Adv. Appl. Probab. **51,** 1179–1208 (2019)
*doi:*10.1017/apr.2019.45

EXACT SAMPLING FOR SOME MULTI-DIMENSIONAL QUEUEING MODELS WITH RENEWAL INPUT

JOSE BLANCHET,* *Stanford University*
YANAN PEI,** *** AND
KARL SIGMAN,** *Columbia University*

Abstract

Using a result of Blanchet and Wallwater (2015) for exactly simulating the maximum of a negative drift random walk queue endowed with independent and identically distributed (i.i.d.) increments, we extend it to a multi-dimensional setting and then we give a new algorithm for simulating exactly the stationary distribution of a first-in–first-out (FIFO) multi-server queue in which the arrival process is a general renewal process and the service times are i.i.d.: the FIFO GI/GI/c queue with $2 \leq c < \infty$. Our method utilizes dominated coupling from the past (DCFP) as well as the random assignment (RA) discipline, and complements the earlier work in which Poisson arrivals were assumed, such as the recent work of Connor and Kendall (2015). We also consider the models in continuous time, and show that with mild further assumptions, the exact simulation of those stationary distributions can also be achieved. We also give, using our FIFO algorithm, a new exact simulation algorithm for the stationary distribution of the infinite server case, the GI/GI/∞ model. Finally, we even show how to handle *fork–join* queues, in which each arriving customer brings c jobs, one for each server.

Keywords: Exact sampling; multi-server queue; random walk; random assignment; dominated coupling from the past

2010 Mathematics Subject Classification: Primary 65C05
Secondary 60K25; 60J05; 60K05; 68U20

1. Introduction

In recent years, the method of *exact simulation* has evolved as a powerful way of sampling from stationary distributions of queueing models for which such distributions cannot be derived explicitly. The main method itself is referred to as *coupling from the past* (CFP), as introduced in Propp and Wilson [17] for finite-state discrete-time Markov chains. Since then, the method has been generalized to cover general state-space Markov chains by using dominating processes; this is known as *dominated coupling from the past* (DCFP), as in Kendall [15]. The main purpose of such methods is to produce a copy by simulation that has exactly (not approximately) the stationary distribution desired. These methods involve simulating processes backwards in time. In the present paper we consider using such methods for the *first-in–first-out*

Received 2 March 2018; revision received 23 July 2019.
* Postal address: Department of Management Science and Engineering, Stanford University, Stanford, CA 94305, USA.
** Postal address: Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027, USA.
*** Email address: yp2342@columbia.edu

(FIFO) multi-server queue, denoted as the FIFO GI/GI/c queue, $2 \leq c < \infty$, where c denotes the number of servers working in parallel, and arriving customers wait in one common queue (line).

The first algorithms yielding exact simulation in stationarity of the FIFO GI/GI/c queue are found in [20] and [21], in which Poisson arrivals are assumed, i.e. the M/G/c case. In [20], a DCFP method is used, but the strong condition of *super-stability* is assumed, $\rho < 1$, instead of $\rho < c$ ($\rho = \mathbb{E}[S]/\mathbb{E}[T]$, where T and S denote an interarrival time and service time respectively; stability only requires that $\rho < c$). As a dominating process, the M/G/1 queue is used under *processor sharing* (PS) together (key) with its time-reversibility properties. In PS, there is no queue: all customers are served simultaneously but at a rate $1/n$ when there are $n \geq 1$ customers in service. Then in [21], the general $\rho < c$ case is covered by using a forward-in-time regenerative method (a general method developed in [2]) and using the M/G/c model under a *random assignment* (RA) discipline as an upper bound – a model in which each arrival joins the ith queue with probability $1/c$ independently. (The general forward-in-time regenerative method in [2] unfortunately always yields infinite expected termination time.) Then in [8], Connor and Kendall generalized the DCFP/PS method in [20] by using the RA model. They accomplished this by first exactly simulating the RA model in stationarity backwards in time under PS at each node, then reconstructing it to obtain the RA model with FIFO at each node and doing so in such a way that a sample path upper bound for the FIFO M/G/c is achieved.

As for renewal arrivals (the general FIFO GI/GI/c queue considered here) the methods used above break down for various reasons, primarily because while under Poisson arrivals the c stations under RA become independent, they are not independent for general renewal arrivals. Also, the time-reversibility property of PS no longer holds, and nor does *Poisson arrivals see time averages* (PASTA). Finally, under general renewal arrivals, the system may never empty once it begins operating. New methods are needed. Blanchet, Dong, and Pei [7] solved the problem by utilizing a vacation model as an upper bound. In the present paper, however, we utilize DCFP by directly simulating the RA model in reverse time (under FIFO at each node). Our method involves extending, to a multi-dimensional setting, a recent result of Blanchet and Wallwater [6] for exactly simulating the maximum of a negative drift random walk endowed with independent and identically distributed (i.i.d.) increments. We also remark on how our approach can lead to new results for other models too, such as multi-server queues under the *last-in–first-out* (LIFO) discipline, or the *randomly choose next* discipline, and even fork–join models (also called split and match models).

2. The FIFO GI/GI/c model

Here we set up the classic first-in–first-out (of queue) (FIFO) multi-server queueing model and its associated Markov chain known as the *Kiefer–Wolfowitz workload vector* (for further details, see e.g. [1, Chapter 12, page 341], and the original paper [16]). In what follows, as input to a c-server in a parallel multi-server queue with $c \geq 2$, we have i.i.d. service times $\{S_n : n \geq 0\}$ distributed as $G(x) = \mathbb{P}(S \leq x)$, $x \geq 0$, with finite and non-zero mean $0 < \mathbb{E}[S] = 1/\mu < \infty$. Independently, the arrival times $\{t_n : n \geq 0\}$ ($t_0 = 0$) of customers to the model form a renewal process with i.i.d. interarrival times $T_n = t_{n+1} - t_n$, $n \geq 0$ distributed as $A(x) = \mathbb{P}(T \leq x)$, $x \geq 0$, and finite non-zero arrival rate $0 < \lambda = \mathbb{E}[T]^{-1} < \infty$. The FIFO GI/GI/$c$ model has only one queue, and customers upon arrival join the end of the queue and then attend service at the station that becomes free first (just like a United States post office). Because of the FIFO assumption, the nth service time initiated for use by a server, S_n, is used on the nth arrival (they arrive at time t_n), so one could equivalently imagine/assume

that S_n is brought to the system by the nth arrival. In this equivalent form, we say that at time t_n, the workload in the system has jumped upward by the amount S_n. Each server is identical in that they process service times at rate 1. We let $\mathbf{W}_n = (W_n(1), \ldots, W_n(c))^T$ denote the Kiefer–Wolfowitz workload vector, defined recursively by

$$\mathbf{W}_{n+1} = \mathcal{R}(\mathbf{W}_n + S_n\mathbf{e} - T_n\mathbf{f})^+, \quad n \geq 0, \tag{1}$$

where $\mathbf{e} = (1, 0, \ldots 0)^T$, $\mathbf{f} = (1, 1, \ldots, 1)^T$, $\mathcal{R}$ places a vector in ascending order, and $^+$ takes the positive part of each coordinate. The vector $\mathbf{W}_n$ shows, in ascending order, how much work at time t_n each server will process from among all work in the system at that time not including S_n. Letting C_n denote the nth arriving customer, $D_n = W_n(1)$ is the customer delay in the queue of C_n, because the server with the least amount of work will be the first to empty in front of C_n. Recursion (1) defines a c-dimensional Markov chain due to the given i.i.d. assumptions. The great importance of the recursion is that it yields $\{D_n : n \geq 0\}$, which is thus a function of a Markov chain.

With stability condition $\rho = \lambda/\mu < c$, it is well known that $\mathbf{W}_n$ converges in distribution as $n \to \infty$ to a proper stationary distribution (hence so does D_n). Let π denote this stationary distribution. Our main objective in the present paper is to provide a simulation algorithm for sampling exactly from π.

3. The RA GI/GI/*c* model

Given a c-server queueing system, the random assignment model (RA) is the case when each of the c servers forms its own FIFO single-server queue, and each arrival to the system, independent of the past, randomly chooses queue i to join with equal probability $1/c$, $1 \leq i \leq c$. In the GI/GI/c case, we refer to this as the RA GI/GI/c model. The following is a special case of Lemma 1.3 of [1, page 342]. (Such results and others even more general are based on [22], [11], and [12].)

Lemma 1. *Let $Q_F(t)$ denote the total number of customers in the system at time $t \geq 0$ for the FIFO GI/GI/c model, and let $Q_{RA}(t)$ denote the total number of customers in the system at time $t \geq 0$ for the corresponding RA GI/GI/c model in which both models are initially empty and fed with exactly the same input of renewal arrivals $\{t_n : n \geq 0\}$ and i.i.d. service times $\{S_n : n \geq 0\}$. Assume further that for both models the service times are used by the servers in the order in which service initiations occur (S_n is the service time used for the nth such initiation). Then*

$$\mathbb{P}(Q_F(t) \leq Q_{RA}(t) \text{ for all } t \geq 0) = 1. \tag{2}$$

The importance of Lemma 1 is that it allows us to jointly simulate versions of the two stochastic processes $\{Q_F(t) : t \geq 0\}$ and $\{Q_{RA}(t) : t \geq 0\}$ while achieving a coupling such that (2) holds. In particular, whenever an arrival finds the RA model empty, the FIFO model is found empty as well. (But we need to impose further conditions if we wish to ensure that indeed the RA GI/GI/c queue will empty with certainty.) Letting time t be sampled at arrival times of customers, $\{t_n : n \geq 0\}$, we thus also have

$$\mathbb{P}(Q_F(t_n-) \leq Q_{RA}(t_n-) \text{ for all } n \geq 0) = 1. \tag{3}$$

In other words, the total number in the system as found by the nth arrival is sample-path ordered as well. Note that for the FIFO model, the nth arriving customer C_n initiates the nth service since FIFO means 'first-in-queue–first-out-of-queue', where by 'queue' we mean the

line before entering service. This means that for the FIFO model we can attach S_n to C_n upon arrival if we so wish when applying Lemma 1. For the RA model, however, customers are not served in the order in which they arrive. For example, consider $c = 2$ servers (system initially empty) and suppose C_1 is assigned to node 1 with service time S_1, and C_2 is also assigned to node 1 (before C_1 departs) with service time S_2. Meanwhile, before C_1 departs, suppose C_3 arrives and is assigned to the empty node 2 with service time S_3. Then S_3 is used for the second service initiation. *For RA, the service times in order of initiation are a random permutation of the originally assigned* $\{S_n\}$.

To use Lemma 1, it is crucial to simply let the server hand out service times one at a time when they are needed for a service initiation. Thus, customers waiting in a queue before starting service do not have a service time assigned until they enter service. In simulation terminology, this amounts to generating the service times in order of when they are needed.

Define the total workload at any time t as the sum of all whole and remaining service times in the system at time t. One disadvantage of generating service times only when they are needed is that it does not allow the workload to be defined – only the amount of work in service. To get around this if need be, one can simply generate service times upon arrival of customers, and give them to the server to be used in order of service initiation. The point is that when C_n arrives, the total work in the system jumps up by the amount S_n. But S_n is not assigned to C_n: it is assigned (perhaps later) to whichever customer initiates the nth service. This allows Lemma 1 to hold true for the total amount of work in the system. If we let $\{V_F(t)\colon t \geq 0\}$ and $\{V_{RA}(t)\colon t \geq 0\}$ denote the total workload in the two models with the service times used in the manner just explained, then in addition to Lemma 1 we have

$$\mathbb{P}(V_F(t) \leq V_{RA}(t) \text{ for all } t \geq 0) = 1, \tag{4}$$

$$\mathbb{P}(V_F(t_n-) \leq V_{RA}(t_n-) \text{ for all } n \geq 0) = 1. \tag{5}$$

It is important, however, to note that what one cannot do is define workload at the individual nodes i by doing this, because that forces us to assign S_n to C_n so that workload at the node that C_n attends (i say) jumps by S_n and C_n enters service using S_n; that destroys the proper coupling needed to obtain Lemma 1. We can only handle the total (sum over all c nodes) workload. In the present paper, our use of Lemma 1 is via a kind of reversal.

Lemma 2. *Let* $\{S'_n\}$ *be an i.i.d. sequence of service times distributed as G, and assign* S'_n *to* C_n *in the RA model. Define* S_n *as the service time used in the nth service initiation. Then* $\{S_n\}$ *is also i.i.d. distributed as G.*

Proof. The key is to note that we are reordering based only on the order in which service times begin to be used, not when they are completed (which would thus introduce a bias). The service time chosen for the next initiation either enters service immediately (e.g. it is one that is routed to an empty queue by an arriving customer) or is chosen from among those waiting in lines, and all those waiting are i.i.d. distributed as G. Let $\hat{t}_n$ denote the time at which the nth service initiation begins. The value S_n of the nth service time chosen (at time $\hat{t}_n$) by a server is independent of the past service time values used before time $\hat{t}_n$, and is distributed as G (the choice of service time chosen as the next to be used is not based on the value of the service time, only its position in the lines). Letting $k(n) =$ the index of the $\{S'_n\}$ that is chosen, i.e. $S_n = S'_{k(n)}$, it is this index (a random variable) that depends on the past, but the value S_n is independent of $k(n)$ since it is a new one. Thus the $\{S_n\}$ are i.i.d. distributed as G. □

The point of the above Lemma 2 is that we can, if we so wish, simulate the RA model by assigning S'_n to C_n (to be used as their service time), but then assigning S_n, i.e. $S'_{k(n)}$, to C_n in the FIFO model. By doing so the requirements of Lemma 1 are satisfied and (2), (3), (4) and (5) hold. Interestingly, however, it is not possible to first simulate the RA model up to a fixed time t, and then stop and reconstruct the FIFO model up to this time t. At time t, there may still be RA customers waiting in lines and hence not enough of the S_n have been determined yet to construct the FIFO model. But all we have to do, if need be, is to continue the simulation of the RA model beyond t until enough S_n have been determined to construct fully the FIFO model up to time t.

4. Simulating exactly from the stationary distribution of the RA GI/GI/c model

By Lemma 1, the RA GI/GI/c queue, which shares the same arrival stream $\{t_n\colon n \geq 0\}$ ($t_0 = 0$) and the same service times in order of service initiations $\{S_n\colon n \geq 0\}$, will serve as a sample path upper bound (in terms of total number of customers in the system and total workload) of the target FIFO GI/GI/c queue. Independent of $\{T_n\colon n \geq 0\}$ and $\{S_n\colon n \geq 0\}$, we let $\{U_n\colon n \geq 0\}$ be an i.i.d. sequence of random variables from discrete uniform distribution on $\{1, 2, \ldots, c\}$; U_n represents the choice that customer C_n makes about which single-server queue to join under RA discipline. Let $\mathbf{V}_n = (V_n(1), \ldots, V_n(c))^T$ denote the workload vector as found by C_n in the RA GI/GI/c model, and for $i = 1, \ldots, c$, $V_n(i)$ is the waiting time of the C_n if he chooses to join the FIFO single-server queue of server i. So, $V_0(i) = 0$ and

$$V_{n+1}(i) = (V_n(i) + S_n I(U_n = i) - T_n)^+, \quad n \geq 0. \tag{6}$$

These c processes are dependent through the common arrival times $\{t_n\colon n \geq 0\}$ (equivalently common interarrival times $\{T_n\colon n \geq 0\}$) and the common $\{U_n\colon n \geq 0\}$ random variables. Because of all the i.i.d. assumptions, $\{\mathbf{V}_n\colon n \geq 0\}$ forms a Markov chain. Define $\tilde{\mathbf{S}}_n = (S_n I(U_n = 1), \ldots, S_n I(U_n = c))^T$ and $\mathbf{T}_n = T_n \mathbf{f}$. Then we can express (6) in vector form as

$$\mathbf{V}_{n+1} = (\mathbf{V}_n + \tilde{\mathbf{S}}_n - \mathbf{T}_n)^+, \quad n \geq 0. \tag{7}$$

Here $\mathbf{V}_n$ uses the same interarrival times $\{T_n\colon n \geq 0\}$ and service times $\{S_n\colon n \geq 0\}$ as we fed $\mathbf{W}_n$ in (1). However, the coordinates of $\mathbf{V}_n$ are not in ascending order, though all of them are non-negative.

