# Scalable Reinforcement Learning for Routing in Ad-hoc Networks Based on Physical-Layer Attributes

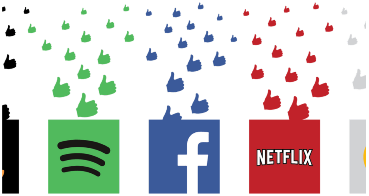
Wei Yu
Joint work with Wei Cui

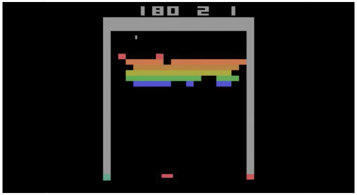
Electrical and Computer Engineering Department
University of Toronto

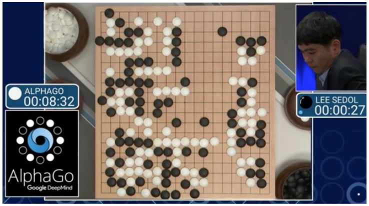
October 28, 2020

## Machine Learning

- Universal function mapping – by supervised or reinforcement learning
- Incorporating vast amount of data over poorly defined problems
- Highly parallel implementation architecture



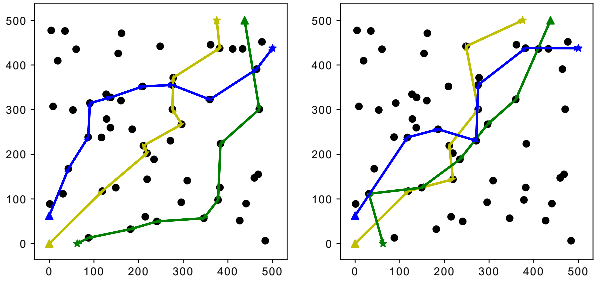




© Images from the web subject to copyrights

## Deep Reinforcement Learning

- Deep Learning: Classification or static optimization probelm
- Reinforcement Learning: Decision or dynamic control problem
- Deep Reinforcement Learning: Incorporate the environment



- This talk: Deep reinforcement learning for routing

## Routing in Wireless Ad-Hoc Networks

- Routing in wireless ad-hoc networks involves sequential decisions in each hop in order to build up a route that optimizes network utilities.
- Multiple flows co-exist in a wireless environment, each consisting of multiple hops from source to destination with intermediate relays.
    - Choice of which neighbouring nodes to serve as relays
    - Amount of interference caused by intermediate nodes
- Due to the lack of centralized control, routing in wireless ad-hoc networks needs to be performed in a distributed manner.
- This paper models routing as a Markov decision process, and use deep reinforcement learning for intelligent routing.

## Markov Decision Process

- Markov Decision Process (MDP) is a process in which an agent acts across time steps in an environment to optimize a reward, defined by:
    - State space: $S$
    - Action space: $A$
    - State-transition probability function: $p(s_{t+1}|s_t, a_t)$
    - Reward function: $r(s_t, a_t)$
- At any time $t$, the agent observes the state $s_t \in S$, executes an action $a_t \in A$, and then receives a reward $r_t$, then goes to a new state $s_{t+1}$.
- Agent's policy $\pi : s \to a$ is a state-to-action mapping, specifying how the agent chooses actions corresponding to every state in MDP.
- Reward is typically cumulative as defined by a discount factor $\gamma$:

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \dots \tag{1}$$

- The MDP control problem is to find an optimal policy $\pi^*$ under which the future cumulative rewards that the agent achieves is maximized.

