Learning Algorithms for Minimizing Queue Length Regret

Thomas Stahlbuhk
Massachusetts Institute of Technology
Cambridge, MA

Brooke Shrader
MIT Lincoln Laboratory
Lexington, MA

Eytan Modiano
Massachusetts Institute of Technology
Cambridge, MA

***Abstract*—We consider a system consisting of a single transmitter and N channels. Packets randomly arrive to the transmitter's queue, and at each time slot a controller can schedule one of the N channels for transmission. The channel's rates are time-varying with unknown statistics and must be learned through observation. Our objective is to minimize the number of packets in the system's queue over T time slots. We define the regret of the system to be the expected difference between the total queue length of a controller that must learn the channels' average rates and a controller that knows the rates, a priori. One approach to solving this problem would be to apply algorithms from the literature that were developed to solve the closely-related stochastic multi-armed bandit problem. However, we show that these methods have $\Omega(\log(T))$ queue length regret. On the other hand, we show that there exists a set of queue-length based policies that are able to obtain order optimal, $O(1)$, regret.**

I. Introduction

In this work, we consider a statistical learning problem that is motivated by the following application. Consider a wireless communication system consisting of a single transmitter and N channels over which the system can communicate. Packets randomly arrive to the transmitter's queue, and at each time step a controller can select one of the N channels to transmit on. The channels' rates vary according to i.i.d. processes with initially unknown statistics. When a channel is selected by the controller, the system can attempt a packet transmission on the channel and use receiver feedback to determine whether the transmission was successful. (If the queue is empty, a dummy/probe packet can be sent to observe a channel.) The objective of the controller is to minimize the packets' delays by quickly identifying the best channel to transmit on.

The channels in the above application behave like servers in a queueing system that, when selected, offer a random amount of service to the queue. Given the above motivation, we consider the problem of identifying the best of N available servers to minimize a queue's backlog. To this end, we associate a unit cost to each time slot that each packet has to wait in the queue. To obtain good performance, the controller must simultaneously schedule the servers (channels) to explore their service rates and also exploit previous observations to minimize the queue backlog. We define the queue length regret as $R^{\pi}(T) \triangleq E\left[\sum_{t=0}^{T-1} Q^{\pi}(t) - \sum_{t=0}^{T-1} Q^{*}(t)\right]$, where $Q^{\pi}(t)$ is the backlog under a learning policy and $Q^{*}(t)$ is the backlog under a controller that knows the best server. Our objective is to find a policy that minimizes this regret.

Our problem is closely related to the stochastic multi-armed bandit problem. In this problem, a player is confronted by a set of N possible actions of which it may select only one at a given time. Each action provides an i.i.d. stochastic reward, but the statistics of the rewards are initially unknown. Over a set of T successive rounds, the player must select from the actions to both explore how much reward each gives as well as exploit its knowledge to focus on the action that appears to give the most reward. Learning policies for solving the multi-armed bandit have long been considered [1]. Historically, the performance of a policy is evaluated using regret, which is defined to be the expected difference between the reward accumulated by the learning policy and a player that knows, a priori, the action with highest mean reward. This quantifies the cost of having to learn the best action. It is well known that there exist policies such that the regret scales on the order of $\log(T)$ and that the order of this bound is tight [2]. In the seminal work of [3], policies for achieving an asymptotically efficient regret rate were derived and subsequent work in [4], [5] have provided simplified policies that are often used in practice.

One approach to solving our problem would be to simply use traditional bandit algorithms.[1] This would maximize the rate of offered service to the queue and, in an infinitely backlogged system, would maximize throughput. The problem with this approach is that it does not exploit the fundamental queueing dynamics of the system. During time periods when the queue is empty, the offered service is unused, and the controller can, therefore, freely poll the servers without hurting its objective. This contrasts with non-empty periods, when exploring can be costly since it potentially uses a suboptimal server. However, a controller cannot restrict its explorations only to time slots when the queue is empty. This is because some servers may have a service rate that is below the packet arrival rate, and if the controller refuses to explore during non-empty periods, it may settle on destabilizing actions that cause the backlog to grow to infinity. What is needed are policies that favor exploration during periods when the queue is empty

This work was sponsored by NSF Grants AST-1547331 and CNS-1701964, and by Army Research Office (ARO) grant number W911NF-17-1-0508. This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.

