Regret of Queueing Bandits

Rudra Laxmi Kanth
dept. Computer Science and Engineering
IIT Madras
Chennai, India
cs20b066@smail.iitm.ac.in

Arunesh J B
dept. Computer Science and Engineering
IIT Madras
Chennai, India
cs20b009@smail.iitm.ac.in

I. Introduction

In this paper we see a variant of the stochastic MAB motivated by queuing applications. Arms are pulled upon arrival of jobs (jobs are waiting in a queue). Each arm is a server which services the arriving jobs. If a arm on pulled gives a positive reward then the job is successfully serviced. Else the service was unsuccessful and the job stays in the queue. Our aim is to choose the arm with highest success probability i.e the arm which most likely services the job successfully.

The basic model lacks a key aspect of service in queueing systems: jobs wait in a queue until they're served. These systems are stateful, as when an arm yields no reward, the job stays in the queue, and the model needs to keep track of pending jobs.

We define a term *queue length*, which is the difference between the cumulative number of arrivals and departures. The queue length measures the quality of the service. We use this queue length to define the notion of regret, which is the expected difference between our queue length and the optimal queue length (the queue length achieved when we pull the best arm).

Our goal is to minimize this regret at finite time t. We later learn that there are "2 stages/phases" in the queuing bandit. In the *early stage* the bandit algorithm is unable to even stabilize the queue i.e on average, the queue length increases over time and is continuously backlogged (accumulated); therefore the queue-regret grows with time. In the *later stage* the bandit algorithm decreases the queue length over time and tends to zero. A stochastically stable queue goes through **Regenerative cycles** – a random cyclical behavior where queues build-up over time, then empty, and the cycle repeats. Therefore the queue regret resets on regular time intervals. We aim to provide lower bounds on queue-regret for both the early and late stages, as well as algorithms that essentially match these lower bounds.

II. Applications

Queueing is employed in modeling a vast range of service systems. In online service platforms such as uber the available supply queues correspond to the available drivers and the arriving demand correspond to the arriving ride requests from customers.

Similarly queuing is also applied in online rental platforms such as Aribnb where we have the available rentals and booking requests. It's used in order flow in financial markets and packet flow in communication networks. Since MAB models are a natural way to capture learning in this entire range of systems, incorporating queueing behavior into the MAB model is an essential challenge.

III. Problem Setting

We consider a discrete-time queuing system with a single queue and K servers. The servers are indexed by k = 1,...,K. Arrivals to the queue and service offered by the links are according to product Bernoulli distribution and i.i.d. across time slots. The mean arrival rate is given by λ and the mean service rates by the vector $\mu = [\mu_k]_{k\in[K]}$, with $\lambda < max_{k\in[K]}\mu_k$. We have to schedule a server to the job in every time slot. $\kappa(t)$ denote the server that is scheduled at time t. $R_k(t) \in \{0, 1\}$ be the reward for this service (rewards follow Bernoulli distribution). S(t), the service offered by service offered by the server $\kappa(t)$ at time t, i.e $S(t) = R_\kappa(t)(t)$. If A(t) in the number of arrivals at time t, then the queue length at time t is given by : $Q(t) = (Q(t-1)+A(t)-S(t))^+$.

Now to evaluate the performance of the queuing bandit we have to define the notion of regret. Let $k^* = argmax_{k\in[K]\mu_k}$ with maximum mean $\mu^* = max_{k\in[K]}\mu_k$. Let $Q^*(t)$ be the queue length under the optimal policy. Therefore the regret is given by $\psi(t) = E[Q(t) - Q^*(t)]$.

Here, $\psi(t)$ is interpreted as the traditional Multi-Armed Bandit (MAB) regret, with the caveat that rewards are accumulated only if there is a job that can benefit from this reward. We refer to $\psi(t)$ as the "queue-regret." Formally, our objective is to develop bandit algorithms that minimize the queue-regret at a finite time t.

IV. Regenerative Cycles

A stochastically stable queue goes through regenerative cycles – a random cyclical behavior where queues build up over time, then empty, and the cycle repeats. The associated

recurring "zero-queue-length" epochs mean that sample-path queue-regret essentially "resets" at (stochastically) regular intervals; i.e., the sample-path queue-regret becomes non-positive at these time instants. Thus, the queue-regret should decrease over time as the algorithm learns.

A. Late Stage

We first consider what happens to the queue regret as $t \to \infty$. And considering a standard bandit algorithm, but where the sample-path queue-regret "resets" at time points of regeneration.

