Importance Sampling for a Non-Premptive Static Priority Queue

Xinyun Chen Jin Guang Guiyu Hong

March 11, 2022

Contents

1 Introduction

Nowadays, the computer network has a large number of switches and routers so that we can send and receive messages stably and with low latency. As the time goes on, this requirement is getting more and more strict. It may require more or faster switches, but this will also increase the cost. One way without increasing the hardware cost is to improve scheduling policies or routing algorithms.

In order to measure the quality of different algorithms, we need to introduce a common performance measure for computer networks: service-level agreement (SLA) [HB21], which is a commitment between the service provider and the client. Specifically, clients hope that the response time, that is, the end-to-end delay, can always be controlled below a certain threshold, such as 10ms, so that clients can have a comfortable experience. However, due to the randomness of the traffic of the computer network, it cannot be completely guaranteed, so the provider introduces SLA, which proposes that the customer's requirements can be satisfied with a high probability like 99.999%.

However, it is quite difficult to find a suitable threshold to meet the SLA, because the probability of not meeting the guarantee is too small. For example, a 99.999% SLA means that if we do 100,000 experiments, on average only one is observed successfully, so the final estimated probability has a large variance.

Importance sampling (IS) is a useful and powerful variance reduction technique in the stochastic simulation field, so it can be used to estimate the above probability with a low variance. A literature review of the IS for the SLA in the computer network is summarized in the another report (see survey), so we will not go over them here.

In the rest of this section, we will first introduce the problem setting about a non-preemptive single station with the static priority scheduling policy in the subsection 1.1. Later, we will later try different but commonly used scheduling policies, like weighted fair queue (WFQ) and deficit round robin (DRR). Since the SLA relies on the stationary response time, we need to use a steady-state simulation technique, regenerative method [RK16], which will be introduced in the subsection 1.2. In order to use the IS method to reduce the variance, we will briefly introduce the IS in the subsection 1.3. Finally, we will introduce the cross-entropy method [RK16] to find the optimal parameters iteratively, in the subsection 1.4.

1.1 Problem Setting

In this report, consider a K-class non-preemptive single station with the static priority scheduling policy, where customers have priorities $w_1 > w_2 > \cdots > w_K$. Since there are huge gaps between the literature and the reality (see survey), in order to simplify the problem, we first assume the Markovian setting, and later extend to the Markov arrival process setting. Specifically, suppose arrival processes following Poisson processes with rates $\lambda_1, \ldots, \lambda_K$ and service times following exponential distributions with rates $\mu_1, \ldots, \mu_K$, respectively.

In the mathematical description, the SLA target is to find a q-tile response time, that is, a small enough threshold γ such that the response time R cannot exceed the threshold with high probability q:

$$\inf\{\gamma : P(R < \gamma) > q\} = \{\gamma : P(R > \gamma) < 1 - q\}. \tag{1}$$

Since the q-tile response time relies on the estimation of the probability of a rare event, we will first focus on how to reduce the variance of the rare event simulation, and then transfer to the q-tile response time.

In the rare event simulation, we usually use the relative error (RE) as the performance measure. Suppose Z is an estimator, the RE will be the standard deviation over the expectation, $\sigma_Z/\mathbb{E}[Z]$. To motivate focusing on the relative error, suppose, for example, that we obtain a point estimator Z of order 10^{-5} and a confidence interval of half-width 10^{-4}. This confidence interval may look narrow, but it does not help to tell whether the true mean $\mathbb{E}[Z]$ is of the magnitude 10^{-4}, 10^{-5}, or even much smaller.

1.2 Regenerative Method

In this subsection, we will introduce the regenerative method , which is a steady-state simulation technique, based on the property of the regenerative process.

A stochastic process $\{X_t\}$ is called regenerative if there exist random time points $T_0 < T_1 < T_2 < \ldots$, such that at each time point the process restarts probabilistically. More precisely, the process $\{X_t\}$ can be split into iid

replicas during intervals, called cycles, of lengths $\tau_i = T_i - T_{i-1}, i = 1, 2, \ldots$. The main strengths of the concept of regenerative processes are that the existence of limiting distributions is guaranteed under very mild conditions and the behavior of the limiting distribution depends only on the behavior of the process during a typical cycle.

If $\mathbb{E}[\tau] < \infty$, the process $\{X_t\}$ has a limiting (or steady-state) distribution, and the steady-state expected value, $\ell = \mathbb{E}[H(X)]$, is given by

$$\ell = \mathbb{E}[H(X)] = \frac{\mathbb{E}[Y]}{\mathbb{E}[\tau]}, \tag{2}$$

where $Y = \sum_{t=1}^{\tau} H(X_t)$. And it also implies that $(Y_i, \tau_i), i = 1, 2, \ldots$, is a sequence of iid random vectors.