[1] One can view the bandit's rewards as offered service in our problem.

978-1-5386-4780-6/18/$31.00©2018 IEEE

but perform enough exploration during non-empty periods to maintain stability. In this work, we show that for systems that have at least one server whose service rate is greater than the arrival rate there exist queue-length based policies such that $R^\pi(T) = O(1)$. Likewise, we show that any traditional bandit learning algorithm, when applied to our setting, may have queue length regret $R^\pi(T) = \Omega(\log(T))$.[2]

The problem considered herein is related to the recent work of [6]. The main difference is that in [6], the controller could only obtain observations of the servers during times when the queue was backlogged. As a result, the policies considered in that work were closely related to those in the bandit literature and focused on maximizing offered service. Under these policies, [6] showed that the tail of $E[Q^\pi(t) - Q^*(t)]$ diminishes as $\frac{1}{t}$, implying that $R^\pi(T)$ is logarithmic. In this work, we show that by exploiting empty periods, the queue length regret is bounded.

II. Problem Setup

We consider a system consisting of a single queue and N servers operating over discrete time slots $t = 0, 1, 2, \ldots$.[3] Packets arrive to the queue as a Bernoulli process, $A(t)$, with rate $\lambda \in (0, 1]$ and can be serviced in the next time slot after their arrival. The amount of service that server $i \in [N]$ can provide follows a Bernoulli process $D^i(t)$ with rate μ_i. The arrival process and server processes are assumed to be independent. We refer to server i as stabilizing if $\mu_i > \lambda$; otherwise, we refer to it as non-stabilizing.

At each time slot, a controller can select one of the N servers to provide service to the queue. We denote the controller's choice at time t as $u(t) \in [N]$ and the service offered to the queue as $D(t)$ which is equal to $D^{u(t)}(t)$. Given the above, the queue backlog, $Q(t)$, evolves as

$$Q(t+1) = (Q(t) - D(t))^+ + A(t), \text{ for } t = 0, 1, 2, \ldots$$

where $(x)^+$ is used to denote the maximum of x and 0. We assume $Q(0) = 0$. An instantiation of the above problem is characterized by the tuple $(\lambda, \vec{\mu})$ where $\vec{\mu}$ is the vector of service rates. Let $\mathcal{P}$ be the set of all problems (i.e., tuples).

The controller cannot observe the values of $D^i(t)$ prior to making its decision $u(t)$. It is then clear that the optimal action, that maximizes expected service, is to select the server

$$i^* \triangleq \arg\max_{i \in [N]} \mu_i$$

to provide service. For simplicity, we assume i^* is always unique.[4] In this work, the controller does not a priori know the values of μ_i and must therefore use observations of $D(t)$ to identify i^*. This implies that the service available from server $u(t)$ is revealed to the controller after time t, but the service that would have been available from all other servers remains hidden. Note that the controller can observe $D(t)$ at all times t, even when $Q(t) = 0$.

Our objective is to design a controller that makes decisions as to which server to schedule at each time slot. Denoting the history of all previous arrivals, service opportunities, and decisions as

$$H(t) = (A(0), D(0), u(0), \ldots, A(t-1), D(t-1), u(t-1)),$$

our objective is to find a controller policy $\pi \triangleq f_0, f_1, \ldots$ where f_t is a (possibly random) mapping from history $H(t)$ to decision $u(t)$. We denote the set of all policies as Π.

Define $Q^*(t)$ to be the queue backlog under the controller that always schedules i^* and $Q^\pi(t)$ the backlog under a policy that must learn the service rates. We will analyze the performance of π using the following definition of queue length regret,

$$R^\pi(T) \triangleq E\left[\sum_{t=0}^{T-1} Q^\pi(t) - \sum_{t=0}^{T-1} Q^*(t)\right]. \quad (1)$$

We will often simply refer to this metric as regret in the rest of the paper. Note that $E[Q^\pi(t)] \geq E[Q^*(t)]$ and a policy that minimizes (1) must also minimize $E[\sum_{t=0}^{T-1} Q^\pi(t)]$. This implies that $R^\pi(T)$ is monotonically increasing for all $\pi \in \Pi$. The focus in this work will be on characterizing how regret may scale with time horizon T. To this end, we will consider policies that are independent of T (i.e., do not optimize for a specific time horizon).

