International Journal of Statistics and Probability; Vol. 4, No. 1; 2015
ISSN 1927-7032 E-ISSN 1927-7040
Published by Canadian Center of Science and Education

Exact Simulation for Fork-Join Networks with Heterogeneous Service

Hongsheng Dai[1]

[1] Department of Mathematical Sciences, University of Essex, Colchester, UK

Correspondence: Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, UK. Tel: 44-1206-873-304. E-mail: hdaia@essex.ac.uk

Received: October 16, 2014 Accepted: October 31, 2014 Online Published: December 19, 2014

doi:10.5539/ijsp.v4n1p19 URL: http://dx.doi.org/10.5539/ijsp.v4n1p19

Abstract

This paper considers a fork-join network with a group of heterogeneous servers in each service station, e.g. servers having different service rate. The main research interests are the properties of such fork-join networks in equilibrium, such as distributions of response times, maximum queue lengths and load carried by servers. This paper uses exact Monte-Carlo simulation methods to estimate the characteristics of heterogeneous fork-join networks in equilibrium, for which no explicit formulas are available. The algorithm developed is based on coupling from the past. The efficiency of the sampling algorithm is shown theoretically and via simulation.

Keywords: Coupling from the past; Fork-join networks; Heterogeneous service; Homogeneous service; Perfect sampling.

1. Introduction

A fork-join network consists of K parallel service stations, where each incoming job is split into K subtasks at the fork station and processed separately in each service station. When all the K subtasks of a job are completed, they will be joined immediately at the join station and leave the system. Such fork-join networks can be used to model manufacturing systems or computing systems (see Ko and Serfozo (2004), Lebrecht and Knottenbelt (2007) and Dai (2011)).

A typical fork-join network is shown in Figure 1, where each service station has waiting capacity N (the number of task-waiting places). The ith service station has s_i servers where the jth server has exponential service times with rate μ_{ij}. If $\mu_{ij} = \mu_i, \forall j$ we say that the network has homogeneous service; if $\mu_{ij}, j = 1, \cdots, s_i$ are not all the same we say that the network has heterogeneous service.

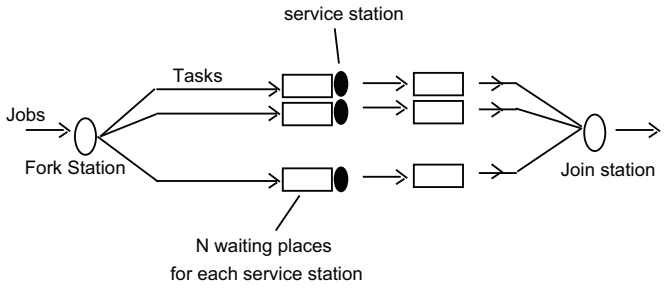

Figure 1. A fork-join network.

Heterogeneity of service is a common feature of many real multi-server queueing situations. It allows tasks to receive different quality of service. The heterogeneous service mechanisms are clearly valid for almost any manufacturing system. Detailed discussions about heterogeneous service can be found in Kumar and Madheswari (2007). However, heterogeneous service were rarely treated in queuing theory (Kumar and Madheswari, 2007), especially for fork-join networks. The difficulty is that the stationary distribution of the fork-join network, even if the network is homogeneous, is intractable Ko and Serfozo (2004). The networks with heterogeneous servers are much more difficult to deal with than the networks with homogeneous servers. This is because for heterogeneous servers when more than one server is available the waiting task chooses which server to occupy must be specified, which makes the Markov model to be multi-dimensional instead of one-dimensional (see discussions for queuing models with heterogeneous servers in Cooper (1976).

For the simplest fork-join network having two service stations with each having a single server, Flatto and Hahn (1984) and Flatto (1985) derived the generating functions of the queue-length distribution. For more general fork-join networks with homogeneous service, existing methods focus on finding approximations or bounds of mean response times and approximations of maximum queue-length distributions with $N = \infty$ (infinite waiting capacity), such as Nelson and Tantawi (1988), Baccelli et al. (1989), Balsoma et al. (1998), Raghavan and Viswanadham (2001) and Ko and Serfozo (2004).

The above methods focused on analytical approximations. However simulation results in Ko and Serfozo (2004) demonstrate that the accuracy of such approximations may be poor in some cases. Dai (2011) considers the use of exact Monte Carlo simulations, based on *coupling from the past* (CFTP) (Propp and Wilson, 1996), to estimate the distributions of response time and queue length for homogeneous fork-join networks with $N < \infty$. Comparing to analytical approximations one advantage of the exact Monte Carlo simulation methods is that the accuracy is controlled by the number of simulated samples. Note that one may consider using approximate simulations (for example Markov chain Monte Carlo methods) which draw samples approximately from equilibrium. However it is very difficult to justify the quality of simulated samples via approximate simulation. On the contrary the exact Monte Carlo simulation methods, which can draw samples exactly from the target distribution (*perfect* samples), are preferable in practice. The other advantage of the exact Monte Carlo simulation methods is that it can provide empirical distribution estimates for response time and maximum queue length. These empirical estimates can further provide all characteristics of these distributions such as mean, median and quartiles. Existing analytical approximations, however, focus on the mean response time approximations which cannot catch much information about the distribution of response time. For example Dai (2011) showed that the distribution of response time has a long tail and is highly skewed.

This paper considers stationary distributions of heterogeneous fork-join networks with $N < \infty$, for which no analytical formulas are available. We consider different strategies to allocate subtasks to idle servers when more than one server is available, 1) a subtask chooses the fastest server (fastest strategy); 2) a subtask chooses the slowest server (slowest strategy); 3) a subtask chooses its server at random from all those idle (random strategy). We propose exact simulation methods, based on CFTP, to generate exact realisations from the equilibrium of such networks. Based on the simulated realisations we provide Monte Carlo estimates for load carried by servers, the distributions of maximum queue lengths and response times.

This paper is organized as follows. In Section 2, we introduce model notations. Then in Section 3 and Section 4 we develop a CFTP algorithm with bounding chains to simulate exact realisations from the equilibrium of heterogeneous fork-join networks with different allocation strategies. Complexities of the algorithms and simulation studies are provided in Section 5. Section 6 provides a discussion.

2. Preliminaries

2.1 Notations

Consider the fork-join network in Figure 1. Jobs arrive at the system according to a Poisson process with rate λ. Each incoming job is split into K subtasks which are simultaneously assigned to K parallel stations for processing. The ith station has s_i servers in which the service rate for the jth server is μ_{ij}. When all the K subtasks of a job are completed, they will be joined immediately at the join station (the service time at the join station is 0) and leave the system. Service station i can hold $N_i = N + s_i$ subtasks at most. Denote $\boldsymbol{N} = (N_1, \cdots, N_K)$. If the subtasks of a job are not all completed, the completed subtasks will wait in the corresponding buffer of the join station. We also assume that the join station has enough places to hold all completed waiting subtasks.

Homogeneous fork-join networks ($\mu_{ij} = \mu_i$, $\forall\, j = 1, \cdots, s_i$) can be represented as a K-dimensional continuous-

time Markov chain $\boldsymbol{Q}(t) = (Q_1(t), \cdots, Q_K(t))$, where $Q_i(t)$ means the number of subtasks at service station i at time t. However, when μ_{ij}s are different, i.e. each server works at its own characteristic rate, the K-dimensional continuous time Markov chain $\boldsymbol{Q}(t)$ cannot totally represent the network. For example, $Q_i(t) = 1$ only means that there is one subtask at service in station i but we do not know which server is active. For such heterogeneous networks the service rate in station i at time t is $\sum_{j=1}^{s_i} \mu_{ij} R_{ij}(t)$, where $R_{ij}(t) = 0$ if the server is idle and $R_{ij}(t) = 1$ is the server is busy at time t (Cooper 1976). Thus the heterogeneous fork-join network can be represented by $R_{ij}(t)$, $j = 1, \cdots, s_i, i = 1, \cdots, K$ together with $R_{i,s_i+1}(t)$, the number of subtasks waiting in the queue.

Let $\boldsymbol{R}_i(t) = (R_{i1}(t), \cdots, R_{i,s_i}(t), R_{i,s_i+1}(t))$. The fork-join network can then be represented by a Markov process $\boldsymbol{R}(t) = (\boldsymbol{R}_1^{\mathrm{T}}(t), \cdots, \boldsymbol{R}_K^{\mathrm{T}}(t))^{\mathrm{T}}$. If we denote the state space of $\boldsymbol{R}(t)$ as $\mathcal{R}$, then its element is a two-dimensional array with K rows and the ith row has $s_i + 1$ elements. The last element $R_{i,s_i+1}(t)$, the number of tasks waiting in the queue of station i, takes integers from 0 to N. Other elements $R_{i1}(t), \cdots, R_{i,s_i}(t)$ take values 0 or 1 which means the corresponding server is idle or active. If some $R_{im}(t) = 0$ for $m \in \{1, \cdots, s_i\}$ then $R_{i,s_i+1}(t)$ must be 0.