1.3 Importance Sampling

Importance sampling (IS) obtains variance reduction by modifying the original distribution f by another distribution g, so that most samples contribute the most to μ:

$$\mu = \mathbb{E}_f[H(X)] = \int H(x)f(x)dx = \int H(x)\frac{f(x)}{g(x)}g(x)dx = \mathbb{E}_g\left[H(X)\frac{f(X)}{g(X)}\right], \tag{3}$$

where $\mathbb{E}_f$ means that we take the expectation over the distribution f. Hence, we have an unbiased estimator $H(X)L(X)$ for μ, named as IS estimator, with X following the IS distribution g, where the likelihood ratio is $L(X) = f(X)/g(X)$. Consequently, if $X_1, X_2, \ldots, X_N$ is random samples from distribution g, that is, $X_1, X_2, \ldots, X_N$ are iid random vectors with density g, then

$$\hat{\mu} = \frac{1}{N}\sum_{k=1}^{N} H(X_k)\frac{f(X_k)}{g(X_k)}. \tag{4}$$

The variance of the IS estimator highly depends on the quality of IS distribution g, and there is a zero-variance distribution if $H(X) \geq 0$:

$$g^*(x) = \frac{H(x)}{\mathbb{E}_f[H(X)]}f(x) = \frac{H(x)}{\int H(x)f(x)dx}f(x), \tag{5}$$

with the likelihood ratio $L^*(X) = \mathbb{E}_f[H(X)]/H(X)$. However, this is unreachable since we does not know the value of $\mathbb{E}_f[H(X)]$.

1.4 Cross-Entropy Algorithm

Although the zero-variance IS distribution is unreachable, we can use cross entropy (CE) or Kullback–Leibler divergence to choose a sub-optimal IS distribution. Specifically, we want to find an IS distribution:

$$\operatorname*{argmin}_h \mathrm{KL}(g^* \| h) = \operatorname*{argmax}_h \int g^*(x)\log h(x)dx = \operatorname*{argmax}_h \int H(x)f(x)\log h(x)dx. \tag{6}$$

If we optimize over all distributions h, then it is immediate that the CE optimal distribution coincides with the optimal distribution g^*.

However, the CE method is not suitable for the rare event simulation directly, since for rare events the sample objective is 0 with high probability. To overcome this difficulty, we will use an adaptive approach to select a good IS distribution by adopting a two-stage procedure where both the threshold γ and the IS distribution are updated.

For example, we want to estimate the probability of the rare event $\{\varphi(X) \geq \gamma\}$ for a large γ. To circumvent this problem, we will use an adaptive approach where we generate a sequence of IS distributions $\{g_t, t \geq 0\}$ and a sequence of thresholds $\{\gamma_t, t \geq 0\}$ while iterating in both γ_t and g_t. Our ultimate goal is to have g_t close to g^* after some number of iterations and finally to use g_t to estimate the probability.

2 Algorithms

In this section, we will first show the procedure of the importance sampling method for this setting in the algorithm 1, and the detailed expression of the likelihood ratio. Then, the cross-entropy method will be used to find an optimal or sub-optimal parameter in the subsection 2.3. Finally, we will give a heuristic approach to convert the tail probability evaluation to the q-tile evaluation in the subsection 2.4.

2.1 Event-based Simulation with Importance Sampling

In order to simulate the queuing system, we use the Python package 'ns.py', which is an event-based simulation framework based on the package 'simpy'. In the algorithm 1, we will show how to use IS method to estimate the nominator of the regenerative method in the equation (2). Besides, the denominator will be estimated by naive simulation, since the cycle length does not need to use IS to estimate.

Note that the regenerative method has the independent sample paths, so the for loop can be implemented using parallelization acceleration.

2.2 Filtration and Likelihood Ratio

Let $A_{k,i}$ be the inter-arrival time between packet $i-1$ and packet i of flow k, and $S_{k,i}$ be the service time for packet i of flow k. Note that there is no packet at time 0, so $A_{k,1}$ is just the arrival time for packet 1 of flow k.

Now consider the filtration and likelihood ratio of the newly served packet of flow j, which starts to be served at time t. Let n_k be the number of packets for flow k, which have been served before the current packet. It can be seen that the filtration relies on the 3 parts:

1. Higher priority. For flow k, this includes any packet $1, 2, \ldots, n_k$ and the inter-arrival of the following packet $n_k + 1$:
$$\sigma(A_{k,1}, \ldots, A_{k,n_k}, A_{k,n_k+1}, S_{k,1}, \ldots, S_{k,n_k})$$
The likelihood ratio corresponds to
$$\frac{d\mathbb{P}(A_{k,1} = a_{k,1}, \ldots, A_{k,n_k} = a_{k,n_k}, S_{k,1} = s_{k,1}, \ldots, S_{k,n_k} = s_{k,n_k})}{d\mathbb{Q}(A_{k,1} = a_{k,1}, \ldots, A_{k,n_k} = a_{k,n_k}, S_{k,1} = s_{k,1}, \ldots, S_{k,n_k} = s_{k,n_k})} \cdot \frac{\mathbb{P}(\sum_{i=1}^{n_k} a_{k,i} + A_{k,n_k+1} \geq s)}{\mathbb{Q}(\sum_{i=1}^{n_k} a_{k,i} + A_{k,n_k+1} \geq s)} \tag{7}$$
where $\mathbb{P}$ is the original probability measure with the distribution f and $\mathbb{Q}$ is the importance probability measure with the IS distribution g.