## Reinforcement Learning

- Reinforcement learning (RL) aims to solve the MDP control problem when the MDP dynamics is unknown.
- The agent perceives and estimates the environment through interactions, then learns $\pi^*$ based on estimations about the MDP.
- Action-value function is the future cumulative reward after the agent executes action $a_t$ at state $s_t$ while following policy $\pi$:

$$\begin{aligned} Q_\pi(s_t, a_t) =& \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots \mid s_t, a_t] && (2) \\ =& \mathbb{E}_\pi[r_t + \gamma Q_\pi(s_{t+1}, a_{t+1}) \mid s_t, a_t] && (3) \end{aligned}$$

- Under $\pi^*$, we denote the corresponding action-value function as $Q^*$:

$$Q^*(s_t, a_t) = \mathbb{E}_{\pi^*}[r_t + \gamma Q^*(s_{t+1}, a_{t+1}) \mid s_t, a_t] \quad (4)$$

## Q-Learning

- Q-Learning is a RL algorithm that learns the $Q^*$ function, then deduces the optimal policy $\pi^*$ by acting greedily based on $Q^*$:

$$\pi^*(s_t) = \text{argmax}_a Q^*(s_t, a) \tag{5}$$

- Q-Learning has the following two key properties:
  - **Model-free**: No explicit modeling of environment.
  - **Value-based**: Learns state-action values, then derives policy.
- Q-Learning is appealing for routing in ad-hoc networks:
  - Ad-hoc networks constantly change due to node mobility. Model-free algorithms allow bypassing the MDP state transition estimation.
  - In routing, actions are the selection of next hop. Q-Learning directly estimates the value of each choice, then select the best hop.

## Classic Q-Learning

**Algorithm 1** Classic Q-Learning

```
1: $\forall s, a$, randomly initialize $Q(s, a)$
2: Set $Q(s_{terminal}, a) = 0 : \quad \forall a$
3: for $i = 1 \rightarrow$ Number_of_Episodes do
4:     Initialize $s_t \leftarrow s_0$
5:     while Episode_Not_Finished do
6:         $\pi(s_t) = \begin{cases} \text{argmax}_a Q(s_t, a) & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$
7:         Generate $a_t \leftarrow \pi(s_t)$, then execute it. Observe $r_t$ and $s_{t+1}$
8:         $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_t + \gamma \text{max}_a Q(s_{t+1}, a) - Q(s_t, a_t)]$
9:         $s_t \leftarrow s_{t+1}$
10:     end while
11: end for
```

Note: $\alpha_t$ is the learning rate. Typically, $\epsilon \rightarrow 0$ as training goes on.

## Deep Q-Learning (DQL)

- In classical Q-Learning, $Q(s, a)$ is stored in tabular form, which becomes intractable for large MDPs or MDPs with continuous states.
- Deep Q-Learning is an algorithm using a deep neural network (referred to as the Deep $Q$ Network (DQN)) to approximate the $Q$ function.
- To train a DQN, we minimize the loss for each state-action $(s_t, a_t)$:

$$\left(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right)^2 \qquad (6)$$

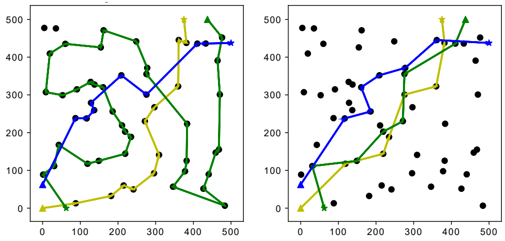
- DQN enjoys generalization ability to unseen state-action pairs.

## DQL for Routing in Ad-Hoc Networks

- Routing is about making decisions in response to local environment for optimizing a global reward, so it is well suited for DQL.
- However, prior works on using DQL for routing assume:
  1. Network model with fixed set of connections;
     *– Does not account for physical-layer SINR based QoS.*
  2. A distinct agent is trained for each node in the network;
     *– Limited scalability and generalization capability.*
  3. The same network is used for both training and testing;
     *– Limited ability to adapt to network changes*
- This work: Same agent for all flows based on physical-layer attributes!

## Ad-Hoc Network Routing Problem

- Consider a wireless ad-hoc network with $N$ mobile nodes, and $M$ data flows, each with a source and a destination at fixed locations.



- Action: To choose an ordered list of intermediary relay nodes.
- State: Network configuration, e.g., pathloss and interference pattern.
- Reward: End-to-end throughput, which depends on bottleneck link.