Each node i as expressed in (6) can be viewed as a FIFO GI/GI/1 queue with common renewal arrival process $\{t_n\colon n \geq 0\}$, but with i.i.d. service times $\{\tilde{S}_n(i) = S_n I(U_n = i)\colon n \geq 0\}$. Across i, the service times $(\tilde{S}_n(1), \ldots, \tilde{S}_n(c))$ are not independent, but they are identically distributed: marginally, with probability $1/c$, $\tilde{S}_n(i)$ is distributed as G, and with probability $(c-1)/c$ it is distributed as the point mass at 0, i.e. $\mathbb{E}[\tilde{S}(i)] = \mathbb{E}[S]/c$. The point here is that we are not treating node i as a single-server queue endowed only with its own arrivals (a thinning of the $\{t_n\colon n \geq 0\}$ sequence) and its own service times i.i.d. distributed as G. Defining i.i.d. increments $\Delta_n(i) = \tilde{S}_n(i) - T_n$ for $n \geq 0$, each node i has an associated negative drift random walk $\{R_n(i)\colon n \geq 0\}$ with $R_0(i) = 0$ and

$$R_n(i) = \sum_{j=1}^{n} \Delta_j(i), \quad n \geq 1.$$

With $\rho = \lambda \mathbb{E}[S] < c$, we define $\rho_i = \lambda \mathbb{E}[\tilde{S}(i)] = \lambda \mathbb{E}[S]/c = \rho/c < 1$; equivalently $\mathbb{E}[\Delta(i)] < 0$ for all $i = 1, \ldots, c$. Let $V^0(i)$ denote a random variable with the limiting (stationary)

distribution of $V_n(i)$ as $n \to \infty$; it is well known (due to the i.i.d. assumptions) that $V^0(i)$ has the same distribution as

$$M(i) \triangleq \max_{m \geq 0} R_m(i)$$

for $i = 1, \ldots, c$.

More generally, even when the increment sequence is just stationary ergodic, not necessarily i.i.d. (hence not time-reversible as in the i.i.d. case), it is the backward-in-time maximum that is used in constructing a stationary version of $\{V_n(i)\}$. We will need this backwards approach in our simulation so we go over it here; it is usually referred to as *Loynes' lemma*. We extend the arrival point process $\{t_n : n \geq 0\}$ to be a two-sided point stationary renewal process $\{t_n : n \in \mathbb{Z}\}$

$$\cdots t_{-2} < t_{-1} < 0 = t_0 < t_1 < t_2 \cdots$$

Equivalently, $T_n = t_{n+1} - t_n$, $n \in \mathbb{Z}$, form i.i.d. interarrival times; $\{T_n : n \in \mathbb{Z}\}$ forms a two-sided i.i.d. sequence.

Similarly, the i.i.d. sequences $\{S_n : n \geq 0\}$ and $\{U_n : n \geq 0\}$ are extended to be two-sided i.i.d., $\{S_n : n \in \mathbb{Z}\}$ and $\{U_n : n \in \mathbb{Z}\}$. These extensions further allow two-sided extension of the i.i.d. increment sequences $\{\Delta_n(i) : n \in \mathbb{Z}\}$ for $i = 1, \ldots, c$, that is,

$$\Delta_n(i) = \tilde{S}_n - T_n = S_n I(U_n = i) - T_n, \quad n \in \mathbb{Z}.$$

Then we define c time-reversed (increments) random walks $\{R_n^{(r)}(i) : n \geq 0\}$ for $i = 1, \ldots, c$, by $R_0^{(r)}(i) = 0$ and

$$R_n^{(r)}(i) = \sum_{j=1}^{n} \Delta_{-j}(i), \quad n \geq 1.$$

A (from-the-infinite-past) stationary version of $\{V_n(i)\}$ denoted by $\{V_n^0(i) : n \leq 0\}$ is then constructed via

$$\begin{aligned}
V_0^0(i) &= \max_{m \geq 0} R_m^{(r)}(i), \\
V_{-1}^0(i) &= \max_{m \geq 1} R_m^{(r)}(i) - R_1^{(r)}(i), \\
V_{-2}^0(i) &= \max_{m \geq 2} R_m^{(r)}(i) - R_2^{(r)}(i), \\
&\vdots \\
V_{-n}^0(i) &= \max_{m \geq n} R_m^{(r)}(i) - R_n^{(r)}(i),
\end{aligned}$$

for all $i = 1, \ldots, c$.

By construction, the process $\mathbf{V}_n^0 = (V_n^0(1), \ldots, V_n^0(c))^T$, $n \leq 0$, is jointly stationary representing a (from-the-infinite-past) stationary version of $\{\mathbf{V}_n : n \leq 0\}$, and satisfies the forward-in-time recursion (7):

$$\mathbf{V}_{n+1}^0 = (\mathbf{V}_n^0 + \tilde{\mathbf{S}}_n - \mathbf{T}_n)^+, \quad n \leq -1. \tag{8}$$

Thus, by starting at $n = 0$ and walking backwards in time, we have (theoretically) a time-reversed copy of the RA model. Furthermore, $\{\mathbf{V}_n^0 : n \leq 0\}$ can be extended to include forward time $n \geq 1$ via using the recursion further:

$$\mathbf{V}_n^0 = (\mathbf{V}_{n-1}^0 + \tilde{\mathbf{S}}_{n-1} - \mathbf{T}_{n-1})^+, \quad n \geq 1, \tag{9}$$

where $\tilde{\mathbf{S}}_n = (S_n I(U_n = 1), \ldots, S_n I(U_n = c))^T$ for $n \in \mathbb{Z}$.

In fact, once we have a copy of just $\mathbf{V}_0^0$, we can start off the Markov chain with it as initial condition and use (9) to obtain a forward-in-time stationary version $\{\mathbf{V}_n^0 \colon n \geq 0\}$.

The above 'construction', however, is theoretical. We do not yet have any explicit way of obtaining a copy of $\mathbf{V}_0^0$, let alone an entire from-the-infinite-past sequence $\{\mathbf{V}_n^0 \colon n \leq 0\}$. In Blanchet and Wallwater [6], a simulation algorithm is given that yields (when applied to each of our random walks), for each $1 \leq i \leq c$, a copy of

$$\{(R_n^{(r)}(i),\, V_{-n}^0(i)) \colon 0 \leq n \leq N\}$$

for any desired $0 \leq N < \infty$ including stopping times N. We modify the algorithm so that it can do the simulation jointly across the c systems, that is, we extend it to a multi-dimensional form.

In particular, it yields an algorithm for obtaining a copy of $\mathbf{V}_0^0$, as well as a finite segment (of length N) of a backward-in-time copy of the RA model, $\{\mathbf{V}_{-n}^0 \colon 0 \leq n \leq N\}$, a stationary into-the-past construction up to discrete time $n = -N$.

Finite exponential moments are not required (because only *truncated* exponential moments are needed, $\mathbb{E}[\mathrm{e}^{\gamma\,\Delta(i)} I\{|\Delta(i)| \leq a\}]$, which in turn allow for the simulation of the exponential tilting of truncated $\Delta(i)$, via acceptance/rejection). To get a finite expected termination time (at each individual node), one needs the service distribution to have finite moment slightly beyond 2: for some (explicitly known) $\epsilon > 0$,

$$\mathbb{E}[S^{2+\epsilon}] < \infty.$$

As our first case, we consider a stopping time N such that $\mathbf{V}_{-N} = \mathbf{0}$. Before giving the definition of the stopping time N, we introduce the main idea of our simulation algorithm.

Let us define the maximum of a sequence of vectors. Suppose we have $\mathbf{Z}_1, \ldots, \mathbf{Z}_k$, where $\mathbf{Z}_i \in \mathbb{R}^d$ with $d \geq 1$ and $k \in \mathbb{N}_+ \cup \{\infty\}$. Define

$$\max(\mathbf{Z}_1, \ldots, \mathbf{Z}_k) = \left(\max_{1 \leq i \leq k} Z_i(1), \ldots, \max_{1 \leq i \leq k} Z_i(d) \right)^T.$$

Next define, for $n \in \mathbb{Z}$,

$$\mathbf{U}_n = (I(U_n = 1), \ldots, I(U_n = c))^T \quad \text{and} \quad \boldsymbol{\Delta}_n = \tilde{\mathbf{S}}_n - \mathbf{T}_n = S_n \mathbf{U}_n - T_n \mathbf{f},$$

where $\{U_n \colon n \in \mathbb{Z}\}$ are i.i.d. from discrete uniform distribution over $\{1, 2, \ldots, c\}$, and independently $\{T_n \colon n \in \mathbb{Z}\}$ are i.i.d. from distribution A (as introduced in Section 2). Our goal is to simulate the stopping time $N \in \mathbb{N}$ such that $\mathbf{V}_{-N}^0 = \mathbf{0}$, defined as

$$N = \inf \left\{ n \geq 0 \colon \mathbf{V}_{-n}^0 = \max_{k \geq n} \mathbf{R}_k^{(r)} - \mathbf{R}_n^{(r)} = \mathbf{0} \right\}, \tag{10}$$

that is, the first time walking in the past, that all coordinates of the workload vector are 0, jointly with $\{(\mathbf{R}_n^{(r)}, \mathbf{V}_{-n}^0) \colon 0 \leq n \leq N\}$. (By convention, the value of any empty sum of numbers is zero, i.e. $\sum_{j=1}^{0} a_j = 0$.)

To ensure that $\mathbb{E}[N] < \infty$, in addition to $\rho < c$ (stability), it is required that $\mathbb{P}(T > S) > 0$ (see the proof of Theorem 2 in [18]), for which the most common sufficient conditions are that T has unbounded support, $\mathbb{P}(T > t) > 0,\ t \geq 0$, or S has mass arbitrarily close to 0, $\mathbb{P}(S < t) > 0$, $t > 0$. But as we shall show in Section 6, given we know that $\mathbb{P}(T > S) > 0$, we can assume without loss of generality that interarrival times are bounded. It is that assumption which makes the extension of [6] to a multi-dimensional form easier to accomplish. Then, we show (in Sections 4.2 and 9) how to simulate from π even when $\mathbb{P}(T > S) = 0$. We do that in two different ways, one as a sandwiching argument and the other involving Harris-recurrent Markov chain regenerations.

4.1. Algorithm for simulating exactly from π for the FIFO GI/GI/c queue: the case $\mathbb{P}(T > S) > 0$

As mentioned earlier, we will assume that $\mathbb{P}(T > S) > 0$, so that the stable ($\rho < c$) RA and FIFO GI/GI/c Markov chains (7) and (1) will visit 0 infinitely often with certainty. (That the RA model empties infinitely often when $\mathbb{P}(T > S) > 0$ is proved, for example, in [18].) We imagine that at the infinite past $n = -\infty$, we start both (7) and (1) from empty. We construct the RA model forwards in time, while using Lemma 2 for the service times for the FIFO model, so that Lemma 1 applies and we have it in the form of (3), for all $t_n \leq 0$ up to and including at time $t_0 = 0$, at which time both models are in stationarity. We might have to continue the construction of the RA model so that $\mathbf{W}_0$ (distributed as π) can be constructed (e.g. enough service times have been initiated by the RA model for using Lemmas 1 and 2). Formally, one can theoretically justify the existence of such infinite-from-the-past versions (that obey Lemma 1), by use of Loynes' lemma. Each model (when started empty) satisfies the monotonicity required to use Loynes' lemma. In particular, noting that $Q_{RA}(t_n-) = 0$ if and only if $\mathbf{V}_n = \mathbf{0}$, we conclude that if at any time n it holds that $\mathbf{V}_n = \mathbf{0}$, then $\mathbf{W}_n = \mathbf{0}$. By the Markov property, given that $\mathbf{V}_n = \mathbf{0} = \mathbf{W}_n$, the future is independent of the past for each model, or said differently, *the past is independent of the future*. This remains valid if n is replaced by a stopping time (strong Markov property).

We outline the simulation algorithm steps as follows.

1. Simulate $\{\{(R_n^{(r)}(i), V_{-n}^0(i))\colon 0 \leq n \leq N\}, 1 \leq i \leq c\}$ with N as defined in (10). If $N = 0$, go to the next step. Otherwise, having stored all data, reconstruct $\mathbf{V}_n^0$ forwards in time from $n = -N$ (initially empty) until $n = 0$, using the recursion (8). During this forward-in-time reconstruction, redefine S_j as the jth service initiation used by the RA model (i.e. we are using Lemma 2 to gather service times in the proper order to feed in the FIFO model, which is why we do the reconstruction). If at time $n = 0$ there have not yet been N service initiations, then continue simulating the RA model out in forward time until finally there is an Nth service initiation, and then stop. This will require, at most, simulating out to t_n with $n = N^{(+)} = \min\{n \geq 0\colon \mathbf{V}_n^0 = \mathbf{0}\}$. Take the vector $(S_{-N}, S_{-N+1}, \ldots, S_{-1})$ and reset $(S_0, S_1, \ldots, S_{N-1}) = (S_{-N}, S_{-N+1}, \ldots, S_{-1})$. Also, store the interarrival times $(T_{-N}, T_{-N+1}, \ldots, T_{-1})$, and reset $(T_0, \ldots, T_{N-1}) = (T_{-N}, T_{-N+1}, \ldots, T_{-1})$.
2. If $N = 0$, then set $\mathbf{W}_0 = \mathbf{0}$ and stop. Otherwise use (1) with $\mathbf{W}_0 = \mathbf{0}$, recursively go forwards in time for N steps until obtaining $\mathbf{W}_N$, via the N reset service $(S_0, S_1, \ldots, S_{N-1})$ and interarrival times $(T_0, \ldots, T_{N-1})$. Reset $\mathbf{W}_0 = \mathbf{W}_N$.
3. Output $\mathbf{W}_0$.

Detailed simulation steps are discussed in Appendix A. Let τ denote the total number of interarrival times and service times to simulate in order to detect the stopping time N. The following proposition shows that our algorithm will terminate in finite expected time, i.e. $\mathbb{E}[\tau] < \infty$. The proof is given in Section B.