2. Same priority. For flow j, this includes any packet $1, 2, \ldots, n_j$ and the inter-arrival of the following packet $n_j + 1$:
$$\sigma(A_{j,1}, \ldots, A_{j,n_j}, A_{j,n_j+1}, S_{j,1}, \ldots, S_{j,n_j}, S_{j,n_j+1})$$
The likelihood ratio corresponds to
$$\frac{d\mathbb{P}(A_{k,1} = a_{k,1}, \ldots, A_{k,n_k} = a_{k,n_k}, A_{k,n_k+1} = a_{k,n_k+1}, S_{k,1} = s_{k,1}, \ldots, S_{k,n_k} = s_{k,n_k}, S_{k,n_k+1} = s_{k,n_k+1})}{d\mathbb{Q}(A_{k,1} = a_{k,1}, \ldots, A_{k,n_k} = a_{k,n_k}, A_{k,n_k+1} = a_{k,n_k+1}, S_{k,1} = s_{k,1}, \ldots, S_{k,n_k} = s_{k,n_k}, S_{k,n_k+1} = s_{k,n_k+1})} \tag{8}$$

3. Lower priority. For flow k, this includes any packet $1, 2, \ldots, n_k$, but if there is no packet served $n_k = 0$, the first one $A_{k,1}$ must need to be considered during the idle system:
$$\sigma(A_{k,1}, \ldots, A_{k,n_k}, S_{k,1}, \ldots, S_{k,n_k}, A_{k,1})$$
The likelihood ratio corresponds to
$$\frac{d\mathbb{P}(A_{k,1} = a_{k,1}, \ldots, A_{k,n_k} = a_{k,n_k}, S_{k,1} = s_{k,1}, \ldots, S_{k,n_k} = s_{k,n_k})}{d\mathbb{Q}(A_{k,1} = a_{k,1}, \ldots, A_{k,n_k} = a_{k,n_k}, S_{k,1} = s_{k,1}, \ldots, S_{k,n_k} = s_{k,n_k})} 1_{n_k \neq 0} + \frac{\mathbb{P}(A_{k,1} > \min_{i=1,\ldots,K} a_{i,1})}{\mathbb{Q}(A_{k,1} > \min_{i=1,\ldots,K} a_{i,1})} 1_{n_k = 0} \tag{9}$$

Finally, multiply these three equations (7)-(9) together, and we will get the likelihood ratio.

Algorithm 1: Importance Sampling for Non-Preemptive Static Priority Queue

Consider a K-type static priority queue, with arrival rates $\lambda_{1,0}, \ldots, \lambda_{K,0}$, and service rates $\mu_{1,0}, \ldots, \mu_{K,0}$.
Input: the number of regenerative cycles N, tilted arrival rates $\lambda_{1,1}, \ldots, \lambda_{K,1}$, and tilted service rates $\mu_{1,1}, \ldots, \mu_{K,1}$.
for $j = 1, 2, \ldots, N$ **do**
  Initialize the current time $t = 0$, the number of packets that have arrived $m_1, \ldots, m_K = 0, \ldots, 0$, the number of packets being served $n_1, \ldots, n_K = 0, \ldots, 0$, the IS distribution phase $\phi = 1$ and the current log-likelihood ratio $L = 0$.
  Generate arrival times of the first packet for each type, $A_{1,1}, \ldots, A_{K,1}$, following the distributions $\exp(\lambda_{1,\phi}), \ldots, \exp(\lambda_{K,\phi})$, respectively.
  while *True* **do**
    Find the next event: type-k arrives (E_1), type-k starts to be served (E_2), type-k departs (E_3).
    if *type-k arrives (E_1)* **then**
      A type-k packet comes with $m_k := m_k + 1$.
      Generate the packet length (service time) S_{k,m_k} following the distribution $\exp(\mu_{k,\phi})$, and record the service phase $\phi^s_{k,m_k} := \phi$.
      if *k is the target: $k = k'$* **then**
        Compute the lower bound of the response time R_{k,m_k} for this packet. Change the phase ϕ to 0 and mark it as the critical packet, if $R_{k,m_k} > \gamma_k$ (policy 1). Note that for the highest priority, this is the exact response time.
      else if *k has a higher priority: $k < k'$* **then**
        Update the lower bound of response times of packets of flow k'. Change the phase ϕ to 0 and mark the packet that meets the condition as the critical packet, if the lower bound of response time of the following packet of flow k' exceeds $\gamma_{k'}$ (policy 2) or there exists a packet i of flow k' with $R_{k',i} > \gamma_{k'}$ (policy 3).
      Generate the inter-arrival times A_{k,m_k+1} following the distributions $\exp(\lambda_{k,\phi})$, and record the arrival phase $\phi^a_{k,m_k+1} := \phi$.
    if *type-k packet i starts to be served (E_2)* **then**
      Add the arrival part for likelihood ratio:

$$L := L + \text{logpdf}(\lambda_{k,0}, A_{k,i}) - \text{logpdf}(\lambda_{k,\phi^a_{k,i}}, A_{k,i}).$$

      if *Current phase ϕ is 0 but the service phase $\phi^s_{k,i}$ is 1 and the critical packet has been served.* **then**
        Re-sample the service time $S_{k,i}$ from the distribution $\exp(\mu_{k,\phi})$, and record phase $\phi^s_{k,i} := \phi$.
      Add the service part for likelihood ratio:

$$L := L + \text{logpdf}(\mu_{k,0}, S_{k,i}) - \text{logpdf}(\mu_{k,\phi^s_{k,i}}, S_{k,i}).$$

      Assign the log-likelihood ratio for this packet: $L_{k,i} := L + \text{tail}(n_1, \ldots, n_K)$.
    if *type-k departs (E_3)* **then**
      A type-k packet departs with $n_k := n_k + 1$.
      Check whether the system is idle. If so, break the loop.
  For each threshold γ_j, compute:

$$Y_j = \sum_{i=1}^{n_{k'}} 1_{\{R_{k',i} > \gamma_j\}} e^{L_{k',i}}$$

Compute the mean and confidence interval.

2.3 Cross-Entropy Method

The importance sampling estimator for the numerator is

$$E_{\mathbb{P}}\left[\sum_{i=1}^{\sigma_{k'}} 1_{R_{k',i}>\gamma_{k'}}\right] = E_{\mathbb{Q}}\left[\sum_{i=1}^{\sigma_{k'}} 1_{R_{k',i}>\gamma_{k'}} L_i\right], \tag{10}$$

where $\sigma_{k'}$ is the number of packets of flow k' in a regenerative cycle, and denote $E_{\mathbb{Q}}[H] = E_{\mathbb{Q}}\left[\sum_{i=1}^{\sigma_{k'}} 1_{R_{k',i}>\gamma_{k'}} L_i\right]$. The corresponding zero-variance IS distribution is

$$\mathbb{Q}^*(dx) = \frac{H(x)}{E_{\mathbb{Q}}[H]}\mathbb{Q}(dx), \tag{11}$$

where x is a sample path. The cross-entropy method is to find a measure $\mathbb{Q}_1$ with density h such that

$$\text{argmin}_{\mathbb{Q}_1} \text{KL}(\mathbb{Q}^* \| \mathbb{Q}_1) = \text{argmax}_h \int \log h(x) \mathbb{Q}^*(dx) = \text{argmax}_h \int H(x) \log h(x) \mathbb{Q}(dx). \tag{12}$$

Suppose we only change the measure of phase 1 to a new phase, say $u(1)$ with $u(0) = 0$, and then the density h of one path x sampled from $\mathbb{Q}$ is

$$h(x) = \prod_{k=1}^{K} \left(\prod_{i=1}^{\sigma_k+1} \lambda_{k,u(\phi^a_{k,i})} \exp(-\lambda_{k,u(\phi^a_{k,i})} a_{k,i}) \prod_{i=1}^{\sigma_k} \mu_{k,u(\phi^s_{k,i})} \exp(-\mu_{k,u(\phi^s_{k,i})} s_{k,i}) \right), \tag{13}$$

and

$$\log h(x) = \sum_{k=1}^{K} \left(\sum_{i=1}^{\sigma_k+1} \left(\log \lambda_{k,u(\phi^a_{k,i})} - \lambda_{k,u(\phi^a_{k,i})} a_{k,i} \right) + \sum_{i=1}^{\sigma_k} \left(\log \mu_{k,u(\phi^s_{k,i})} - \mu_{k,u(\phi^s_{k,i})} s_{k,i} \right) \right),$$

and hence, the objective is to maximize

$$E_{\mathbb{Q}}\left[H(X) \sum_{k=1}^{K} \left(\sum_{i=1}^{\sigma_k+1} \left(\log \lambda_{k,u(\phi^a_{k,i})} - \lambda_{k,u(\phi^a_{k,i})} A_{k,i} \right) + \sum_{i=1}^{\sigma_k} \left(\log \mu_{k,u(\phi^s_{k,i})} - \mu_{k,u(\phi^s_{k,i})} S_{k,i} \right) \right) \right]. \tag{14}$$