## Physical Layer Attributes

- The link rate from node $i$ to node $j$:

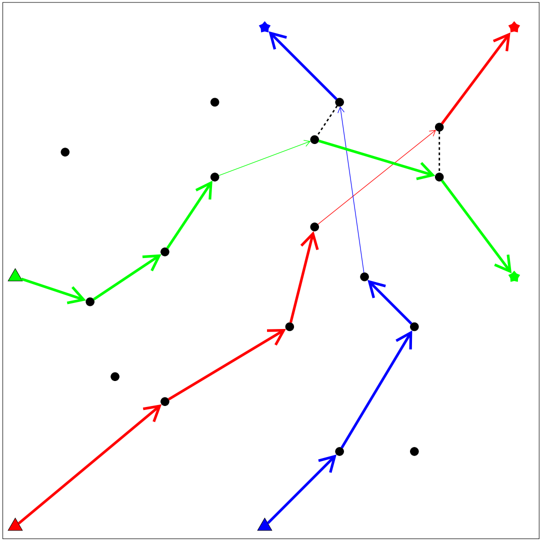
$$R_{(i,j)} = W \log \left( 1 + \frac{|h_{ij}|^2 p_i x_i}{\sum_{\substack{k \neq i,j \\ k \in \mathcal{T} \cup \mathcal{N}}} |h_{kj}|^2 p_k x_k + \sigma^2} \right) \tag{7}$$

  where $x_i \in \{0, 1\}$ indicates whether node $n_i$ is transmitting or idle as determined by the routing solution.

- For a data flow $f$ taking the route $n_1 \rightarrow \dots n_i \rightarrow \dots n_r$, the overall throughput is determined by its bottleneck rate:

$$R_f = \min_{i=1,2,\dots,r-1} R_{(n_i, n_{i+1})} \tag{8}$$

## Bottlenecks in Routing



## Network Objective

- Multiple flows in a network compete with each other:
  - A node cannot serve as a relay for two flows;
  - The flows produce interference with each other.
- Possible network objective functions:

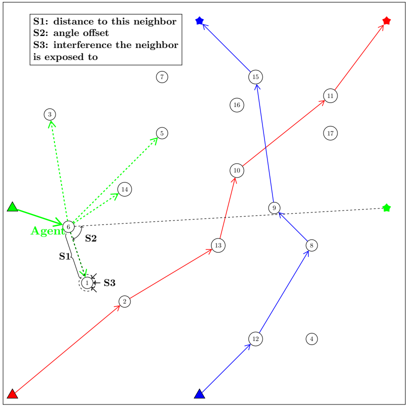
$$\text{Objective A: Sum-Rate} \quad \sum_{f\in\mathcal{F}} R_f \tag{9}$$

$$\text{Objective B: Min-Rate} \quad \min_{f\in\mathcal{F}} R_f \tag{10}$$

- Optimizing a single flow is already challenging: Discrete & Nonconvex
- Optimizing multiple flows is even more so due to their interactions.
- This work adopts a sequential order for establishing routes.

## Associate an Agent at Frontier Node of Each Flow

- The agent moves along the route to decide the best next hop.


S1: distance to this neighbor
S2: angle offset
S3: interference the neighbor is exposed to
Agent
S1
S2
S3
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

- The same agent is used for all nodes along the route and for all flows!

## States and Actions

- **Action space:**
  - Consider $c$ nearest neighbors as candidates for next-hop.
  - One action for reprobing: if none of the $c$ strongest neighbors is a suitable next hop, it proceeds to probe the next $c$ strongest neighbors.
- **State space:** The agent gathers information for each $c$ neighbors:
  1. The distance between the neighbor and the frontier node.
  2. The angle between direction to neighbor and direction to destination.
  3. The total interference the neighbor is exposed to.

