A Constrained Bandit Approach for Online Dispatching

Xin Liu
University of Michigan, Ann Arbor
xinliuee@umich.edu

Bin Li
University of Rhode Island
binli@uri.edu

Pengyi Shi
Purdue University
shi178@purdue.edu

Lei Ying
University of Michigan, Ann Arbor
leiying@umich.edu

ABSTRACT

This paper considers constrained online dispatching with unknown arrival, reward and constraint distributions [4]. The problem is formulated as a constrained bandits problem. We propose a novel online learning-based algorithm, named POND, standing for Pessimistic-Optimistic oNline Dispatching, which achieves $O(\sqrt{T})$ regret and $O(1)$ constraint violation. Both bounds are sharp. Our experiments on synthetic and real datasets show that POND achieves low regret with minimal constraint violations.

1. PROBLEM FORMULATION

Online dispatching refers to the process (or an algorithm) that dispatches incoming jobs to available servers in real-time. The problem arises in many different fields. Examples include routing customer calls to representatives in a call center, assigning patients towards in a hospital, dispatching goods to different shipping companies, scheduling packets over multiple frequency channels in wireless communications, routing search queries to servers in a data center, selecting an advertisement to display to an Internet user, and allocating jobs to workers in crowdsourcing.

In this paper, we consider the following discrete-time model over a finite horizon T for the online dispatching problem. We assume there are N types of jobs, the set of jobs is denoted by $\mathcal{N} = \{1, 2\cdots, N\}$, and M types of servers, the set of servers is denoted by $\mathcal{M} = \{1, 2\cdots, M\}$. Here a job may represent a patient who comes to an emergency room and needs to be hospitalized, an Internet user who browses a webpage, or a job submitted to a crowdsourcing platform; and a server may represent a hospital ward or a doctor in the emergency room, an advertisement of a product, or a worker registered at the crowdsourcing platform. We assume that jobs of type i arrive at each time slot t according to a stochastic process $\Lambda_i(t)$ with unknown mean $\mathbb{E}[\Lambda_i(t)] = \lambda_i$. The online dispatcher sends $x_{i,j}(t)$ of the $\Lambda_i(t)$ jobs to server j, and receives reward $R_{i,j,s}(t)$ for the s^{th} job. We model $R_{i,j,s}(t)$ to be a random variable in with an unknown distribution with $\mathbb{E}[R_{i,j,s}(t)] = r_{i,j}, \forall s$ which is the case in many applications such as order dispatching in logistics, online advertising, crowdsourcing, and patient assignment in healthcare (e.g. click-through-rates are unknown to advertising platforms, average job completion quality is unknown to crowdsourcing platforms and quality of treatment is unknown to hospital operators). Furthermore, we assume the arrival processes $\{\Lambda_i(t)\}$ and the reward processes $\{R_{i,j,s}(t)\}$ are i.i.d across job types, servers and time slots.

The goal of this paper is to develop learning-based online dispatching algorithms to maximize the cumulative reward over T time slots:

$$\sum_{t=0}^{T-1} \mathbb{E}\left[\sum_{i,j}\sum_{s=1}^{x_{i,j}(t)} R_{i,j,s}(t)\right] \tag{1}$$

The objective function in (1) is equivalent to

$$\sum_{t=0}^{T-1} \mathbb{E}\left[\sum_{i,j} r_{i,j}x_{i,j}(t)\right] \tag{2}$$

because $x_{i,j}(t)$ is independent of $R_{i,j,s}(t), \forall s$.

For a resource-constrained server system, we aim to maximize the objective (2) subject to a set of constraints $\mathcal{K}$, including the capacity, fairness and resource budget constraints (all these constraints are unified into general forms), formulated as follows:

$$\max_{\mathbf{x}(t)} \sum_{t=0}^{T-1} \mathbb{E}\left[\sum_{i,j} r_{i,j}x_{i,j}(t)\right] \tag{3}$$

$$\text{s.t. } \sum_{j} x_{i,j}(t) = \Lambda_i(t), \forall i, x_{i,j}(t) \geq 0, \forall i, j. \tag{4}$$

$$\sum_{t=0}^{T-1} \mathbb{E}\left[\sum_{i} w_{i,j}^{(k)}(t)x_{i,j}(t) - \rho_j^{(k)}(t)\right] \leq 0, \ \forall j, k, \tag{5}$$

where $x_{i,j}(t)$ is the number of type-i jobs assigned to server j at time slot t and $\mathbf{x}(t)$ is its matrix version in which the (i,j)th entry is $x_{i,j}(t)$; (4) represents the allocating conservation for job arrivals; and (5) can represent the capacity, fairness and resource budget constraints, where $w_{i,j}^{(k)}(t)$ is the "weight" of a type-i job to server j and $\rho_j^{(k)}(t)$ is the corresponding "requirement". $w_{i,j}^{(k)}(t)$ and $\rho_j^{(k)}(t)$ are i.i.d across i, j, and t.