2.2 Allocation Strategies

If we label the servers from 1 (the fastest) to s_i (the slowest), the fastest strategy (a subtask chooses the fastest idle server) can be viewed as that a newly arrived subtask goes to the idle server with the smallest label or will wait in a queue if all servers are busy. Similarly for the slowest strategy with slowest server labelled as 1 and the fastest labelled as s_i. Denote $\mathcal{R}$ as the sate space of the network and suppose that the current state of the network is $\boldsymbol{r} \in \mathcal{R}$. The transitions of a new job arrival under the fastest/slowest strategy can be denoted as $\boldsymbol{r}' = \boldsymbol{r} \oplus \mathbf{1}$, where $\mathbf{1}$ is a K-dimensional vector with all elements equal to 1 and

$$\begin{aligned} & \boldsymbol{r}' = \boldsymbol{r} \oplus \mathbf{1} \Leftrightarrow \boldsymbol{r}_i' = \boldsymbol{r}_i \oplus 1 \text{ for all } i; \\ & \boldsymbol{r}_i' = \boldsymbol{r}_i \oplus 1 \\ \Leftrightarrow \quad & r_{ij}' = \begin{cases} r_{ij} + 1 & \text{if } j \text{ is the smallest server label such that } r_{ij} = 0; \\ r_{ij} + 1 & \text{if } j = s_i + 1 \text{ and } r_{im} > 0 \text{ for all } m \in \{1, \cdots, s_i\}; \\ r_{ij} & \text{for other label } j. \end{cases} \end{aligned} \tag{1}$$

For the random strategy a subtask chooses its server at random from all those idle. We denote the transitions of a new job arrival under the random strategy as $\boldsymbol{r}' = \boldsymbol{r} \uplus \mathbf{1}$, which is defined as $\boldsymbol{r}_i' = \boldsymbol{r}_i \uplus 1$ for all i and

$$\begin{aligned} & \boldsymbol{r}_i' = \boldsymbol{r}_i \uplus 1 \\ \Leftrightarrow \quad & r_{ij}' = \begin{cases} r_{ij} + 1 & \text{if } j \text{ is chosen randomly from those such that } r_{ij} = 0, j \le s_i; \\ r_{ij} + 1 & \text{if } j = s_i + 1 \text{ and } r_{im} > 0 \text{ for all } m \in \{1, \cdots, s_i\}; \\ r_{ij} & \text{for other label } j. \end{cases} \end{aligned} \tag{2}$$

2.3 Partial Order

We consider the following partial orders for the state space, which will be used later when we develop CFTP algorithms. For $\boldsymbol{r}, \boldsymbol{v} \in \mathcal{R}$, define $\boldsymbol{r} \prec \boldsymbol{v}$ if $r_{ij} \le v_{ij}, \forall i, j$ and $r_{ij} < v_{ij}$ for at least a pair of (i, j). Define $\boldsymbol{r} = \boldsymbol{v}$ if $r_{ij} = v_{ij}, \forall i, j$.

Note that for $\boldsymbol{r} \in \mathcal{R}$, its elements r_{ij}, $j = 1, \cdots, s_i, i = 1, \cdots, K$ are bounded since they are all $0, 1$ values. But $r_{i,s_i+1}, i = 1, \cdots, K$ can take values from 0 to N which represent the number of waiting tasks. Thus when $N = \infty$ there is no maximum element in $\mathcal{R}$.

3. Exact Simulation for Heterogeneous Networks with the Fastest/Slowest Strategy

We assume throughout this paper that $\lambda < \sum_{j=1}^{s_i} \mu_{ij}$ for all $i = 1, \cdots, K$. We can simulate $\boldsymbol{R}(t)$ exactly, via the CFTP algorithm Propp and Wilson (1996), which runs Markov chains starting at all different states from the past and if all chains coalesce before time 0 we collect the sample at time 0 which is exactly from equilibrium. Such CFTP algorithm is not practical since it requires heavy computation to monitoring coalescence of all different chains.

Propp and Wilson (1996) also proposed the monotone CFTP algorithm. If the Markov chains preserve some partial order we only need to run two chains, the upper chain from the maximum state and the lower chain from the minimum state. All chains are bounded by them and if the two chains coalesce all other chains coalesce. Such monotone CFTP algorithm is much more efficient than the simple CFTP. When $N = \infty$, for the partial order defined in Section 2.3 there is no maximum element in $\mathcal{R}$ since r_{i,s_i+1} ranges from 0 to N. Therefore the monotone CFTP algorithm is not readily available. It is very challenging to deal with $N = \infty$ although it might be possible to use

the dominated CFTP method in Kendall and Moller (2000). We will provide a discussion on $N = \infty$ in the end of this paper, but here we simply assume that $N < \infty$, which is reasonable in practice. Then the elements in $\mathcal{R}$ are bounded. However, with $N < \infty$ the partial order defined in Section 2.3 is not always preserved by the Markov process $\mathcal{R}(t)$, which is shown later in this section. Therefore the monotone CFTP algorithm is still not available for $N < \infty$. But since $\mathcal{R}$ is bounded, we can use CFTP with bounding chains. Examples of CFTP with bounding chains can be found in Dai (2008), Dai (2011) and Huber (2004).

3.1 Continuous Time Markov Chain Simulation for $\boldsymbol{R}(t)$

The transition rate for $\boldsymbol{R}(t)$ from a state $\boldsymbol{r}$ to another state $\boldsymbol{v}$ is given by

$$f(\boldsymbol{r},\boldsymbol{v}) = \begin{cases} \lambda & \text{if } \boldsymbol{v} = \boldsymbol{r} \oplus \mathbf{1} \text{ and } r_{i,s_i+1} < N, \text{ for all } i, \\ r_{ij}\mu_{ij} & \text{if } \boldsymbol{v} = \boldsymbol{r} \ominus \boldsymbol{e}_{ij}, j = 1, \cdots, s_i \text{ and } r_{i,s_i+1} = 0 \\ \sum_{m=1}^{s_i} r_{im}\mu_{im} & \text{if } \boldsymbol{v} = \boldsymbol{r} \ominus \boldsymbol{e}_{i,s_i+1}, \text{ and } r_{i,s_i+1} > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $\boldsymbol{e}_{ij}$ has the same dimensions as the element in $\mathcal{R}$ and it has all elements equal to 0 except that the jth element in the ith row is equal to 1. Note that in (3) the transition $\boldsymbol{v} = \boldsymbol{r} \oplus \mathbf{1}$ means a new job arrives, thus the transition rate is λ. Also note that in (3) $\boldsymbol{v} = \boldsymbol{r} \ominus \boldsymbol{e}_{ij}$ corresponds to a subtask completion, defined as

$$\begin{aligned} & \boldsymbol{v} = \boldsymbol{r} \ominus \boldsymbol{e}_{ij} \\ \Leftrightarrow \quad & v_{ij} = \begin{cases} \max\{0, r_{ij} - 1\} & \text{if } j \in \{1, \cdots, s_i\} \text{ and } r_{i,s_i+1} = 0; \\ r_{ij} - 1 & \text{if } j = s_i + 1 \text{ and } r_{i,s_i+1} > 0; \end{cases} \\ & v_{km} = r_{km}, \quad \text{for all } m \neq j \text{ or } k \neq i \end{aligned} \tag{4}$$

In the first line of equation (4) $v_{km} = \max\{0, r_{km} - 1\}$ means that either a subtask is completed or no transitions at all. This can be explained as follows. In the first line the condition $r_{i,s_i+1} = 0$ means that no subtask is waiting in the queue of station i. Thus $r_{ij} = 1$ and $v_{ij} = 0$ correspond to the subtask in the jth server of the ith station is completed. In addition $r_{ij} = 0$ and $v_{ij} = 0$ correspond to no transitions. The corresponding transition rate is $r_{ij}\mu_{ij}$ in (3), which is μ_{ij} if $r_{ij} = 1$ and is 0 if $r_{ij} = 0$.

In the second line of equation (4), $r_{i,s_i+1} > 0$ means that there are subtasks waiting in the queue of station i. Thus $r_{im} = 1$ for all $m \in \{1, \cdots, s_i\}$ (all servers are active in station i). If the subtask in any server, say server m, of the ith station is completed then a subtask in the queue occupies the mth server immediately and the transition is from r_{i,s_i+1} to $v_{i,s_i+1} = r_{i,s_i+1} - 1$, which has transition rate $\sum_{m=1}^{s_i} r_{im}\mu_{im}$ in (3).

