Better than the Best: Gradient-based Improper Reinforcement Learning for Network Scheduling

Mohammadi Zaki, Avi Mohan, Aditya Gopalan and Shie Mannor

ABSTRACT

We consider the problem of scheduling in constrained queueing networks with a view to minimizing packet delay. We formulate a novel top down approach to scheduling where, given an unknown network and a set of scheduling policies, we use a policy gradient based reinforcement learning algorithm that produces a scheduler that performs better than the available atomic policies. We derive convergence results and analyze finite time performance of the algorithm. Simulation results show that the algorithm performs well even when the arrival rates are nonstationary and can stabilize the system even when the constituent policies are unstable. Link to paper: `https://arxiv.org/pdf/2102.08201.pdf`

1. INTRODUCTION

The design of communication networks has traditionally involved fine-grained modeling of traffic and network characteristics, followed by devising scheduling and routing protocols optimized for the models. This constituted a "bottom-up" approach, with resource allocation algorithms being tightly coupled to network model assumptions, and worked very well with the relatively simple requirements of yesteryear networks comprising mostly homogeneous traffic sources. It has yielded a readily available storehouse of design principles and rules of thumb that can be used to generate with little effort, a menu of schedulers with desirable properties, e.g., MaxWeight and variants [5, 3].

Modern communication systems, on the other hand, are becoming increasingly complex, and are required to handle multiple types of traffic with widely varying characteristics (such as arrival rates and service times). This, coupled with the need for rapid network deployment, render such a granular model-based and bottom-up approach infeasible. At the same time, we would prefer to retain the favorable principles underlying the design of existing scheduling algorithms and not give them up altogether. In this regard, this paper advocates a novel "top-down" approach to designing effective and adaptable scheduling strategies, in which, given an unknown network and a set of scheduling policies, we aim to learn a scheduler that is either better than all the atomic policies or as good as the best atomic policy (which is a priori unknown for the network setting at hand). We achieve this using a gradient-based optimization algorithm over an improper mixture class of policies constructed using the given base controllers. We employ tools from recent analyses of policy gradient methods to derive convergence results and analyze finite-time performance of the proposed algorithm in this new setting. Simulation results show that the algorithm performs well even when the arrival rates are nonstationary, and can stabilize the system even when the constituent policies are unstable.

Related work is discussed in detail in Sec. 1.1 of our tech report [6] and is omitted here due to paucity of space.

2. SYSTEM SETTING

To describe our approach in detail, we focus on the well-known setting of a single server attending to N queues in discrete time, where for example, each queue models packets waiting on a communication link. We note, however, that our algorithmic framework applies more generally to policy optimization for any Markov decision process (MDP), including one with continuous state/action spaces, as long as a set of policies is given for it (please refer to [6] for the complete formulation). The server decides which queues are to be scheduled for service in each slot, based on service constraints (e.g., at most 1 queue to be scheduled each time). The indicator random variable $D_i(t)$ denotes whether queue i is scheduled in slot t or not. For concreteness, we also assume (1) IID Bernoulli(λ_i) arrivals $\{A_i(t)\}_{t\in\mathbb{N}}$ to each queue i, (2) deterministic, single-packet service for each queue when scheduled, and (3) a scheduling constraint of at most 1 queue per time slot. Note, however, that our policy optimization approach extends to general arrival processes or interference graphs (i.e., scheduling constraints). Hence, the queue lengths evolve as $Q_i(t+1) = (Q_i(t) - D_i(t))^+ + A_i(t+1),\ i \in [N],\ t \in \mathbb{N}$, where $(x)^+ := \max\{0, x\},\ \forall\ x \in \mathbb{R}$. Note that the arrival rates $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_N]$ are a priori unknown to the scheduler/learner. The learner (i.e., scheduling algorithm at the server) needs to decide which of the N queues it intends to serve in a given slot. The server's decision at each slot can be denoted by the vector $\mathbf{D}(t) = (D_i(t))_{i\in N}$ taking values in the action space $\mathcal{A} := \{[0, \cdots, 0], [1, 0, \cdots, 0], [0, 0, \cdots, 1]\}$, where a "1" denotes service and a "0" denotes lack thereof.