Proposition 1. *If $\rho = \lambda/\mu < c$, $\mathbb{P}(T > S) > 0$, and there exists some $\epsilon > 0$ such that $\mathbb{E}[S^{2+\epsilon}] < \infty$, then*

$$\mathbb{E}[N] < \infty \quad \textit{and} \quad \mathbb{E}[\tau] < \infty.$$

4.2. A more efficient algorithm: sandwiching

In this section we no longer even need to assume that $\mathbb{P}(T > S) > 0$. (Another method allowing for $\mathbb{P}(T > S) = 0$ involving Harris-recurrent regeneration is given later in Section 9.) Instead of waiting for the workload vector of the GI/GI/c queue under RA discipline to become $\mathbf{0}$, we choose an 'inspection time' $t_{-\kappa} < 0$ for some $\kappa \in \mathbb{Z}_+$ to stop the backward simulation of the RA GI/GI/c queue, then construct two bounding processes of the target FIFO GI/GI/c queue and evolve them forwards in time, using the same stream of arrivals and service time requirements (in order of service initiations), until coalescence or time zero. In particular, we let the upper bound process be a FIFO GI/GI/c queue starting at time $t_{-\kappa}$ with workload vector $\mathbf{V}^0_{-\kappa}$, and let the lower bound process be a FIFO GI/GI/c queue starting at the same time $t_{-\kappa}$ from empty, i.e. with workload vector $\mathbf{0}$.

Let $\mathbf{W}(t)$ denote the ordered (ascendingly) workload vector of the original FIFO GI/GI/c queueing process, starting from the infinite past, evaluated at time t. For $t \geq t_{-\kappa}$, we define $\mathbf{W}^u_{-\kappa}(t)$ and $\mathbf{W}^l_{-\kappa}(t)$ to be the ordered (ascendingly) workload vectors of the upper bound and lower bound processes, initiated at the inspection time $t_{-\kappa}$, evaluated at time t. By our construction and Theorem 3.3 in [8],

$$\mathbf{W}^u_{-\kappa}(t_{-\kappa}) = \mathcal{R}(\mathbf{V}^0_{-\kappa}) \geq \mathbf{W}(t_{-\kappa}) \geq \mathbf{W}^l_{-\kappa}(t_{-\kappa}) = \mathbf{0},$$

and for all $t > t_{-\kappa}$,

$$\mathbf{W}^u_{-\kappa}(t) \geq \mathbf{W}(t) \geq \mathbf{W}^l_{-\kappa}(t),$$

where all the above inequalities hold coordinate-wise.

Note that we can evolve the ordered workload vectors of the two bounding processes as follows: for $t_{n-1} \leq t < t_n$ when $-\kappa < n \leq -1$,

$$\begin{aligned} \mathbf{W}^u_{-\kappa}(t) &= \mathcal{R}(\mathbf{W}^u_{-\kappa}(t_{n-1}) + S_{n-1}\mathbf{e} - (t - t_{n-1})\mathbf{f})^+, \\ \mathbf{W}^l_{-\kappa}(t) &= \mathcal{R}(\mathbf{W}^l_{-\kappa}(t_{n-1}) + S_{n-1}\mathbf{e} - (t - t_{n-1})\mathbf{f})^+. \end{aligned} \tag{11}$$

Similarly, let $Q(t)$ denote the number of customers in the original FIFO GI/GI/c queueing process, starting from the infinite past, evaluated at time t. For $t \geq t_{-\kappa}$, we let $Q^u_{-\kappa}(t)$ and $Q^l_{-\kappa}(t)$ denote the number of customers in the upper and lower bound queueing processes respectively, both initiated at the inspection time $t_{-\kappa}$, evaluated at time t. If at some time $\tau \in [t_{-\kappa}, 0]$ we observe that $\mathbf{W}^u_{-\kappa}(\tau) = \mathbf{W}^l_{-\kappa}(\tau)$, then it must be true that $\mathbf{W}(\tau) = \mathbf{W}^u_{-\kappa}(\tau) = \mathbf{W}^l_{-\kappa}(\tau)$ and $Q(\tau) = Q^u_{-\kappa}(\tau) = Q^l_{-\kappa}(\tau)$ (because the ordered remaining workload vectors of two bounding processes can only meet when they both have idle servers). We call such time τ 'coalescence time' and from then on we have full information of the target FIFO GI/GI/c queue, hence we can continue to simulate it forwards in time until time 0.

However, if coalescence does not happen by time 0, we can adopt the so-called 'binary back-off' method by letting the arrival time $t_{-2\kappa}$ be our new inspection time and redo the above procedure to detect coalescence. Theorem 3.3 in [8] ensures that, for any $t_{-\kappa} \leq t \leq 0$,

$$\mathbf{W}^u_{-\kappa}(t) \geq \mathbf{W}^u_{-2\kappa}(t) \geq \mathbf{W}(t) \geq \mathbf{W}^l_{-2\kappa}(t) \geq \mathbf{W}^l_{-\kappa}(t).$$

We summarize the sandwiching algorithm as follows.

1. Simulate $\{(\mathbf{R}^{(r)}_n, \mathbf{V}^0_{-n}) \colon 0 \leq n \leq \kappa\}$ with all data stored.
2. Use the stored data to reconstruct $\mathbf{V}^0_n$ forwards in time from $n = -\kappa$ until $n = 0$, using (8), and redefine S_j as the jth service initiation used by the RA model.

3. Set $\mathbf{W}^u_{-\kappa}(t_{-\kappa}) = \mathcal{R}(\mathbf{V}^0_{-\kappa})$ and $\mathbf{W}^l_{-\kappa}(t_{-\kappa}) = \mathbf{0}$. Then use the same stream of interarrival times $(T_{-\kappa}, T_{-\kappa+1}, \ldots, T_{-1})$ and service times $(S_{-\kappa}, S_{-\kappa+1}, \ldots, S_{-1})$ to simulate $\mathbf{W}^u_{-\kappa}(t)$, $\mathbf{W}^l_{-\kappa}(t)$ forwards in time using (11).

4. If at some time $t \in [t_{-\kappa}, 0]$ we detect $\mathbf{W}^u_{-\kappa}(t) = \mathbf{W}^l_{-\kappa}(t)$, set $\tau = t$, $\mathbf{W}(\tau) = \mathbf{W}^u_{-\kappa}(\tau)$, $Q(\tau) = \sum_{i=1}^{c} I(\mathbf{W}(\tau; i) > 0)$, where $\mathbf{W}(t; i)$ is the ith entry of vector $\mathbf{W}(t)$. Then use the remaining interarrival times and service times to evolve the original FIFO GI/GI/c queue forwards in time until time $t_0 = 0$, output $(\mathbf{W}(0), Q(0))$ and stop.

5. If no coalescence is detected by time 0, set $\kappa = 2\kappa$, then continue to simulate the backward RA GI/GI/c process until $(-\kappa)$th arrival, i.e. $\{(\mathbf{R}^{(r)}_n, \mathbf{V}^0_{-n}) \colon 0 \le n \le \kappa\}$, with all data stored. Go to step 2.

Next we analyze properties of the coalescence time. Define

$$\kappa^*_- = \inf\Big\{n \ge 0 \colon \inf_{t_{-n} \le t \le 0} \|\mathbf{W}^u_{-n}(t) - \mathbf{W}^l_{-n}(t)\|_\infty = 0\Big\}.$$

If at time $t_{-\kappa^*_-}$ we start an upper bound FIFO GI/GI/c queue with workload vector $\mathbf{W}^u_{-\kappa^*_-}(t_{-\kappa^*_-})$ and a lower bound FIFO GI/GI/c queue with workload vector $\mathbf{0}$, they will coalesce by time $t_0 = 0$. Therefore if we simulate the RA system backwards in time to $t_{-\kappa^*_-}$, we will be able to detect a coalescence. We next show that $\mathbb{E}[-t_{-\kappa^*_-}] < \infty$.

By stationarity we have that κ^*_- is equal in distribution to

$$\kappa^*_+ = \inf\Big\{n \ge 0 \colon \inf_{0 \le t \le t_n} \|\mathbf{W}^u_0(t) - \mathbf{W}^l_0(t)\|_\infty = 0\Big\},$$

hence $-t_{-\kappa^*_-} \overset{d}{=} t_{\kappa^*_+}$.

Proposition 2. *If* $\rho = \mathbb{E}[S]/\mathbb{E}[T] < c$ *and there exists some* $\epsilon > 0$ *such that* $\mathbb{E}[S^{2+\epsilon}] < \infty$ *and* $\mathbb{E}[T^{2+\epsilon}] < \infty$, *then*

$$\mathbb{E}[t_{\kappa^*_+}] < \infty.$$

The proof follows the same argument as in the proof of Proposition 3 in [7], so we give a brief proof outline in Section B.

5. Continuous-time stationary constructions

For a stable FIFO GI/GI/1 queue, let D denote stationary customer delay (time spent in queue); that is, it has the limiting distribution of $D_{n+1} = (D_n + S_n - T_n)^+$ as $n \to \infty$.

Independently, let S_e denote a random variable distributed as the *equilibrium distribution* G_e of service time distribution G,

$$G_e(x) = \mu \int_0^x \mathbb{P}(S > y)\, \mathrm{d}y, \quad x \ge 0,$$

where $S \sim G$. Let $V(t)$ denote the total work in the system at time t, the sum of all whole or remaining service times in the system at time t. $D_n = V(t_n-)$, and one can construct $\{V(t)\}$ via

$$V(t) = (D_n + S_n - (t - t_n))^+, \quad t_n \le t < t_{n+1}.$$

(It is to be continuous from the right with left limits.) Let V denote stationary workload; that is, it has the limiting distribution

$$\mathbb{P}(V \le x) = \lim_{t \to \infty} \frac{1}{t} \int_0^t \mathbb{P}(V(s) \le x)\, \mathrm{d}s, \quad x \ge 0.$$

It is well known that the following holds (see e.g. [19, Sections 6.3 and 6.4]):

$$\mathbb{P}(V > x) = \rho \mathbb{P}(D + S_e > x), \quad x \geq 0.$$

Letting $F_D(x) = \mathbb{P}(D \leq x)$ denote the probability distribution of D, letting δ_0 be the point mass at 0, and letting $*$ be convolution of distributions, this means that the distribution of V can be written as a mixture:

$$(1 - \rho)\delta_0 + \rho F_D * G_e.$$

This leads to the following.

Proposition 3. *For a stable* $(0 < \rho < 1)$ *FIFO GI/GI/1 queue, if* ρ *is explicitly known, and one can exactly simulate from* D *and* G_e*, then one can exactly simulate from* V*.*

Proof. 1. Simulate a Bernoulli (ρ) random variable B.

2. If $B = 0$, then set $V = 0$. Otherwise, if $B = 1$, then simulate D and independently simulate a copy $S_e \sim G_e$. Set $V = D + S_e$. Stop. □

Another algorithm requiring the ability to simulate from A_e (equilibrium distribution of the interarrival time distribution A) instead of G_e follows from another known relation:

$$V \stackrel{d}{=} (D + S - T_e)^+, \tag{12}$$

where D, S, and $T_e \sim A_e$ are independent (see e.g. equation (88) of [23, page 426]). Thus, by simulating D, S, and T_e, simply set $V = (D + S - T_e)^+$. Equation (12) extends analogously to the FIFO GI/GI/c model, where our objective is to exactly simulate from the time-stationary distribution of the continuous-time Kiefer–Wolfowitz workload vector, $\mathbf{W}(t) = (W(t;1), \ldots, W(t;c))^T$, $t \geq 0$, where it can be constructed via

$$\mathbf{W}(t) = \mathcal{R}(\mathbf{W}_n + S_n \mathbf{e} - (t - t_n)\mathbf{f})^+, \quad t_n \leq t < t_{n+1}.$$

It is to be continuous from the right with left limits, $\mathbf{W}_n = \mathbf{W}(t_n -)$. Total workload $V(t)$, for example, is obtained from this via

$$V(t) = \sum_{i=1}^{c} W(t; i).$$

Letting $\mathbf{W}^*$ have the time-stationary distribution of $\mathbf{W}(t)$ as $t \to \infty$, and letting $\mathbf{W}_0$ have the discrete-time stationary distribution π and letting S, T_e, and $\mathbf{W}_0$ be independent, then

$$\mathbf{W}^* \stackrel{d}{=} \mathcal{R}(\mathbf{W}_0 + S\mathbf{e} - T_e \mathbf{f})^+. \tag{13}$$

So once we have a copy of $\mathbf{W}_0$ (distributed as π) from our algorithm in Section 4.1 or Section 4.2, we can easily construct a copy of $\mathbf{W}^*$ as long as we can simulate from A_e. Of course, if arrivals are Poisson then the distribution of $\mathbf{W}^*$ is identical to that of $\mathbf{W}_0$ by PASTA, but otherwise we can use (13).

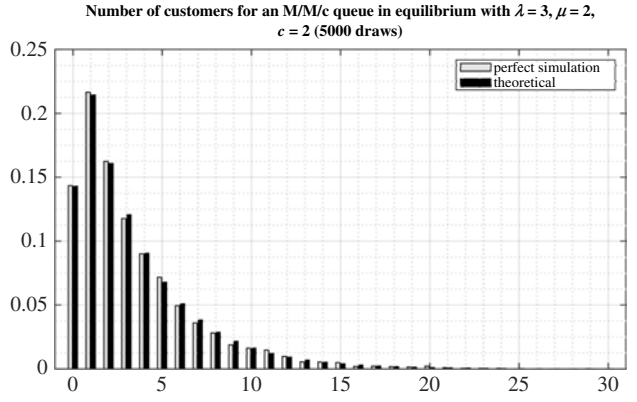

FIGURE 1: Number of customers for an M/M/c queue in stationarity when $\lambda = 3$, $\mu = 2$, and $c = 2$.

5.1. Numerical results

As a sanity check, we have implemented our perfect sampling algorithm in MATLAB® for the case of an Erlang (k_1, λ)/ Erlang $(k_2, \mu)/c$ queue. We provide our implementation codes for both algorithms in the online appendix of this paper, available at https://github.com/yanan-pei/exact-sampling-multiserver-queue.

First we consider M/M/c queues, which are special cases of Erlang (k_1, λ)/ Erlang $(k_2, \mu)/c$ with $k_1 = k_2 = 1$. For the quantity of interest, the number of customers in the FIFO M/M/c queue at stationary, we obtain its empirical distribution from a large number of independent runs of our algorithm and compare it to the theoretical distribution which has a well-established closed form:

$$\pi_0 = \left(\sum_{k=0}^{c-1} \frac{\rho^k}{k!} + \frac{\rho^c}{(c-1)!} \frac{1}{c-\rho} \right)^{-1},$$

$$\pi_k = \begin{cases} \pi_0 \cdot \rho^k / k! & \text{if } 0 < k < c, \\ pi_0 \cdot \rho^k c^{c-k} / c! & \text{if k} \geq c, \end{cases}$$

where $\rho = \lambda/\mu < c$.

As an example, Figure 1 shows the result of such a test when $\lambda = 3$, $\mu = 2$, and $c = 2$. Gray bars are the empirical results of 5 000 draws using our algorithm and black bars are the theoretical distribution number of customers in the system from stationarity. A Pearson's chi-squared test between the theoretical and empirical distributions gives a p-value equal to 0.8781, indicating close agreement (i.e. we cannot reject the null hypothesis that there is no difference between these two distributions). For another set of parameters $\lambda = 10$, $\mu = 2$, and $c = 10$, the results are shown in Figure 2 with a p-value of 0.6069 for the chi-squared fitness test.