To simulate $\boldsymbol{R}(t)$, we need to simulate the holding time, the amount of time that the continuous Markov chain $\boldsymbol{R}(t)$ spends in the current state $\boldsymbol{r}$, and simulate transitions. From (3) we know that the holding time is exponentially distributed with rate

$$e_{\boldsymbol{r}} := \begin{cases} \lambda + \sum_i \sum_j r_{ij}\mu_{ij}, & \text{if } r_{i,s_i+1} < N, \text{ for all } i; \\ \sum_i \sum_j r_{ij}\mu_{ij}, & \text{otherwise.} \end{cases} \tag{5}$$

At the end of the holding time, $\boldsymbol{R}(t)$ jumps to another state $\boldsymbol{v}$ with transition probability

$$p_{\boldsymbol{r},\boldsymbol{v}} = \begin{cases} \lambda/e_{\boldsymbol{r}} & \text{if } \boldsymbol{v} = \boldsymbol{r} \oplus \mathbf{1} \text{ and } r_{i,s_i+1} < N, \text{ for all } i, \\ r_{ij}\mu_{ij}/e_{\boldsymbol{r}} & \text{if } \boldsymbol{v} = \boldsymbol{r} \ominus \boldsymbol{e}_{ij}, j \in \{1, \cdots, s_i\} \text{ and } r_{i,s_i+1} = 0, \\ \sum_{m=1}^{s_i} r_{im}\mu_{im}/e_{\boldsymbol{r}} & \text{if } \boldsymbol{v} = \boldsymbol{r} \ominus \boldsymbol{e}_{i,s_i+1} \text{ and } r_{i,s_i+1} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

It is straightforward to simulate holding times and transitions according to (5) and (6). Suppose the current time is τ_l and current state of $\boldsymbol{R}(t)$ is $\boldsymbol{r}$. We simulate random numbers $\mathcal{U}^{(l)} := \{\xi_0^{(l)} \sim \exp(\lambda)$ and $\xi_{ij}^{(l)} \sim \exp(\mu_{ij}), j = 1, \cdots, s_i, i = 1, \cdots, K\}$. If $r_{i,s_i+1} < N$ then the holding time of $\boldsymbol{R}(t)$ can be simulated as $T_l = \min\{\xi_0^{(l)}, \xi_{ij}^{(l)}/r_{ij}, j = 1, \cdots, s_i, i = 1, \cdots, K\}$, since $T_l \sim \exp(e_{\boldsymbol{r}})$ (see Ross (2007) for more details about the properties of exponential distributions). CFTP algorithm in Propp and Wilson (1996) needs to monitor coalescence of all different Markov chains. With the above simulation method, different chains have different holding times T_l, which makes it extremely difficult to monitor coalescence. Thus in order to develop a CFTP algorithm we consider a thinning algorithm in the following section instead.

3.2 A Thinning Algorithm

We consider possible holding times, defined as $T_l^{\min} = \min\{\xi_0^{(l)}, \xi_{ij}^{(l)}, j = 1, \cdots, s_i, i = 1, \cdots, K\}$, which is the same for all different Markov chains. This means that we will consider transitions at the same time $\tau_{l+1} = T_l^{\min} + \tau_l$ for all different chains. This makes it possible to monitor coalescence. The time points $\boldsymbol{\tau} = \{\cdots, \tau_l, \tau_{l+1}, \cdots\}$ are possible transition times of $\boldsymbol{R}(t)$. The thinning algorithm can select transition times of $\boldsymbol{R}(t)$ from $\boldsymbol{\tau}$ according to probability $e_{\boldsymbol{r}}/(\lambda + \sum_i \sum_{j=1}^{s_i} \mu_{ij})$. Proofs of such thinning algorithms can be found in Ross (2007).

The algorithm includes two parts. One is the case when new jobs can enter the network (there are waiting places available), the other is the case when new jobs cannot enter the network (no waiting places available).

Case A. First consider that $\boldsymbol{r}$ is such that $r_{i,s_i+1} < N$, for all i. Since $P(T_l^{\min} = T_l) = e_{\boldsymbol{r}}/(\lambda + \sum_i \sum_{j=1}^{s_i} \mu_{ij})$, if $T_l^{\min} = T_l$ then $T_l^{\min}$ will be the holding time and we simulate transitions at $\tau_{l+1} = T_l^{\min} + \tau_l$; if $T_l^{\min} < T_l$ then the holding time is greater than $T_l^{\min}$ and there is no transition at τ_{l+1}.

If $T_l^{\min} = T_l$ then $T_l^{\min}$ must be equal to $\xi_0^{(l)}$ or $\xi_{ij}^{(l)}$ for some i, j such that $r_{ij} = 1$. We simulate the transitions of $\boldsymbol{R}(t)$ as follows. If $T_l^{\min} = T_l = \xi_0^{(l)}$ (a new job arrives) then let $\boldsymbol{R}(\tau_{l+1}) = \boldsymbol{R}(\tau_l) \oplus \mathbf{1}$ since $P(T_l^{\min} = T_l = \xi_0^{(l)}) = \lambda/e_{\boldsymbol{r}}$. If $T_l^{\min} = T_l = \xi_{ij}^{(l)}$ for some i, j, then (a) let $\boldsymbol{R}(\tau_{l+1}) = \boldsymbol{R}(\tau_l) \ominus \boldsymbol{e}_{ij}$ provided that $r_{i,s_i+1} = 0$ since $P(T_l^{\min} = T_l = \xi_{ij}^{(l)}) = r_{ij}\mu_{ij}/e_{\boldsymbol{r}}$; or (b) let $\boldsymbol{R}(\tau_{l+1}) = \boldsymbol{R}(\tau_{l+1}) \ominus \boldsymbol{e}_{i,s_i+1}$ provided that $r_{i,s_i+1} > 0$ since $P(\cup_{m=1}^{s_i}\{T_l^{\min} = T_l = \xi_{im}^{(l)}\}) = \sum_m r_{im}\mu_{im}/e_{\boldsymbol{r}}$. So the transitions are correctly simulated according to (6).

Case B. Now consider that the current state $\boldsymbol{r}$ is such that some $r_{i,s_i+1} = N$. This means that new jobs cannot enter the network since no waiting place is available. Since $P(T_l^{\min} = T_l \neq \xi_0^{(l)}) = \sum_i \sum_{j=1}^{s_i} r_{ij}\mu_{ij}/(\lambda + \sum_i \sum_{j=1}^{s_i} \mu_{ij})$, if $T_l^{\min} = T_l \neq \xi_0^{(l)}$ then $T_l^{\min}$ will be the holding time. We then simulate transitions at $\tau_{l+1} = T_l^{\min} + \tau_l$, which can be done similarly as Case A. If $T_l^{\min} < T_l$ then the holding time is greater than $T_l^{\min}$ and there is no transition at $\tau_{l+1} = T_l^{\min} + \tau_l$; if $T_l^{\min} = T_l = \xi_0^{(l)}$ then there is no transition at τ_{l+1} since the new job cannot enter the network due to no waiting place available.

The above method tells us how to simulate $\boldsymbol{R}(t)$ between τ_l and τ_{l+1}. Due to the memoryless property of exponential distribution we can repeat the whole procedure to simulate $\boldsymbol{R}(t)$ after τ_{l+1}. Above all, the thinning method correctly simulates $\boldsymbol{R}(t)$ according to the transition rate function (3).

3.3 Updating Functions and the Partial Order

The updating rules for $\boldsymbol{R}(t)$ based on the thinning method in previous section can be denoted as a deterministic function ϕ of the random numbers $\mathcal{U}^{(l)} = \{\xi_0^{(l)}, \xi_{ij}^{(l)}, j = 1, \cdots, s_i, i = 1, \cdots, K\}$,

$$\boldsymbol{R}(\tau_l + a) = \phi(\boldsymbol{R}(\tau_l), \mathcal{U}^{(l)}, a) \tag{7}$$

$$= \begin{cases} \boldsymbol{R}(\tau_l), & \text{if } 0 < a < T_l^{\min}, \\ \boldsymbol{R}(\tau_l) \oplus \mathbf{1}, & \text{if } a = T_l^{\min},\ T_l^{\min} = T_l = \xi_0^{(l)} \text{ and } r_{i,s_i+1} < N \text{ for all } i, \\ \boldsymbol{R}(\tau_l), & \text{if } a = T_l^{\min},\ \ T_l^{\min} = T_l = \xi_0^{(l)} \text{ and } r_{i,s_i+1} = N \text{ for some } i, \\ \boldsymbol{R}(\tau_l) \ominus \boldsymbol{e}_{ij}, & \text{if } a = T_l^{\min}, T_l^{\min} = T_l = \xi_{ij}^{(l)} \text{ and } r_{i,s_i+1} = 0 \\ \boldsymbol{R}(\tau_l) \ominus \boldsymbol{e}_{i,s_i+1}, & \text{if } a = T_l^{\min}, T_l^{\min} = T_l = \xi_{ij}^{(l)} \text{ and } r_{i,s_i+1} > 0 \\ \boldsymbol{R}(\tau_l), & \text{if } a = T_l^{\min} \text{ and } T_l^{\min} < T_l, \end{cases}$$

The following theorem tells us that in some cases the partial order between a lower chain $\boldsymbol{R}(t)$ and an upper chain $\boldsymbol{R}'(t)$ is preserved by the updating function ϕ.