Consider the empirical objective, and then the derivative and expectation can be exchanged:

$$\nabla_{\lambda_{k,u(1)}} \text{obj} = \frac{1}{N} \sum_{j=1}^{N} \left[H(x_j) \sum_{i=1}^{\sigma_k+1} \left(\frac{1}{\lambda_{k,u(1)}} - a^j_{k,i} \right) 1_{\phi^a_{k,i}=1} \right] = 0$$

$$\nabla_{\mu_{k,u(1)}} \text{obj} = \frac{1}{N} \sum_{j=1}^{N} \left[H(x_j) \sum_{i=1}^{\sigma_k} \left(\frac{1}{\mu_{k,u(1)}} - s^j_{k,i} \right) 1_{\phi^s_{k,i}=1} \right] = 0,$$

and hence, the cross-entropy parameters are

$$\lambda_{k,u(1)} = \frac{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k+1} 1_{\phi^a_{k,i}=1}}{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k+1} a^j_{k,i} 1_{\phi^a_{k,i}=1}}, \quad k = 1, \ldots, K \tag{15}$$

and

$$\mu_{k,u(1)} = \frac{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k} 1_{\phi^s_{k,i}=1}}{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k} s^j_{k,i} 1_{\phi^s_{k,i}=1}}, \quad k = 1, \ldots, K. \tag{16}$$

Therefore, the cross-entropy method can be summarized as the following algorithm. Note that θ can be updated serveral times although $\gamma_t \geq \gamma_{k'}$.

Algorithm 2: Cross-Entropy Method for Priority Queue

Consider a K-type priority queue, with arrival rates $\lambda_{1,0}, \ldots, \lambda_{K,0}$, and service rates $\mu_{1,0}, \ldots, \mu_{K,0}$.
Input: the number of sample paths N, the quantile hyper-parameter ρ and the target flow k'.
Initialize CONTINUE=True and $t = 0$. Start from the original probability measure $\mathbb{P}$ with distribution f where $h_{\theta_0} = f$.
while *CONTINUE* **do**
$t := t + 1$.
Adaptive updating of γ_t. For a fixed θ_{t-1}, derive N sample paths $x_1, \ldots, x_N$ from $h_{\theta_{t-1}}$ with the length $\sigma_{k',1}, \ldots, \sigma_{k',N}$ and then the γ_t will be the $(1-\rho)$-quantile of

$$\left\{ \max_{i=1,\ldots,\sigma_{k',j}} R_{k',i} : j = 1, \ldots, N \right\}.$$

if $\gamma_{k'} < \gamma_t$ **then**
$\gamma_t = \gamma_{k'}$ and set CONTINUE as False.
Adaptive updating of θ_t. For fixed γ_t and θ_{t-1}, draw N sample paths $x_1, \ldots, x_N$ from $h_{\theta_{t-1}}$ with the length $\sigma_{k',1}, \ldots, \sigma_{k',N}$ and then derive θ_t as

$$\lambda_{k,t} = \frac{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k+1} 1_{\phi^a_{k,i}=1}}{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k+1} a^j_{k,i} 1_{\phi^a_{k,i}=1}}, \quad k = 1, \ldots, K \tag{17}$$

and

$$\mu_{k,t} = \frac{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k} 1_{\phi^s_{k,i}=1}}{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k} s^j_{k,i} 1_{\phi^s_{k,i}=1}}, \quad k = 1, \ldots, K, \tag{18}$$

where $h(x) = \sum_{i=1}^{\sigma_{k'}} 1_{R_{k',i} > \gamma_t} L_i$.
Result: A good change of measure with arrival rates $\lambda_{1,t}, \ldots, \lambda_{K,t}$, and service rates $\mu_{1,t}, \ldots, \mu_{K,t}$.

2.4 q-tile Response time Evaluation

In order to get a q-tile response time evaluation, the first thing is to find a good parameter by using cross-entropy method. Next, use this parameter to evaluate probabilities of many thresholds, and then we can estimate a confidence interval of the q-tile response time by them.

Naive simulation for the cycle length. Before finding a good parameter, we need run a naive simulation to get an estimation of the cycle length, which is the denominator in the regenerative method. Note that we do not use IS method here, since the cycle length doesn't have much variance in the naive simulation.

Find a good parameter. The cross-entropy method is shown in the following algorithm 3. Not only a good parameter can be found, we can also give an interval which contains the q-tile response time. However, in practice, running the same simulation twice in a loop is too wasteful, so we change their order, i.e., update the parameters θ first and then update the threshold γ. Moreover, since the cross-entropy method should converge to the optimal parameter, we update the parameter θ several times without updating the threshold.