Let H_t denote the state-action history (historical trajectory) until time t, and $\mathcal{P}(\mathcal{A})$ the space of all probability distributions on $\mathcal{A}$. We aim to find a policy $\pi = [\pi_1, \pi_2, \cdots]$, where $\pi_t : H_t \to \mathcal{P}(\mathcal{A})$, to minimize the ∞-horizon discounted system backlog given by

$$J_\pi(\mathbf{q}) := \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^{N} Q_i(t) \mid \mathbf{Q}(0) = \mathbf{q} \right]. \qquad (1)$$

Note that we are using system backlogs (queue lengths) as a proxy for packet delays as is commonly done; a more finer performance criterion involving the actual packet delays can also be optimized if the MDP is suitably redefined. Any policy π with $J_\pi(\mathbf{Q}(0)) < \infty$, $\forall \mathbf{Q}(0) \in \mathbb{Z}_+^2$ is said to be *stabilizing* (or, equivalently, a *stable* policy). The capacity region [5] of this network can be seen to be $\left\{\boldsymbol{\lambda} \in \mathbb{R}_+^N \mid \sum_{i\in[N]} \lambda_i < 1\right\}$.

This problem can be viewed as one of finding an ∞-horizon, γ-discounted *reward* optimal policy in the MDP with state space $\mathcal{S} = \mathbb{N}^N$, i.e., all possible values of the queue lengths $\mathbf{Q}(t) \equiv (Q_i(t))_{i\in N}$, action space $\mathcal{P}(\mathcal{A})$, single stage reward $r(\mathbf{Q}(t), \mathbf{D}(t)) = -\sum_{i=1}^N Q_i(t)$, and an appropriately defined probability transition kernel $\mathtt{P}$ following the Bernoulli arrival process [6]. In keeping with standard reinforcement learning parlance, we will refer to the negative discounted system backlog $-J_\pi(\cdot)$ as the *value function* $V^\pi(\cdot)$ of policy π, to be maximized over policies π. Moreover, due to the Markov nature of the system, we consider only policies that depend on the current state, i.e., $\pi_t : \mathbf{Q}(t) \to \mathcal{P}(\mathcal{A})$.

Improper Learning. We assume that we are provided with a finite number of controllers/policies $\mathcal{C} := \{K_1, \cdots, K_M\}$. We aim to identify the best policy for the given queueing network within a class, i.e.,

$$\pi^* = \underset{\pi\in\mathcal{I}_{soft}(\mathcal{C})}{\operatorname{argmin}} \; V^\pi(\rho), \tag{2}$$

where $\mathcal{I}_{soft}(\mathcal{C})$ is a parameterized, improper, policy class that we define as follows.

The Softmax Policy Class. Each policy in the softmax policy class $\mathcal{I}_{soft}(\mathcal{C})$ is parameterized by weights $\theta := [\theta_1, \cdots, \theta_M] \in \mathbb{R}^M$. The policy $\pi_\theta \in \mathcal{I}_{soft}(\mathcal{C})$, given a state $s \in \mathcal{S}$, plays an action by (a) first choosing a controller drawn from $\mathtt{softmax}(\theta)$, i.e., the probability of choosing controller K_m is given by,

$$\pi_\theta(m) := \frac{e^{\theta_m}}{\sum_{m'=1}^{M} e^{\theta_{m'}}}, \tag{3}$$

and (b) then choosing an action by applying the sampled controller at the state s. Note, therefore, that in every round, our algorithm decides which action to apply only *through* the controller sampled in the first step of that round. In the rest of the paper, we will deal exclusively with a fixed base policy class $\mathcal{C}$ and the resultant $\mathcal{I}_{soft}(\mathcal{C})$. We use the notation $\pi_\theta(a|s)$ for any $a \in \mathcal{A}$ and $s \in \mathcal{S}$ to denote the probability with which the softmax policy π_θ, as defined above, chooses action a in state s. Hence, we have that for any $\theta \in \mathbb{R}^m$,

$$\pi_\theta(a|s) = \sum_{m=1}^{M} \pi_\theta(m) K_m(s,a), \tag{4}$$

where $K_m(s,a)$ is the probability with which the policy K_m plays a in state s. Since we deal with gradient-based methods in the sequel, we define the *value gradient* of policy $\pi_\theta \in \mathcal{I}_{soft}$, by $\nabla_\theta V^{\pi_\theta}$.