III. Regret of Traditional Bandit Algorithms

We establish that the regret of any $\pi \in \Pi$ that is independent of the arrival process $A(t)$ (i.e., f_t is only a function of the previous decisions and observed offered service) must have a regret that grows logarithmically for some $\vec{\mu}$. Note that this result applies to any policy borrowed from the stochastic multi-armed bandit literature that maximizes offered service to the queue without regard to queue state. Without loss of generality, the theorem is established for a system with two servers. The complete proof of the theorem can be found in [7]. Let $\mu_{(j)}$ be the j^{th} server when the servers are sorted by rate.

Theorem 1: For any $\pi \in \Pi$ that is independent of $A(t)$ and $Q(t)$, $\exists \vec{\mu}$ with server rates $\mu_{(1)}$ and $\mu_{(2)}$ ($\mu_{(2)} > \mu_{(1)}$) such that for any $\lambda \in \big(0, \mu_{(2)}\big)$, $R^\pi(T) = \Omega(\log(T))$.

IV. Regret of Queue-Length Based Policies

In this section, we examine the asymptotic scaling of regret $R^\pi(T)$ for policies that make decisions using $Q(t)$ (or equivalently, $A(t)$). We do so by considering the problem for different subsets of $\mathcal{P}$. This section will build to the main result of this work, Theorem 4, which states that there exists a $\pi \in \Pi$ such that for any problem $(\lambda, \vec{\mu})$ with $\mu_{i^*} > \lambda$, $R^\pi(T) = O(1)$.

We begin our analysis in Subsection IV-A under the assumption that every server is stabilizing (i.e., $\mu_i > \lambda$, $\forall i \in [N]$). Under this assumption, the controller does not need to account for the possibility of system instability. As a result, the controller can limit itself to performing exploration only on

[2]Recall, for functions f and g, $f(x) = O(g(x))$ *iff* $\exists M > 0$ and $\exists x_0$ such that $\forall x > x_0$, $|f(x)| \leq Mg(x)$. Likewise, $f(x) = \Omega(g(x))$ *iff* $\exists M > 0$ such that $\forall x_0$, $\exists x > x_0$ with $|f(x)| \geq Mg(x)$.

[3]We assume $N \geq 2$. Otherwise, the problem is trivial.

[4]Then, we let $\mathcal{P}$ be the set of all tuples such that i^* is unique.

time slots in which the queue is empty. During time slots in which the queue is backlogged, the controller will exploit its previously obtained knowledge to schedule the server it believes to be best.

In Subsection IV-B, we allow for both stabilizing and non-stabilizing servers in the system (i.e., we allow $\mu_i \leq \lambda$ for a subset of the servers). However, we will assume that the controller is given a randomized policy that achieves an offered service rate that is greater than the arrival rate to the queue. Note that the controller does not need to learn this policy and can use it to return the queue to the empty state. The given randomized policy is not required to have any relationship to i^* and will not, in general, minimize queue length regret. As a result, to minimize queue length regret, the controller will not want to excessively rely upon it.

In Subsection IV-C, we further relax our assumptions on the problem and require that the controller learn which servers are and are not stabilizing while simultaneously trying to minimize $R^{\pi}(T)$. This will require a policy that does not destabilize the system. To this end, the controller will have to explore the servers' offered service rates during both time slots when the queue is empty and backlogged. Explorations during time slots when the queue is backlogged, in general, waste work and should therefore be performed sparingly. Intuitively, as the controller identifies which subset of servers have service rates $\mu_i > \lambda$, it can focus its explorations on time slots when the queue is empty.

The above three cases build upon one another. The insight from one will point to a policy for the next, and we will therefore analyze the above cases in sequence. Under each of the above assumptions, we will find that there exists a policy such that the regret converges for all $(\lambda, \vec{\mu})$ meeting the assumption. In contrast, in Section V, we show that there does not exist a policy that can achieve convergent regret over the class of all problems for which $\mu_i \leq \lambda$, $\forall i \in [N]$. The complete proof of all lemmas and Theorem 5 can be found in [7].

A. All Servers Are Stabilizing

In this subsection, we assume all servers are stabilizing.

Assumption 1: $(\lambda, \vec{\mu}) \in \mathcal{P}_1 \triangleq \{\mathcal{P} : \mu_i > \lambda, \forall i \in [N]\}$.