Algorithm 3: q-tile Cross-Entropy Method for Priority Queue

Consider a K-type priority queue, with arrival rates $\lambda_{1,0}, \ldots, \lambda_{K,0}$, and service rates $\mu_{1,0}, \ldots, \mu_{K,0}$.
Input: the number of sample paths N, the quantile hyper-parameter ρ and the target flow k'.
Initialize CONTINUE=True and $t = 0$. Start from the original probability measure $\mathbb{P}$ with distribution f where $h_{\theta_0} = f$.
while *CONTINUE* **do**
$t := t + 1$.
Adaptive updating of γ_t. Generate N sample paths $x_1, \ldots, x_N$ from the distribution $h_{\theta_{t-1}}$ with the length $\sigma_{k',1}, \ldots, \sigma_{k',N}$, and compute the sample $(1-\rho)$-quantile γ_t, according to

$$\left\{ \max_{i=1,\ldots,\sigma_{k',j}} R_{k',i} : j = 1, \ldots, N \right\}.$$

Adaptive updating of θ_t. Use the same sample paths $x_1, \ldots, x_N$ to solve the stochastic version of the program (14). The solution is

$$\lambda_{k,t} = \frac{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k+1} 1_{\phi^a_{k,i}=1}}{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k+1} a^j_{k,i} 1_{\phi^a_{k,i}=1}}, \quad k = 1, \ldots, K, \tag{19}$$

and

$$\mu_{k,t} = \frac{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k} 1_{\phi^s_{k,i}=1}}{\sum_{j=1}^{N} H(x_j) \sum_{i=1}^{\sigma_k} s^j_{k,i} 1_{\phi^s_{k,i}=1}}, \quad k = 1, \ldots, K, \tag{20}$$

where $H(x) = \sum_{i=1}^{\sigma_{k'}} 1_{R_{k',i} > \gamma_t} L_i$. At the same time, evaluate the sample mean and the confidence interval of $H(x_1), \ldots, H(x_N)$.
if *the upper confidence interval of sample probability, from $H(x_1), \ldots, H(x_N)$ and cycle length in the naive simulation, is smaller than* $(1-q)$ **then**
Save γ_t as the upper bound of the threshold γ_u for the q-tile.
Set CONTINUE as False.
else
Save γ_t as the lower bound of the threshold γ_l for the q-tile.
Result: A good change of measure with arrival rates $\lambda_{1,t}, \ldots, \lambda_{K,t}$, and service rates $\mu_{1,t}, \ldots, \mu_{K,t}$. And the upper and lower bound of the threshold γ_u, γ_l for q-tile waiting time.

Narrow the lower and upper bound of the q-tile response time. The range of two bounds, $\gamma_u - \gamma_l$, is often very large, because the threshold increases very fast in the CE algorithm, which is why the CE has few iterations to converge. In order to narrow the bound, we can evaluate many thresholds at once, $\gamma_1, \ldots, \gamma_M$ in (γ_l, γ_u). Note that here we need to balance the accuracy and time of the search, that is, the finer the search, the more time we need. Therefore, in practice, we use few cycles for 100 thresholds uniformly in (γ_l, γ_u) to roughly narrow the range. After roughly narrowing down the range a few times, we do a precise search with more cycles. Specifically, we use lots of cycles for 100 thresholds uniformly in (γ_l, γ_u), which now is has small range.

Note that during this stage, we can also use the cross-entropy method to update the parameter, since the maximum threshold γ_u changes.

q-tile estimation. Let $\hat{F}(\gamma)$ be the sample mean of tail probability at γ, and $\bar{F}(\gamma), \underline{F}(\gamma)$ be the upper and lower confidence interval of tail probability at γ. The quantile estimator will be $\widehat{F}^{-1}(q) = \inf\{s : F(s) \geq q\}$, with the confidence interval $\left[\underline{F}^{-1}(q), \bar{F}^{-1}(q)\right]$.

3 Experiments

In this section, we consider several settings for the 2-class static priority queue. For each experiment, we will find the q-tile response time γ_q first ($q = 0.999, 0.99999$), and then evaluate the relative error of the probability $P(R > \gamma_q)$ to compare with the naive simulation. If unspecified, some hyper-parameters default to the following: the number of cycles in the naive simulation is 5,000; the extra iteration in the cross-entropy method is 2; the number of cycles in the rough search is 10,000; the condition to exit the rough search is $\gamma_u - \gamma_l < 1$; the number of cycles in the precise search and final evaluation is 100,000; the number of thresholds in both rough and precise search is 100; the number of CPUs for regeneration cycles is 16.