For the general Erlang(k_1, λ)/ Erlang$(k_2, \mu)/c$ queue when $k_1 > 1$ and $k_2 > 1$ when $\rho/c = \lambda k_2/(c\mu k_1) = 0.9$, we compare the empirical distribution of number of customers in the system at stationarity, obtained from a large number of runs of our perfect sampling algorithm, to the numerical results (with precision at least 10^{-4}) provided in Table III of [14]. The results for an Erlang$(2, 9)$/ Erlang$(2, 5)/c$ queue are given in Figure 3. Gray bars are the empirical results of 5 000 draws using our algorithm and black bars are the numerical values given in [14], and they are very close to each other. The Pearson's chi-squared test gives a p-value of 0.9464, so we cannot reject the null hypothesis that these two distributions agree well.

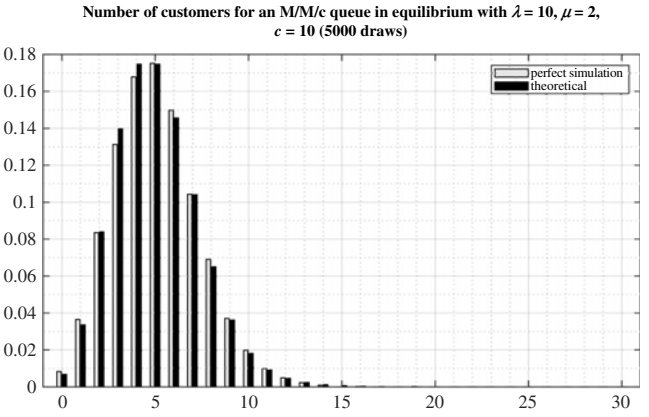

FIGURE 2: Number of customers for an M/M/c queue in stationarity when $\lambda = 10$, $\mu = 2$, and $c = 10$.

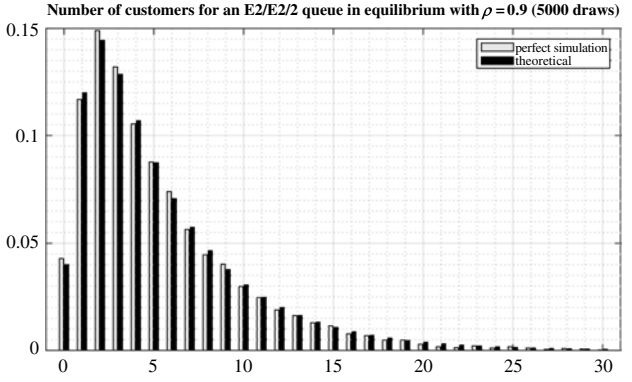

FIGURE 3: Number of customers for an Erlang (k_1, λ)/ Erlang $(k_2, \mu)/c$ queue in stationarity when $k_1 = 2$, $\lambda = 9$, $k_2 = 2$, $\mu = 5$, $c = 2$, and $\rho/c = 0.9$.

Next we run a numerical experiment comparing the computational efficiency of the first algorithm in Section 4.1 and the second sandwiching algorithm in Section 4.2. We measure the computational efficiency in two aspects. The first one is how far in the past we need to simulate the dominating process to detect coalescence (counting the total number of arrivals sampled backwards). The second aspect involves actual computation time in seconds. Figure 4 depicts such a comparison for an M/M/c queue with parameters $\lambda = 10$, $\mu = 2$, and $c = 10$, from 5 000 runs. Both results indicate that the second algorithm (sandwiching) is significantly more efficient than the first one.

Finally we study how the computational complexity of our sandwiching algorithm compares to the algorithm given in [7]. Note that these two algorithms look similar: they both use back-off strategies to run two bounding processes from some inspection time and check if they meet before time 0. The difference is that in [7] they use a so-called 'vacation system' to construct upper bound process, whereas we use the same queue but under RA discipline instead. In the following numerical experiment, we define the computational complexity as the total

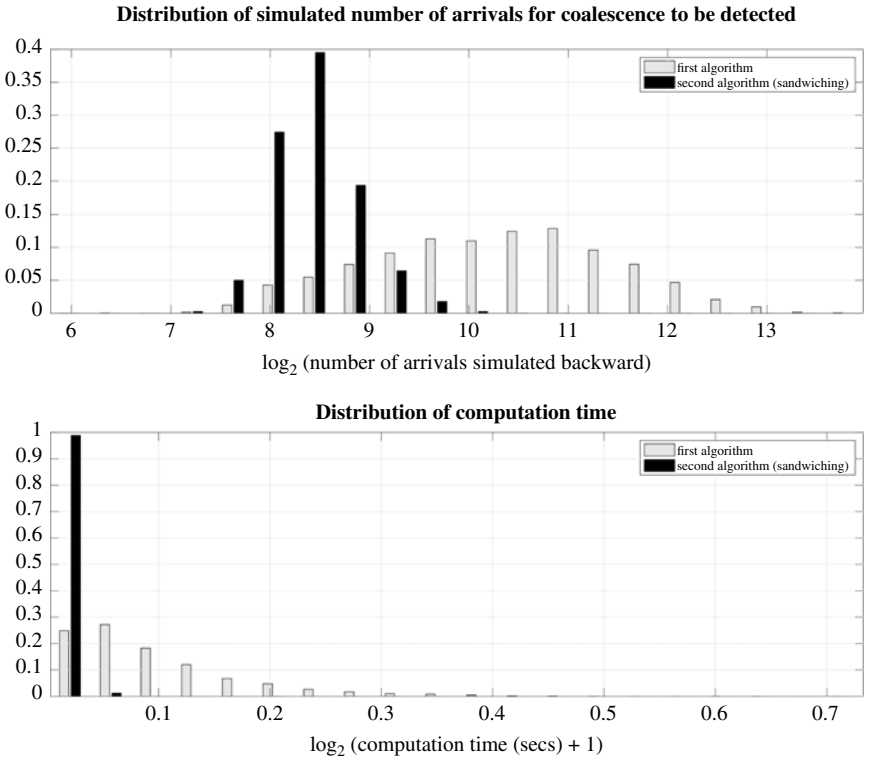

FIGURE 4: Computational efficiency comparison between two algorithms for an M/M/c queue.

TABLE 1. Simulation results for computational complexities with varying traffic intensities for an M/M/c queue with fixed $\mu = 5$ and $c = 2$.

		95% confidence interval of number of arrivals simulated backwards	
λ	ρ/c	Algorithm in Section 4.2	Algorithm in [7]
5	0.5	54.8194 ± 0.5758	146.5618 ± 2.3598
6	0.6	86.5394 ± 1.0536	308.4448 ± 4.9413
7	0.7	152.6552 ± 2.2695	730.1130 ± 11.2783
8	0.8	337.9544 ± 6.3021	2201.8254 ± 32.1556
9	0.9	1521.3502 ± 31.8267	12277.8686 ± 161.5824

number of arrivals each algorithm samples backwards to detect coalescence. Table 1 shows how they vary with traffic intensity, $\rho/c = \lambda/(c\mu)$, based on 5 000 independent runs of both algorithms using the same back-off strategy with the same initial $\kappa = 1$. The result suggests that our second algorithm (sandwiching) outperforms the one proposed in [7] as the magnitude of our computational complexity does not increase as fast as theirs when traffic intensity increases.

6. Why we can assume that interarrival times are bounded

Lemma 3. *Consider the recursion*

$$D_{n+1} = (D_n + S_n - T_n)^+, \quad n \geq 0,$$

where both $\{T_n\}$ and $\{S_n\}$ *are non-negative random variables, and* $D_0 = 0$.

Suppose that, for another sequence of non-negative random variables $\{\hat{T}_n\}$,

$$\mathbb{P}(\hat{T}_n \leq T_n,\ n \geq 0) = 1.$$

Then, for the recursion

$$\hat{D}_{n+1} = (\hat{D}_n + S_n - \hat{T}_n)^+, \quad n \geq 0,$$

with $\hat{D}_0 = 0$,

$$\mathbb{P}(D_n \leq \hat{D}_n,\ n \geq 0) = 1.$$

Proof. The proof is by induction on $n \geq 0$: because (with probability 1 in the following arguments) $\hat{T}_0 \leq T_0$, we have

$$D_1 = (S_0 - T_0)^+ \leq (S_0 - \hat{T}_0)^+ = \hat{D}_1.$$

Now suppose the result holds for some $n \geq 0$. Then $D_n \leq \hat{D}_n$ and by assumption $\hat{T}_n \leq T_n$; hence

$$D_{n+1} = (D_n + S_n - T_n)^+ \leq (\hat{D}_n + S_n - \hat{T}_n)^+ = \hat{D}_{n+1},$$

and the proof is complete. □

Proposition 4. *Consider the stable RA GI/GI/c model in which* $\mathbb{P}(T > S) > 0$. *In order to use this model to simulate from the corresponding stationary distribution of the FIFO GI/GI/c model as explained in the Section 4.1, without loss of generality we can assume that the interarrival times* $\{T_n\}$ *are bounded: there exists* $b > 0$ *such that*

$$\mathbb{P}(T_n \leq b,\ n \geq 0) = 1.$$

Proof. By stability, $c\mathbb{E}[T] > \mathbb{E}[S]$, and by assumption $\mathbb{P}(T > S) > 0$. If the $\{T_n\}$ are not bounded, then for $b > 0$, define $\hat{T}_n = \min\{T_n, b\}$, $n \geq 0$, i.e. truncated T_n. Choose b sufficiently large that $c\mathbb{E}[\hat{T}] > \mathbb{E}[S]$ and $\mathbb{P}(\hat{T} > S) > 0$ still hold. Now use the $\{\hat{T}_n\}$ in place of the $\{T_n\}$ to construct an RA model, denoted by $\widehat{RA}$. Denote this by

$$\hat{\mathbf{V}}_n = (\hat{V}_n(1), \ldots, \hat{V}_n(c)),$$

where it satisfies the recursion (7) in the form

$$\hat{\mathbf{V}}_{n+1} = (\hat{\mathbf{V}}_n + \tilde{\mathbf{S}}_n - \hat{\mathbf{V}}_n)^+, \quad n \geq 0,$$

where $\hat{\mathbf{T}}_n = \hat{T}_n \cdot \mathbf{f}$.

Starting from $\mathbf{V}_0 = \hat{\mathbf{V}}_0 = \mathbf{0}$, then from Lemma 3, it holds (coordinate-wise) that

$$\mathbf{V}_n \leq \hat{\mathbf{V}}_n, \quad n \geq 0,$$

and thus, if for some $n \geq 0$ it holds that $\hat{\mathbf{V}}_n = \mathbf{0}$, then $\mathbf{V}_n = \mathbf{0}$ and hence $\mathbf{W}_n = \mathbf{0}$ (as explained in our previous section). Since b was chosen ensuring that $c\mathbb{E}[\hat{T}] > \mathbb{E}[S]$ and $\mathbb{P}(\hat{T} > S) > 0$, $\{\hat{\mathbf{V}}_n\}$

is a stable RA GI/GI/c queue that will indeed empty infinitely often. Thus we can use it to do the backward-in-discrete-time stationary construction until it empties, at time (say) $-\hat{N}$, where $\hat{N} = \min\{n \geq 0 \colon \hat{\mathbf{V}}_{-n} = 0\}$. Then, we can reconstruct the original RA model (starting empty at time $-\hat{N}$) using the (original untruncated) $\hat{N}$ interarrival times $(T_{-\hat{N}}, T_{-\hat{N}+1}, \ldots, T_{-1})$ in lieu of $(\hat{T}_{-\hat{N}}, \hat{T}_{-\hat{N}+1}, \ldots, \hat{T}_{-1})$, so as to collect $\hat{N}$ reordered S_n needed in the construction of $\mathbf{W}_0$ for the FIFO model. □

Remark 1. One would expect the reconstruction of the original RA model in the above proof to be unnecessary, that instead we only need to reconstruct the $\widehat{RA}$ model until we have $\hat{N}$ service initiations from it, as opposed to $\hat{N}$ service initiations from the original RA model. Although this might be true, the subtle problem is that the order in which service times are initiated in the $\widehat{RA}$ model will typically be different than for the original RA model; they have different arrival processes (counter-examples are easy to construct). Thus it is not clear how one can utilize Lemma 1 and Lemma 2 and so on. One would need to generalize Lemma 1 to account for truncated arrival times used in the RA model, but not the FIFO model, in perhaps a form such as a variation of (3),

$$\mathbb{P}(Q_F(t_n-) \leq Q_{\widehat{RA}}(\hat{t}_n-) \text{ for all } n \geq 0) = 1,$$

where $\{\hat{t}_n\}$ is the truncated renewal process. We have not explored this further.

7. Infinite server systems and other service disciplines

In this section we sketch how one can utilize our FIFO GI/GI/c results to obtain exact sampling of some other models including the infinite server queue, and the multi-server queue under other disciplines.

In [4] an exact simulation algorithm is presented for simulating from the stationary distribution of the infinite server queue, the GI/GI/∞. Here we sketch how to utilize our new FIFO GI/GI/c results to accomplish this by using a FIFO GI/GI/c model as an upper bound. The GI/GI/∞ model has an infinite number of servers, there is no line, every arrival enters service immediately upon arrival; the nth customer arrives at time t_n and departs at time $t_n + S_n$.

For $0 < \rho = \lambda/\mu < \infty$, this model is always stable. Note that any $c > \rho$ can be chosen for this construction. A larger value of c will result in a smaller number of arrivals necessary to detect coalescence, up to a certain point, since there will be less congestion in the bounding systems and the customers will leave faster. On the other hand, the actual simulation time may increase just as a consequence of simulating a c-dimensional random walk. We suggest a rule consistent with square-root staffing, $c = \rho + \sqrt{\rho}$, since this is well known to trade quality and capacity costs, which in our setting precisely translate to trading faster coalescence with cost-per-replication costs (see [13]).

Letting $V_\infty(t)$ denote the total amount of work in the GI/GI/∞ model, and letting $V_c(t)$ denote the total amount of work in the (necessarily stable) FIFO GI/GI/c model being fed exactly the same input (of service times and interarrival times), and both starting initially empty, the following is easily established:

$$\mathbb{P}(V_\infty(t) \leq V_c(t) \text{ for all } t \geq 0) = 1,$$

hence

$$\mathbb{P}(V_\infty(t_n-) \leq V_c(t_n-) \text{ for all } n \geq 0) = 1.$$

(Note that both models use the service times in the same order of initiation, which makes the coupling easy from the start.)

Thus, if, for example, $\mathbb{P}(T > S) > 0$, then the FIFO model will empty and can be used to detect times when the GI/GI/∞ model will empty. Let $L_\infty(t_n-)$ denote the total number of busy servers in the GI/GI/∞ model as found by C_n.

Simulating the FIFO model backwards in time in stationarity (using our previous algorithm), until it first empties, can then be used to detect a time when the GI/GI/∞ model is empty, and then one can construct it back up to time $t = 0$ to obtain a stationary copy of $V_\infty(t_n-)$ and of $L_\infty(t_n-)$.

