Regret of Queuing Bandits

Arunesh J B & Rudra Laxmi Kanth

Indian Institute of Technology
Madras

03-11-2023

Introduction

- We will consider a variant of the stochastic multi-armed bandits problem motivated by queuing applications.
- We will try to understand the notion of regret for this multi-armed bandit setting.

Outline

- Queuing Bandits
- Notion of Regret
- Regenerative Cycles
- Problem Setting
- Algorithm
- Conclusion
- Applications

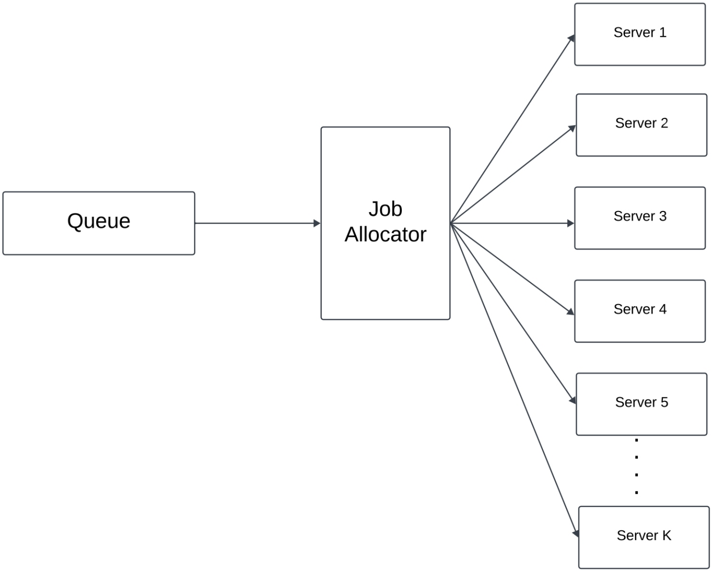
Recall the Stochastic MAB

- In the general MAB setting we aim to find the best arm. At each discrete time an algorithm pulls an arm and receives a reward (Bernoulli).
- The notion of regret was the loss incurred when compared to a genie policy (Cumulative Regret).

Introducing ”Queuing Bandits”

- It is variant of the stochastic MAB with queuing applications.
- Here arms correspond to servers.
- Arms are pulled on arrival of jobs, each server (arm) is chosen to service the arriving job.

Queueing System



Notion of 'Reward'

- In this model, the stochastic reward described above is equivalent to service.
- If the arm (server) that is chosen results in positive reward, the job is successfully completed and departs the system.
- When the chosen arm results in zero reward, the job being served remains in the queue.

Notion of 'Regret'

- How should we define the notion of regret ?
- How to quantify the performance of the system ?
- "Queue Length"
- The difference between the cumulative number of arrivals and departures
- the queue length is the most common measure of the quality of the service strategy being employed.

Notion of 'Regret' (continued)

- Let $Q(t)$ be the queue length at time t under a given bandit algorithm.
- let $Q^*(t)$ be the corresponding queue length under the "genie" policy that always schedules the optimal server
- The regret is given by $\psi(t) = E[Q(t) - Q^*(t)]$

Explore vs Exploit

- This problem clearly has the explore-exploit tradeoff similar to the Stochastic MAB case.
- Since the success probabilities across different servers are unknown, there is a tradeoff between learning these success probabilities (exploring) and servicing these jobs using the most promising server from past observations (exploiting).

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.

- A queuing bandit problem goes through two stages.
  - Early Stage.
  - Late Stage.

Early Stage

- Period preceding the algorithm's ability to stabilize queues.
- Lower bound established: $\Omega(\log t / \log\log t)$ during the early stage.
- Exploration of the heavily loaded regime, a fundamental asymptotic regime for studying queueing systems.
- Period preceding the algorithm's ability to stabilize queues.
- Time to switch from early to late stage scales as $t = \Omega(K/\epsilon)$, where is the gap between arrival and service rates.

Late Stage

- In the late stage, the system has gained more knowledge and may have identified the optimal arms for efficient service.
- Optimal cumulative regret scales logarithmically with time (log t), leading to the expectation of queue-regret scaling as 1/t.
- Queue-regret for α -consistent policies is at least C/t infinitely often, where C is a constant independent of time (t).