3.1 Setting 1: small packet has high priority

A good policy for priority queue is assign the smaller packet with the higher priority, so we first consider the setting where packets with the higher priority have more demanding. Specifically, set arrival rates $\lambda_1 = 0.6$ and $\lambda_2 = 0.2$, service rates $\mu_1 = 2$ and $\mu_2 = 1$, for higher and lower priority class, respectively.

Evaluate 0.999-tile response time of the higher priority. First, run the naive simulation to get the estimation of the cycle length. The experiment shows the sample mean of higher priority is 1.49824 with RE 0.00399, and that of lower priority is 0.5011 with RE 0.00609. It turns out that the relative error of the cycle length in the naive simulation is quite small. Second, run the cross-entropy method to update the parameter and find the lower and upper bounds of the 0.999-tile response time, as shown in the table 3.1. Then the bound of the 0.999-tile response time is $(\gamma_l.\gamma_u) = (5.637, 8.362)$. Third, narrow the bound of 0.999-tile response time. One-step rough search will result in the bound $(6.7929, 6.9856)$, and the precise search will give the estimation of the 0.999-tile response time 6.9009 and the confidence interval $(6.8649, 6.9350)$. Note that the above three steps spend 144.6s.

iteration	λ_1	λ_2	μ_1	μ_2	delay γ	$P(R > \gamma)$	RE	running time
0	0.6	0.2	2	1	0	0.999838	0.0187	4.19s
1	0.733	0.215	1.479	0.569	1.756	0.183043	0.0385	4.17s
2	0.843	0.297	1.036	0.425	3.260	0.038570	0.0470	4.26s
3	0.878	0.391	1.029	0.347	5.637	0.003305	0.0472	4.35s
4	0.924	0.483	0.992	0.238	8.362	0.000203	0.0651	5.20s
5	0.908	0.554	1.001	0.183	8.362	0.000220	0.0596	5.15s
6	0.939	0.613	1.002	0.182	8.362	0.000247	0.1108	5.37s
7	0.772	0.578	1.063	0.197	8.362	0.000233	0.0968	5.25s
8	0.891	0.618	1.093	0.184	8.362	0.000245	0.0564	5.25s
9	0.937	0.667	1.084	0.174	8.362	0.000259	0.1711	5.47s

Table 1: The evolution for parameters and thresholds in the cross-entropy method.

Finally, evaluate the probability $P(R > 6.9009)$ to compare with the naive simulation. The experiment spends 27.08s and shows the estimation is 1.0054e-3 with RE 0.0152, but the naive simulation with the same cycles spends 13.29s and has the estimation 7.4754e-4 with RE 0.1825. It can be seen that although the IS method takes twice longer than the naive simulation, the RE of IS is only 1/12 of that of the naive simulation. So it's increased by about 72 times.

Evaluate 0.999-tile response time of the lower priority. First, run the naive simulation to get the estimation of the cycle length. Second, run the cross-entropy method to update the parameter and find the lower and upper bounds of the 0.999-tile response time, as shown in the table 3.1. Then the bound of the 0.999-tile response time is $(\gamma_l.\gamma_u) = (10.942, 21.590)$. Third, narrow the bound of 0.999-tile response time. One-step rough search will result in the bound $(16.750018, 17.180237)$, and the precise search will give the estimation of the 0.999-tile response time 16.8304 and the confidence interval $(16.7631, 16.9108)$. Note that the above three steps spend 113.91s.

Finally, evaluate the probability $P(R > 16.8304)$ to compare with the naive simulation. The experiment spends 23.58s and shows the estimation is 1.0078e-3 with RE 0.0129, but the naive simulation with the same cycles spends

iteration	λ_1	λ_2	μ_1	μ_2	delay γ	$P(R > \gamma)$	RE	running time
0	0.6	0.2	2	1	0	0.987011	0.0275	4.15s
1	0.534	0.403	1.430	0.805	1.766	0.378682	0.0317	4.14s
2	0.635	0.619	1.180	0.536	4.527	0.105789	0.0431	4.36s
3	0.771	0.646	1.223	0.483	10.942	0.008589	0.0532	4.69s
4	0.842	0.624	1.360	0.478	21.590	0.000190	0.0691	5.12s
5	0.849	0.608	1.363	0.452	21.590	0.000164	0.0603	4.98s
6	0.866	0.597	1.352	0.447	21.590	0.000182	0.0671	4.95s
7	0.858	0.588	1.370	0.445	21.590	0.000176	0.0599	4.98s
8	0.857	0.577	1.359	0.449	21.590	0.000182	0.0788	4.97s
9	0.864	0.552	1.339	0.444	21.590	0.000180	0.0663	5.09s

Table 2: The evolution for parameters and thresholds in the cross-entropy method.

13.48s and has the estimation 9.7785e-4 with RE 0.2684. It can be seen that although the IS method takes twice longer than the naive simulation, the RE of IS is only 1/21 of that of the naive simulation. So it's increased by about 220 times.

