Elastic Job Scheduling with Unknown Utility Functions

Xinzhe Fu and Eytan Modiano

MIT LIDS

November 11, 2021
Elastic Job Scheduling

Problem Setting:

- Decide on sizes of incoming jobs.
- Route the jobs to servers.
- Maximize the total utility.
- Receive observations of utility values upon jobs’ completion.

Applications:

- Jobs are malleable in nature.
  - ML-training tasks [1].
- Utility functions are unknown apriori.

Problem Formulation

Consider a discrete-time system with \( K \) classes, for \( t = 1, \ldots, T \):

- \( a_k(t) \) jobs of classes \( k \) arrive at dispatcher \( u_k \).
- \( u_k \) is connected to subset of servers \( S_{u_k} \).
- Decide the “job-size” \( x_k(t) \in [0, B] \).
- Route the jobs to servers.
- Server \( s_m \)'s offered service rate \( c_m(t) \).
- Queue length at server \( s_m \):
  \[
  Q_m(t + 1) = [Q_m(t) + Arr_m(t) - c_m(t)]^+.
  \]
Problem Formulation

Assumptions:

• Each class $k$ is associated with a utility function $f_k$.
  • Utility functions are unknown apriori.
  • Each $f_k$ is non-decreasing, concave and $L$-Lipschitz-continuous.

• For a class $k$ job of size $x_k$, we observe and obtain $f_k(x_k) + \epsilon_i$ when the job is completed.
  • $\epsilon_i$’s are i.i.d. noises.
  • $\mathbb{E}[\epsilon_i] = 0$, $|\epsilon_i| \leq C$.

• Arrivals $a_k(t)$’s are i.i.d. Offered services $c_m(t)$’s are i.i.d.
  • $\mathbb{E}[a_k(t)] = \lambda_k$, $\mathbb{E}[c_m(t)] = \mu_m$. $0 \leq a_k(t), c_m(t) \leq C$.
  • $a_k(t)$’s, $c_m(t)$’s are observable but $\lambda_k$’s, $\mu_m$’s are unknown.
Problem Formulation

Goal:
- Design a policy $\pi$ that decides on the sizes and designated servers of incoming jobs such that the expected utility obtained by the end of the time horizon $T$ is maximized.

Performance Metric: Regret

- $U(\pi, T)$: the expected utility obtained under policy $\pi$.
- $\pi^*$: the policy that maximizes the expected utility.
- Regret of $\pi$: $R(\pi, T) = U(\pi^*, T) - U(\pi, T)$. 
Bounds on the Utility

Consider the set $\Lambda$:

$$\Lambda := \left\{ (x_1, \ldots, x_K) \mid \begin{array}{l}
\exists \{\alpha\}_{km}, \text{s.t.} \\
\sum_k \alpha_{km} \lambda_k x_k \leq \mu_m \quad \forall s_m, \\
\alpha_{km} = 0, \quad \forall s_m \notin S_{uk} \\
\sum_m \alpha_{km} = 1, \forall k \\
\alpha_{km} \geq 0, \quad \forall k, m \\
0 \leq x_k \leq B, \quad \forall k
\end{array} \right\},$$

- Stability region of the network.
- $\Lambda$ involves unknown parameters $\lambda, \mu$. 
 Bounds on the Utility

Consider the optimization problem $\mathcal{P}$:

$$\mathcal{P} : \max \sum_{k=1}^{K} \lambda_k f_k(x_k)$$

s.t. $(x_1, \ldots, x_K) \in \Lambda$

$$x_k \in [0, B], \quad \forall k.$$ 

**Theorem**

$$U(\pi^*, T) \leq T \cdot OPT(\mathcal{P}).$$

- Proof using Jensen’s inequality.
Bounds on the Utility

**Theorem**

For any policy $\pi$, $U(\pi, T) \geq$

$$\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k \mathbb{E}[f_k(x_k(t))] - L \cdot \sum_{t=1}^{T} \sum_{m=1}^{M} \mathbb{E}[Q_m(T)].$$

- Expected utility of dispatched jobs minus the incomplete jobs in the queues.