Now we consider alternatives disciplines to FIFO for the GI/GI/c model. It is immediate that when service times are generated only when needed by a server, the total number of customers in the system process $\{Q(t)\}$ remains the same under FIFO as under last-in–first-out (LIFO), in which the next customer to enter service is the one at the bottom of the line, or random selection next (RS), in which the next customer to enter service from the line is selected at random by the server. Thus, they all share the same stationary distribution of $Q(t)$ as $t \to \infty$, as well as the stationary distribution of $Q(t_n-)$ as $n \to \infty$. Let Q_0 have this limiting (as $n \to \infty$) distribution. This fact can be used to exactly simulate, for example, stationary delay D under LIFO or RS (they are not the same as for FIFO). The method (sketch) is as follows. Simulate a copy of Q_0, jointly with the remaining service times of those in service, by assuming FIFO. This represents the distribution of the system as found in stationarity (at time 0) by arrival C_0. Consider RS, for example. If the line is empty, then define $D_{RS} = 0$; C_0 enters service immediately. Otherwise, place C_0 in the line, and continue simulating but now using RS instead of FIFO. As soon as C_0 enters service, stop and define D_{RS} as that length of time.

8. Fork–join models

The RA recursion (7),

$$\mathbf{V}_{n+1} = (\mathbf{V}_n + \mathbf{S}_n - \mathbf{T}_n)^+, \quad n \geq 0, \tag{14}$$

is actually a special case for the modeling of *fork–join* (FJ) queues (also called *split and match*) with c nodes. In an FJ model, each arrival is a 'job' with c components, the ith component requiring service at the ith FIFO queue. So upon arrival at time t_n, the job splits into its c components to be served. As soon as all c components have completed service, then and only then does the job depart. Such models are useful in manufacturing applications. The nth job (C_n) thus arrives with a service time vector attached of the form $\mathbf{S}_n = (S_n(1), \ldots, S_n(c))$. Let us assume that the vectors are i.i.d., but otherwise each vector's coordinates can have a general joint distribution; for then (14) still forms a Markov chain. We will denote this model as the GI/GI/c-FJ model. The sojourn time of the ith component is given by $V_n(i) + S_n(i)$, and thus the sojourn time of the nth job, C_n, is given by

$$H_n = \max_{1 \leq i \leq c} \{V_n(i) + S_n(i)\}.$$

Of great interest is obtaining the limiting distribution of H_n as $n \to \infty$; we denote a random variable with this distribution as H^0. FJ models are notoriously difficult to analyze analytically. Even the special case of Poisson arrivals and i.i.d. exponential service times is non-trivial because of the dependence of the c queues through the common arrival process. (A classic paper is that of Flatto [10].) In fact, when $c \geq 3$, only bounds and approximations are available. As for exact simulation, there is a paper by Hongsheng Dai [9] in which Poisson arrivals and independent exponential service times are assumed. Because of the *continuous-time Markov chain* (CTMC) model structure, Dai was able to construct (simulate) the time-reversed CTMC

to use in a coupling from the past algorithm. But with general renewal arrivals and or general distribution service times, such CTMC methods can no longer be used.

Our simulation method for the RA model outlined in Section 4, however, yields an exact copy of H^0 for the general GI/GI/c-FJ model, under the condition that there exists $\boldsymbol{\theta} > \mathbf{0}, \boldsymbol{\theta} \in \mathbb{R}^c$ such that

$$\mathbb{E}[\exp(\boldsymbol{\theta}^T(\mathbf{S}_1 - \mathbf{T}_1))] < \infty.$$

First we simulate $\mathbf{V}_0^0$ exactly using exponential change of measure method introduced in [3] (we use the same technique for multi-dimensional simulation in Section 4.1), then simulate a vector of service times $\mathbf{S} = (S(1), \ldots, S(c))$ independently and set

$$H^0 = \max_{1 \le i \le c} \{V_0^0(i) + S(i)\}.$$

Even when the service time components within $\mathbf{S}$ are independent, or the case when service time distributions are assumed to have a finite moment generating function (in a neighborhood of the origin), such results are new and non-trivial.

9. The case when $\mathbb{P}(T > S) = 0$: Harris-recurrent regeneration

For a stable FIFO GI/GI/c queue, the stability condition can be rewritten as $\mathbb{E}[T_1 + \cdots + T_c] > \mathbb{E}[S]$, which implies also that $\mathbb{P}(T_1 + \cdots + T_c > S) > 0$. Thus assuming that $\mathbb{P}(T > S) > 0$ is not necessary for stability. When $\mathbb{P}(T > S) = 0$, the system will never empty again after starting, and so using consecutive visits to $\mathbf{0}$ as regeneration points is not possible. But the system does regenerate in a more general way via the use of Harris-recurrent Markov chain theory; see [18] for details and history of this approach. The main idea is that while the system will not empty infinitely often, the number in system process $\{Q_F(t_n-)\colon n \ge 0\}$ will visit an integer $1 \le j \le c-1$ infinitely often.

For illustration here, we will consider the $c = 2$ case (for the general case $c \ge 2$ the specific regeneration points analogous to what we present here are carefully given in equation (4.6) of [18, page 396]). Let us assume that $1 < \rho < 2$. (Note that if $\rho < 1$, then equivalently $\mathbb{E}[T] > \mathbb{E}[S]$ and so $\mathbb{P}(T > S) > 0$; that is why we rule out $\rho < 1$ here.) We now assume that $\mathbb{P}(T > S) = 0$. This implies that for $\underline{s} \triangleq \inf\{s > 0\colon \mathbb{P}(S > s) > 0\}$ and $\bar{t} \triangleq \sup\{t > 0\colon \mathbb{P}(T > t) > 0\}$, we must have $0 < \bar{t} < \underline{s} < \infty$. It is shown in [18] that for $\epsilon > 0$ sufficiently small, the following event will happen infinitely often (in n) with probability 1:

$$\{Q_{RA}(t_n-) = 1,\ V_n(1) = 0,\ V_n(2) \le \epsilon,\ T_n > \epsilon,\ U_n = 1\}. \tag{15}$$

If n is such a time, then at time $n+1$ we have

$$\{Q_{RA}(t_{n+1}-) = 1,\ V_{n+1}(2) = 0,\ V_{n+1}(1) = (S_n - T_n)\}. \tag{16}$$

The point is that C_n finds one server (server 1) empty, and the other queue with only one customer in it, and that customer is in service with a remaining service time $\le \epsilon$. C_n then enters service at node 1 with service time S_n; but since $T_n > \epsilon$, C_{n+1} arrives finding the second queue empty, and the first server has remaining service time $S_n - T_n$ conditional on $T_n > \epsilon$. Under the coupling of Lemma 1, the same will be so for the FIFO model (see Remark 2 below). At such a time n,

$$\{Q_F(t_n-) = 1,\ W_n(1) = 0,\ W_n(2) \le \epsilon,\ T_n > \epsilon\}, \tag{17}$$

and at time $n+1$ we have

$$\{Q_F(t_{n+1}-) = 1,\ W_n(1) = 0,\ W_n(2) = (S_n - T_n)\}. \tag{18}$$

Equations (16) and (18) define positive recurrent regeneration points for the two models (at time $n+1$); the consecutive times at which regenerations occur form a (discrete-time) positive recurrent renewal process (see [18]).

To put this to use, we change the stopping time N given in (10) to

$$N+1 = \min\{n \geq 1 \colon Q^0_{RA}(t_{-(n+1)}-) = 1,\ V^0_{-(n+1)}(1) = 0,\\ V^0_{-(n+1)}(2) \leq \epsilon,\ T_{-(n+1)} > \epsilon,\ U_{-(n+1)} = 1\}.$$

Then we do our reconstructions for the algorithm in Section 4.1 by starting at time $-N$, with both models starting with the same starting value

$$\{Q_{RA}(t_{-N}-) = 1,\ V^0_{-N}(2) = 0,\ V^0_{-N}(1) = (S_{-(N+1)} - T_{-(N+1)}) \mid T_{-(N+1)} > \epsilon\}, \tag{19}$$

$$\{Q_F(t_{-N}-) = 1,\ W_{-N}(1) = 0,\ W_{-N}(2) = (S_{-(N+1)} - T_{-(N+1)}) \mid T_{-(N+1)} > \epsilon\}. \tag{20}$$

Remark 2. The service time used in (19) and (20) for coupling via Lemma 2, $S_{-(N+1)}$, is in fact identical for both systems for the following subtle reason. At time $-(N+1)$, both systems have only one customer in the system, and thus the total work is in fact equal to the remaining service time, so we use (5) to conclude that both remaining service times (even if different) are $\leq \epsilon$ (for example, that is why (17) follows from (15)). Meanwhile, $C_{-(N+1)}$ enters service immediately across both systems, so it is indeed the same service time $S_{-(N+1)}$ used for this initiation. Coalescence is detected in finite expected time because of the positive recurrence property underlying the definition of the regeneration points from (16) and (18).

Appendix A. Detailed algorithm steps in Section 4.1

To simulate the process $\{(\mathbf{R}^{(r)}_n, \mathbf{V}^0_{-n})\colon 0 \leq n \leq N\}$ with the time N defined in (10) as

$$N = \inf\left\{n \geq 0 \colon \mathbf{V}^0_{-n} = \max_{k \geq n} \mathbf{R}^{(r)}_k - \mathbf{R}^{(r)}_n = \mathbf{0}\right\},$$

we must sample the running time maxima (entry by entry) of the c-dimensional random walk

$$\mathbf{R}^{(r)}_n = \sum_{i=1}^{n} \mathbf{\Delta}_{-i} = \sum_{i=1}^{n} (\tilde{\mathbf{S}}_{-i} - \mathbf{T}_{-i}), \quad n \geq 0.$$

We will find a sequence of random times $\{N_n \colon n \geq 1\}$ such that

$$\max_{n \leq k \leq N_n} \mathbf{R}^{(r)}_k \geq \max_{k \geq N_n} \mathbf{R}^{(r)}_k.$$

Hence, we will be able to find the running time maxima by only sampling the random walk on a finite time interval, that is, N_n is such that

$$\max_{k \geq n} \mathbf{R}^{(r)}_k = \max_{n \leq k \leq N_n} \mathbf{R}^{(r)}_k.$$

To achieve this, we first decompose the random walk into two random walks and then construct a sequence of 'milestone' events for each of these two random walks to detect the N_n. We will elaborate the detailed implementations in the following context.

Because of the stability condition $\rho = \lambda/\mu < c$, we can find some value $a \in (1/\mu, c/\lambda)$. For any $n \geq 0$, define

$$\mathbf{X}_{-n} = \sum_{j=1}^{n} (S_{-j} - a)\mathbf{U}_{-j}, \tag{21}$$

$$\mathbf{Y}_{-n} = \sum_{j=1}^{n} (a\mathbf{U}_{-j} - \mathbf{T}_{-j}), \tag{22}$$

hence $\mathbf{R}_n^{(r)} = \sum_{j=1}^{n} \boldsymbol{\Delta}_{-j} = \mathbf{X}_{-n} + \mathbf{Y}_{-n}$ and $\max_{k\geq n}\mathbf{R}_k^{(r)} = \max_{k\geq n}(\mathbf{X}_{-n} + \mathbf{Y}_{-n})$.

For all $n \geq 0$, let

$$N_n^X = \inf\left\{n' \geq n \colon \max_{k\geq n'} \mathbf{X}_{-k} \leq \mathbf{X}_{-n}\right\}, \tag{23}$$

$$N_n^Y = \inf\left\{n' \geq n \colon \max_{k\geq n'} \mathbf{Y}_{-k} \leq \mathbf{Y}_{-n}\right\}, \tag{24}$$

$$N_n = \max\{N_n^X, N_n^Y\}. \tag{25}$$

Then, by the definitions above,

$$\max_{k\geq N_n} \mathbf{R}_k^{(r)} \leq \max_{k\geq N_n} \mathbf{X}_{-k} + \max_{k\geq N_n} \mathbf{Y}_{-k} \leq \mathbf{X}_{-n} + \mathbf{Y}_{-n} = \mathbf{R}_n^{(r)}.$$

Therefore, to get the running time maximum $\max_{k\geq n}\mathbf{R}_k^{(r)}$ for each $n \geq 0$, we only need to sample the random walk from step n to N_n, because

$$\max_{k\geq n} \mathbf{R}_k^{(r)} = \max\left\{\max_{n\leq k\leq N_n} \mathbf{R}_k^{(r)}, \max_{n\geq N_n} \mathbf{R}_k^{(r)}\right\} = \max_{n\leq k\leq N_n} \mathbf{R}_k^{(r)}.$$

Next we describe how to sample N_n along with the multi-dimensional random walks

$$\{\mathbf{X}_{-n} \colon n \geq 0\} \quad \text{and} \quad \{\mathbf{Y}_{-n} \colon n \geq 0\}.$$

A.1. Simulation algorithm for the process $\{\mathbf{Y}_{-n} : n \geq 0\}$

We first consider simulating the c-dimensional random walk $\{\mathbf{Y}_{-n} \colon n \geq 0\}$ with $\mathbf{Y}_0 = \mathbf{0}$. For each $j \geq 1$, $\mathbb{E}[a\mathbf{U}_{-j} - \mathbf{T}_{-j}] < \mathbf{0}$, we can simulate the running time maximum $\max_{k\geq n}\mathbf{Y}_{-k}$ jointly with the path $\{\mathbf{Y}_{-k} \colon 0 \leq k \leq n\}$ via the method developed in [3], with the following assumptions.

Assumption (A1). There exists $\boldsymbol{\theta} > \mathbf{0}$, $\boldsymbol{\theta} \in \mathbb{R}^c$ such that

$$\mathbb{E}[\exp(\boldsymbol{\theta}^T(a\mathbf{U}_{-j} - \mathbf{T}_{-j}))] < \infty.$$

Assumption (A1b). Suppose that in every dimension $i = 1, \ldots, c$, there exists $\theta^* \in (0, \infty)$ such that

$$\phi_i(\theta^*) := \log \mathbb{E}[\exp(\theta^*(aI(U_{-j} = i) - T_{-j}))] = 0.$$

Because for each $j \geq 1$, $aI(U_{-j} = i) - T_{-j}$ are marginally identically distributed across i, so θ^* would work for all $i = 1, \ldots, c$.

Remark 3. In our setting, since $\mathbf{U}_{-j}$ is bounded, assumption (A1) always holds. Assumption (A1b) is known as Cramer's condition in the large deviations literature, and it is a strengthening of assumption (A1). We shall explain briefly at the end of this section that it is possible to relax this assumption to (A1) by modifying the algorithm without affecting the exactness or computational cost of the algorithm. For the moment we continue to describe the main algorithmic idea under assumption (A1b).

For any $\mathbf{s} \in \mathbb{R}^c$ and $\mathbf{b} \in \mathbb{R}^c_+$, define

$$\begin{aligned} T_{\mathbf{b}} &= \inf\{n \geq 0\colon\ Y_{-n}(i) > b(i) \text{ for some } i \in \{1, \ldots, c\}\},\\ T_{-\mathbf{b}} &= \inf\{n \geq 0\colon\ Y_{-n}(i) < -b(i) \text{ for all } i = 1, \ldots, c\},\\ P_{\mathbf{s}}(\cdot) &= \mathbb{P}(\cdot \mid \mathbf{Y}_0 = \mathbf{s}). \end{aligned}$$

We will use these definitions in Algorithm LTGM given in Section A.1.1.