Under this assumption we will prove the following theorem, which states that there exists a policy such that, for any problem from the set $\mathcal{P}_1$, the regret converges.

Theorem 2: Under Assumption 1, $\exists \pi \in \Pi$ such that, for each $(\lambda, \vec{\mu}) \in \mathcal{P}_1$, $R^{\pi}(T) = O(1)$.

To prove Theorem 2, we analyze the policy shown in Fig. 1 on an arbitrary $(\lambda, \vec{\mu}) \in \mathcal{P}_1$. This policy maintains a sample mean $\hat{\mu}_i$ that estimates μ_i using observations of server i's offered service. Note that under this policy π_1, the queue backlog will transition through alternating time intervals, wherein the queue is empty ($Q(t) = 0$) and busy ($Q(t) > 0$). We enumerate these periods using positive integers $p = 1, 2, \ldots$. Under the policy, the first time slot of each empty period is used to update an estimate of one of the servers chosen uniformly at random, and during each busy period the queue is serviced by only one server (namely, the one with highest sample mean). Therefore, the policy performs no exploration during busy periods and instead focuses on exploiting the observations it has already made.

for Empty period p **do**
  At the first time slot of the period, set $u(t) = i$ uniformly at random over $[N]$ and update $\widehat{\mu}_i$ with the observed state of $D(t)$
end for
for Busy period p **do**
  Schedule $\arg\max_{i \in [N]} \widehat{\mu}_i$ until the queue empties
end for

Fig. 1. Policy π_1 for achieving Theorem 2. We assume $\widehat{\mu}_i$ is set to zero, if server i has not yet been observed.

Given that the policy schedules server i during busy period p, the duration of the busy period is given by random variable X_i with mean $\overline{X}_i$. By Assumption 1, for all i, $\overline{X}_i$ is finite. Furthermore, the integral of the queue backlog over the busy period scheduled to server i is given by random variable Z_i with mean $\overline{Z}_i$. Thus, for a busy period starting at time τ under service from i, $Z_i \triangleq \sum_{t=\tau}^{\tau+X_i-1} Q(t)$. Finally, we use $S_i^{\pi_1}(P)$ to denote the number of busy periods in which i has been selected during the first P busy periods.

The proof of Theorem 2 now proceeds through four lemmas. We begin with Lemma 1, which shows that the busy periods over which we schedule i^* do not contribute to the regret. The proof follows from a sample path argument that shows, that for any outcome ω, over any busy period that π_1 schedules i^*, $Q^{\pi_1}(t, \omega) \leq Q^*(t, \omega)$. This gives the following bound on regret. We let $\mathbf{1}\{\cdot\}$ be the indicator random variable.

Lemma 1:

$$R^{\pi_1}(T) \leq E\left[\sum_{t=0}^{T-1} \sum_{i \in [N]-i^*} Q^{\pi_1}(t) \mathbf{1}\left\{u(t) = i\right\}\right].$$

Next, in Lemma 2, the expected value of Z_i is shown to be finite. This is because Z_i may be bounded by X_i^2 which has a finite expectation.

Lemma 2: $\overline{Z}_i < \infty$.

Now, no more than T busy periods can occur in T time slots. Therefore, we can bound the total queue backlog summed over times when we were scheduling server i up until T with the following.

Lemma 3:

$$E\left[\sum_{t=0}^{T-1} Q^{\pi_1}(t) \mathbf{1}\left\{u(t) = i\right\}\right] \leq \overline{Z}_i E\left[S_i^{\pi_1}(T)\right].$$

Finally, in Lemma 4, we show that the expected number of busy periods in which a sub-optimal server is chosen over the first T busy periods is bounded by a finite constant which is independent of T. Since we obtain a new observation during each empty period p, using Hoeffding's inequality we can show that the probability that a sample mean $\widehat{\mu}_i \geq \widehat{\mu}_{i^*}$, for

for Empty period p **do**
  At the first time slot of the period, set $u(t) = i$ uniformly at random over $[N]$ and update $\widehat{\mu}_i$ with the observed state of $D(t)$
end for
for Busy period p **do**
  Schedule $\arg\max_{i\in[N]} \widehat{\mu}_i$ for first p time slots or until the queue empties
  if The queue does not empty during the first p time slots **then**
    At each time slot, schedule server i with probability α_i until the queue empties
  end if
end for

Fig. 2. The policy π_2 for achieving Theorem 3. We assume $\widehat{\mu}_i$ is set to zero, if server i has not yet been observed.