3. SCHEDULING VIA POLICY GRADIENTS

The Policy Gradient Approach. The Policy Gradient (PG) method has, following stunning success with applications such as game playing, has become a cornerstone of reinforcement learning [4]. In general, PG methods involve parameterizing the control policy and optimizing the parameter using a gradient ascent algorithm of the form

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \nabla_{\theta_t} V^{\pi_{\theta_t}} \tag{5}$$

When the value function and its gradient are computable in closed form, we propose an algorithm, *SoftMax PG*, that provably converges to the best parameter, $\boldsymbol{\theta}^*$, within $\mathcal{I}_{soft}(\mathcal{C})$.

Algorithm 1 Softmax Policy Gradient (SoftMax PG)

Input: learning rate $\eta > 0$, initial state distribution μ
Initialize each $\theta_m^1 = 1$, for all $m \in [M]$, $s_1 \sim \mu$
for $t = 1$ **to** T **do**
  Choose controller $m_t \sim \pi_t$.
  Play action $a_t \sim K_{m_t}(s_t, :)$.
  Observe $s_{t+1} \sim \mathtt{P}(.|s_t, a_t)$.
  Update: $\theta_{t+1} = \theta_t + \eta . \nabla_{\theta_t} V^{\pi_{\theta_t}}$.
end for

3.1 Convergence of SoftMax PG

For a given policy π and initial state distribution μ, the quantity $d_\mu^\pi(\cdot) := \mathbb{E}_{\mathbf{Q}_0\sim\mu}\left[(1-\gamma)\sum_{t=0}^{\infty} \mathbb{P}\left[\mathbf{Q}_t = \cdot \;\middle|\; \mathbf{Q}_o, \pi, \mathtt{P}\right]\right]$, defines a distribution over $\mathcal{S}$, called the *discounted state visitation measure.*

THEOREM 3.1 (RATE OF CONVERGENCE). *Let $|\mathcal{S}\times\mathcal{A}| < \infty$, $\mathbf{Q}(0) \sim \mu$ and assume that the scheduler is provided with M stationary controllers. With $\{\theta_t\}_{t\geqslant 1}$ generated as in Algorithm 1 and using a learning rate $\eta = \frac{(1-\gamma)^2}{7\gamma^2+4\gamma+5}$, for all $t \geqslant 1$,*

$$V^{\pi_{\theta_t}}(\rho) - V^{\pi^*}(\rho) \leqslant \frac{1}{t} M \left(\frac{7\gamma^2+4\gamma+5}{c^2(1-\gamma)^3}\right) \left\|\frac{d_\mu^{\pi^*}}{\mu}\right\|_\infty^2 \left\|\frac{1}{\mu}\right\|_\infty .$$

Here, $c = \inf_{t\geqslant 1} \min\{\pi_{\theta_t}(m) : m \in [M], \pi^*(m) > 0\}$ is the minimum probability that Algorithm 1 puts on the controllers on which the best mixture π^* is supported.