We next construct a sequence of upward and downward 'milestone' events for this multi-dimensional random walk. The construction is completely analogous to the classical ladder height decomposition of one-dimensional random walks: we introduce a parameter, m, in order to facilitate a certain acceptance/rejection step to be explained in the next subsection. Let

$$m = \lceil \log(c)/\theta^* \rceil. \tag{26}$$

Define $D_0 = 0$ and $\Gamma_0 = \infty$. For $k \geq 1$, let

$$D_k = \inf\{n \geq D_{k-1} \vee \Gamma_{k-1} I(\Gamma_{k-1} < \infty)\colon\ Y_{-n}(i) < Y_{-D_{k-1}}(i) - m \text{ for all } i\}, \tag{27}$$

$$\Gamma_k = \inf\{n \geq D_k\colon\ Y_{-n}(i) > Y_{-D_k}(i) + m \text{ for some } i\}. \tag{28}$$

Note that, by convention, $\Gamma_k I(\Gamma_k < \infty) = 0$ if $\Gamma_k = \infty$ for any $k \geq 0$. We let $\mathbf{B} \in \mathbb{R}^c$, initially set as $(\infty, \ldots, \infty)^T \in \mathbb{R}^c$, to be the running time upper bound of process $\{\mathbf{Y}_{-n}\colon\ n \geq 0\}$. Let $\mathbf{m} = m\mathbf{f}$, where $\mathbf{f} = (1, \ldots, 1)^T$ as defined in Section 2. From the construction of 'milestone' events in (27) and (28), we know that if $\Gamma_k = \infty$ for some $k \geq 1$, the process will never cross over the level $\mathbf{Y}_{-D_k} + \mathbf{m}$ after D_k coordinate-wise, that is, for $i = 1, \ldots, c$,

$$Y_{-n}(i) \leq Y_{-D_k}(i) + m \quad \text{for all } n \geq D_k.$$

Hence, in this case we update the upper bound vector $\mathbf{B} = \mathbf{Y}_{-D_k} + \mathbf{m}$.

A.1.1. *Global maximum simulation.* Define

$$\Lambda = \inf\{D_k\colon\ \Gamma_k = \infty, k \geq 1\}.$$

By the construction of 'milestone' events, for all $n \geq \Lambda$,

$$\mathbf{Y}_{-n} \leq \mathbf{Y}_{-\Lambda} + \mathbf{m} < \mathbf{0} = \mathbf{Y}_0.$$

Hence, we can evaluate the global maximum level of the process $\{\mathbf{Y}_{-n}\colon\ n \geq 0\}$ to be

$$\mathbf{M}_0 := \max_{k \geq 0} \mathbf{Y}_{-k} = \max_{0 \leq k \leq \Lambda} \mathbf{Y}_{-k},$$

and we give the detailed sampling procedure in the following algorithm. The algorithm has elements, such as sampling from $P_{\mathbf{0}}(T_{\mathbf{m}} < \infty)$, which will be explained below.

Algorithm 1. (LTGM.) Simulate the global maximum of c-dimensional process $\{\mathbf{Y}_{-n}\colon\ n \geq 0\}$ jointly with the subpath and the subsequence of 'milestone' events.

Input: $a \in (1/\mu, c/\lambda)$ satisfies assumption (A1b), m as in (26).

1. (*Initialization*) Set $n = 0$, $\mathbf{Y}_0 = \mathbf{0}$, $\mathbf{D} = [0]$, $\mathbf{\Gamma} = [\infty]$, $\mathbf{L} = \mathbf{0}$, and $\mathbf{B} = \infty\mathbf{f}$.
2. Generate $U \sim \text{Unif}\{1, \ldots, c\}$ and let $\mathbf{U} = (I(U = 1), \ldots, I(U = c))^T$. Independently sample $T \sim A$ and let $\mathbf{T} = T\mathbf{f}$. Set $n = n + 1$, $\mathbf{Y}_{-n} = \mathbf{Y}_{-(n-1)} + a\mathbf{U} - \mathbf{T}$, $U_{-n} = U$, and $T_{-n} = T$.
3. If there is some $1 \le i \le c$ such that $Y_{-n}(i) \ge L(i) - m$, then go to step 2; otherwise set $\mathbf{D} = [\mathbf{D}, n]$ and $\mathbf{L} = \mathbf{Y}_{-n}$.
4. Independently sample $J \sim \text{Ber}\,(P_{\mathbf{0}}(T_{\mathbf{m}} < \infty))$.
5. If $J = 1$, simulate a new conditional path $\{(\mathbf{y}_{-k}, u_{-k}, t_{-k})\colon 1 \le k \le T_{\mathbf{m}}\}$ with $\mathbf{y}_0 = \mathbf{0}$, following the conditional distribution of $\{\mathbf{Y}_{-k}\colon 0 \le k \le T_{\mathbf{m}}\}$ given $T_{\mathbf{m}} < \infty$. Set $\mathbf{Y}_{-(n+k)} = \mathbf{Y}_{-n} + \mathbf{y}_{-k}$, $U_{-(n+k)} = u_{-k}$, $T_{-(n+k)} = t_{-k}$ for $1 \le k \le T_m$. Set $n = n + T_{\mathbf{m}}$, $\mathbf{\Gamma} = [\mathbf{\Gamma}, n]$. Go to step 2.
6. If $J = 0$, set $\Lambda = n$, $\mathbf{\Gamma} = [\mathbf{\Gamma}, \infty]$, and $\mathbf{B} = \mathbf{L} + \mathbf{m}$.
7. Output $\{(\mathbf{Y}_{-k}, U_{-k}, T_{-k})\colon 1 \le k \le \Lambda\}$, $\mathbf{D}$, $\mathbf{\Gamma}$, and global maximum $\mathbf{M}_0 = \max_{0 \le k \le \Lambda} \mathbf{Y}_{-k}$.

Now we explain how to execute steps 4 and 5 in the above algorithm. The procedure is similar to the multi-dimensional procedure given in [3], so we describe it briefly here. As $P_{\mathbf{0}}(\cdot)$ denotes the canonical probability, we let $P^*_{\mathbf{0}}(\cdot) = P_{\mathbf{0}}(\,\cdot\,|T_{\mathbf{m}} < \infty)$. Our goal is to simulate from the conditional law of $\{\mathbf{Y}_{-k}\colon 0 \le k \le T_{\mathbf{m}}\}$ given that $T_{\mathbf{m}} < \infty$ and $\mathbf{Y}_0 = \mathbf{0}$, i.e. to simulate from $P^*_{\mathbf{0}}$. We will use acceptance/rejection by letting $P'_{\mathbf{0}}(\cdot)$ denote the proposal distribution. A typical element ω' sampled under $P'_{\mathbf{0}}(\cdot)$ is of the form $\omega' = ((\mathbf{Y}_{-k}\colon k \ge 0), \text{index})$, where index $\in \{1, \ldots, c\}$, and it indicates the direction we pick to do exponential tilting; it is the coordinate in which we change its measure to increase the chance of its hitting the upward 'milestone' as defined in (28). Given the value of index, the process $(\mathbf{Y}_{-k}\colon k \ge 0)$ remains a random walk. We now describe $P'_{\mathbf{0}}$ by explaining how to sample ω'. First,

$$P'_{\mathbf{0}}(\,\text{index} = i) = w_i := \frac{1}{c}. \tag{29}$$

Then, conditioning on index $= i$, for every set $A \in \sigma(\{\mathbf{Y}_{-k}\colon 0 \le k \le n\})$,

$$P'_{\mathbf{0}}(A \mid \text{index} = i) = E_{\mathbf{0}}[\,\exp\,(\theta^* Y_{-n}(i))I_A]. \tag{30}$$

To obtain the induced distribution for U and T, we study the moment generating function induced by definition (30). Given $\boldsymbol{\eta} \in \mathbb{R}^c$ in a neighborhood of the origin,

$$\begin{aligned}&\frac{E_{\mathbf{0}}[\,\exp\,(\boldsymbol{\eta}^T(a\mathbf{U} - \mathbf{T}) + \theta^* e_i^T(a\mathbf{U} - \mathbf{T}))]}{E_{\mathbf{0}}[\,\exp\,(\theta^* e_i^T(a\mathbf{U} - \mathbf{T}))]} \\ &\quad = \frac{E_{\mathbf{0}}[\,\exp\,((\boldsymbol{\eta} + \theta^* e_i)^T a\mathbf{U})]}{E_{\mathbf{0}}[\,\exp\,(\theta^* e_i^T a\mathbf{U})]} \cdot \frac{E_{\mathbf{0}}[\,\exp\,(-(\boldsymbol{\eta} + \theta^* e_i)^T\mathbf{T})]}{E_{\mathbf{0}}[\,\exp\,(-\theta^* e_i^T\mathbf{T})]}.\end{aligned}$$

The above expression indicates that under $P'_{\mathbf{0}}(\cdot)$, T and U are independent. Moreover, we have

$$E_{\mathbf{0}}[\,\exp\,(\theta^* e_i^T a\mathbf{U})] = \frac{\exp\,(\theta^* a) + c - 1}{c}.$$

Therefore,

$$P'_{\mathbf{0}}(U = j \mid \text{index} = i) = \begin{cases} \dfrac{\exp(\theta^* a)}{\exp(\theta^* a) + c - 1} & \text{if } j = i, \\ \dfrac{1}{\exp(\theta^* a) + c - 1} & \text{if } j \neq i. \end{cases} \tag{31}$$

On the other hand, conditional on index $= i$, the distribution of a generic interarrival time T is obtained by exponential tilting such that

$$\mathrm{d}P_{\mathbf{0}}(T \mid \text{index} = i) = \mathrm{d}P_{\mathbf{0}}(T) \cdot \frac{\exp(-\theta^* T)}{E_{\mathbf{0}}[\exp(-\theta^* T)]} = \mathrm{d}P_{\mathbf{0}}(T) \cdot \frac{\exp(a\theta^*) + c - 1}{c \exp(\theta^* T)}, \tag{32}$$

where the second equation follows from assumption (A1b).

Following assumption (A1b) and because $\mathrm{Var}(aI(U_{-j} = i) - T_{-j}) > 0$, by convexity,

$$E'_{\mathbf{0}}[Y_{-n}(\text{index})] = \sum_{i=1}^{c} E_{\mathbf{0}}[Y_{-n}(i) \exp(\theta^* Y_{-n}(i))] P'_{\mathbf{0}}(\text{index} = i) = \frac{1}{c} \sum_{i=1}^{c} \frac{\mathrm{d}\phi_i(\theta^*)}{\mathrm{d}\theta} > 0,$$

so $Y_{-n}(\text{index}) \to \infty$ as $n \to \infty$ almost surely under $P'_{\mathbf{0}}(\cdot)$, hence $T_{\mathbf{m}} < \infty$ with probability one under $P'_{\mathbf{0}}(\cdot)$. Now, to verify that $P_{\mathbf{0}}(\cdot)$ is a valid proposal for acceptance/rejection method, we must verify that $\mathrm{d}P^*_{\mathbf{0}}/\mathrm{d}P'_{\mathbf{0}}$ is bounded by a constant, that is,

$$\begin{aligned} &\frac{\mathrm{d}P^*_0}{\mathrm{d}P'_0}(\mathbf{Y}_{-k} \colon 0 \le k \le T_{\mathbf{m}}) \\ &\quad = \frac{1}{P_0(T_{\mathbf{m}} < \infty)} \cdot \frac{\mathrm{d}P_0}{\mathrm{d}P'_0}(\mathbf{Y}_{-k} \colon 0 \le k \le T_{\mathbf{m}}) \\ &\quad = \frac{1}{P_0(T_{\mathbf{m}} < \infty)} \cdot \frac{1}{\sum_{i=1}^{c} w_i \exp(\theta^* Y_{-T_{\mathbf{m}}}(i))} \\ &\quad \le \frac{1}{P_0(T_{\mathbf{m}} < \infty)} \cdot \frac{c}{\exp(\theta^* m)} \\ &\quad < \frac{1}{P_0(T_{\mathbf{m}} < \infty)}, \end{aligned}$$

where the last inequality is guaranteed by (26). So, acceptance/rejection is valid.

Moreover, the overall probability of accepting the proposal is precisely $P_{\mathbf{0}}(T_{\mathbf{m}} < \infty)$. Thus, we execute not only step 5 but simultaneously also step 4. We use this acceptance/rejection method to replace steps 4 and 5 in Algorithm LTGM as follows.

4′. Sample $\{(\mathbf{y}_{-k}, u_{-k}, t_{-k}) \colon 0 \le k \le T_{\mathbf{m}})\}$ with $\mathbf{y}_0 = \mathbf{0}$ from $P'_0(\cdot)$ as indicated via (29), (31), and (32). Sample a Bernoulli J with success probability

$$\frac{c}{\sum_{i=1}^{c} \exp(\theta^* y_{-T_{\mathbf{m}}}(i))}.$$

5′. If $J = 1$, set $\mathbf{Y}_{-(n+k)} = \mathbf{Y}_{-n} + \mathbf{y}_{-k}$, $U_{-(n+k)} = u_{-k}$, $T_{-(n+k)} = t_{-k}$ for $1 \le k \le T_{\mathbf{m}}$. Set $n = n + T_{\mathbf{m}}$ and $\mathbf{\Gamma} = [\mathbf{\Gamma}, n]$. Go to step 2.

A.1.2. *Simulate* $\{\mathbf{Y}_{-n}\colon n \geq 0\}$ *jointly with 'milestone' events.* In this section we provide an algorithm to sequentially simulate the multi-dimensional random walk $\{\mathbf{Y}_{-n}\colon n \geq 0\}$ along with its downward and upward 'milestone' events as defined in (27) and (28). We first extend Lemma 3 in [5] to a multi-dimensional version as follows.

