. However, this definition has the drawback that such rewards and the resulting Q-values could well be determined by an earlier link, making them independent of the agent's choice of actions. Consequently, it makes it impossible for the agent to interpret the rewards during training.

To address this problem, we propose the following novel reward definition: during training, each state-action pair for a node is assigned a reward value of the **bottleneck link rate from that node *onwards* in the route**. Specifically, consider the data flow $f \in \mathcal{F}$ taking the route $n_1 \to \dots n_i \to \dots n_h$. At any node n_k in f, $(0 \le k < h)$, the agent observing state s_k and taking the action a_k (leading to n_{k+1}) is assigned a reward, denoted as $\widetilde{\mathbf{Q}}$, of

$$\widetilde{Q}(s_k, a_k) = \min_{i=k\dots h-1} R_{(n_i, n_i+1)} \tag{4}$$

Note that we denote the reward as $\widetilde{Q}$, because it also plays the role of Q values for Q-learning that incorporates the future information (and no past information) and being the metric for determining optimal actions. This definition of $\widetilde{Q}$ allows its value to be learned and interpreted by the agent based on the observed state-action pairs during training. During training, we use the novel $\widetilde{Q}$ signals in place of the Q values in conventional reinforcement learning.

In the works on classical Q-learning [17] and DQN [21, 23, 24], the target for $Q(s_t, a_t)$ is computed based on the Bellman equation [25], in which $Q(s_{t+1}, a_{t+1})$ is also needed and is computed based on the estimation of the model. Such an procedure is commonly referred to as *bootstrapping*, which can cause biases. Instead, our $\widetilde{Q}$ signals are directly computed by (4). Therefore, as we train our model with $\widetilde{Q}$ values, there is no bias within targets caused by bootstrapping. We do note that our target computation resembles the single-trajectory *Monte-Carlo approximation*, therefore can lead to more variance in our targets, which is a price for eliminating bias.

3.3. Deep Q-Learning based Routing with Novel Rewards

Since the states are continuous variables, the agent needs to be able to generalize over unseen portions of the state space $\mathcal{S}$. To this end, we utilize deep reinforcement learning, specifically deep Q-learning [21], in which a neural network, namely *DQN*, is trained to predict Q values (in our case, the $\widetilde{Q}$ values) given the state-action inputs.

We format each state input as a vector of length $3c$ (three features for each of the c strongest neighbors). These input vectors are processed with fully connected layers with non-linearities. We adopt the state-of-the-art *dueling-DQN* network architecture [23] for our agent, which is trained following the experience-replay technique as in [21], with uniform sampling in the replay buffer. The agent follows the *ϵ-greedy* [26] policy for gathering experiences. With probability $1-\epsilon$, the agent acts greedily according to the agent's current estimation; the action with the highest $\widetilde{Q}$ value is selected to generate the next step. With probability ϵ, the agent randomly selects a node as the next hop. Given this randomly selected node, we then produce a series of agent transactions (actions along with old and new states) to be stored in the agent's replay buffer. In either case, if the selected node is not within the c strongest neighbors to the agent, we store a reprobing transaction and search for the next c strongest neighbors. This allows us to obtain enough reprobe transactions.

To showcase the generalization ability of our method, we only train the agent on one specific data flow during experience gathering. Specifically, we first use a simple *closest-to-destination among the strongest neighbors* heuristic to produce routes of all but one data flow. We then assign the agent to the last flow to gather experiences and conduct training. As we shall see in Section 4, such lightweight training strategy is already sufficient to produce an agent that generalizes well across all data-flows as well as across different ad-hoc networks. In testing, the trained agent simply selects the next hop (or reprobe) based on $\widetilde{Q}$ values. It does not need to be re-trained when the network topology changes, unlike in methods such as in [10].

3.4. Agent Policy Enhancements

In addition to $\widetilde{Q}$ values based routing, from domain knowledge, we propose two enhancements to the agent's policy:

- If the agent chooses a neighbor that does not have the strongest channel to the frontier node, then all neighbors with stronger channels to the frontier node are excluded for future consideration for the next hop.
- If the destination node appears within the agent's exploration scope (i.e. one of the c strongest neighbors), then the agent would not choose any neighbor which has a weaker channel to the frontier node than the destination node.

Both enhancements effectively prevent agents from taking non-essential back-and-forth hops.

3.5. Multi-Agent Distributed Optimization

The agent learns and acts according to $\widetilde{Q}$ in (4) in order to optimize the data rate of its own flow. If instead we optimize with respect to a global reward signal, we can encounter the *credit mis-assignment* problem [27]. We observe, however, that in a wireless ad-hoc network, where the major performance limiting factor is the mutual interference, route optimization in a flow tends to lead to interference avoidance which benefits other data flows as well. Therefore in our problem, designing the agent to act selfishly in multiple rounds actually allows every agent to adjust according to each other's routes, thus achieving *implicit cooperation*.