What about convergence ?

- After each cycle, the cycle length (c(t)) will decrease.

Goal

- We have to come up with a bandit algorithm that minimizes the queue-regret at a finite time t.

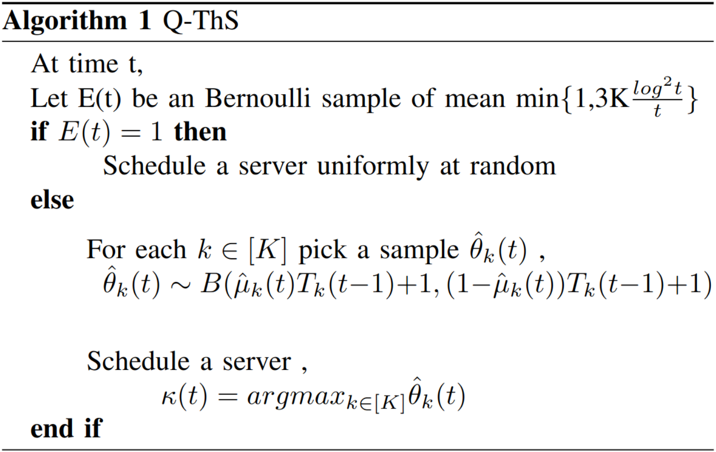
Problem Setting

- We consider a discrete-time queueing system with a single queue and K servers (1,2,...K)
- Arrivals to the queue and service offered by the links follow Bernoulli distribution and i.i.d. across time slot.
- The mean arrival rate is λ and the service rate for each arm $\mu = [\mu_k]_{k \in [K]}$
- The server scheduled at time t is given by $\kappa(t)$
- let k^* be the best arm and μ^* be it's mean. $(\mu^* = \mu_{k^*})$

Algorithm

- How should we design the algorithm ?
- ETC ?
- ϵ-greedy ?
- "Structured Exploration"
- We exploit the fact that the queue regenerates regularly. Therefore we have to explore more systematically and aggressively (at regular time intervals).

Algorithm

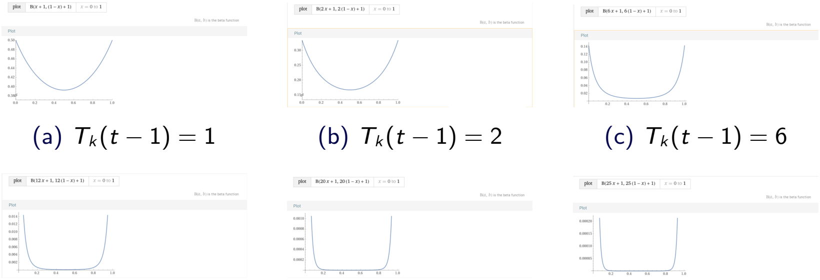


Algorithm 1 Q-ThS

```
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
        Schedule a server uniformly at random
else
      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
```

Beta function



plot B(x + 1, (1 − x) + 1) x = 0 to 1

plot B(2 x + 1, 2 (1 − x) + 1) x = 0 to 1

plot B(6 x + 1, 6 (1 − x) + 1) x = 0 to 1

plot B(12 x + 1, 12 (1 − x) + 1) x = 0 to 1

plot B(20 x + 1, 20 (1 − x) + 1) x = 0 to 1

plot B(25 x + 1, 25 (1 − x) + 1) x = 0 to 1

B(a, b) is the beta function

(a) $T_k(t-1) = 1$

(b) $T_k(t-1) = 2$

(c) $T_k(t-1) = 6$

(d) $T_k(t-1) = 12$

(e) $T_k(t-1) = 20$

(f) $T_k(t-1) = 25$

Figure 1: $\hat{\theta}_k(t)$ vs $\hat{\mu}_k(t)$ for different values of $T_k(t-1)$

Recall

- $\hat{\theta}_k(t) \sim B(\hat{\mu}_k(t) T_k(t-1) + 1, (1 - \hat{\mu}_k(t)) T_k(t-1) + 1)$

A few more parameters

- The load on the system is given by $\epsilon = (\mu^* - \lambda)$
- The minimum difference between rates of the best and next best servers is given by $\Delta = \mu^* - max_{k \neq k^*} \mu_k$

Lower Bounds on the Queue Regret

- We will now try to briefly go through the lower bounds on the queue regret for the Late Stage and the Early Stage

The Late Stage

- We show that for a given (λ, μ) the regret under Q-ThS scales as O (poly(log t)/t).