$i \neq i^*$, decays exponentially in p. The result then follows from the convergence of geometric series.

Lemma 4: $E\left[S_i^{\pi_1}(T)\right] = O(1)$, $\forall i \in [N] - i^*$.

Given the above four lemmas, we are now ready to establish the theorem.

Proof of Theorem 2: Combining Lemmas 1 and 3, $R^{\pi_1}(T) \leq \sum_{i\in[N]-i^*} \overline{Z}_i E\left[S_i^{\pi_1}(T)\right]$. Then by Lemmas 2 and 4, $R^{\pi_1}(T) = O(1)$ giving the result. ∎

B. Non-Stabilizing Servers

In this subsection, we relax Assumption 1 to allow for non-stabilizing servers; i.e., we will now allow for $\mu_i \leq \lambda$ for some strict subset of the servers. Importantly, we will assume that the controller is given a known convex summation over the servers' rates that strictly dominates the arrival rate to the queue. Then, by randomizing over the servers using this convex summation, the controller can always stabilize the system. Concretely, we make the following assumption.

Assumption 2: For given $\alpha_i \geq 0$ and $\sum_{i\in[N]} \alpha_i = 1$,

$$(\lambda, \vec{\mu}) \in \mathcal{P}_2 \triangleq \Big\{\mathcal{P} : \sum_{i\in[N]} \alpha_i \mu_i > \lambda\Big\}.$$

Note that this assumption implies that the controller knows one stationary, randomized policy that has an offered service rate that is greater than the arrival rate, and that this randomized policy does not need to be learned. Further note that α_i is not required to have any special relationship to i^* and that α_{i^*} can be zero. Therefore, the randomized policy will not generally minimize regret and its use should not be overly relied upon. We then have the following theorem.

Theorem 3: Under Assumption 2, $\exists \pi \in \Pi$ such that, for each $(\lambda, \vec{\mu}) \in \mathcal{P}_2$, $R^{\pi}(T) = O(1)$.

A policy that achieves Theorem 3 is given in Fig. 2 and is referred to as π_2 throughout this subsection. The policy is similar to the policy π_1 of Subsection IV-A in that it iterates over empty and busy periods. The main difference however is that each busy period has a time-out threshold. If by the time-out threshold, the queue has not emptied, the policy uses the randomized policy defined by Assumption 2 to bring the queue backlog back to the empty state. The time-out threshold grows linearly in p, and therefore with each busy period, the controller becomes more reluctant to call upon the randomized policy.

Theorem 3 is established by analyzing π_2 on an arbitrary $(\lambda, \vec{\mu}) \in \mathcal{P}_2$. The proof is derived through three lemmas. We introduce some additional notation to facilitate understanding. For each time t, define $p(t)$ to be the period number for the empty or busy period in which t resides. Furthermore, let $C(p) \in \{0, 1\}$ be an indicator that takes the value 1 if either: a server not equal to i^* is first scheduled at the start of the busy period p or if the time-out threshold is hit in busy period p. Otherwise, $C(p) = 0$. Note that by the definition of π_2, if any server $i \neq i^*$ is scheduled during busy period p, then $C(p)$ must equal 1. Then, we have the following lemma, which states that the regret is upper bounded by the sum of queue backlogs over those busy periods in which indicator $C(p) = 1$. The proof is similar to that of Lemma 1.

Lemma 5: $R^{\pi_2}(T) \leq E\left[\sum_{t=0}^{T-1} Q^{\pi_2}(t) C\left(p(t)\right)\right]$.

Now, to upper bound the right-hand side of Lemma 5, we will want to bound the probability that indicator $C(p) = 1$. Note, $C(p)$ may equal 1 if at the start of the busy period, there exists an $i \neq i^*$ such that $\widehat{\mu}_i \geq \widehat{\mu}_{i^*}$ or if we schedule i^* for the start of the busy period but still cross the time-out threshold. As with the policy of the previous subsection, the probability of the former case diminishes exponentially in p. Likewise, for the latter case, since Assumption 2 implies $\mu_{i^*} > \lambda$, the probability of i^* not emptying the queue in p time slots also diminishes exponentially in p. This gives the following lemma.