## Novel Reward

- Bottleneck throughput of the route is not known until the last hop!
- Further, the reward could be determined by an earlier link, making the rewards (and Q-values) independent of the agent's future actions.
- Instead, we propose the novel Q-value definition as the **bottleneck link rate from that node *onwards* in the route**:

$$\widetilde{Q}(s_k, a_k) = \min_{i=k\ldots h-1} R_{(n_i, n_{i+1})} \tag{11}$$

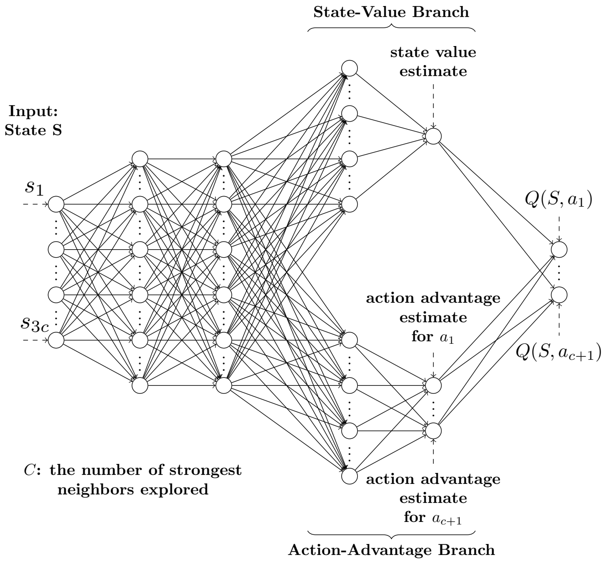
  - It incorporates the future information (and no past information).
  - It is a metric for determining optimal actions.
  - Unlike in usual Q-learning, it does not stem from a reward definition.

- Note that $\widetilde{Q}$ can be determined only after we complete the route.

## Deep Q-Learning for Routing

- Since the states are continuous variables, we utilize deep Q-Learning so the agent can generalize over unseen portions of the state space.
  - The input is a vector of length $3c$ (three features for each neighbor).
  - The inputs are processed by fully connected layers with nonlinearities.
  - We adopt state-of-the-art dueling-DQN network architecture.
  - The output is the Q-value for different actions.
- The dueling-DQN is trained using the experience-replay technique.
  - We only train the agent on one data flow, while fixing other flows.
  - Initially, random explorations are performed and stored in replay buffer.
  - Then, the agent follows the *$\epsilon$-greedy* policy to gather more experiences. If the selected node is not within $c$ strongest neighbors, we get reprobe.
  - Objective is to predict the correct $\tilde{Q}(s, a)$, (i.e., future bottleneck rate).

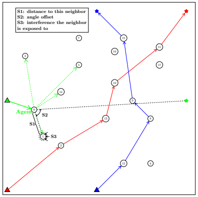
## Dueling DQN Agent

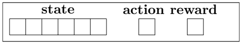

State-Value Branch
state value estimate
Input: State S
$s_1$
$s_{3c}$
$Q(S, a_1)$
$Q(S, a_{c+1})$
action advantage estimate for $a_1$
action advantage estimate for $a_{c+1}$
$C$: the number of strongest neighbors explored
Action-Advantage Branch

## DQN Training with Replay Buffer

**States:** Physical-Layer Attributes
**Action:** Routing Choices
**Reward:** Future Bottleneck Rate

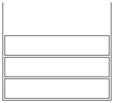
**Gather Experience**


S1: distance to this neighbor
S2: angle offset
S3: interference the neighbor is exposed to
Agent
S1
S2
S3



**Store**

**Act (with $\epsilon$-greedy)**

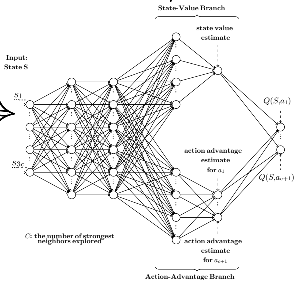


**Replay Buffer**

(FIFO, size 30K)

**Sample Experience to Train**

(mini-batch, size 1K)