$\psi(t)$ scales as O(poly(logt)/t)

Theorem

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* $v(t) = \frac{24}{\epsilon^2}logt + \frac{60K}{\epsilon}\frac{v'(t)log^2t}{t}$ *then under Q-ThS the regret* $\psi(t)$ *satisfies,*

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

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

- Let's now look at how this theorem is proved

Proof idea

The proof has two parts:

- In the first part we make use of the structured exploration component ($\hat{\theta}_k(t)$) of Q-ThS to show that all the arms, including the sub-optimal ones, are sampled a sufficiently large number of times to give a good estimate of the service rates.
- In the second part we prove a high probability bound on the last time instant when the queue length was zero i.e at any time, the beginning of the current regenerative cycle is not very far in time.
- The proof of the theorem proceeds by combining the two parts above to show that the main contribution to the queue-regret comes from the structured exploration component in the current regenerative cycle, which gives the stated result.

Heavily Loaded Regime

- We will now try to analyze a special case.
- "Heavily Loaded System"
- A system is said to be heavily loaded when the arrival rate approaches the service rate of the optimal server.
- Mathematically $\lambda \to \mu^*$ i.e $\epsilon \to 0$

Bounds for Heavily Loaded Regime

Theorem

Let $w(t) = exp\Big(\Big(\frac{2logt}{\Delta}\Big)^{2/3}\Big)$ *then,*

$$\psi(t) \geq O\big(K\frac{log^3 t}{\epsilon^2 t}\big)$$

for all t such that
$\frac{w(t)}{logt} \geq \frac{2}{\epsilon}, \frac{t}{w(t)} \geq max\big\{\frac{24K}{\epsilon}, 15K^2 logt\big\}, t \geq \exp(6/\Delta^2), \frac{t}{logt} \geq \frac{198}{\epsilon^2}$

The Early Stage

Theorem

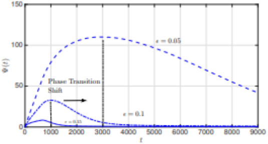
For given any problem instance (λ, μ) and for any α-consistent policy and $\gamma > \frac{1}{1-\alpha}$ the regret $\psi(t)$ satisfies

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

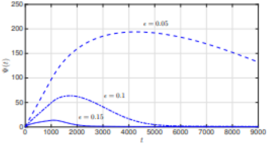
for $t \in \left[max C_1 K^{\gamma}, \tau, (K-1)\frac{D(\mu)}{2\epsilon}\right]$ where $D(\mu)$ is defined below, and τ and C_1 are constants that depend on α, γ and the policy.

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

Simulation Results



(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 $\epsilon \in \{0.05, 0.1, 0.15\}$

- We can see that there is a phase transition in both figures.
- We can observe that regret decays faster in the smaller system.
- Regret grows with decreasing ϵ. And the phase transition too shifts to the right as ϵ decreases.

Conclusion

This paper provides the first regret analysis of the queueing bandit problem. The preceding discussion highlights why minimization of queue-regret presents a subtle learning problem.

1. On one hand, if the queue has been stabilized, the presence of regenerative cycles allows us to establish that queue regret must eventually decay to zero under optimal policy.
2. On the other hand, to actually have regenerative cycles in the first place, a learning algorithm needs to exploit enough to actually stabilize the queue (the early stage).

In this way the queuing bandit is a remarkable example of the tradoff between the exploration and exploitation tradoff.