**Theorem**

For any policy $\pi$, $R(\pi, T) \leq$

$$\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k \mathbb{E}[f_k(x^*_k) - f_k(x_k(t))] + L \cdot \sum_{t=1}^{T} \sum_{m=1}^{M} \mathbb{E}[Q_m(T)].$$

- Utility Regret: $\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k \mathbb{E}[f_k(x^*_k) - f_k(x_k(t))]$.
- Queueing Regret: $\sum_{t=1}^{T} \sum_{m=1}^{M} \mathbb{E}[Q_m(T)]$.
- Convex optimization with bandit feedback and unknown (stochastic) constraints.
- Regret lower bound: $\Omega(\sqrt{T})$. 

Related Works

First-order Methods

• Use utility observations to construct approximate gradients of the utility functions [2][3].

• Cannot achieve order-optimal regret due to the variance of approximate gradients.

Zeroth-order Methods

• Maintain a target region that contains the optimal solution, and directly use utility observations to shrink the target region. Robust to noise, but cannot handle stochastic constraints [4].

• Main Contribution: Zeroth-order algorithm for elastic job scheduling with order-optimal regret.


The SCBA Algorithm

Stochastic Convex Bandit Algorithm [4]:

$$\max_x F(x)$$

s.t.  \( x \in \mathcal{X} \)

- \( F \) is concave and Lipschitz-continuous. item Query \( x(t) \), observe \( F(x(t)) + \epsilon_t \).
- Achieves \( \tilde{O}(\sqrt{T}) \)-regret if \( \mathcal{X} \) is known.
- Output a sequence of query points \( x_1, \ldots, x_T \) that
  \[ \sum_{t=1}^{T} \mathbb{E}[F(x^*) - F(x(t))] \leq \tilde{O}(\sqrt{T}) \].

The SCBA Algorithm

Stochastic Convex Bandit Algorithm:

- Maintain a target region (initialized as $\mathcal{X}$).
- Repeatedly query a set of points in the target region.
- One query corresponds to one $x_t$. The $x_t$ for multiple $t$’s may take the same value.
- Construct confidence intervals of function values of the queried points.
- Eliminate a fraction of the target region (thereby shrink the target region) based on the constructed confidence intervals.
Transformed Optimization Problem

\[ \mathcal{P} : \max_x \sum_{k=1}^{K} \lambda_k f_k(x_k) \]
\[ \text{s.t. } x \in \Lambda, \quad x_k \in [0, B], \quad \forall k. \]

\[ \tilde{\mathcal{P}} : \max_x F(x) := \sum_{k=1}^{K} \lambda_k f_k(x_k) - L \cdot \Delta(x, \Lambda) \]
\[ \text{s.t. } x_k \in [0, B], \quad \forall k. \]

- \( \Delta(\cdot, \Lambda) \) denotes the \( l_1 \)-distance from \( x \) to \( \Lambda \).

**Theorem**

\( \mathcal{P} \) and \( \tilde{\mathcal{P}} \) are equivalent.

- \( \mathcal{P} \) and \( \tilde{\mathcal{P}} \) have the same optimal solution.
- \( \tilde{\mathcal{P}} \) is a convex problem with Lipschitz-continuous objective function.
Transformed Optimization Problem

\[ \tilde{P} : \max_x F(x) := \sum_{k=1}^{K} \lambda_k f_k(x_k) - L \cdot \Delta(x, \Lambda) \]

s.t. \( x_k \in [0, B], \quad \forall k. \)

- \( \tilde{P} \) does not involve stochastic constraints.
- However, there is no unbiased observations available for the objective function \( F(x) \).
  - Design a special procedure to construct confidence intervals for \( F(x) \).
Confidence Interval Construction

If $x(t) = x$ for $t \in \{t_1, t_1 + 1, \ldots, t_2\}$, how to construct confidence intervals around $F(x)$?