In this case, the queue-regret is approximately a (discrete) derivative of the cumulative regret. Since the optimal cumulative regret scales like $\log t$, asymptotically the optimal queue-regret should scale like $1/t$. Indeed, we show that the queue-regret for α-consistent policies is at least C/t infinitely often, where C is a constant independent of t. Further, we introduce an algorithm called Q-ThS for the queueing bandit (a variant of Thompson sampling with explicit structured exploration), and show an asymptotic regret upper bound of $O(\text{poly}(\log t)/t)$ for Q-ThS, thus matching the lower bound up to poly-logarithmic factors in t. Q-ThS exploits structured exploration: we exploit the fact that the queue regenerates regularly to explore more systematically and aggressively.

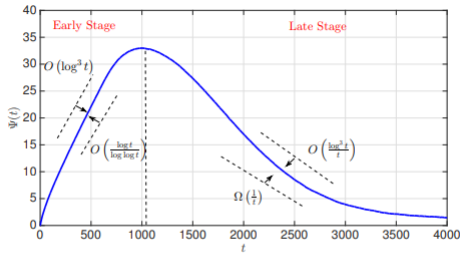


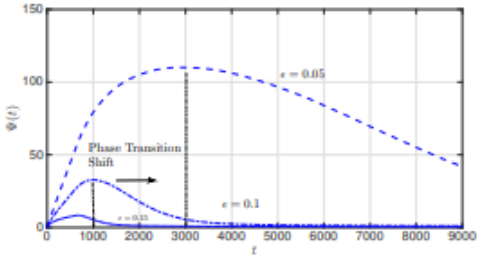
Figure 1: Queue-regret $\Psi(t)$ under Q-ThS in a system with $K = 5$, $\epsilon = 0.1$ and $\Delta = 0.17$

B. Early Stage

The early stage of the system, which is defined as the period preceding the algorithm's ability to stabilize the queues. This early stage is particularly crucial in heavily loaded systems, where the rate of incoming requests approaches the rate at which the optimal server can process them efficiently. In our research, we establish a lower bound of $\Omega(\log t/\log\log t)$ on the queue-regret during this initial phase. Interestingly, we observe that, up to a logarithmic factor of $\log\log t$, the behavior of early stage regret closely resembles that of cumulative regret, which scales with the logarithm of time ($\log t$). To delve deeper into our findings, we explore the heavily loaded regime, a fundamental asymptotic regime for studying queueing systems, and one that has been extensively examined in existing literature. This report aims to provide insights and analysis into the behavior of queue-regret during the early stage in heavily loaded systems.

Analysis shows that the time to switch from the early stage to the late stage scales at least as t = Ω(K/ε), where ε is the gap between the arrival rate and the service rate of the optimal server; thus $\varepsilon \to 0$ in the heavy-load setting.

The early stage lower bound of $\Omega(\log t/\log\log t)$ is valid up to t = O(K/ε).In the heavy-load limit, depending on the relative scaling between K and ε, the regret of Q-ThS scales like O(poly(log t)/ε^2t). For times that are arbitrarily close to Ω(K/ε). In Q-ThS is nearly optimal in the time it takes to "switch" from the early stage to the late stage.



(a) Queue-Regret under Q-ThS for a system with 5 servers with $\epsilon \in \{0.05, 0.1, 0.15\}$

V. Algorithm

Algorithm 1 Q-ThS

At time t,
Let E(t) be an Bernoulli sample of mean min{1,3K$\frac{log^2 t}{t}$}
if $E(t) = 1$ **then**
  Explore :
    Schedule a server uniformly at random
else
  Exploit :

  For each $k \in [K]$ pick a sample $\hat{\theta}_k(t)$,
  $\hat{\theta}_k(t) \sim B(\hat{\mu}_k(t)T_k(t-1)+1, (1-\hat{\mu}_k(t))T_k(t-1)+1)$

  Schedule a server ,
    $\kappa(t) = argmax_{k\in[K]}\hat{\theta}_k(t)$
end if

In the above algorithm B is the Beta function.
Q-ThS exploits structured exploration: we exploit the fact that the queue regenerates regularly to explore more systematically and aggressively. This is implemented using the Beta function which ensures that the we employ aggressive exploring regularly.
We analyze the performance of a scheduling algorithm with respect to queue-regret as a function of time and system parameters like: (a) the load on the system $\epsilon = (\mu^* - \lambda)$, and (b) the minimum difference between the rates of the best and the next best servers $\Delta = \mu^* - max_{k\neq k}\mu_k$.