Suppose that the current time is τ_l and that $\boldsymbol{R}(\tau_l) = \boldsymbol{r} \preceq \boldsymbol{R}'(\tau_l) = \boldsymbol{r}'$.

Condition 3.1 *T*he upper chain state $\boldsymbol{r}'$ is such that $r'_{i,s_i+1} < N$ for all i.

Theorem 3.1 *I*f Condition 3.1 is true (there are waiting places free for a new job in the upper chain $\boldsymbol{R}'(t)$) then for any chain $\boldsymbol{v}(t)$ such that $\boldsymbol{R}(\tau_l) = \boldsymbol{r} \preceq \boldsymbol{v}(\tau_l) \preceq \boldsymbol{R}'(\tau_l) = \boldsymbol{r}'$ we have that given the exponential random numbers $\mathcal{U}^{(l)}$,

$$\begin{aligned} \boldsymbol{R}(\tau_l + a) = \phi(\boldsymbol{R}(\tau_l), \mathcal{U}^{(l)}, a) &\preceq \boldsymbol{v}(\tau_l + a) = \phi(\boldsymbol{v}(\tau_l), \mathcal{U}^{(l)}, a) \\ &\preceq \boldsymbol{R}'(\tau_l + a) = \phi(\boldsymbol{R}'(\tau_l), \mathcal{U}^{(l)}, a), \end{aligned} \tag{8}$$

for $0 < a \leq T_l^{\min}$.

If Condition 3.1 is not true, i.e. when a new job arrives the upper chain cannot increase due to no waiting place available, the partial order of chains $\boldsymbol{v}(t)$ between $\boldsymbol{R}(t)$ and $\boldsymbol{R}'(t)$ may not be preserved since the lower chain may still increase. Although the partial order is not always preserved, we can still find two bounding chains, an upper-bound chain $\bar{\boldsymbol{R}}(t)$ and a lower-bound chain $\underline{\boldsymbol{R}}(t)$, which bound all Markov chains. When the two bounding chains coalesce, all Markov chains coalesce. This is shown in the following section.

3.4 CFTP with Bounding Chains

Define an upper-bound chain $\bar{\boldsymbol{R}}(t)$ and a lower-bound chain $\underline{\boldsymbol{R}}(t)$, which are updated simultaneously as follows. The upper chain updating function is

$$\bar{\boldsymbol{R}}(\tau_l + a) = \bar{\phi}(\bar{\boldsymbol{R}}(\tau_l), \mathcal{U}^{(l)}, a) \tag{9}$$

$$= \begin{cases} \bar{\boldsymbol{R}}(\tau_l), & \text{if } 0 < a < T_l^{\min}, \\ \bar{\boldsymbol{R}}(\tau_l) \oplus \mathbf{1}, & \text{if } a = T_l^{\min},\ T_l^{\min} = \xi_0^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) < N \text{ for all } i, \\ \bar{\boldsymbol{R}}(\tau_l) \bar{\oplus} \mathbf{1}, & \text{if } a = T_l^{\min},\ T_l^{\min} = \xi_0^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) = N \text{ for some } i, \\ \bar{\boldsymbol{R}}(\tau_l) \ominus \boldsymbol{e}_{ij}, & \text{if } a = T_l^{\min}, T_l^{\min} = \xi_{ij}^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) = 0, \\ \bar{\boldsymbol{R}}(\tau_l) \ominus \boldsymbol{e}_{i,s_i+1}, & \text{if } a = T_l^{\min}, T_l^{\min} = \xi_{ij}^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) > 0, \end{cases}$$

where $\bar{\boldsymbol{R}}(\tau_l) \bar{\oplus} \mathbf{1}$ is defined as $\bar{\boldsymbol{R}}_i(\tau_l) \bar{\oplus} 1$ for each i and

$$\bar{\boldsymbol{R}}_i(\tau_l) \bar{\oplus} 1$$

$$\Leftrightarrow \begin{cases} \bar{R}_{ij}(\tau_l) + 1 & \text{if } j \text{ is the smallest server label such that } \bar{R}_{ij}(\tau_l) = 0; \\ \min\{N, r_{ij} + 1\} & j = s_i + 1;\ \text{if } \bar{R}_{im}(\tau_l) > 0 \text{ for all } m = 1, \cdots, s_i; \\ \bar{R}_{ij}(\tau_l) & \text{for other label } j. \end{cases} \tag{10}$$

The lower-bound chain updating function is

$$\underline{\boldsymbol{R}}(\tau_l + a) = \underline{\phi}(\underline{\boldsymbol{R}}(\tau_l), \bar{\boldsymbol{R}}(\tau_l), \mathcal{U}^{(l)}, a) \tag{11}$$

$$= \begin{cases} \underline{\boldsymbol{R}}(\tau_l), & \text{if } 0 < a < T_l^{\min}, \\ \underline{\boldsymbol{R}}(\tau_l) \oplus \mathbf{1}, & \text{if } a = T_l^{\min},\ T_l^{\min} = \xi_0^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) < N \text{ for all } i, \\ \underline{\boldsymbol{R}}(\tau_l), & \text{if } a = T_l^{\min},\ T_l^{\min} = \xi_0^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) = N \text{ for some } i, \\ \underline{\boldsymbol{R}}(\tau_l) \ominus \boldsymbol{e}_{ij}, & \text{if } a = T_l^{\min}, T_l^{\min} = \xi_{ij}^{(l)} \text{ and } \underline{R}_{i,s_i+1}(\tau_l) = 0, \\ \underline{\boldsymbol{R}}(\tau_l) \ominus \boldsymbol{e}_{i,s_i+1}, & \text{if } a = T_l^{\min}, T_l^{\min} = \xi_{ij}^{(l)} \text{ and } \underline{R}_{i,s_i+1}(\tau_l) > 0. \end{cases}$$

The following theorem holds for such upper-bound and lower-bound chains.

Theorem 3.2 *S*uppose that the upper-bound chain and lower-bound chain are define with update functions (9) and (11) respectively as above. If $\underline{\boldsymbol{R}}(t) \preceq \boldsymbol{R}(t) \preceq \bar{\boldsymbol{R}}(t)$ then $\underline{\boldsymbol{R}}(t + a) \preceq \boldsymbol{R}(t + a) \preceq \bar{\boldsymbol{R}}(t + a)$ for $a > 0$.

Define the maximum point as $\bar{\boldsymbol{r}}$ which is such that $\bar{r}_{ij} = 1, j = 1, \cdots, s_i$ and $\bar{r}_{i,s_i+1} = N$. Define the minimum point as $\underline{\boldsymbol{r}}$ with all elements equal to 0. Suppose that the upper-bound chain $\bar{\boldsymbol{R}}(t)$ starting from $\bar{\boldsymbol{R}}(-M) = \bar{\boldsymbol{r}}$ and the lower-bound chain $\underline{\boldsymbol{R}}(-M) = \underline{\boldsymbol{r}}$. Here M is a large fixed number. Then any Markov chain $\boldsymbol{R}(-M)$ starting from an element $\boldsymbol{r} \in \mathcal{R}$ is bounded by $\bar{\boldsymbol{R}}(-M)$ and $\underline{\boldsymbol{R}}(-M)$. According to Theorem 3.2 all different chains $\{\boldsymbol{R}(t), t \in [-M, 0]\}$ will be bounded by the two bounding chains. Thus we can run the upper-bound chain and lower-bound chain simultaneously and when the two chains coalesce, all Markov chains coalesce. After coalescence, say coalescence time $-\Delta$, we simply run $\boldsymbol{R}(t)$ from time $-\Delta$ to 0 with $\boldsymbol{R}(-\Delta) = \bar{\boldsymbol{R}}(-\Delta)$ and $\boldsymbol{R}(0)$ is from equilibrium distribution.