Table 1. Design Parameters for the DDQN Agent

Parameters	Number of Neurons	
Initial Main-Branch fully-connected layers	1st	120
	2nd	120
State-Value Function fully-connected layers	1st	75
	2nd	1 (1 state value)
Action-Advantage Function fully-connected layers	1st	75
	2nd	7 (7 actions)

Table 2. Average Sum-Rate and Min-Rate Performances (kbps)

Methods	Sum Rate	Min Rate
DDQN Agent	450.6	60.86
Closest-to-destination among strongest neighbors	149.7	18.65
Best-direction among strongest neighbors	156.1	16.22
Largest-data-rate among strongest neighbors	67.4	1.32
Strongest neighbor	56.1	0.63
Lowest-interference among strongest neighbors	32.4	2.72

4. SIMULATION RESULTS

We consider wireless ad-hoc networks in a $500 \times 500 m^2$ region with $M = 3$ data flows. The node density profile is specified over nine equally divided sub-regions, with $(5, 10, 7, 6, 7, 4, 8, 3, 6)$ nodes located randomly within each sub-region. We consider the short-range outdoor model ITU-1411 with a distance-dependent path-loss to model all wireless channels, over 5MHz bandwidth at 2.4GHz carrier frequency. All antennas are with 1.5m height. We assume the transmit power level at 30dBm for all nodes; the background noise level at -150dBm/Hz. We randomly generate 520K ad-hoc network layouts for training. Among these, 20K layouts are used for random exploration to generate the agent's initial experience. The remaining 500K layouts are used for the ϵ-greedy policy based training. For testing, we randomly generate 500 new ad-hoc network layouts.

We use $c = 6$ as the number of the strongest neighbors the agent explores each time. Correspondingly, the inputs to the dueling-DQN are 18-component vectors. The state inputs are processed by sequential fully-connected layers, organized into a state-value prediction branch and an action-advantage prediction branch, as in [23]. The specification for our neural network is summarized in Table 1.

We present the sum-rate and min-rate testing results for our agent (named DDQN) as compared to a comprehensive list of benchmarks. These benchmarks are greedy in nature, since to our knowledge there currently does not exist efficient non-greedy algorithm for ad-hoc network routing at the physical layer. We follow the multi-round sequential routing for all methods. In simulations, two rounds appear to be sufficient already. As shown in Table 2, our proposed method achieves significantly better performance than all benchmarks in both sum-rate and min-rate performances. We emphasize that this performance is achieved by re-using a single agent across all three data flows, over all testing network layouts.

To better understand how our method excels, we provide a visualization of routes formed by our agent together with selected benchmarks over a random ad-hoc network layout in Fig. 1. As shown, our agent intelligently spreads out all data flows to mitigate mutual interference, while still maintaining strong and properly directed links to the destination to form the routes.

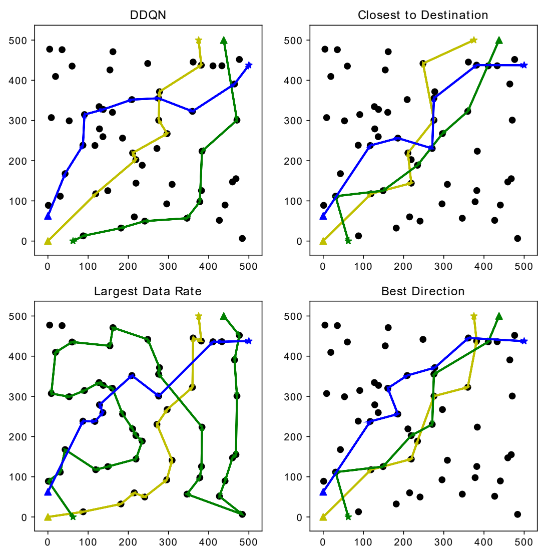

Fig. 1. Routes achieved on three data flows.

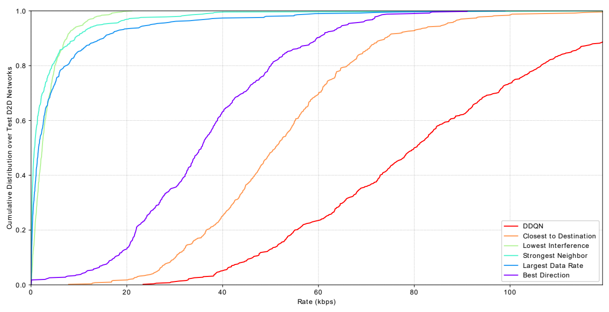

Fig. 2. Cumulative distribution function of sum rate over 10 flows in 500 large-scale networks.

To test the generalizability of our method, we directly take the agent trained under the original setting and reuse it for a much larger ad-hoc network of 10 data flows in a $1500 \times 1500 m^2$ region. We place an arbitrarily selected and much larger number of $(36, 34, 42, 38, 46, 40, 54, 45, 42)$ nodes over nine evenly divided sub-regions. The sum-rate results are shown in Fig. 2. Our approach still significantly outperforms the benchmarks. This illustrates that the proposed method generalizes well for larger networks with more data flows in sum-rate optimization.