Evaluate 0.99999-tile response time of the higher priority. First, run the naive simulation to get the estimation of the cycle length. Second, run the cross-entropy method to update the parameter and find the lower and upper bounds of the 0.99999-tile response time, as shown in the table 3.1. Then the bound of the 0.99999-tile response time is $(\gamma_l.\gamma_u) = (8.362, 12.034)$. Third, narrow the bound of 0.99999-tile response time. One-step rough search will result in the bound $(11.255, 11.478)$, and the precise search will give the estimation of the 0.99999-tile response time 11.478 and the confidence interval $(11.440, 11.478)$. Note that the above three steps spend 244.37s.

iteration	λ_1	λ_2	μ_1	μ_2	delay γ	$P(R > \gamma)$	RE	running time
0	0.6	0.2	2	1	0	0.999838	0.0187	4.16s
1	0.733	0.215	1.479	0.569	1.756	0.183043	0.0385	4.27s
2	0.843	0.297	1.036	0.425	3.260	0.038570	0.0470	4.26s
3	0.878	0.391	1.029	0.347	5.637	0.003305	0.0472	4.66s
4	0.924	0.483	0.992	0.238	8.362	0.000203	0.0651	5.09s
5	0.908	0.554	1.001	0.183	12.034	0.000006	0.0832	6.76s
6	0.969	0.589	1.049	0.129	12.034	0.000006	0.0837	6.49s
7	0.884	0.620	0.966	0.125	12.034	0.000006	0.1250	6.60s
8	0.944	0.684	1.038	0.124	12.034	0.000005	0.0700	6.97s
9	0.983	0.688	1.030	0.124	12.034	0.000005	0.0647	6.70s
10	0.958	0.696	0.976	0.105	12.034	0.000006	0.0789	6.37s

Table 3: The evolution for parameters and thresholds in the cross-entropy method.

Finally, evaluate the probability $P(R > 11.478)$ to compare with the naive simulation. The experiment spends 72.76s and shows the estimation is 1.01045e-5 with RE 0.0257, but the naive simulation with the 1,000,000 cycles spends 98.97s and has the estimation 4.1369e-06 with RE 0.3953. It can be seen that although the IS method takes near 3/4 the naive simulation, and the RE of IS is only 1/15 of that of the naive simulation. So it's increased by about 330 times.

Evaluate 0.99999-tile response time of the lower priority. First, run the naive simulation to get the estimation of the cycle length. Second, run the cross-entropy method to update the parameter and find the lower and upper bounds of the 0.99999-tile response time, as shown in the table 3.1. Then the bound of the 0.99999-tile response time is $(\gamma_l.\gamma_u) = (21.590, 37.506)$. Third, narrow the bound of 0.99999-tile response time. One-step rough search will result

in the bound $(29.146, 29.789)$, and the precise search will give the estimation of the 0.99999-tile response time 29.675 and the confidence interval $(29.510, 29.789)$. Note that the above three steps spend 148.66s.

iteration	λ_1	λ_2	μ_1	μ_2	delay γ	$P(R > \gamma)$	RE	running time
0	0.6	0.2	2	1	0	0.9870113	0.0275	4.19s
1	0.534	0.403	1.430	0.805	1.766	0.3786823	0.0317	4.27s
2	0.635	0.619	1.180	0.536	4.527	0.1057886	0.0431	4.36s
3	0.771	0.646	1.223	0.483	10.942	0.0085885	0.0532	4.54s
4	0.842	0.624	1.360	0.478	21.590	0.0001899	0.0691	5.01s
5	0.849	0.608	1.363	0.452	37.506	0.0000006	0.0939	5.67s
6	0.868	0.587	1.359	0.431	37.506	0.0000005	0.0636	5.65s
7	0.873	0.564	1.347	0.428	37.506	0.0000007	0.1231	5.69s
8	0.869	0.578	1.355	0.430	37.506	0.0000005	0.0747	5.75s
9	0.862	0.572	1.336	0.439	37.506	0.0000005	0.0627	5.65s
10	0.870	0.553	1.354	0.437	37.506	0.0000007	0.1317	5.68s

Table 4: The evolution for parameters and thresholds in the cross-entropy method.

Finally, evaluate the probability $P(R > 29.675)$ to compare with the naive simulation. The experiment spends 33.45s and shows the estimation is 9.8405e-06 with RE 0.0297. The threshold is too large and the naive simulation only exceeds the threshold once in 1,000,000 rounds, so the RE is very large, and the results obtained are very unreliable. Therefore, we skip this part of the comparison, but it can be imagined that the improvement must be even higher.

References

[HB21] Mor Harchol-Balter. "Open problems in queueing theory inspired by datacenter computing". In: *Queueing Systems* 97.1 (2021), pp. 3–37.

[RK16] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*. Vol. 10. John Wiley & Sons, 2016.