4. Perfect Sampling for Heterogeneous Networks with the Random Strategy

For random allocation strategy the transition rate function for $\boldsymbol{R}(t)$ becomes

$$f(\boldsymbol{r}, \boldsymbol{v}) = \begin{cases} \lambda & \text{if } \boldsymbol{v} = \boldsymbol{r} \uplus \mathbf{1} \text{ and } r_{i,s_i+1} < N, \text{ for all } i, \\ r_{ij}\mu_{ij} & \text{if } \boldsymbol{v} = \boldsymbol{r} \ominus \boldsymbol{e}_{ij}, j = 1, \cdots, s_i \text{ and } r_{i,s_i+1} = 0 \\ \sum_{m=1}^{s_i} r_{im}\mu_{im} & \text{if } \boldsymbol{v} = \boldsymbol{r} \ominus \boldsymbol{e}_{i,s_i+1}, \text{ and } r_{i,s_i+1} > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where $\uplus$ is defined in (2).

We can still use formula (5) to simulate holding times, since different allocation strategies only change the transition probabilities but not changing holding time distributions. At the end of the holding time, $\boldsymbol{R}(t)$ jumps from $\boldsymbol{r}$ to $\boldsymbol{v}$

with transition probability

$$p_{\boldsymbol{r},\boldsymbol{v}} = \begin{cases} \lambda/e_{\boldsymbol{r}} & \text{if } \boldsymbol{v} = \boldsymbol{r} \uplus \mathbf{1} \text{ and } r_{i,s_i+1} < N, \text{ for all } i, \\ r_{ij}\mu_{ij}/e_{\boldsymbol{r}} & \text{if } \boldsymbol{v} = \boldsymbol{r} \ominus \boldsymbol{e}_{ij},\ j = 1, \cdots, s_i \text{ and } r_{i,s_i+1} = 0 \\ \sum_{m=1}^{s_i} r_{im}\mu_{im}/e_{\boldsymbol{r}} & \text{if } \boldsymbol{v} = \boldsymbol{r} \ominus \boldsymbol{e}_{i,s_i+1}, \text{ and } r_{i,s_i+1} > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

which is similar to (6). The only difference is that in the above equation we use $\uplus$ instead of $\oplus$ when a new job arrives. Thus we can use a similar thinning algorithm as that in Section 3.2 to simulate the Markov chains, except that when a new job arrives random allocation (operator $\uplus$) need to be considered.

At time τ_l apart from the simulated random number $\mathcal{U}^{(l)}$, we also need to simulate the random allocation '$\uplus$' of the new job. We simulate a sequence of random integers for each station i, denoted as $\boldsymbol{\eta}_i^{(l)} = \{Y_{i1}^{(l)}, Y_{i2}^{(l)}, Y_{i3}^{(l)}, \cdots\}$, where each $Y_{ig}^{(l)}$ is taken uniformly from the server labels $\{1, \cdots, s_i\}$ of station i. Suppose the current state is $\boldsymbol{R}(\tau_l) = \boldsymbol{r}$. If $\boldsymbol{r}$ is such that at least one $r_{ij} = 0, j = 1, \cdots, s_i$ then in the sequence $\boldsymbol{\eta}_i^{(l)}$, $Y_{ig^*}^{(l)}$ is the label of the randomly chosen server for the new subtask, where $g^* = \min\{g : \text{ such that } r_{i,Y_{ig}^{(l)}} = 0\}$. Denote this randomly chosen label as a function, $g^* = J(\boldsymbol{\eta}_i^{(l)}, \boldsymbol{r}_i)$. Such a method is a rejection sampling method, which draws a server randomly from $\{1, \cdots, s_i\}$ and accepts it if this server is free. Then following (2), $\boldsymbol{r}_i' = \boldsymbol{r}_i \uplus 1$ can be further written as

$$\begin{aligned} & \boldsymbol{r}_i' = \boldsymbol{r}_i \uplus 1 \\ \Leftrightarrow \quad & r_{ij}' = \begin{cases} r_{ij} + 1 & \text{if at least one } r_{im} = 0, m \in \{1, \cdots, s_i\} \text{ and } j = J(\boldsymbol{\eta}_i^{(l)}, \boldsymbol{r}_i); \\ r_{ij} + 1 & \text{if } j = s_i + 1 \text{ and } r_{im} > 0 \text{ for all } m \in \{1, \cdots, s_i\}; \\ r_{ij} & \text{for other label } j. \end{cases} \end{aligned} \tag{14}$$

Such a rejection sampling method guarantees that the partial order is preserved by $\uplus$ under some condition. This is given by the following lemma.

Lemma 4.1 *S*uppose that $\boldsymbol{r} \preceq \boldsymbol{r}'$ and Condition 3.1 holds for $\boldsymbol{r}'$. Then $\boldsymbol{r} \uplus \mathbf{1} \preceq \boldsymbol{r}' \uplus \mathbf{1}$.

With the definition of random allocation $\uplus$, the updating function for $\boldsymbol{R}(t)$ can be written as

$$\begin{aligned} & \boldsymbol{R}(\tau_l + a) = \psi(\boldsymbol{R}(\tau_l), \mathcal{U}^{(l)}, \boldsymbol{\eta}^{(l)}, a) \\ = \quad & \begin{cases} \boldsymbol{R}(\tau_l), & \text{if } 0 < a < T_l^{\min}, \\ \boldsymbol{R}(\tau_l) \uplus \mathbf{1}, & \text{if } a = T_l^{\min},\ T_l^{\min} = T_l = \xi_0^{(l)} \text{ and } r_{i,s_i+1} < N \text{ for all } i, \\ \boldsymbol{R}(\tau_l), & \text{if } a = T_l^{\min},\ \ T_l^{\min} = T_l = \xi_0^{(l)} \text{ and } r_{i,s_i+1} = N \text{ for some } i, \\ \boldsymbol{R}(\tau_l) \ominus \boldsymbol{e}_{ij}, & \text{if } a = T_l^{\min}, T_l^{\min} = T_l = \xi_{ij}^{(l)} \text{ and } r_{i,s_i+1} = 0 \\ \boldsymbol{R}(\tau_l) \ominus \boldsymbol{e}_{i,s_i+1}, & \text{if } a = T_l^{\min}, T_l^{\min} = T_l = \xi_{ij}^{(l)} \text{ and } r_{i,s_i+1} > 0 \\ \boldsymbol{R}(\tau_l), & \text{if } a = T_l^{\min} \text{ and } T_l^{\min} < T_l. \end{cases} \end{aligned} \tag{15}$$

Define the upper-bound chain and lower-bound chain updating functions as follows. The upper chain updating function is

$$\begin{aligned} & \bar{\boldsymbol{R}}(\tau_l + a) = \bar{\psi}(\bar{\boldsymbol{R}}(\tau_l), \mathcal{U}^{(l)}, \boldsymbol{\eta}^{(l)}, a) \\ = \quad & \begin{cases} \bar{\boldsymbol{R}}(\tau_l), & \text{if } 0 < a < T_l^{\min}, \\ \bar{\boldsymbol{R}}(\tau_l) \uplus \mathbf{1}, & \text{if } a = T_l^{\min},\ T_l^{\min} = \xi_0^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) < N \text{ for all } i, \\ \bar{\boldsymbol{R}}(\tau_l) \bar{\uplus} \mathbf{1}, & \text{if } a = T_l^{\min},\ \ T_l^{\min} = \xi_0^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) = N \text{ for some } i, \\ \bar{\boldsymbol{R}}(\tau_l) \ominus \boldsymbol{e}_{ij}, & \text{if } a = T_l^{\min}, T_l^{\min} = \xi_{ij}^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) = 0, \\ \bar{\boldsymbol{R}}(\tau_l) \ominus \boldsymbol{e}_{i,s_i+1}, & \text{if } a = T_l^{\min}, T_l^{\min} = \xi_{ij}^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) > 0, \end{cases} \end{aligned} \tag{16}$$

where

$$\begin{aligned} & \bar{\boldsymbol{R}}_i(\tau_l) \bar{\uplus} 1 \\ \Leftrightarrow \quad & \begin{cases} \bar{R}_{ij}(\tau_l) + 1 & j = J(\boldsymbol{\eta}_i^{(l)}, \bar{\boldsymbol{R}}_i(\tau_l)); \text{ if at least one } r_{im} = 0 \text{ for } m \in \{1, \cdots, s_i\}; \\ \min\{N, r_{ij} + 1\} & j = s_i + 1; \text{ if } r_{im} > 0 \text{ for all } m \in \{1, \cdots, s_i\}; \\ r_{ij} & \text{for other label } j. \end{cases} \end{aligned} \tag{17}$$