5. CONCLUSION

This paper proposes a physical-layer based reinforcement learning approach to the wireless ad-hoc network routing problem. By training a universal agent along the flow that takes the physical environment as the input, along with a novel reward function definition, we have arrived at a highly scalable routing algorithm which can be adapted to the varying network layout characteristics and generalized to large-scale networks.

6. REFERENCES

[1] E. Gafni and D. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *IEEE Trans. Commun.*, pp. 11–18, Jan. 1981.

[2] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," *Proc. SIGCOMM '94*, pp. 234–244, Oct. 1994.

[3] S. Murthy and J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *Mobile Netw. Appl.*, pp. 183–197, June 1996.

[4] R. Sivakumar, B. S. Das, and V. Bharghavan, "An improved spine-based infrastructure for routing in ad hoc networks," *Proc. IEEE Symp. Computers and Communications*, 1998.

[5] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in ad hoc networks," *IEEE J Sel. Area Comm.*, vol. 17, no. 8, 1999.

[6] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad-hoc wireless networks," in *IEEE INFOCOM*, Apr. 2001.

[7] R. Bellman, "A markovian decision process," *J. of Math. and Mech.*, vol. 6, no. 5, pp. 679–684, 1957.

[8] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," *Adv. in Neural Inf. Process. Syst.*, vol. 6, Oct. 1999.

[9] S. Choi and D. Yeung, "Predictive q-routing: A memory-based reinforcement learning approach to adaptive traffic control," *Adv. in Neural Inf. Process. Syst.*, vol. 8, Dec. 1999.

[10] Y. Chang, T. Ho, and L.P. Kaelbling, "Mobilized ad-hoc networks: a reinforcement learning approach," in *Int. Conf. on Auton. Comput. (ICAC)*, May 2004.

[11] A. Forster and A. L. Murphy, "A feedback-enhanced learning approach for routing in wsn," in *Commun. in Distrib. Syst.*, Apr. 2007.

[12] G. Santhi, A. Nachiappan, M. Z. Ibrahime, R. Raghunadhane, and M. K. Favas, "Q-learning based adaptive QoS routing protocol for manets," in *Int. Conf. on Recent Trends in Inf. Technol. (ICRTIT)*, June 2011.

[13] A. Alharbi, A. Al-Dhalaan, and M. Al-Rodhaan, "Q-routing in cognitive packet network routing protocol for MANETs," *NCTA 2014 - Proc. of the Int. Conf. on Neural Comput. Theory and Appl.*, pp. 234–243, Jan. 2014.

[14] V. K. Sharma, S. S. P. Shukla, and V. Singh, "A tailored Q-learning for routing in wireless sensor networks," in *Int. Conf. on Parallel, Distrib. and Grid Comput.*, Dec. 2012.

[15] L. R. S. Campos, R. D. Oliveira, J. D. Melo, and A. D. Doria Neto, "Overhead-controlled routing in WSNs with reinforcement learning," in *Int. Conf. on Intell. Data Eng. and Autom. Learn. (IDEAL)*, 2012.

[16] R. Desai and B. P. Patil, "Cooperative reinforcement learning approach for routing in ad hoc networks," in *Int. Conf. on Pervasive Comput. (ICPC)*, Jan. 2015.

[17] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[18] P. Wang and T. Wang, "Adaptive routing for sensor networks using reinforcement learning," in *Int. Conf. on Comput. and Inf. Technol. (CIT)*, Sept. 2006.

[19] M. G. Lagoudakis and R. Parr, "Model-free least squares policy iteration," in *Neural Inf. Process. Syst. (NIPS)*, 2002.

[20] N. Abuzainab, T. Erpek, K. Davaslioglu, Y. Sagduyu, Y. Shi, S. Mackey, M. Patel, F. Panettieri, M. Qureshi, V. Isler, and A. Yener, "QoS and jamming-aware wireless networking using deep reinforcement learning," Oct. 2019, [Online] Available: https://arxiv.org/abs/1910.05766.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[22] P. Nurmi, "Reinforcement learning for routing in ad hoc networks," in *Int. Symp. on Modeling and Optim. in Mobile, Ad Hoc and Wireless Netw. and Workshops*, Apr. 2007.

[23] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," *Proc. of Mach. Learn. Res.*, vol. 48, pp. 1995–2013, 2016.

[24] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. of Assoc. for the Adv. of Artif. Intell.*, Feb. 2016.

[25] R. Bellman, "On the theory of dynamic programming," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 38, no. 8, pp. 716–719, 1952.

[26] C. Watkins, *Learning From Delayed Rewards*, Ph.D. thesis, University of Cambridge, Jan. 1989.

[27] Y. Chang, T. Ho, and L. Kaelbling, "All learning is local: Multi-agent learning in global reward games," in *Adv. in Neural Inf. Process. Syst. (NeurIPS)*, 2003.