State-Value Branch
state value estimate
Input: State S
$S1$
$S3c$
$Q(S,a_1)$
$Q(S,a_{c+1})$
action advantage estimate for $a_1$
action advantage estimate for $a_{c+1}$
$C$: the number of strongest neighbors explored
Action-Advantage Branch

**Same Agent for All Nodes in All Flows!**

## Agent Policy Enhancements

- From domain knowledge, we propose enhancements to agent's policy:

  1. If the agent chooses a neighbor that does not have the strongest channel from the frontier node, then all neighbors with stronger channels are excluded from future consideration for the next hop.
  2. If the destination node is within the agent's $c$ strongest neighbors, then the agent would not choose any neighbor which has a weaker channel.

- These are to prevent non-essential back-and-forth hops.

## Ad-Hoc Wireless Network Model

- Wireless ad-hoc network in 500×500$m^2$ area with $M = 3$ data flows.
- Full frequency reuse with 5MHz bandwidth at 2.4GHz carrier frequency; 1.5m antenna height and 2.5dB antenna gain. Full duplex.
- Additive white Gaussian noise at -150dBm/Hz.
- Max transmit power is set to be constant across each link at 30dBm.
- Short-range outdoor model ITU-1411 distance-dependent pathloss.
- The density of nodes in different regions may be different: nine sub-regions, with $(5, 10, 7, 6, 7, 4, 8, 3, 6)$ nodes in each sub-region.

## Neural Network Configuration

- The agent can explore $c = 6$ strongest neighbors.
- The specifications of neural network layers are as following:

| **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) |

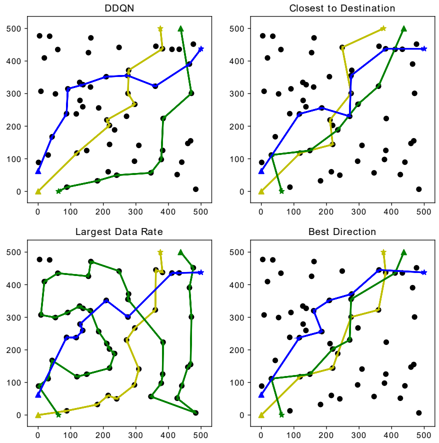
- We randomly generate 520K ad-hoc network layouts to train our agent: 20K are used for random exploration to generate initial experience; 500K are used to train the agent using the $\epsilon$-greedy policy.

## Sum-Rate and Min-Rate Results

Table: 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 |

## Analyzing the Routing Strategies



Our agent spreads out all data flows to mitigate mutual interference, while still maintaining strong and properly directed links to form the routes.

## Generalization to Larger Ad-hoc Networks

We directly use the same model previously trained under the original setting for the routing of 10 data flows in an area of 1.5km x 1.5km.

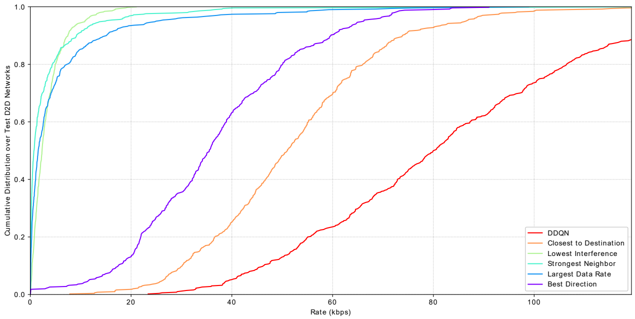


Figure: Cumulative distribution function of sum rate in 500 large-scale networks.

## Conclusion

- We propose 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 definition, we have arrived at a highly scalable routing algorithm.
- The agent can be reused among all data flows, adapted to the varying network layout characteristics, and generalized to large-scale networks.
- Our hope is to show eventually that deep Q-learning can be used to tackle the fundamental spectrum sharing problem in dense networks.

[Reference] Wei Cui and Wei Yu, "Scalable Reinforcement Learning for Routing in Ad Hoc Networks Based on Physical-Layer Attributes", submitted to IEEE ICASSP, 2021.