We note this problem (3)-(5) is a constrained contextual bandit problem where $\mathcal{N}$ is the set of contexts and $\mathcal{M}$ is the set of actions (called "One Bandit per Context" in [3]). When action j is applied to a context of type-i, the learner receives reward $R_{i,j}$ and incurs cost $w_{i,j}$. We next propose a novel online learning algorithm to maximize the cumulative reward while keeping constraints satisfied.

2. POND ALGORITHM

There are two major challenges in solving (3)-(5) in real time: unknown reward distributions, and unknown statistics of arrival processes and constraint parameters. To tackle unknown reward distributions, we utilize UCB learning [1] to estimate $r_{i,j}$. To deal with unknown arrival processes and stochastic constraints, we maintain virtual queues on the server side. The virtual queues are related to dual variables [5, 6], which are used to track constraint violations.

Virtual Queues:

$$Q_j^{(k)}(t+1) = \left[Q_j^{(k)}(t) + \sum_i w_{i,j}^{(k)}(t)x_{i,j}(t) - \rho_j^{(k)}(t) + \epsilon\right]^+ \quad (6)$$

for any $j \in \mathcal{M}$ and $k \in \mathcal{K}$. The operator $(x)^+ = \max(x, 0)$. $Q_j^{(k)}(t)$ is the virtual queue associated to the k^{th} constraint imposed on server j. $\sum_i w_{i,j}^{(k)}(t)x_{i,j}(t)$ is the "total weight" (e.g. capacity or budget consumption) on server j and $\rho_j^{(k)}(t)$ is the "requirement" (e.g. capacity or budget limit) on the server j. ϵ is a tightness constant that decides the trade-off between the regret and constraint violations, which we will specify in the proof later. This idea of adding tightness was inspired by the adaptive virtual queue (AVQ) used for the Internet congestion control [2]. By choosing $\epsilon = O(1/\sqrt{T})$, the algorithm presented next can achieve $O(\sqrt{T})$ regret and $O(1)$ constraint violations.

To maximize the cumulative reward in (3) while keeping constraint violations reasonably small, we incorporate the learned reward and virtual queues in (6) to design POND - Pessimistic-Optimistic oNline Dispatching (Algorithm 1). In Algorithm 1, we first utilize the classic UCB algorithm to learn the reward $\hat{r}_{i,j}(t)$, then allocate the incoming jobs according to a "max-weight" algorithm, and finally update virtual queues and reward estimation according to the max-weight dispatching decisions. Note that $\hat{r}_{i,j}(t) = \infty$ when $N_{i,j}(t-1) = 0$, which implies that $\eta_{i,j}(t) = \infty$. When multiple $\eta_{i,j}(t) = \infty$, we break the tie uniformly at random. In weight $\eta_{i,j}(t) = V\hat{r}_{i,j}(t) - \sum_k w_{i,j}^{(k)}(t)Q_j^{(k)}(t)$, parameter V is chose to be $O(\sqrt{T})$ to balance the reward and virtual queues (constraint violations). When the virtual queue $Q_j^{(k)}(t)$ associated to capacity constraint is large (capacity constraint of server j is violated too often), which implies the algorithm allocates too many jobs to server j, weight $\eta_{i,j}(t)$ tends to be small so POND is less likely to allocate new incoming jobs to server j. Similarly, when virtual queue $Q_j^{(k)}(t)$ associated to fairness constraint is large (fairness constraint of server j has been violated), which implies server j has not received sufficient number of jobs, weight $\eta_{i,j}(t)$ tends to be large (recall $w_{i,j}^{(k)}(t) = -1$ in fairness constraints) so POND is more likely to allocate new incoming jobs to server j.

We remark that by replacing UCB learning with MOSS learning $\hat{r}_{i,j}(t) = \bar{r}_{i,j}(t-1) + \sqrt{\frac{2}{N_{i,j}(t-1)} \log \frac{T}{M \cdot N_{i,j}(t-1)}}$, POND can achieve the tight regret bound $O(\sqrt{T})$, and UCB achieves regret bound $O(\sqrt{T \log T})$. However, in practice, MOSS learning might explore too much and suffer from sub-optimality and instability.

Algorithm 1: POND Algorithm.