Lemma 4. *Let* $0 < \mathbf{a} < \mathbf{b} \leq \infty\mathbf{f}$ *(coordinate-wise) and consider any sequence of bounded positive measurable functions* $f_k\colon \mathbb{R}_{c\times(k+1)} \to [0,\infty)$,

$$E_0[f_{T_{-\mathbf{a}}}(\mathbf{Y}_0,\ldots,\mathbf{Y}_{-T_{-\mathbf{a}}}) \mid T_{\mathbf{b}} = \infty] = \frac{E_0[f_{T_{-\mathbf{a}}}(\mathbf{Y}_0,\ldots,\mathbf{Y}_{-T_{-\mathbf{a}}}) \cdot I(Y_{-j}(i) \leq b(i), 0 \leq j < T_{-\mathbf{a}}, 1 \leq i \leq c)] \cdot P_{\mathbf{Y}_{-T_{-\mathbf{a}}}}(T_{\mathbf{b}} = \infty)}{P_0(T_{\mathbf{b}} = \infty)}.$$

Therefore, if $P_0^{**}(\cdot) := P_0(\cdot \mid T_{\mathbf{b}} = \infty)$, *then*

$$\frac{\mathrm{d}P_0^{**}}{\mathrm{d}P_0} = \frac{I(Y_{-j}(i) \leq b(i) \text{ for all } j < T_{-\mathbf{a}}, 1 \leq i \leq c) \cdot P_{\mathbf{Y}_{-T_{-\mathbf{a}}}}(T_{\mathbf{b}} = \infty)}{P_0(T_{\mathbf{b}} = \infty)} \leq \frac{1}{P_0(T_{\mathbf{b}} = \infty)}.$$

Lemma 4 enables us to sample a downward patch by using the acceptance/rejection method with the nominal distribution P_0 as proposal. Suppose our current position is $\mathbf{Y}_{-D_j}$ (for some $j \geq 1$) and we know that the process will never go above the upper bound $\mathbf{B}$ (coordinate-wise). Next we simulate the path up to time D_{j+1}. If we can propose a downward patch $(\mathbf{y}_{-1},\ldots,\mathbf{y}_{-T_{-m}}) := (\mathbf{Y}_{-1},\ldots,\mathbf{Y}_{-T_{-\mathbf{m}}})$, under the unconditional probability given $\mathbf{y}_0 = \mathbf{0}$ and $\mathbf{y}_{-k} \leq \mathbf{m}$ for $1 \leq k \leq T_{-\mathbf{m}}$, then we accept it with probability $P_0(T_{\boldsymbol{\sigma}} = \infty)$, where $\boldsymbol{\sigma} = \mathbf{B} - \mathbf{Y}_{-D_j} - \mathbf{y}_{-T_{-\mathbf{m}}}$. A more efficient way to sample is to sequentially generate $(\mathbf{y}_{-1},\ldots,\mathbf{y}_{-\Lambda})$ with $\mathbf{y}_0 = \mathbf{0}$ as long as $\mathbf{m}_0 := \max_{0\leq k\leq\Lambda}\mathbf{y}_{-k} \leq \mathbf{m}$ coordinate-wise, then concatenate the sequence to the previously sampled subpath. We give the efficient implementation procedure in the next algorithm.

Algorithm 2. (LTRW.) Continue to sample the process $\{(\mathbf{Y}_{-k}, U_{-k}, T_{-k})\colon 0 \leq k \leq n\}$ jointly with the partially sampled 'milestone' event lists $\mathbf{D}$ and $\boldsymbol{\Gamma}$, until a stopping criterion is met.

Input: a, m, previously sampled partial process $\{(\mathbf{Y}_{-j}, U_{-j}, T_{-j})\colon 0 \leq j \leq l\}$, partial 'milestone' sequences $\mathbf{D}$ and $\boldsymbol{\Gamma}$, and stopping criterion $\mathcal{H}$. (Note that if there is no previous simulated random walk, we initialize $l = 0$, $\mathbf{D} = [0]$, and $\boldsymbol{\Gamma} = [\infty]$.)

1. Set $n = l$. If $n = 0$, call Algorithm LTGM to get Λ, $\{(\mathbf{Y}_{-k}, U_{-k}, T_{-k})\colon 0 \leq k \leq \Lambda\}$, $\mathbf{D}$ and $\boldsymbol{\Gamma}$. Set $n = \Lambda$.
2. While the stopping criterion $\mathcal{H}$ is not satisfied:
   (a) Call Algorithm LTGM to get $\tilde{\Lambda}$, $\{(\tilde{\mathbf{Y}}_{-j}, \tilde{U}_{-j}, \tilde{T}_{-j})\colon 0 \leq j \leq \tilde{\Lambda}\}$, $\tilde{\mathbf{D}}$, $\tilde{\boldsymbol{\Gamma}}$, and $\tilde{\mathbf{M}}_0$.
   (b) If $\tilde{\mathbf{M}}_0 \leq \mathbf{m}$, accept the proposed sequence and concatenate it to the previous subpath, that is, set $\mathbf{Y}_{-(n+j)} = \mathbf{Y}_{-n} + \tilde{\mathbf{Y}}_{-j}$, $U_{-(n+j)} = \tilde{U}_{-j}$, $T_{-(n+j)} = \tilde{T}_{-j}$ for $1 \leq j \leq \tilde{\Lambda}$. Update the sequences of 'milestone' events to be $\mathbf{D} = [\mathbf{D}, n + \tilde{\mathbf{D}}(2\colon \mathrm{end})]$, $\boldsymbol{\Gamma} = [\boldsymbol{\Gamma}, n + \tilde{\boldsymbol{\Gamma}}(2\colon \mathrm{end})]$ and set $n = n + \tilde{\Lambda}$.
3. Output $\{(\mathbf{Y}_{-k}, U_{-k}, T_{-k})\colon 0 \leq k \leq n\}$ with updated 'milestone' event sequences $\mathbf{D}$ and $\boldsymbol{\Gamma}$.

For $n \geq 0$, define

$$d_1(n) = \inf\{D_k \geq n \colon \mathbf{Y}_{-D_k} \leq \mathbf{Y}_{-n}\}, \tag{33}$$

$$d_2(n) = \inf\{D_k > d_1(n) \colon \Gamma_k = \infty\}, \tag{34}$$

and $d_2(n)$ is an upper bound of N_n^Y defined in (24) because

$$\max_{k \geq d_2(n)} \mathbf{Y}_{-k} \leq \mathbf{Y}_{-d_2(n)} + \mathbf{m} < \mathbf{Y}_{-d_1(n)} \leq \mathbf{Y}_{-n}.$$

Remark 4. Since assumption (A1b) is a strengthening of assumption (A1), we can accommodate our algorithms under the general assumption (A1). The implementation details are the same as those mentioned in the remark section of [3, page 15].

A.2. Simulation algorithm for the process $\{\mathbf{X}_{-n} : n \geq 0\}$

Recall from (21) that, for $n \geq 0$,

$$X_{-n}(i) = \sum_{j=1}^{n} (S_{-j} - a) I(U_{-j} = i) \quad \text{for } i = 1, \ldots, c.$$

Define

$$N_k(i) = \sum_{j=1}^{k} I(U_{-j} = i), \tag{35}$$

$$L_n(i) = \inf\{k \geq 0 \colon N_k(i) = n\} \ (L_0(i) = 0), \tag{36}$$

$$\hat{S}_{-n}^{(i)} = S_{-L_n(i)}, \tag{37}$$

for $k \geq 0$, $n \geq 0$, and $i = 1, \ldots, c$. $N_k(i)$ denotes the total number of customers routed to server i among the first k arrivals counting backwards in time. $L_n(i)$ denotes the index of the nth customer that gets routed to server i in the common arrival stream, counting backwards in time. $\hat{S}_{-n}^{(i)}$ denotes the service time of the nth customer that gets routed to server i, counting backwards in time.

For each $i = 1, \ldots, c$, let $\{\hat{X}_{-n}^{(i)} \colon n \geq 0\}$ with $\hat{X}_0^{(i)} = 0$ be an auxiliary process such that

$$\hat{X}_{-n}^{(i)} := \sum_{j=1}^{n} (\hat{S}_{-j}^{(i)} - a) = X_{-L_n(i)}(i). \tag{38}$$

For $n \geq 0$ and $1 \leq i \leq c$, define

$$\hat{N}_n(i) = \inf\left\{n' \geq N_n(i) \colon \max_{k \geq n'} \hat{X}_{-k}^{(i)} \leq \hat{X}_{-N_n(i)}^{(i)}\right\}, \tag{39}$$

and hence, by definition, in (23), we have

$$N_n^X = \max\{L_{\hat{N}_n(1)}(1), \ldots, L_{\hat{N}_n(c)}(c)\}. \tag{40}$$

First we develop simulation algorithms for each of the c one-dimensional auxiliary processes $\{(\hat{X}_{-n}^{(i)} \colon n \geq 0) \colon 1 \leq i \leq c\}$. Next we use the common server allocation sequence $\{U_{-n} \colon n \geq 0\}$ (sampled jointly with the process $\{\mathbf{Y}_{-n} \colon n \geq 0\}$ in Section A.1) with (35), (36), and (37) to find N_n^X via (40) for each $n \geq 0$.

A.2.1. *'Milestone' construction and global maximum simulation.* For each one-dimensional auxiliary process $\{\hat{X}^{(i)}_{-n}\colon n \geq 0\}$ with $i = 1, \ldots, c$, we adopt the algorithm developed in [6] by choosing any $m' > 0$ and $L' \geq 1$ properly and define the sequences of upward and downward 'milestone' events by letting $D^{(i)}_0 = 0$, $\Gamma^{(i)}_0 = \infty$, and for $j \geq 1$,

$$D^{(i)}_j = \inf\left\{n^{(i)} \geq \Gamma^{(i)}_{j-1} I(\Gamma^{(i)}_{j-1} < \infty) \vee D^{(i)}_{j-1}\colon \hat{X}^{(i)}_{-n^{(i)}} < \hat{X}^{(i)}_{-D^{(i)}_{j-1}} - L'm'\right\}, \tag{41}$$

$$\Gamma^{(i)}_j = \inf\left\{n^{(i)} \geq D^{(i)}_j\colon \hat{X}^{(i)}_{-n^{(i)}} - \hat{X}^{(i)}_{-D^{(i)}_j} > m'\right\}, \tag{42}$$

with the convention that if $\Gamma^{(i)}_j = \infty$, then $\Gamma^{(i)}_j I(\Gamma^{(i)}_j < \infty) = 0$ for any $j \geq 0$.

For each $i = 1, \ldots, c$, define

$$\Lambda^{(i)} = \inf\{D^{(i)}_k\colon \Gamma^{(i)}_k = \infty, k \geq 1\}.$$

By the 'milestone' construction in (41) and (42), for all $n \geq \Lambda^{(i)}$,

$$\hat{X}^{(i)}_{-n} \leq \hat{X}^{(i)}_{-\Lambda^{(i)}} + m' < 0 = \hat{X}^{(i)}_0.$$

Therefore we can evaluate the global maximum of the infinite-horizon process $\{\hat{X}^{(i)}_{-n}\colon n \geq 0\}$ in finite steps, that is,

$$M^{(i)}_0 := \max_{k \geq 0} \hat{X}^{(i)}_{-k} = \max_{0 \leq k \leq \Lambda^{(i)}} \hat{X}^{(i)}_{-k}.$$

We summarize the simulation details in the following algorithm.

Algorithm 3. (GGM.) Simulate the global maximum of the one-dimensional process

$$\{(\hat{X}^{(i)}_{-n}, \hat{S}^{(i)}_{-n})\colon n \geq 0\}$$

jointly with the subpath and the subsequence of 'milestone' events.

Input: a, m', L'.

1. (*Initialization*) Set $n = 0$, $\hat{X}^{(i)}_0 = 0$, $\mathbf{D}^{(i)} = [0]$, $\mathbf{\Gamma}^{(i)} = [\infty]$, and $L^{(i)} = 0$.
2. Generate $S \sim G$. Set $n = n + 1$, $\hat{X}^{(i)}_{-n} = \hat{X}^{(i)}_{-(n-1)} + S$, and $\hat{S}^{(i)}_{-n} = S$.
3. If $\hat{X}^{(i)}_{-n} \geq L^{(i)} - L'm'$, go to step 2; otherwise set $\mathbf{D}^{(i)} = [\mathbf{D}^{(i)}, n]$ and $L^{(i)} = \hat{X}^{(i)}_{-n}$.
4. Call Algorithm 1 of [6, page 10] and obtain (J, ω).
5. If $J = 1$, set

   $$\hat{X}^{(i)}_{-(n+l)} = L^{(i)} + \omega(l), \quad \hat{S}^{(i)}_{-(n+l)} = \hat{X}^{(i)}_{-(n+l)} - \hat{X}^{(i)}_{-(n+l-1)} + a \quad \text{for } l = 1, \ldots, \text{length}(\omega).$$

   Set $n = n + \text{length}\,(\omega)$, $\mathbf{\Gamma}^{(i)} = [\mathbf{\Gamma}^{(i)}, n]$ and go to step 2.
6. If $J = 0$, set $\Lambda^{(i)} = n$ and $\mathbf{\Gamma}^{(i)} = [\mathbf{\Gamma}^{(i)}, \infty]$.
7. Output

   $$\{(\hat{X}^{(i)}_{-k}, \hat{S}^{(i)}_{-k})\colon 1 \leq k \leq \Lambda^{(i)}\},$$

   $\mathbf{D}^{(i)}$, $\mathbf{\Gamma}^{(i)}$, and global maximum $M^{(i)}_0 = \max_{0 \leq k \leq \Lambda^{(i)}} \hat{X}^{(i)}_{-k}$.

A.2.2. *Simulate* $\{\mathbf{X}_{-n}\colon n\geq 0\}$ *jointly with 'milestone' events.* In this section we first explain how to sample the auxiliary one-dimensional processes $\{\hat{X}^{(i)}_{-n}\colon n\geq 0\}$ along with the 'milestone' events defined in (41) and (42). Next we will need the service allocation information $\{U_{-n}\colon n\geq 0\}$, from the simulation procedure of process $\{\mathbf{Y}_{-n}\colon n\geq 0\}$, to recover the multi-dimensional process of interest $\{\mathbf{X}_{-n}\colon n\geq 0\}$ via (38).

The following algorithm gives the the sampling procedure for each auxiliary one-dimensional process $\{\hat{X}^{(i)}_{-n}\colon n\geq 0\}$ for $i=1,\ldots,c$. The simulation steps are the same as the procedure given in Algorithm 3 of [6, page 16].

Algorithm 4. (GRW.) Continue to sample the process $\{(\hat{X}^{(i)}_{-k}, \hat{S}^{(i)}_{-k})\colon 0\leq k\leq n\}$ jointly with the partially sampled 'milestone' event lists $\mathbf{D}^{(i)}$ and $\mathbf{\Gamma}^{(i)}$, until a stopping criterion is met.

Input: a, m', L', previously sampled partial process $\{(\hat{X}^{(i)}_{-j}, \hat{S}^{(i)}_{-j})\colon 0\leq j\leq l\}$, partial 'milestone' sequences $\mathbf{D}^{(i)}$ and $\mathbf{\Gamma}^{(i)}$, and stopping criterion $\mathcal{H}^{(i)}$. (Note that if there is no previously simulated random walk, we initialize $l=0$, $\mathbf{D}^{(i)}=[0]$, and $\mathbf{\Gamma}^{(i)}=[\infty]$.)

1. Set $n=l$. If $n=0$, call Algorithm GGM to get $\Lambda^{(i)}$, $\{(\hat{X}^{(i)}_{-k}, \hat{S}^{(i)}_{-k})\colon 0\leq k\leq \Lambda^{(i)}\}$, $\mathbf{D}^{(i)}$, and $\mathbf{\Gamma}^{(i)}$. Set $n=\Lambda^{(i)}$.
2. While the stopping criterion $\mathcal{H}^{(i)}$ is not satisfied:
   (a) Call Algorithm GGM to get $\tilde{\Lambda}^{(i)}$, $\{(\tilde{X}^{(i)}_{-j}, \tilde{S}^{(i)}_{-j})\colon 0\leq j\leq \tilde{\Lambda}^{(i)}\}$, $\tilde{\mathbf{D}}$, $\tilde{\mathbf{\Gamma}}$, and $\tilde{M}^{(i)}_0$.
   (b) If $\tilde{M}^{(i)}_0\leq m'$, accept the proposed sequence and concatenate it to the previous subpath, that is, set $\hat{X}^{(i)}_{-(n+j)}=\hat{X}^{(i)}_{-n}+\tilde{X}^{(i)}_{-j}$, $\hat{S}^{(i)}_{-(n+j)}=\tilde{S}^{(i)}_{-j}$ for $1\leq j\leq \tilde{\Lambda}^{(i)}$. Update the sequences of 'milestone' events to be $\mathbf{D}^{(i)}=[\mathbf{D}^{(i)}, n+\tilde{\mathbf{D}}^{(i)}(2\colon \text{end})]$, $\mathbf{\Gamma}^{(i)}=[\mathbf{\Gamma}^{(i)}, n+\tilde{\mathbf{\Gamma}}^{(i)}(2\colon \text{end})]$ and set $n=n+\tilde{\Lambda}^{(i)}$.
3. Output $\{(\hat{X}^{(i)}_{-k}, \hat{S}^{(i)}_{-k})\colon 0\leq k\leq n\}$ with updated 'milestone' event sequences $\mathbf{D}^{(i)}$ and $\mathbf{\Gamma}^{(i)}$.