Lemma 6: There exist positive constants M_0, χ, and p_0 such that for all $p \geq p_0$, $P\left(C(p) = 1\right) \leq M_0 e^{-\chi p}$.

Now, let variable $Z^{\pi_2}(p) \in \{1, 2, \dots\}$ denote the integral of the queue backlog over busy period p. In Lemma 7, we show $E\left[Z^{\pi_2}(p) | C(p) = 1\right]$ grows at most quadratically with p. This is because the queue backlog can grow at most linearly over the first p time slots, and given that the time-out threshold is hit, the required amount of time that the randomized policy will need to empty the queue will also grow in expectation at most linearly with p. Thus, the integral will be at most on the order of p^2.

Lemma 7: There exist positive constants M_1 and β_1 such that for all p, $E\left[Z^{\pi_2}(p) | C(p) = 1\right] \leq M_1 p^2 + \beta_1$.

We are now ready to establish the theorem.

Proof of Theorem 3: Using Lemma 5 and the fact that no more than T busy periods can occur in T time slots,

$$R^{\pi_2}(T) \leq E\left[\sum_{t=0}^{T-1} Q^{\pi_2}(t) C\left(p(t)\right)\right] \leq E\left[\sum_{p=1}^{T} Z^{\pi_2}(p) C(p)\right] \tag{2}$$

Recall that $C(p)$ is an indicator. When $C(p) = 0$, $Z^{\pi_2}(p)C(p) = 0$. Thus,

$$(2) = \sum_{p=1}^{T} P(C(p) = 1) E\left[Z^{\pi_2}(p) | C(p) = 1\right]$$

for Time slot $n = 0, 1, \dots$ until queue empties **do**
  if n is a dedicated exploration time slot **then**
    Choose $u(n) = i$ uniformly at random over $[N]$
    Update $\widehat{\mu}_i^p$ with the observed state of $D(n)$
  else
    Schedule $u(n) = \arg\max_{i \in [N]} \widehat{\mu}_i^p$
  end if
end for

Fig. 3. Learning algorithm for bringing the queue back to the empty state. Time n is normalized to when the algorithm is called. We assume $\widehat{\mu}_i^p$ is set to zero, if server i has not yet been observed during this algorithm's call. Note that $\widehat{\mu}_i^p$ is a separate variable from $\widehat{\mu}_i$ and is only used during busy period p.

$$\leq \sum_{p=1}^{p_0-1} \left(M_1 p^2 + \beta_1\right) + \sum_{p=p_0}^{T} \left(M_1 p^2 + \beta_1\right) M_0 e^{-\chi p}$$

where we have applied Lemmas 6 and 7 to obtain the last inequality. Since $\sum_{p=0}^{\infty} \left(M_1 p^2 + \beta_1\right) M_0 e^{-\chi p} < \infty$ we see that $R^{\pi_2}(T) = O(1)$ which gives the result. ∎

C. Learning Stabilizing Policies

We now relax Assumption 2 and no longer assume that a randomized policy is given to the controller, a priori. Instead, we will only assume that there exists at least one stabilizing server.

Assumption 3: $(\lambda, \vec{\mu}) \in \mathcal{P}_3 \triangleq \{\mathcal{P} : \mu_{i^*} > \lambda\}$.

We then have the following theorem, which is analogous to the previous subsections.

Theorem 4: Under Assumption 3, $\exists \pi \in \Pi$ such that, for each $(\lambda, \vec{\mu}) \in \mathcal{P}_3$, $R^{\pi}(T) = O(1)$.

To prove the theorem, we will analyze the following policy π_3 on an arbitrary $(\lambda, \vec{\mu}) \in \mathcal{P}_3$. Under π_3 we follow the policy described in Fig. 2 with the following minor change. When the time-out threshold of a busy period is reached, rather than relying on the known randomized policy given by Assumption 2, π_3 uses the method of Fig. 3 to bring the queue back to the empty state. The method in Fig. 3 has dedicated exploration time slots. During these time slots, the policy chooses from the servers uniformly at random and updates new variables $\widehat{\mu}_i^p$ based off of its observations of the resulting offered service. The location of the dedicated exploration time slots are predetermined by the controller. Let $V(n)$ be the number of dedicated explorations made by time slot n. Then for our proof of Theorem 4, we require that the dedicated exploration times are chosen such that

