Data-Driven Stochastic Network Control via Reinforcement Learning

Qiaomin Xie, Cornell ORIE
YEQT, June 9th 2021


 Longer talks for 1 & 2: In SIGMETRICS workshop “Reinforcement Learning in Networks and Queues”, June 14, 2021
Example I: Multi-Class Single Server

- A discrete-time system with two infinite-buffer queues
  - Unbounded state space: \( q = (q_1, q_2) \in \mathbb{N} \times \mathbb{N} \)

- Scheduling decision/action
  - \( A = \{1, 2\} \), i.e., which queue to serve

- Goal: minimize average total queue length (i.e., delay)
Example II: Multi-Class Parallel-Servers

Parallel server scheduling

Input 1
- $\lambda_{1,1} = 0.4$
- $\lambda_{1,2} = 0.4$

Input 2
- $\lambda_{2,1} = 0.2$
- $\lambda_{2,2} = 0.1$

Matching

Switch scheduling

Routing/load balancing
Challenges

"Model-driven" approach
1. Accurate stochastic modeling of system
2. Rely on intuition or a flash of genius to guess a good algorithm
3. Test/tune the algorithm
4. Prove performance guarantees

Challenge 1: Lack of accurate models
- Unknown system parameters
- Time-varying dynamics

Challenge 2: Optimal policies difficult to find
- Even for simple, known models
- More so for: jobs with multiple dependent tasks, heterogeneity of servers/jobs, general service time, etc.
Data-Driven Approach

- Opportunity: availability of fine-grained data or system-level simulators
- A data-driven framework: **Reinforcement Learning** (RL)

![Diagram]

- Challenge 1: Lack of accurate models
  - **Learn system dynamics from data**

- Challenge 2: Optimal policies difficult to find
  - **Discover new policies**
RL for Learning to Schedule

- Learn to schedule purely from data
  - System model unknown

- Optimize a given criterion
  - e.g., minimize average/discounted queue lengths

- Key characteristic: unbounded state space
  - e.g., \( q = (q_1, q_2) \in \mathbb{N} \times \mathbb{N} \)
Challenges of Unbounded State Space

- Insufficiency of *offline*-training-then-deploy
  - Using finite samples
  - Reach a previously *unobserved state*
  - Might have undesirable behavior
    - e.g., serving an empty queue while the other queue is large; assign slow server to busy queue

- Require *online training*: decide action when encountering new states

![Diagram of an unobserved state](image)
Summary of Our Results

- A notion of *stability* to quantify “goodness” of RL algorithm
  - Applies to general systems with unbounded state space
  - Stability provides a first-order optimality

- An *online* RL algorithm that achieves stability
  - Sample complexity bounds

- From stability to optimality
Markov Decision Process (MDP)

- Infinite horizon discounted MDP: \((S, A, p, R, \gamma)\)
  - \(S\): **unbounded** state space
  - \(A\): **finite** action space
  - \(p(s'|s, a)\): transition kernel
  - \(R(s, a)\): one-stage reward
  - \(\gamma \in (0,1)\): discount factor

- (Stochastic) Policy \(\pi\): \(S \rightarrow \Delta(A)\)
- State-action value function (Q-function)

\[
Q^\pi(s, a) = E_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a \right]
\]

- Optimal Q-function

\[
Q^*(s, a) = \max_\pi Q^\pi(s, a)
\]
Minimizing queue length requires keeping queue length finite. **Stability** is a necessary first step towards optimality.

**Definition (Stability)**

We call a policy \( \{\pi_t\} \) stable, if \( \forall \theta \in (0,1) \), there exists a bounded set \( S(\theta) \subset S \) s.t.

1. **Boundedness:**
   \[
   \liminf_{t \to \infty} \mathbb{P}(s_t \in S(\theta) | s_0 = s) \geq 1 - \theta, \forall s \in S.
   \]

2. **Recurrence:**
   Let \( T(s, t, \theta) = \min\{k \geq 0: s_{t+k} \in S(\theta) | s_t = s\} \),
   \[
   \sup_t \mathbb{E}[T(s, t, \theta) | s_t = s] < \infty, \forall s \in S.
   \]
Question: For unbounded state space, how to learn a stable policy in a data-driven manner?

We use a Monte-Carlo simulation and search method.
A Monte Carlo Approach

- At each time step $t$
  - Query a Monte Carlo (MC) oracle
    - Input: state $s_t$
    - Output: a probability distribution over actions $\mu_t$
  - Take action $a_t \sim \mu_t$ and reach state $s_{t+1}$
A Monte Carlo Approach

- At each time step $t$
  - Query a Monte Carlo (MC) oracle
    - Input: state $s_t$
    - Output: a probability distribution over actions $\mu_t$
  - Take action $a_t \sim \mu_t$ and reach state $s_{t+1}$

Using finite samples

\[ a_0 \sim \mu_0 \rightarrow s_0 \]
\[ a_1 \sim \mu_1 \rightarrow s_1 \]
\[ \vdots \]
\[ a_t \sim \mu_t \rightarrow s_t \]
Monte Carlo Oracles