The lower-bound chain updating function is

$$\underline{\boldsymbol{R}}(\tau_l + a) = \underline{\psi}(\underline{\boldsymbol{R}}(\tau_l), \bar{\boldsymbol{R}}(\tau_l), \mathcal{U}^{(l)}, \boldsymbol{\eta}^{(l)}, a) \tag{18}$$

$$= \begin{cases} \underline{\boldsymbol{R}}(\tau_l), & \text{if } 0 < a < T_l^{\min}, \\ \underline{\boldsymbol{R}}(\tau_l) \uplus \mathbf{1}, & \text{if } a = T_l^{\min},\ T_l^{\min} = \xi_0^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) < N \text{ for all } i, \\ \underline{\boldsymbol{R}}(\tau_l), & \text{if } a = T_l^{\min},\ \ T_l^{\min} = \xi_0^{(l)} \text{ and } \bar{R}_{i,s_i+1}(\tau_l) = N \text{ for some } i, \\ \underline{\boldsymbol{R}}(\tau_l) \ominus \boldsymbol{e}_{ij}, & \text{if } a = T_l^{\min}, T_l^{\min} = \xi_{ij}^{(l)} \text{ and } \underline{R}_{i,s_i+1}(\tau_l) = 0, \\ \underline{\boldsymbol{R}}(\tau_l) \ominus \boldsymbol{e}_{i,s_i+1}, & \text{if } a = T_l^{\min}, T_l^{\min} = \xi_{ij}^{(l)} \text{ and } \underline{R}_{i,s_i+1}(\tau_l) > 0. \end{cases}$$

Theorem 4.1 *S*uppose that the upper-bound chain and lower-bound chain are define with update functions (16) and (18) respectively as above. For $\boldsymbol{R}(t)$ updated according to (15), if $\underline{\boldsymbol{R}}(t) \preceq \boldsymbol{R}(t) \preceq \bar{\boldsymbol{R}}(t)$ then $\underline{\boldsymbol{R}}(t+a) \preceq \boldsymbol{R}(t+a) \preceq \bar{\boldsymbol{R}}(t+a)$ for $a > 0$.

With Theorem 4.1 we can also use CFTP with bounding chains to simulate $\boldsymbol{R}(t)$ from equilibrium with the random allocation strategy.

5. Algorithm Complexity and Simulation Studies

5.1 Algorithm Complexity

In the CFTP algorithms we only need to simulate transitions at discrete time points $\boldsymbol{\tau} = \{\cdots, \tau_l, \tau_{l+1}, \cdots\}$, although the upper-bound and lower-bound chains are continuous time chains. Thus to monitor coalescence we need to consider coalescence of the discrete time chains $\bar{\boldsymbol{R}}^{(l)} = \bar{\boldsymbol{R}}(\tau_l)$ and $\underline{\boldsymbol{R}}^{(l)} = \underline{\boldsymbol{R}}(\tau_l)$. Let ν_c be the number of steps needed for $\bar{\boldsymbol{R}}^{(l)}$ and $\underline{\boldsymbol{R}}^{(l)}$ to coalesce. Define the distance between $\bar{\boldsymbol{R}}^{(l)}$ and $\underline{\boldsymbol{R}}^{(l)}$ as $d(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}) := \sum_i \sum_j \left| \bar{R}_{ij}^{(l)} - \underline{R}_{ij}^{(l)} \right|$. Coalescent occurs when the distance becomes 0. We know that in the CFTP algorithms $\bar{\boldsymbol{R}}^{(l)}$ starts from the maximum and $\underline{\boldsymbol{R}}^{(l)}$ starts from the minimum, which have distance $\mathbb{N} = \sum_{i=1}^{K} N_i$. We will expect that the larger the value of $\mathbb{N}$ the longer the coalescent time. In this section we will show that the expected coalescence steps $E\nu_c$ is bounded by a polynomial function of $\mathbb{N}$, which means that the running time of the CFTP algorithm only increases polynomially as $\mathbb{N}$ increases. This implies the algorithm is very efficient.

According to

$$E\nu_c = \sum_{l=1}^{\infty} P(\nu_c \geq l) = \sum_{l=1}^{\infty} P\left(d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right) \geq 1\right) \leq \sum_{l=1}^{\infty} E\left[d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right)\right], \tag{19}$$

to calculate a bound for $E\nu_c$, we need to work out a bound for $E\left[d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right)\right]$.

Following from the updating rules for the upper-bound and lower-bound chains, from step $l-1$ to step l we have three possible cases: event $A_1^{(l)}$ for distance not changing, $d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right) = d\left(\bar{\boldsymbol{R}}^{(l-1)}, \underline{\boldsymbol{R}}^{(l-1)}\right)$; event $A_2^{(l)}$ for distance decreasing, $d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right) = d\left(\bar{\boldsymbol{R}}^{(l-1)}, \underline{\boldsymbol{R}}^{(l-1)}\right) - 1$; or event $A_3^{(l)}$ for distance increasing, $d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right) \leq d\left(\bar{\boldsymbol{R}}^{(l-1)}, \underline{\boldsymbol{R}}^{(l-1)}\right) + K$. Event $A_3^{(l)}$ corresponds to that when a new job arrives the upper-bound chain increases but the lower-bound chain does not move. Following the proof of Lemma 4 in Dai (2011) we have that the probability $P(A_3^{(l)})$ converges to 0 at an exponential rate as $l \to \infty$, provided that $\lambda < \sum_j \mu_{ij}$ for $i = 1, \cdots, K$. The rate of $P(A_3^{(l)})$ only depends on the network parameters λ and μ_{ij}. On the other hand event $A_2^{(l)}$ corresponds to that a subtask is completed in a server in the upper-bound chain but no subtask is completed in the lower-bound chain since the corresponding server is idle in the lower-bound chain. Thus the probability $P(A_2^{(l)}) \geq \Psi = \min\{\mu_{ij}\}/\Lambda$ where $\Lambda = \lambda + \sum_{ij} \mu_{ij}$. Event $A_1^{(l)}$ corresponds to all other possibilities.

With the above arguments, we have

$$\begin{aligned} & E\left[d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right)\right] = E\left\{E\left[d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right) \middle| \bar{\boldsymbol{R}}^{(l-1)}, \underline{\boldsymbol{R}}^{(l-1)}\right]\right\} \\ \leq\ & P(A_1^{(l)}) E d\left(\bar{\boldsymbol{R}}^{(l-1)}, \underline{\boldsymbol{R}}^{(l-1)}\right) + P(A_2^{(l)}) E\left[d\left(\bar{\boldsymbol{R}}^{(l-1)}, \underline{\boldsymbol{R}}^{(l-1)}\right) - 1\right] \\ & + P(A_3^{(l)}) E\left[d\left(\bar{\boldsymbol{R}}^{(l-1)}, \underline{\boldsymbol{R}}^{(l-1)}\right) + K\right] \\ =\ & E d\left(\bar{\boldsymbol{R}}^{(l-1)}, \underline{\boldsymbol{R}}^{(l-1)}\right) - P(A_2^{(l)}) + K P(A_3^{(l)}) \\ \leq\ & E d\left(\bar{\boldsymbol{R}}^{(l-1)}, \underline{\boldsymbol{R}}^{(l-1)}\right) \left[1 - \frac{P(A_2^{(l)}) - K P(A_3^{(l)})}{\mathbb{N}}\right] \end{aligned} \tag{20}$$

Since $P(A_3^{(l)})$ converges to 0 at an exponential rate, there exist an large constant H (only depends on λ and μ_{ij}) and when $l \geq H$ we have $P(A_2^{(l)}) - KP(A_3^{(l)}) \geq \Psi - KP(A_3^{(l)}) \geq \Psi/2$. Thus from (20) and $d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right) \leq \mathbb{N}$ we have, for $l \geq H$,

$$E\left[d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right)\right] \quad \leq \quad \mathbb{N}\left(1 - \frac{\Psi}{2\mathbb{N}}\right)^{l-H}. \tag{21}$$

Then (19) and (21) imply that

$$E\nu_c \leq \sum_{l=1}^{\infty} E\left[d\left(\bar{\boldsymbol{R}}^{(l)}, \underline{\boldsymbol{R}}^{(l)}\right)\right] \leq H\mathbb{N} + 2\mathbb{N}^2/\Psi = O(\mathbb{N}^2). \tag{22}$$

Therefore the expected coalescence time is bounded by a polynomial function of $\mathbb{N}$.

The running time comparisons for different values of K and N are summarized in Table 1. The simulations are performed on a desktop with a 2.13 GHz Intel Core Duo processor and 3.2G memory. We can see that the algorithm is very efficient.

Note that the above proof is correct for all the three allocation strategies. This is because the coalescence steps only depends on the probabilities that the distance between the two bounding chains changes (increasing, decreasing or not changing), which does not depend on the allocation strategies. Therefore the running time for exact Monte Carlo simulation algorithm of the three allocation strategies should be similar. Of course, the algorithm for the random allocation strategy should take slightly longer as it involves the extra rejection sampling steps. Table 1 demonstrates that the algorithm for the fastest strategy and slowest strategy have similar running times, but the algorithm for the random allocation strategy has longer running times.