3.2 Estimating Value Gradients

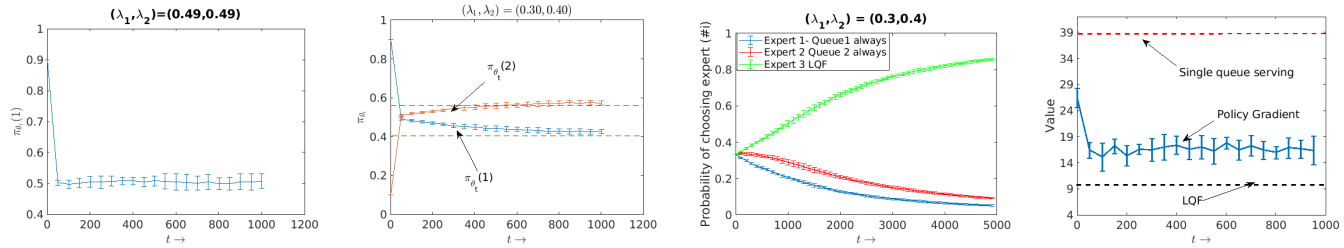
In most RL problems, neither value functions nor value gradients are available in closed-form. To expanding the applicability of SoftMax PG to such situations, we propose a gradient estimation subroutine called *GradEst* (Algorithm 2). This uses a combination of (1) rollouts to estimate the value of the current (improper) policy and (2) a stochastic perturbation-based approach to estimate its value gradient.

Specifically, in order to estimate the value gradient, we use the approach of Flaxman et al [1], noting that for $V : \mathbb{R}^M \to \mathbb{R}$, the gradient $\nabla V(\theta) \approx \mathbb{E}\left[(V(\theta+\alpha.u) - V(\theta))\, u\right].\frac{M}{\alpha}$. where $\alpha \in (0,1)$. If u is chosen to be uniformly random on unit sphere, the second term is zero, i.e.,
$\mathbb{E}\left[(V(\theta+\alpha u) - V(\theta))\, u\right].\frac{M}{\alpha} = \mathbb{E}\left[(V(\theta+\alpha u))\, u\right].\frac{M}{\alpha}$.

The expression above requires evaluation of the value function at the point $(\theta + \alpha.u)$. Since the value function may not be explicitly computable, we employ rollouts for its evaluation.

4. EXPERIMENTS

In this section, we show the efficacy of our policy gradient algorithm through simulations in multiple scenarios. We



(a) $\boldsymbol{\lambda} = (0.49, 0.49)$, $\mathcal{C}$ = {'always serve 1', 'always serve 2'}

(b) $\boldsymbol{\lambda} = (0.3, 0.4)$, $\mathcal{C}$ = {'always serve 1', 'always serve 2'}

(c) $\boldsymbol{\lambda} = (0.3, 0.4)$, $\mathcal{C}$ = {'serve 1', 'serve 2', LQF}

(d) Expected discounted queue length cost for 1c

Figure 1: Softmax policy gradient with gradient estimation finds the best mixture policy for various base policies in a 2-queue network.

Algorithm 2 GradEst

Input: Policy parameters θ, parameter $\alpha > 0$.
for $i = 1$ **to** `#runs` **do**
  $u^i \sim Unif(\mathbb{S}^{M-1})$.
  $\theta_\alpha = \theta + \alpha.u^i$
  $\pi_\alpha = \texttt{softmax}(\theta_\alpha)$
  for $l = 1$ **to** `#rollouts` **do**
    Generate trajectory according to the policy π_α : $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{lt}, a_{lt}, r_{lt})$
    $\texttt{reward}(l) = \sum_{j=0}^{lt} \gamma^j r_j$
  end for
  $\texttt{mr}(i) = \texttt{mean}(\texttt{reward})$
end for
$\texttt{GradValue} = \frac{1}{\texttt{\#runs}} \sum_{i=1}^{\texttt{\#runs}} \texttt{mr}(i).u^i.\frac{M}{\alpha}$.
return `GradValue`

study the performance of softmax PG with gradient estimation (GradEst) over two different settings: (1) when the packet arrival rates, and therefore the optimal controller, are fixed and (2) where they are time varying. In all our experiments, we consider a system with $N = 2$ queues and a fully connected interference graph. We provide all details about hyperparameters in [6, Sec. E].