Input: V, ϵ, $Q_j^{(k)}(0) = 0$ and $\bar{r}_{i,j}(-1) = N_{i,j}(-1) = 0$.

for $t = 1, \cdots, T-1$ **do**

UCB learning: $\hat{r}_{i,j}(t) = \bar{r}_{i,j}(t-1) + \sqrt{\frac{\log T}{N_{i,j}(t-1)}}$;

Compute the weight of a type-i job to server j :

$$\eta_{i,j}(t) = V\hat{r}_{i,j}(t) - \sum_k w_{i,j}^{(k)}(t)Q_j^{(k)}(t).$$

Observe jobs arrival $\Lambda_i(t)$ and do max-weight allocation:

$$x_{i,j}(t) \in \underset{\Lambda_i(t)=\sum_j x_{i,j}}{\arg\max} \sum_{i,j} \eta_{i,j}(t)x_{i,j}.$$

Update virtual queues according to (6).

Update the estimation of $\bar{r}_{i,j}(t)$ according to the rewards received:

$$N_{i,j}(t) = N_{i,j}(t-1) + x_{i,j}(t),$$

$$\bar{r}_{i,j}(t) = \frac{\bar{r}_{i,j}(t-1)N_{i,j}(t-1) + \sum_{s=1}^{x_{i,j}(t)} R_{i,j,s}(t)}{N_{i,j}(t)}.$$

end

3. MAIN RESULTS

To analyze the performance of POND, we compare it with an offline optimization problem given the reward, arrival, and constraint parameters. By abuse of notation, define $x_{i,j} = \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[x_{i,j}(t)]$, $w_{i,j}^{(k)} = \mathbb{E}[w_{i,j}^{(k)}(t)]$ and $\rho_j^{(k)} = \mathbb{E}[\rho_j^{(k)}(t)]$ in the optimization problem (3)-(5). We consider the following offline optimization problem (or fluid optimization problem):

$$\max_{\mathbf{x}} \sum_{i,j} r_{i,j}x_{i,j} \quad (7)$$

$$\text{s.t. } \lambda_i = \sum_j x_{i,j}, \ \forall i, \ x_{i,j} \geq 0, \forall i, j, \quad (8)$$

$$\sum_i w_{i,j}^{(k)}x_{i,j} \leq \rho_j^{(k)}, \ \forall j, k. \quad (9)$$

where $x_{i,j}$ corresponds to the average number of type-i jobs assigned to server j per time slot; (8) includes throughput constraints; (9) includes capacity constraints, fairness constraints and resource budget constraints.

Next, we define performance metrics including regret and constraint violation, and present an informal version of the main theorem.

Regret: Let $\mathcal{X}$ be the feasible set and $\mathbf{x}^*$ be the solution to the offline problem (7)-(9). We define the regret of an online dispatching algorithm to be

$$\mathcal{R}(T) = T\sum_{i,j} r_{i,j}x_{i,j}^* - \sum_{t=0}^{T-1} \mathbb{E}\left[\sum_{i,j} r_{i,j}x_{i,j}(t)\right].$$

Constraint violation: We define constraint violations to be

$$\mathcal{V}(T) = \sum_j \sum_k \left(\sum_{t=0}^{T-1} \mathbb{E}\left[\sum_i w_{i,j}^{(k)}(t)x_{i,j}(t) - \rho_j^{(k)}(t)\right]\right)^+,$$

which includes violations from capacity, fairness, and budget constraints. Note that $\mathcal{V}(T) \leq C$ implies that each constraint violation is bounded by C.

THEOREM 1 (INFORMAL STATEMENT). *Assuming bounded arrivals and rewards and let* $V = O(\sqrt{T})$ *and* $\epsilon = O(1/\sqrt{T})$, *the regret and constraint violations under POND are*

$$\mathcal{R}(T) = O(\sqrt{T}) \text{ and } \mathcal{V}(T) = O(1).$$

The formal statement of the theorem can be found [4].

4. EXPERIMENTS

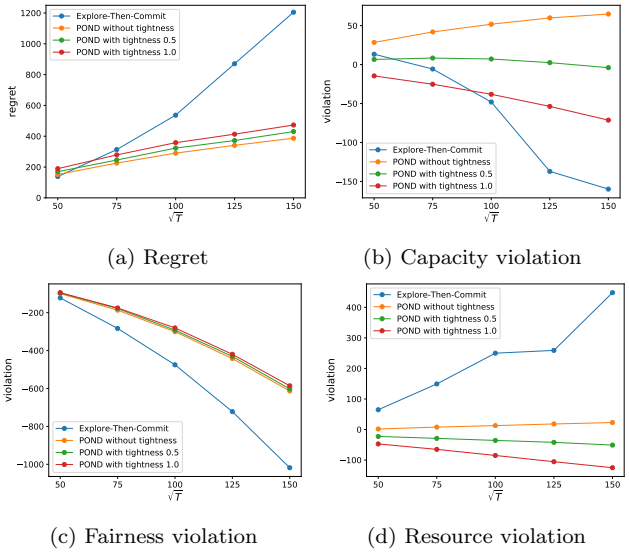
In this section, we present simulation results that demonstrate the performance of our POND algorithm. In particular, we show that POND achieves the $O(\sqrt{T})$ regret and $O(1)$ constraint violations with "tightness", while without tightness, the algorithm achieves $O(\sqrt{T})$ regret and $O(\sqrt{T})$ constraint violations. We also see that POND outperforms the Explore-Then-Commit algorithm (baseline) significantly.