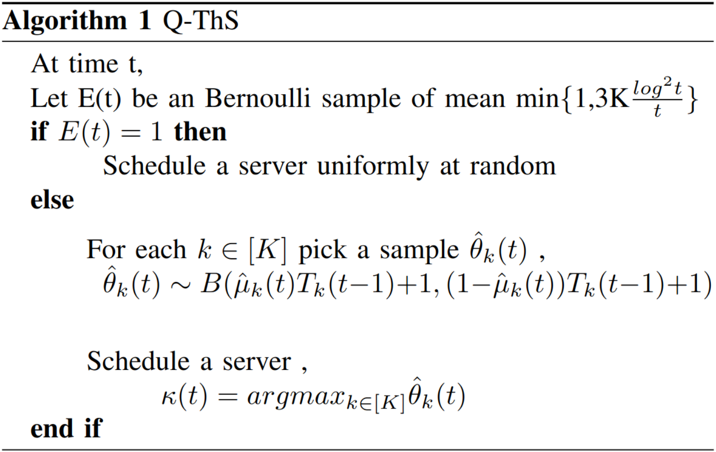
Can we further generalize?

- What if there are n queues ?
- How will the problem setting change ?
- What about the algorithm ?

Modified Problem Setting

- The service rate would no longer be a vector, it would be a matrix $\mu = [\mu_{uk}]_{u \in [U], k \in [K]}$
- At each time instance we have to find an optimal matching in the bipartite graph between queues and servers.

Recall the Algorithm

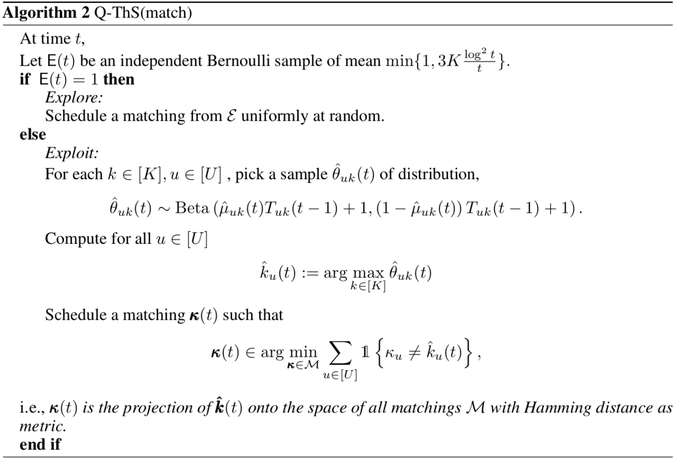


Algorithm 1 Q-ThS

```
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**
        Schedule a server uniformly at random
**else**
        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**
```

Modified Algorithm



Algorithm 2 Q-ThS(match)

```
At time $t$,
Let $\mathsf{E}(t)$ be an independent Bernoulli sample of mean $\min\{1, 3K\frac{\log^2 t}{t}\}$.
**if** $\mathsf{E}(t) = 1$ **then**
    *Explore:*
    Schedule a matching from $\mathcal{E}$ uniformly at random.
**else**
    *Exploit:*
    For each $k \in [K], u \in [U]$ , pick a sample $\hat{\theta}_{uk}(t)$ of distribution,
        $$\hat{\theta}_{uk}(t) \sim \text{Beta}\left(\hat{\mu}_{uk}(t)T_{uk}(t-1)+1, \left(1-\hat{\mu}_{uk}(t)\right)T_{uk}(t-1)+1\right).$$
    Compute for all $u \in [U]$
        $$\hat{k}_u(t) := \arg\max_{k \in [K]} \hat{\theta}_{uk}(t)$$
    Schedule a matching $\boldsymbol{\kappa}(t)$ such that
        $$\boldsymbol{\kappa}(t) \in \arg\min_{\boldsymbol{\kappa} \in \mathcal{M}} \sum_{u \in [U]} \mathbb{1}\left\{\kappa_u \neq \hat{k}_u(t)\right\},$$
i.e., $\boldsymbol{\kappa}(t)$ *is the projection of* $\hat{\mathbf{k}}(t)$ *onto the space of all matchings* $\mathcal{M}$ *with Hamming distance as metric.*
**end if**
```

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.

References

Regret of Queueing Bandits by Subhashini Krishnasamy, Rajat Sen, Ramesh Johari, Sanjay Shakkottai.

Thank You