With the service allocation information $\{U_{-n}\colon n\geq 0\}$, we can construct the c-dimensional process $\{\mathbf{X}_{-n}\colon n\geq 0\}$ ($\mathbf{X}_0=\mathbf{0}$) from the auxiliary processes $\{(\hat{X}^{(i)}_{-n}, \hat{S}^{(i)}_{-n})\colon n\geq 0\}$, $i=1,\ldots,c$. For $n\geq 1$,

$$S_{-n}=\hat{S}^{(U_{-n})}_{\sum_{j=1}^{n} I(U_{-j}=U_{-n})}, \tag{43}$$

$$X_{-n}(i)=\begin{cases}[c]lrX_{-(n-1)}(i) & \text{if } i\neq U_{-n},\\ X_{-(n-1)}+S_{-n}-a & \text{if } i=U_{-n}.\end{cases} \tag{44}$$

By the definition of 'milestone' events in (41) and (42), for each $n\geq 0$, let

$$d^{(i)}_1(n)=\inf\left\{D^{(i)}_k\geq n\colon \hat{X}^{(i)}_{-D^{(i)}_k}\leq \hat{X}^{(i)}_{-n}\right\}, \tag{45}$$

$$d^{(i)}_2(n)=\inf\left\{D^{(i)}_k> d^{(i)}_1(n)\colon \Gamma^{(i)}_k=\infty\right\}. \tag{46}$$

Since

$$\max_{k\geq d^{(i)}_2(N_n(i))}\hat{X}^{(i)}_{-k}\leq \hat{X}^{(i)}_{-d^{(i)}_2(N_n(i))}+m'<\hat{X}^{(i)}_{-d^{(i)}_1(N_n(i))}\leq \hat{X}^{(i)}_{-N_n(i)},$$

we conclude that $\hat{N}_n(i) \le d_2^{(i)}(N_n(i))$ and hence

$$N_n^X \le \max\left\{L_{d_2^{(1)}(N_n(1))}(1), \ldots, L_{d_2^{(c)}(N_n(c))}(c)\right\}.$$

A.3. Simulation algorithm for $\{\mathbf{R}_n^{(r)}: n \ge 0\}$ and coalescence detection

We shall combine the simulation algorithms in Sections A.1 and A.2 for processes

$$\{((\hat{X}_{-n}^{(i)}, \hat{S}_{-n}^{(i)}): n \ge 0), 1 \le i \le c\} \quad \text{and} \quad \{(\mathbf{Y}_{-n}, U_{-n}, T_{-n}): n \ge 0\}$$

together to exactly simulate the multi-dimensional random walk $\{\mathbf{R}_n^{(r)}: n \ge 0\}$ until coalescence time N defined in (10). To detect the coalescence, we start from $n = 0$ to compute $d_2(n)$ and $d_2^{(i)}(N_n(i))$ (as defined in (34) and (36) respectively). If

$$\max_{n \le k \le d_2(n)} \mathbf{Y}_{-k} = \mathbf{Y}_{-n}, \tag{47}$$

and

$$\max_{N_n(i) \le k \le d_2^{(i)}(N_n(i))} \hat{X}_{-k}^{(i)} = \hat{X}_{-N_n(i)}^{(i)} \tag{48}$$

for all $i = 1, \ldots, c$, we set the coalescence time $N \leftarrow n$ and stop. Otherwise we increase n by 1 and repeat the above procedure until the first time that (47) and (48) are satisfied.

In the following algorithm we give the simulation procedure to detect coalescence while sampling the time-reversed multi-dimensional process $\{\mathbf{R}_n^{(r)}: n \ge 0\}$.

Algorithm 5. (CD.) Sample the coalescence time N jointly with the process $\{\mathbf{R}_n^{(r)}: n \ge 0\}$.

Input: a, m, m', L'.

1. (*Initialization*) Set $n = 0$. Set $l = 0$, $\mathbf{Y}_0 = \mathbf{0}$, $\mathbf{D} = [0]$, and $\mathbf{\Gamma} = [\infty]$. Set $l_i = 0$, $\hat{X}_0^{(i)} = 0$, $\mathbf{D}^{(i)} = [0]$, and $\mathbf{\Gamma}^{(i)} = [\infty]$ for all $i = 1, \ldots, c$.
2. Call Algorithm LTRW to further sample $\{(\mathbf{Y}_{-j}, U_{-j}, T_{-j}): 0 \le j \le l\}$, $\mathbf{D}$, and $\mathbf{\Gamma}$ with the stopping criterion $\mathcal{H}$ being $\sum_{j=1}^{l} I(U_{-j} = i) > l_i$ for all $i = 1, \ldots, c$ and $\mathbf{Y}_{-\mathbf{D}(\text{end}-1)} \le \mathbf{Y}_{-n}$.
3. For each $i = 1, \ldots, c$:
   (a) Set $n_i = \sum_{j=1}^{n} I(U_{-j} = i)$.
   (b) Call Algorithm GRW to further sample $\{(\hat{X}_{-k}^{(i)}, \hat{S}_{-k}^{(i)}): 0 \le k \le l_i\}$, $\mathbf{D}^{(i)}$, and $\mathbf{\Gamma}^{(i)}$ with the stopping criterion $\mathcal{H}^{(i)}$ being $\sum_{j=1}^{l} I(U_{-j} = i) \le l_i$ and $\hat{X}^{(i)}_{-\mathbf{D}^{(i)}(\text{end}-1)} \le \hat{X}^{(i)}_{-n_i}$.
4. If $\max_{n \le k \le \mathbf{D}(\text{end})} \mathbf{Y}_{-k} \le \mathbf{Y}_{-n}$ and $\max_{n_i \le k \le \mathbf{D}^{(i)}(\text{end})} \hat{X}_{-k}^{(i)} \le \hat{X}_{-n_i}^{(i)}$ for all $i = 1, \ldots, c$, go to the next step. Otherwise set $n = n + 1$ and go to step 2.
5. For $1 \le k \le n$, recover S_{-k} and $\mathbf{X}_{-k}$ from the auxiliary processes via (43) and (44).
6. Output coalescence time $N = n$, the sequence $\{(U_{-k}, T_{-k}, S_{-k}): 0 \le k \le n\}$, and process $\{\mathbf{R}_k^{(r)}: 0 \le k \le n\}$.

Appendix B. Proofs

B.1. Proof of Proposition 1

Proof. Firstly, $\mathbb{E}[N] < \infty$ holds true under assumptions $\rho < c$ and $\mathbb{P}(T > S) > 0$ (proved in [18]). Next we shall prove the computational cost τ has finite expectation as well.

For $n \geq 0$, we have N_n^X, N_n^Y, and N_n defined in (23)–(25) such that

$$\max_{k \geq N_n} \mathbf{R}_k^{(r)} \leq \max_{k \geq N_n} \mathbf{X}_{-k} + \max_{k \geq N_n} \mathbf{Y}_{-k} \leq \mathbf{X}_n + \mathbf{Y}_n = \mathbf{R}_n^{(r)}.$$

Therefore, in order to evaluate the running time maximum over the infinite horizon $\max_{k \geq n} \mathbf{R}_k^{(r)}$, it only requires sampling from n to N_n backwards in time, that is,

$$\max_{k \geq n} \mathbf{R}_k^{(r)} = \max \left\{ \max_{n \leq k \leq N_n} \mathbf{R}_k^{(r)}, \max_{k \geq N_n} \mathbf{R}_k^{(r)} \right\} = \max_{n \leq k \leq N_n} \mathbf{R}_k^{(r)}.$$

An easy upper bound for τ is given by $\tilde{\tau} = \sum_{n=0}^{N} N_n$. By Wald's identity, it suffices to show that $\mathbb{E}[N_n] < \infty$ for any $n \geq 0$.

By the 'milestone' events construction for multi-dimensional process $\{\mathbf{Y}_{-n} \colon n \geq 0\}$ in (27), (28) and because $d_2(n)$ is an upper bound of N_n^Y, $\mathbb{E}[N_n^Y] \leq \mathbb{E}[d_2(n)] < \infty$ follows directly from elementary properties of compound geometric random variables (see Theorem 1 of [3]).

For the other process $\{\mathbf{X}_{-n} \colon n \geq 0\}$, we simulate each of its c entries separately, that is, $\{\{\hat{X}_{-n}^{(i)} \colon n \geq 0\} \colon 1 \leq i \leq c\}$ in Section A.2. Equation (40) gives

$$N_n^X = \max\{L_{\hat{N}_n(1)}(1), \ldots, L_{\hat{N}_n(c)}(c)\} \leq \sum_{i=1}^{c} L_{\hat{N}_n(i)}(i),$$

where $\hat{N}_n(i)$ is defined in (39). By Theorem 2.2 of [6], $\mathbb{E}[\hat{N}_n(i)] < \infty$. Because

$$L_{\hat{N}_n(i)}(i) = \inf \left\{ k \geq 0 \colon \sum_{j=1}^{k} I(U_{-j} = i) = \hat{N}_n(i) \right\} \sim \text{NegBinomial}\left(\hat{N}_n(i); 1 - \frac{1}{c} \right) + \hat{N}_n(i),$$

hence

$$\mathbb{E}[L_{\hat{N}_n(i)}(i)] = (c-1)\mathbb{E}[\hat{N}_n(i)] + \mathbb{E}[\hat{N}_n(i)] = c\mathbb{E}[\hat{N}_n(i)] < \infty,$$

and

$$\mathbb{E}[N_n^X] \leq \sum_{i=1}^{c} \mathbb{E}[L_{\hat{N}_n(i)}(i)] < \infty.$$

Therefore

$$\mathbb{E}[N_n] \leq \mathbb{E}[N_n^X] + \mathbb{E}[N_n^Y] < \infty.$$

□

B.2. Proof of Proposition 2

Proof. By Wald's identity, it suffices to show that $\mathbb{E}[\kappa_+^*] < \infty$ because $\mathbb{E}[T] < \infty$. Next we only provide a proof outline here since it follows the same argument as in the proof of Proposition 3 in [7].

Firstly, we construct a sequence of events $\{\Omega_k : k \geq 1\}$, which leads to the occurrence of κ_+^*. Secondly, we split the process $\{\mathbf{W}_0^u(t_n) : n \geq 0\}$ into cycles with bounded expected cycle length. We also ensure the probability that the event happens during each cycle is bounded from below by a constant, which allows us to bound the number of cycles we need to check before finding κ_+^* by a geometric random variable. Finally, we could establish an upper bound for $\mathbb{E}[\kappa_+^*]$ by applying Wald's identity again. □

Acknowledgements

Support from the NSF via grants CMMI-1538217, DMS-1820942, and DMS-1838676 is gratefully acknowledged.

References

[1] ASMUSSEN, S. (2008). *Applied Probability and Queues* (Applications of Mathematics: Stochastic Modelling and Applied Probability 51). Springer.
[2] ASMUSSEN, S., GLYNN, P. W. AND THORISSON, H. (1992). Stationarity detection in the initial transient problem. *ACM Trans. Model. Comput. Simul.* **2**, 130–157.
[3] BLANCHET, J. AND CHEN, X. (2015). Steady-state simulation of reflected Brownian motion and related stochastic networks. *Ann. Appl. Prob.* **25**, 3209–3250.
[4] BLANCHET, J. AND DONG, J. (2015). Perfect sampling for infinite server and loss systems. *Adv. Appl. Prob.* **47**, 761–786.
[5] BLANCHET, J. H. AND SIGMAN, K. (2011). On exact sampling of stochastic perpetuities. *J. Appl. Prob.* **48**, 165–182.
[6] BLANCHET, J. AND WALLWATER, A. (2015). Exact sampling of stationary and time-reversed queues. *ACM Trans. Model. Comput. Simul.* **25**, 26.
[7] BLANCHET, J., DONG, J. AND PEI, Y. (2018). Perfect sampling of GI/GI/*c* queues. *Queueing Systems* **90**, 1–33.
[8] CONNOR, S. B. AND KENDALL, W. S. (2015). Perfect simulation of M/G/*c* queues. *Adv. Appl. Prob.* **47**, 1039–1063.
[9] DAI, H. (2011). Exact Monte Carlo simulation for fork–join networks. *Adv. Appl. Prob.* **43**, 484–503.
[10] FLATTO, L. AND HAHN, S. (1984). Two parallel queues created by arrivals with two demands, I. *SIAM J. Appl. Math.* **44**, 1041–1053.
[11] FOSS, S. (1980). Approximation of multichannel queueing systems. *Siberian Math. J.* **21**, 851–857.
[12] FOSS, S. G. AND CHERNOVA, N. I. (2001). On optimality of the FCFS discipline in multiserver queueing systems and networks. *Siberian Math. J.* **42**, 372–385.
[13] HALFIN, S. AND WHITT, W. (1981). Heavy-traffic limits for queues with many exponential servers. *Operat. Res.* **29**, 567–588.
[14] HILLIER, F. S. AND LO, F. D. (1972). *Tables for multiple-server queueing systems involving Erlang distributions*. Technical report, Stanford University, CA.
[15] KENDALL, W. S. AND MØLLER, J. (2000). Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. Adv. Appl. Prob. 32, 844–865.
[16] KIEFER, J. AND WOLFOWITZ, J. (1955). On the theory of queues with many servers. *Trans. Amer. Math. Soc.* **78**, 1–18.
[17] PROPP, J. G. AND WILSON, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures Algorithms* **9**, 223–252.
[18] SIGMAN, K. (1988). Regeneration in tandem queues with multiserver stations. *J. Appl. Prob.* **25**, 391–403.
[19] SIGMAN, K. (1995). *Stationary Marked Point Processes: An Intuitive Approach* (Stochastic Modeling Series 2). Taylor & Francis.
[20] SIGMAN, K. (2011). Exact simulation of the stationary distribution of the FIFO M/G/*c* queue. *J. Appl. Prob.* **48**, 209–213.
[21] SIGMAN, K. (2012). Exact simulation of the stationary distribution of the FIFO M/G/*c* queue: the general case for. *Queueing Systems* **70**, 37–43.
[22] WOLFF, R. W. (1987). Upper bounds on work in system for multichannel queues. *J. Appl. Prob.* **24**, 547–551.
[23] WOLFF, R. W. (1989). *Stochastic Modeling and the Theory of Queues* (Prentice Hall International Series in Industrial and Systems Engineering 14). Prentice Hall, Englewood Cliffs, NJ.