Table 1: Running time comparisons for simulating 10,000 realisations from a fork-join network with $\lambda = 1$, $s_i = 3, i = 1, \cdots, K$ and service rates are 0.5, 1.0, 1.5 for the three servers in each station. I: the slowest strategy; II: the fastest strategy; III: the random strategy.

	I			II			III		
N	500	1000	1500	500	1000	1500	500	1000	1500
$K = 5$	278s	560s	1090s	281s	569s	1068s	318s	641s	1298s
$K = 10$	822ss	1617s	3241s	773s	1666s	2945s	906s	1751s	3489s
$K = 15$	1431s	2908s	6048s	1501s	2862s	6060s	1677s	3370s	6664s

5.2 A Toy Example

Before introducing comprehensive simulation studies, we provide a toy example to illustrate how to carry out the CFTP method. We consider the simplest case with one station (no fork-join but a simple queuing system), which has two servers with service rates μ_{11} and μ_{12} respectively. For simplicity, we assume that there is no waiting space. Therefore the system can take 2 jobs at most by the two servers. Therefore, the network has the following four possible states $(1, 1)$, $(1, 0)$ and $(0, 1)$ and $(0, 0)$, where 0 means the corresponding server is idle and 1 means it is busy. When using CFTP, we should run four Markov chains (starting from the four different states) simultaneously from a starting time say $-M$ in the past, using the same random numbers.

We generate $\xi_{11}^{(1)}$ as the service time for server I, $\xi_{12}^{(1)}$ as the service time for server II and $\xi_0^{(1)}$ as the next job arrival time. The first potential transition time will occur at $T_1^{\min} = \min\{\xi_{11}^{(1)}, \xi_{12}^{(1)}, \xi_0^{(1)}\}$. Suppose that $T_1^{\min} = \xi_{11}^{(1)}$, then the job in server I (if server I is busy) will be finished at time $-M + \xi_{11}^{(1)}$. Therefore, if the system starts from $R(-M) = (1, 1)$ (or $R(-M) = (1, 0)$), it will move to $R(-M + \xi_{11}^{(1)}) = (0, 1)$ (or $R(-M + \xi_{11}^{(1)}) = (0, 0)$). On the other hand, if the system starts from $R(-M) = (0, 1)$ (or $R(-M) = (0, 0)$) it will not change at time $-M + \xi_{11}^{(1)}$, since server I is idle at time $-M$. Therefore, at time $-M + \xi_{11}^{(1)}$ the four chains coalesce into two chains $(0, 1)$ and $(0, 0)$.

Then we generate $\xi_{11}^{(2)}$, $\xi_{12}^{(2)}$ and $\xi_0^{(2)}$ for the second potential transition time, which will occur at the $T_2^{\min} = \min\{\xi_{11}^{(2)}, \xi_{12}^{(2)}, \xi_0^{(2)}\}$. Suppose that $T_2^{\min} = \xi_{12}^{(2)}$, then the job in server II (if server II is busy) will be finished at time $-M + \xi_{11}^{(1)} + \xi_{12}^{(2)}$. Therefore, the system moves from $R(-M + \xi_{11}^{(1)}) = (0, 1)$ to $R(-M + \xi_{11}^{(1)} + \xi_{12}^{(2)}) = (0, 0)$, or the system stays at $R(-M + \xi_{11}^{(1)} + \xi_{12}^{(2)}) = (0, 0)$ if $R(-M + \xi_{11}^{(1)}) = (0, 0)$. This means that the chains will coalesce

into a single chain $(0,0)$ at time $-M+\xi_{11}^{(1)}+\xi_{12}^{(2)}$. If the coalescence time is less than 0, we can keep running the chain and collect a sample at time 0.

If using the CFTP algorithm with bounding chains, we only need to run the upper chain $\bar{R}(-M)=(1,1)$ and $\underline{R}(-M)=(0,0)$. At time $-M+\xi_{11}^{(1)}$ the upper chain becomes $\bar{R}(-M+\xi_{11}^{(1)})=(0,1)$ and the lower chain does not change $\underline{R}(-M+\xi_{11}^{(1)})=(0,0)$. Then at the next transition $\bar{R}(-M+\xi_{11}^{(1)}+\xi_{12}^{(2)})=(0,0)$ and the lower chain is still $\underline{R}(-M+\xi_{11}^{(1)}+\xi_{12}^{(2)})=(0,0)$. The upper and lower chains coalesce. For the more general fork-join network presented in this paper, we can follow the formulas presented in previous sections to update $\bar{\boldsymbol{R}}(t)$ and $\underline{\boldsymbol{R}}(t)$.

5.3 Simulation Studies

In this section, we carried out simulation studies for distributions of maximum queue lengths, work loads for each server and response times.

Let $\mathbb{R}_i(t)$ be the number of uncompleted subtasks (including jobs at service or waiting) in station i. Then $\mathbb{R}(t)=\max_i \mathbb{R}_i(t)$ represents the number of uncompleted jobs in the network. We are interested in the distribution of $\mathbb{R}(t)$ in equilibrium. Consider a network with $K=3$ stations and each station has 4 ($s_1=s_2=s_3=4$) servers. The servers in Station One have service rates 0.2, 0.4, 0.6 and 0.8 respectively. The servers in Station Two have service rates 0.3, 0.6, 0.9 and 1.2 respectively. Station Three is the same as Station Two. The Monte Carlo simulation results for the maximum queue length distribution are summarized in Table 2.

Table 2: The entries in the table are the probabilities $P(\mathbb{R}(t)=k), k=0,1,2,\cdots$. I: the slowest strategy; II: the fastest strategy; III: the random strategy. The network has $K=3$ stations and each has 4 servers. The results are based on 10,000 simulations.

	I		II		III	
$\mathbb{R}(t)=$	N=100	N=300	N=100	N=300	N=100	N=300
0	0.01182	0.01093	0.07934	0.08072	0.02552	0.02568
1	0.12919	0.13052	0.25684	0.25693	0.18196	0.18140
2	0.30050	0.29821	0.28438	0.28328	0.30468	0.30546
3	0.27541	0.27932	0.19338	0.19167	0.24462	0.24650
4	0.14187	0.14396	0.09407	0.09594	0.12499	0.12449
5	0.07165	0.07005	0.04651	0.04695	0.06134	0.06029
6	0.03424	0.03312	0.02295	0.02274	0.02915	0.02918
7	0.01725	0.01712	0.01206	0.01111	0.01475	0.01433
8	0.00909	0.00839	0.00538	0.00542	0.00667	0.00642
9	0.00428	0.00433	0.00247	0.00245	0.00298	0.00320
10	0.00223	0.00199	0.00127	0.00139	0.00170	0.00156

From the results we can see that $N=100$ and $N=300$ provide similar results. This is because the maximum queue length is almost no more than 10 and waiting capacity 100 is large enough to hold all waiting jobs in equilibrium. The results also show that on average the network in equilibrium under the slowest strategy will have the most jobs, the network in equilibrium under the fastest strategy will have the least jobs.

For the above scenario we also consider the work load analysis for each server. The results are summarized in Table 3, where the entries are the probabilities of the corresponding server being active. Again the workloads for each server are similar under $N=100$ and $N=300$, but the workloads are very different under different allocation strategies. Servers with smaller service rate have larger workload under the slowest strategy than under the fastest strategy, which is as we expected. Note that in Table 3 we only provided workloads for servers in Station One and Station Two since Station Three is the same as Station Two and they have similar results.

Using the simulated realisations of queue lengths from equilibrium, we can easily simulate response times. For the above scenario with $N=100$, based on 10,000 simulations from equilibrium, Figure 2 provides the fitted density curves for response times under different allocation strategies. The mean response times under the three strategies are 2.18, 3.53 and 4.14. We can see that the distribution of response times are highly skewed and the distribution under the slowest strategy has a much longer tail than the others. By simply looking at the mean response times we do not have any information about the properties of the distribution tails, which represent how extreme the response time could be. The 90th-percentiles for the response time distribution under different allocation strategies are estimated as 4.12, 7.08 and 8.76. This suggests that under the slowest strategy the 10% longest jobs would take

Table 3: The entries are the probabilities of the corresponding server being active. I: the slowest strategy; II: the fastest strategy; III: the random strategy. The results are based on 10,000 simulations.

		Station	One			Station	Two	
Server rates	0.2	0.4	0.6	0.8	0.3	0.6	0.9	1.2
I, N=100	0.8529	0.6926	0.5009	0.3148	0.7771	0.5456	0.3072	0.1376
I, N=300	0.8518	0.6919	0.5000	0.3186	0.7791	0.5449	0.3064	0.1325
II, N=100	0.3913	0.4019	0.4821	0.5903	0.1540	0.2019	0.3091	0.4599
II, N=300	0.3896	0.4009	0.4827	0.5891	0.1565	0.2034	0.3103	0.4613
III, N=100	0.6901	0.5598	0.4804	0.4373	0.5372	0.3901	0.3086	0.2724
III, N=300	0.6890	0.5576	0.4789	0.4375	0.5350	0.3896	0.3124	0.2726

time longer than 8.76, but under the fastest strategy 10% longest jobs just take time longer than 4.12.