We considered a model with two types of jobs and four servers. In particular, we assumed geometric arrivals with mean $\boldsymbol{\lambda} = [1.0, 2.0]$, Bernoulli rewards with mean

$$\mathbf{r} = \begin{bmatrix} 0.5 & 0.6 & 0.1 & 0.2 \\ 0.2 & 0.6 & 0.5 & 0.2 \end{bmatrix}.$$

We assumed capacity constraints $\sum_{i=1}^{2} x_{i,j} \leq \mu_j$, fairness constraints $\sum_{i=1}^{2} x_{i,j} \geq d_j \sum_{i=1}^{2} \lambda_i$, and resource constraints $\sum_{i=1}^{2} w_{i,j} x_{i,j} \leq \rho_j$ where we set $\mathbf{d} = [0.25, 0.25, 0.20, 0.20]$, $\boldsymbol{\mu} = [0.85, 0.85, 0.8, 0.8]$ and $\mathbf{w} = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 4 & 4 & 4 & 3.5 \end{bmatrix}$ and $\boldsymbol{\rho} = [3, 3, 2.5, 2.5]$ (these values are deterministic in the simulation). Let $V = 2\sqrt{T}$. We compared POND algorithm with "tightness" $\epsilon = O(1/\sqrt{T})$, and "no tightness", i.e. $\epsilon = 0$.

We simulated POND and Explore-Then-Commit over T time slots with $T = [50^2, 75^2, 100^2, 125^2, 150^2]$, where 500 independent trials were averaged for each T. We plotted the regret, capacity violation, fairness violation and resource violation against $\sqrt{T}$ in Figure 1, where we used the *maximum* average violation among four servers for each type of constraint violations, i.e., $\max_j \left(\sum_{t=0}^{T-1} \sum_i w_{i,j}^{(k)}(t) x_{i,j}(t) - \rho_j^{(k)}(t) \right)$. Figure 1 shows that using POND with tightness constants $\epsilon = 0.5/\sqrt{T}$ and $1/\sqrt{T}$, POND achieved $O(\sqrt{T})$ regret as in Figure 1a and $O(1)$ constraint violation as in Figure 1b-1d. Without the tightness constant, POND achieved $O(\sqrt{T})$ regret but $O(\sqrt{T})$ constraint violation as shown by the orange curve in Figure 1b. These numerical results are consistent with our theoretical analysis. The experimental results also show that using the tightness constant is critical for achieving the $O(1)$ constraint violations. Moreover, POND performed much better than Explore-Then-Commit by achieving both lower regret and constraint violations.



(a) Regret (b) Capacity violation

(c) Fairness violation (d) Resource violation

Figure 1: Regret and constraint violation v.s. T.

5. CONCLUSION

In this paper, we developed a novel online dispatching algorithm, POND, to maximize cumulative reward over a finite time horizon, subject to general constraints that arise from resource capacity and fairness considerations. Given unknown arrival, reward, and constraint distributions, POND leverages the UCB approach to learn the reward while using the MaxWeight algorithm to make the dispatching decision with virtual queues tracking the constraint violations. POND achieves $O(\sqrt{T})$ regret and $O(1)$ constraint violation with the key being introducing a "tightness" constant to balance between the regret and constraint violation.

Acknowledge

This work has been supported in part by NSF grants CNS-1815563, CNS-2001687, and CNS-2002608.

6. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, May 2002.

[2] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue (avq) algorithm for active queue management. *ACM SIGCOMM Computer Communication Review*, 31(4):123–134, 2001.

[3] T. Lattimore and C. Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[4] X. Liu, B. Li, P. Shi, and L. Ying. POND: Pessimistic-Optimistic oNline Dispatching. *arXiv preprint arXiv:2010.09995*, 2020.

[5] M. J. Neely. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks*, 3(1):1–211, 2010.

[6] R. Srikant and L. Ying. *Communication Networks: An Optimization, Control and Stochastic Networks Perspective*. Cambridge University Press, 2014.