VI. Theorems

A. Late Stage

Theorem 1: For any problem instance (λ, μ) and any α-consistent policy, the regret $\psi(t)$ satisfies

$$\psi(t) \geq \left(\frac{\lambda}{4} D(\mu)(1-\alpha)(K-1)\right) \frac{1}{t}$$

for infinitely many t, where

$$D(\mu) = \frac{\Delta}{KL(\mu_{min}, \frac{\mu^*+1}{2})}$$

Proof idea: The proof of the lower bound consists of three main steps. First we show that the regret at any time-slot is lower bounded by the probability of a sub-optimal schedule in that time-slot (up to a constant factor that is dependent on the problem instance). In the second step, the lower bound on the regret in terms of the probability of a sub-optimal schedule enables us to obtain a lower bound on the cumulative queue-regret in terms of the number of sub-optimal schedules. We then use a lower bound on the number of sub-optimal schedules for $\alpha - consistent$ policies to obtain a lower bound on the cumulative regret. In the final step, we use the lower bound on the cumulative queue-regret to obtain an infinitely often lower bound on the queue-regret.

Theorem 2: Consider a problem instance (λ, μ). Let w(t) = $exp\left(\left(\frac{2logt}{\Delta}\right)^{2/3}\right)$, v'(t) = $\frac{6K}{\epsilon} w(t)$ and v(t) = $\frac{24}{\epsilon^2} logt + \frac{60K}{\epsilon} \frac{v'(t) log^2 t}{t}$ then under Q-ThS the regret $\psi(t)$ satisfies,

$$\psi(t) \geq O\left(\frac{Kv(t)log^2 t}{t}\right)$$

for all t such that $\frac{w(t)}{logt} \geq \frac{2}{\epsilon}, t \geq \exp(6/\Delta^2), v(t) + v'(t) \leq t/2$

Proof idea: The proof consists of two main parts – one which gives a high probability result on the number of sub-optimal schedules in the exploit phase in the late stage, and the other which shows that at any time, the beginning of the current regenerative cycle is not very far in time.

B. Early Stage

Theorem 3: For given any problem instance (λ, μ) and for any α-consistent policy and $\gamma > \frac{1}{1-\alpha}$ the regret $\psi(t)$ satisfies

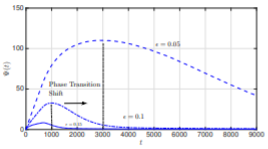
$$\psi(t) \geq \frac{D(\mu)}{2}(K-1)\frac{log\ t}{log\ log\ t}$$

for $t \in \left[max C_1 K^\gamma, \tau, (K-1)\frac{D(\mu)}{2\epsilon}\right]$ where D(μ) is defined in theorem 1, and τ and C_1 are constants that depend on α, γ and the policy.

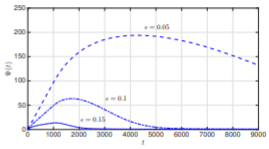
Proof idea : As in Theorem 1, we then use a lower bound on the number of sub-optimal schedules for α-consistent policies to obtain a lower bound on the queue-regret.

VII. Simulation Results

In this section, we present the simulation results for various queueing bandit systems with K servers.Our simulations demonstrate a phase transition from unstable to stable behavior, consistent with our theoretical predictions. We assess the performance of Algorithm while varying system parameters, such as traffic (ϵ), the gap between optimal and suboptimal servers (δ), and the system size (K).



(a) Queue-Regret under Q-ThS for a system with 5 servers with $\epsilon \in \{0.05, 0.1, 0.15\}$



(b) Queue-Regret under Q-ThS for a a system with 7 servers with $\epsilon \in \{0.05, 0.1, 0.15\}$

In This Figure , we observe the evolution of regret ($R(t)$) in systems of size 5 and 7. It is evident that regret decays faster in the smaller system, This demonstrates the impact of different traffic settings on system performance, showing that regret increases as ϵ decreases.we note that the time of the phase transition shifts to the right as ϵ decreases.

VIII. Conclusion

1) The paper presents a novel regret analysis of the queueing bandit problem, offering insights into regret in early and late stages, as well as the switching time. An asymptotically optimal algorithm, Q-ThS, is introduced, which exhibits the desired switching behavior.
2) Open questions remain regarding the development of a single adaptive algorithm that performs well throughout the entire duration, particularly addressing early stage regret.
3) A key challenge in finding a single optimal algorithm is establishing concentration results for suboptimal arm pulls within regenerative cycles. These results are necessary for both asymptotic late stage regret analysis and early stage regret in heavily loaded scenarios.

Acknowledgment

This report is based on the paper published by Subhashini Krishnasamy, Rajat Sen, Ramesh Johari, and Sanjay Shakkottai. We would like to express our gratitude to these authors for their valuable contribution to the field. The link to the paper can be found at: **Link**