$$rn^{\epsilon} - b_1 \leq V(n) \leq rn^{\epsilon} + b_2, \text{ for } n = 0, 1, 2, \dots \quad (3)$$

for positive constants b_1, b_2, r, and $\epsilon \in (0, 1)$. For example, one choice could be to have dedicated explorations occur at time slots $n = k^2$ for $k = 0, 1, 2, \dots$. Note that (3) requires that the frequency of explorations diminishes with n and eventually falls below $1 - \frac{\lambda}{\mu_{i^*}}$. Subject to (3), the exact choice of the dedicated exploration times is left to the designer.

In contrast to policy π_1 of Subsection IV-A, π_3 performs some exploration when the queue is backlogged. This allows the controller to continuously empty the queue even if it has not had enough empty periods, so far, to learn the best server. However, similar to policy π_2, the controller becomes increasingly reluctant to explore during busy periods as time progresses.

As policy π_3 is similar to π_2, the proof of Theorem 4 closely follows the proof of Theorem 3 with Lemma 8 below replacing Lemma 7. For policy π_3, let $Z^{\pi_3}(p) \in \{1, 2, \dots\}$ denote the integrated queue backlog of busy period p. Note that $Z^{\pi_3}(p)$ is analogous to $Z^{\pi_2}(p)$ of the previous subsection. Then, we have the following.

Lemma 8: There exist positive constants M_2 and β_2 such that for all p, $E\left[Z^{\pi_3}(p) \middle| C(p) = 1\right] \leq M_2 p^2 + \beta_2$.

The proof of Theorem 4 then follows the proof of Theorem 3 with $Z^{\pi_3}(p)$ replacing $Z^{\pi_2}(p)$.

V. Systems without Stabilizing Servers

In the previous section, we saw that the existence of a stabilizing server (i.e., $\mu_{i^*} > \lambda$) allowed for policies that had $R^{\pi}(T) = O(1)$. A natural question is whether we can obtain similar results for the subset of problems $(\lambda, \vec{\mu})$ that do not have stabilizing servers. We proceed to show that there cannot exist a policy that can achieve $O(1)$ regret over this entire subset. The proof can be found in [7].

Assumption 4: $(\lambda, \vec{\mu}) \in \mathcal{P}_4 \triangleq \{\mathcal{P} : \mu_i \leq \lambda, \forall i \in [N]\}$.

Given this, we have the following theorem.

Theorem 5: For any policy $\pi \in \Pi$ there exists a $(\lambda, \vec{\mu}) \in \mathcal{P}_4$ such that $R^{\pi}(T) = \Omega(T)$.

VI. Conclusion

This work considered the problem of learning service rates to minimize queue length regret. We showed that queue-length based policies can have a queue length regret that is order optimal, $O(1)$, for all systems such that $\mu_{i^*} > \lambda$, while traditional bandit algorithms may have a queue length regret that is $\Omega(\log(T))$. This shows that to optimize queue length regret, learning algorithms must take into account the system's queueing dynamics and cannot simply maximize offered service to the queue. The policies considered herein were chosen for their analytical tractability, and we believe there are opportunities to improve upon their rates of convergence.

References

[1] W. Thompson, "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples," *Bulletin of the American Mathematics Society*, vol. 25, no. 3/4, pp. 285–294, 1933.

[2] S. Bubeck and N. Cesa-Bianchi, "Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems," *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[3] T. L. Lai and H. Robbins, "Asymptotically Efficient Adaptive Allocation Rules," *Advances in Applied Mathematics*, vol. 6, pp. 4–22, 1985.

[4] R. Agrawal, "Sample Mean Based Index Policies with $O(\log n)$ Regret for the Multi-Armed Bandit Problem," *Advances in Applied Probability*, vol. 27, no. 4, pp. 1054–1078, 1995.

[5] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, vol. 47, no. 2–3, pp. 235–256, 2002.

[6] S. Krishnasamy et al., "Regret of Queueing Bandits," in *Proc. Neural Information Processing Systems*, 2016, pp. 1669–1677.

[7] T. Stahlbuhk, *Control of Wireless Networks Under Uncertain State Information*, Doctoral Thesis, Massachusetts Institute of Technology, 2018.