- Sparse-Sampling Oracle [Kearns-Mansour-Ng, ‘02]
- Monte Carlo Tree Search [Kocsis-Szepesvari, ‘06] [Shah-X-Xu, ’20]
- Oracle Approximation Guarantees for MCTS

**Theorem** [Shah-X-Xu ‘20]

With appropriate parameters, with probability at least $1 - \delta$,

$$|\hat{Q}(s, a) - Q^*(s, a)| \leq \varepsilon, \forall a.$$  

**Corollary**

With softmax policy $\mu(s, a) \propto e^{\hat{Q}(s,a)/\tau}$, we have

$$||\mu(s,:) - \pi^*(s)||_{TV} \leq c_1 \frac{e^{\varepsilon/\tau} - 1}{e^{\varepsilon/\tau} + 1} + c_2 e^{-\frac{c_3}{\tau}},$$

where $c_1, c_2, c_3 > 0$ are constants.

Can be small with small $\varepsilon$ and $\tau$
From Approximation to Stability

Questions:
- When is the policy $\{\mu_t\}$ stable?
- What is the sample complexity of each oracle query?

When is stability possible
- The Markov chain $M^*$ under the optimal policy $\pi^*$ is positive recurrent
- A necessary and sufficient condition for positive recurrence of a Markov chain is the existence of a Lyapunov function\(^1\)
- We assume that $M^*$ satisfies a Lyapunov Drift Condition

Assumption: Lyapunov Function

**Assumption**

There exists a function $L: S \rightarrow \mathbb{R}^+$ such that the Markov chain under $\pi^*$ satisfies that

1. change of $L$ for any transition is bounded,
2. has a negative drift $-\alpha$ when $L(s) > B$.

**Example**: single-server two-queue system

- Optimal policy $\pi^*$: $c\mu$ rule
- $L(q_1, q_2) = \frac{q_1}{\mu_1} + \frac{q_2}{\mu_2}$ satisfies the assumption

**Remark**: Algorithm *not* need to know the Lyapunov function
Main Results

Theorem (Stability)

Under the Lyapunov assumption, with proper parameters, the resulting policy \( \{\mu_t\} \) sequence is stable.

Theorem (Sample complexity)

Sample complexity per time step for small \( \alpha \) scales as

\[
O \left( \left( \frac{1}{\alpha^4} \log \frac{1}{\alpha} \right)^{\log \frac{1}{\alpha}} \right)
\]
Refinements

- **Adaptive version**
  - Automatically discover the appropriate tuning parameters $\varepsilon, \tau$
  - Using a statistical hypothesis test for growing queue length

- **Sample-efficient version**
  - Small $\alpha$: high load regime in queueing
  - From super-polynomial to polynomial

$$O \left( \left( \frac{1}{\alpha^4 \log \frac{1}{\alpha}} \right)^{\log \frac{1}{\alpha}} \right) \rightarrow O \left( \frac{1}{\alpha^{2d+4}} \right)$$
From Stability to Optimality

- Given a **stable** policy, can we learn the **optimal** policy? *Yes!*

**Our Approach:**

For the **outside** states
- Apply default **stable** policy $\pi^{\text{stable}}$
  - e.g. policy we already know
  - Or, use stable RL
- Cost for other states can be controlled

For the **truncated state space**
- Apply model-based RL policy $\pi^{\text{RL}}$
- Converge to optimal policy $\tilde{\pi}^*$

20
Theoretical Guarantee

**Theorem** [Liu-X-Modiano, ‘19]

Under Lyapunov assumption, with state space truncated at $U$, the average queue length of our algorithm approaches the optimal queue length exponentially fast.

\[
\lim_{T \to \infty} \frac{\mathbb{E} \left[ \sum_{t=1}^{T} \sum_i Q_i(t) \right]}{T} = \rho^* + \mathcal{O} \left( \frac{1}{\exp(U)} \right)
\]
Simulation: Scheduling with Connectivity

- $\pi_0$: Serve-Longest-Connected Queue
- $U=5$: converge to 3.93
- $U=10$: converge to 3.75


RL as Performance Benchmarks

- RL methods can achieve state-of-art performance for complex stochastic network control problems
- Switch Scheduling:
RL as Performance Benchmarks

- OR-Suites: OR version of OpenAI gym
  - Ongoing with S. Sinclair, C. Lee Yu and S. Banerjee
  - RL Benchmarks for operations research applications
    - Rideshare matching
    - Ambulance routing
    - Revenue management
    - Foodbank allocation
    - ...

- Demo at RLNQ Workshop
Summary

- RL for stochastic network control with unbounded state space
  - A notion of stability suitable for RL setting
  - Achieve stability by Monte Carlo planning
  - From stability to optimality

- Future work
  - Combined with function approximation and policy optimization
  - Complex stochastic networks: synthesizing model-driven & data-driven approaches
Thank you!