- Assume (for now) utility observation is immediately available after the corresponding job size decision.
- Using empirical means, we have:
  - Confidence interval of $f_k(x_k)$: $[f_k^l(x_k), f_k^u(x_k)]$.
  - Confidence interval of $\lambda_k$: $[\lambda_k^l, \lambda_k^u]$.
  - Confidence interval of $\mu_m$: $[\mu_m^l, \mu_m^u]$.

- These confidence intervals can propagate to a confidence interval of $F(x)$. 
Confidence Interval Construction

Recall the set \( \Lambda \):

\[
\Lambda := \left\{ (x_1, \ldots, x_K) \right\} \quad \begin{array}{l}
\exists \{\alpha\}_{km}, \text{ s.t.} \\
\sum_k \alpha_{km} \lambda_k x_k \leq \mu_m \quad \forall s_m, \\
\alpha_{km} = 0, \quad \forall s_m \notin S_{u_k} \\
\sum_m \alpha_{km} = 1, \forall k \\
\alpha_{km} \geq 0, \quad \forall k, m \\
0 \leq x_k \leq B, \quad \forall k
\end{array}
\]

- \( \Delta(x, \Lambda) \) is parameterized by \( \Delta(x, \lambda, \mu) \), where \( \lambda = (\lambda_1, \ldots, \lambda_K) \), \( \mu = (\mu_1, \ldots, \mu_M) \).
Confidence Interval Construction

- \( \Delta(x, \Lambda) \) is parameterized by \( \Delta(x, \lambda, \mu) \), where \( \lambda = (\lambda_1, \ldots, \lambda_K) \), \( \mu = (\mu_1, \ldots, \mu_M) \).
- \( \Delta(x, \lambda^u, \mu^l) \geq \Delta(x, \lambda, \mu) \geq \Delta(x, \lambda^l, \mu^u) \).
- \( F^l(x) := \sum_k \lambda^l_k f^l_k(x_k) - L \cdot \Delta(x, \lambda^u, \mu^l) \).
- \( F^u(x) := \sum_k \lambda^u_k f^u_k(x_k) - L \cdot \Delta(x, \lambda^l, \mu^u) \).
- \([F^l(x), F^u(x)]\) satisfies the requirements of SCBA.
- SCBA algorithm embedded with the confidence interval construction procedure can be used to solve \( \tilde{\mathcal{P}} \).
The CI-JSQ Policy

Routing

• The routing is done based on the Join-the-Shortest-Queue (JSQ) rule.

• Send the jobs to the server with the smallest $Q_m(t)$.

Confidence Interval Join-the-Shortest-Queue (CI-JSQ) Policy:

• For each time $t$, decide on the job size based on the output of SCBA embedded with the confidence interval construction procedure.

• Route the jobs based on the JSQ rule.
The CI-JSQ Policy

The CI-JSQ policy achieves $\tilde{O}(\sqrt{T})$-regret.

\begin{itemize}
    \item $\sum_{t=1}^{T} \mathbb{E}[F(x^*(t)) - F(x(t))] \leq \tilde{O}(\sqrt{T})$ from the guarantee of SCBA.
    \item Can show that:
        \begin{itemize}
            \item $\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k \mathbb{E}[f_k(x^*_k) - f_k(x_k(t))] \leq \tilde{O}(\sqrt{T})$ (utility regret).
            \item $\sum_{t=1}^{T} \mathbb{E}[\Delta(x(t), \Lambda)] \leq \tilde{O}(\sqrt{T})$ (constraint violation).
        \end{itemize}
    \item Using properties of JSQ, the queueing regret is also in $\tilde{O}(\sqrt{T})$ from the constrain violation bound.
    \item An episodic scheme (with the same order of regret) that deals with feedback delay.
\end{itemize}
• We considered the problem of elastic job scheduling with unknown utility functions.
• The problem can be viewed as a problem of convex optimization with bandit feedback and stochastic constraints.
• We combined the SCBA with a confidence interval construction procedure to form a policy with order-optimal regret for the elastic job scheduling problem.
Thank You!

– xinzhe@mit.edu