4.1 Constant Arrival Rates

In this case, we first simulate a network where the the optimal policy ($\pi_{\boldsymbol{\theta}^*}$) is a strict improper combination of the available controllers and later, a network where it is at a corner point, i.e., one of the available controllers itself is optimal. Our simulations show that in both the cases, softmax PG converges to the correct controller distribution in $\mathcal{I}_{soft}$.

The scheduler is given two base/atomic controllers $\mathcal{C} := \{K_1, K_2\}$, i.e. $M = 2$. Controller K_i serves Queue i with probability 1, $i = 1, 2$. As can be seen in Fig. 1a when $\boldsymbol{\lambda} = [0.49, 0.49]$, softmax PG converges to the improper mixture policy that serves each queue independently with probability $[0.5, 0.5]$, which is the delay-optimal controller in $\mathcal{I}_{soft}(\mathcal{C})$. Interestingly, the mixture stabilizes the system whereas both base controllers lead to instability because of insufficient service to some queue. Fig. 1b shows that with unequal arrival rates too, Softmax-PG with GradEst quickly converges to the correct improper combination.

Fig. 1d shows the evolution of the value function (system backlog) of GradEst (blue) compared with those of the base

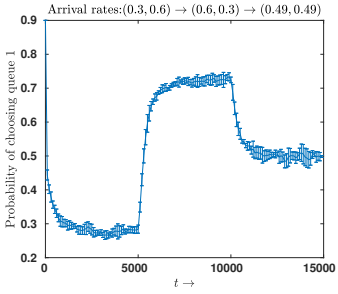


Figure 2: Plot showing that GradEst *adapts* to varying arrival rates over time.

controllers (red) and the *Longest Queue First* policy (LQF) which, as the name suggests, always serves the longest queue in the system (black). LQF, like any work-conserving policy, is known to be delay optimal [2].

Finally, Fig. 1c shows the result of the second experimental setting with three atomic controllers, one of which is delay optimal. The first two are K_1, K_2 as before and the third controller, K_3, is LQF. Notice that K_1, K_2 are both queue length-agnostic, meaning they could attempt to serve empty queues as well. LQF, on the other hand, always and only serves nonempty queues. Hence, in this case the optimal policy is attained at one of the corner points, i.e., $[0, 0, 1]$. The plot shows the GradEst converging to the correct point on the simplex.

4.2 Time-varying Arrival Rates

We consider a modification to the system in Sec. 6 wherein the arrival rates $\boldsymbol{\lambda}$ to the two queues vary over time (adversarially). In particular, $\boldsymbol{\lambda}$ varies from $(0.3, 0.6) \to (0.6, 0.3) \to (0.49, 0.49)$. Our PG algorithm successfully *tracks* this change and *adapts* to the optimal improper stationary policies in each case as shown in Fig. 2. In all three cases a mixed controller is optimal, and is successfully tracked by our PG algorithm.

5. CONCLUSION

Our results show that our new improper learning algorithmic framework is able to efficiently learn optimal mixtures of given policies. This paves the way for (a) building more refined theory towards understanding the convergence behavior of such schemes, and (b) benchmarking it more extensively in diverse RL settings including problems of robotic control, computer game playing, etc. This will form the subject of future work.

6. REFERENCES

[1] Flaxman, A. D., Kalai, A. T., and McMahan, H. B. Online convex optimization in the bandit setting: Gradient descent without a gradient. SODA '05, Society for Industrial and Applied Mathematics, p. 385–394.

[2] Mohan, A., Chattopadhyay, A., and Kumar, A. Hybrid MAC protocols for low-delay scheduling. In *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)* (Los Alamitos, CA, USA, oct 2016), IEEE Computer Society, pp. 47–55.

[3] Shakkottai, S., and Stolyar, A. L. Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *Translations of the American Mathematical Society-Series 2 207* (2002), 185–202.

[4] Sutton, R. S., and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[5] Tassiulas, L., and Ephremides, A. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control 37*, 12 (1992), 1936–1948.

[6] Zaki, M., Mohan, A., Gopalan, A., and Mannor, S. Improper learning with gradient-based policy optimization. *arXiv preprint arXiv:2102.08201* (2021).