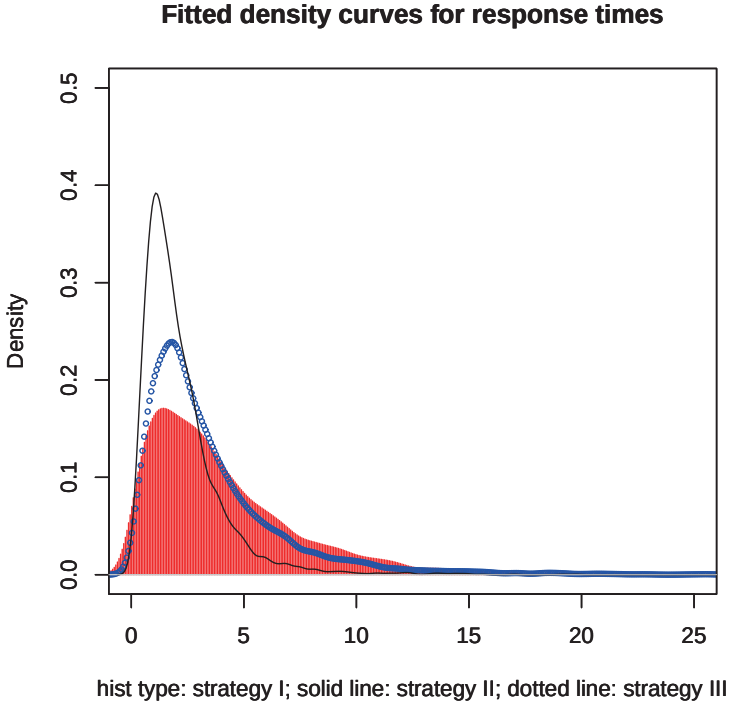

Figure 2. Density fitting for response times.

6. Discussion

We presented a perfect sampling method based on CFTP to draw exact realisations from the equilibrium of a fork-join network with heterogeneous service. The proposed method is very important as there is no analytical method available for such problems. Similarly as that in Dai (2011) the methods only work for fork-join network with finite capacity $N \leq \infty$. When $N = \infty$ there is no maximum point in the state space, therefore such CFTP with bounding chains is not readily available. Although $N < \infty$ is reasonable in practice, theoretically it is worth looking for methods for $N = \infty$.

A feasible method to solve the problem with $N = \infty$ is to use the dominated CFTP in Kendall and Moller (2000). The idea requires to finding a dominating process which is reversible, bounds the target chains and can be simulated easily from its equilibrium in reverse time. In terms of the process $\boldsymbol{R}(t)$ in this paper, $R_{i,s_i+1}(t), i = 1, \cdots, K$ are not bounded. One possible way to apply dominated CFTP is to find a univariate dominating process $X(t)$ to bound $\max_i R_{i,s_i+1}(t)$. Note that we can easily find a univariate process $X_i(t)$ (for example the birth-death process with birth rate λ and death rate $\min_j \mu_{ij}$) to bound $R_{i,s_i+1}(t)$. However if $X_i(t), i = 1, \cdots, K$ are independent then $X(t) = \max_i X_i(t)$ may not bound $\max_i R_{i,s_i+1}(t)$. This is because when $X_i(t)$ increases by 1 (a new job arrives)

$X(t)$ may not increase. On the other hand, all $R_{i,s_i+1}(t)$ will increase by 1 and thus $\max_i R_{i,s_i+1}(t)$ increases by 1. Therefore the dominated CFTP algorithm is not readily available. We leave it as a future research work to solve the problem with $N = \infty$.

In realistic situations, the service rates or arrival rates may be unknown and need to be estimated. This will be a much more challenging problem, since for such fork-join networks the likelihood function is not available explicitly which is required by traditional likelihood or Bayesian methods. A possible solution will be using approximate Bayesian computation (ABC) to estimate the service or arrival rates. The fundamental idea of ABC is to simulate data based on given parameter values. If the simulated data are *close* to the real data, the given parameters will be treated as a good estimate. Therefore the proposed Monte Carlo simulation method would find its application in the area of ABC.

A.1. Proof of Theorem 3.1

Proof. Define T_l and T'_l as,

$$\begin{aligned} T_l &= \min\{\xi_0^{(l)}, \xi_{ij}^{(l)}/r_{im}, m = 1, \cdots, s_i, i = 1, \cdots, K\}, \\ T'_l &= \min\{\xi_0^{(l)}, \xi_{ij}^{(l)}/r'_{im}, m = 1, \cdots, s_i, i = 1, \cdots, K\}. \end{aligned} \tag{23}$$

We only need to consider $\boldsymbol{r} \prec \boldsymbol{r}'$ which tells us $T_l > T'_l \geq T_l^{\min}$.

If $a < T_l^{\min}$, we have that $\boldsymbol{R}(\tau_l + a) = \boldsymbol{R}(\tau_l) \prec \boldsymbol{R}'(\tau_l + a) = \boldsymbol{R}'(\tau_l)$.

If $a = T_l^{\min}$ and $T_l^{\min} = \xi_0^{(l)}$, then $T_l^{\min} = T_l = T'_l = \xi_0^{(l)}$. To show $\boldsymbol{R}(\tau_l + a) = \boldsymbol{r} \oplus \mathbf{1} \preceq \boldsymbol{R}'(\tau_l + a) = \boldsymbol{r}' \oplus \mathbf{1}$, we only need to show that $\boldsymbol{R}_i(\tau_l + a) = \boldsymbol{r}_i \oplus 1 \preceq \boldsymbol{R}'_i(\tau_l + a) = \boldsymbol{r}'_i \oplus 1$ for each i. Let j be the smallest server label in station i such that $r_{ij} = 0$, or $j = s_i + 1$ if $r_{ij} = 1$ for all $j \in \{1, \cdots, s_i\}$. Let h be the smallest server label in station i such that $r'_{ih} = 0$ or $h = s_i + 1$ if $r'_{ih} = 1$ for all $h \in \{1, \cdots, s_i\}$. Then the operation $\oplus$ means that r_{ij} is added by 1, r'_{ih} is added by 1 and all other elements in row i will not change. If $j = h$ then the partial order $\preceq$ holds. On the other hand if $j \neq h$ we must have that $r_{ij} = 0$, $r'_{ij} = 1$, $r_{ih} = 0$ and $r'_{ih} = 0$ due to $\boldsymbol{r} \prec \boldsymbol{r}'$. Then the partial order $\preceq$ also holds.

If $a = T_l^{\min}$ and $T_l^{\min} = \xi_{ij}^{(l)} = T'_l = T_l$, there are three possible cases. (1) If $r'_{i,s_i+1} = 0$ then $r_{i,s_i+1} = 0$. Thus the partial order $\boldsymbol{R}(\tau_l + a) = \boldsymbol{r} \ominus \boldsymbol{e}_{ij} \preceq \boldsymbol{R}'(\tau_l + a) = \boldsymbol{r}' \ominus \boldsymbol{e}_{ij}$ holds. (2) If $r_{i,s_i+1} > 0$ then $r'_{i,s_i+1} > 0$. Thus the partial order $\boldsymbol{R}(\tau_l + a) = \boldsymbol{r} \ominus \boldsymbol{e}_{i,s_i+1} \preceq \boldsymbol{R}'(\tau_l + a) = \boldsymbol{r}' \ominus \boldsymbol{e}_{i,s_i+1}$ holds. (3) If $r_{i,s_i+1} = 0$ and $r'_{i,s_i+1} > 0$, we also have $\boldsymbol{R}(\tau_l + a) = \boldsymbol{r} \ominus \boldsymbol{e}_{ij} \preceq \boldsymbol{R}'(\tau_l + a) = \boldsymbol{r}' \ominus \boldsymbol{e}_{i,s_i+1}$. Thus partial order is preserved for all cases.

If $a = T_l^{\min}$ and $T_l^{\min} = \xi_{ij}^{(l)} = T'_l < T_l$, then we have that $R_{ij}(\tau_l) = 0$ and $R_{i,s_i+1}(\tau_l) = 0$ and no transitions at time τ_{l+1}. We also have that either $R'_{ij}(t)$ decreases from 1 to 0 or $R'_{i,s_i+1}(t)$ decreases by 1 at time τ_{l+1}. Therefore $\boldsymbol{R}(\tau_l + a) = \boldsymbol{r} \preceq \boldsymbol{R}'(\tau_l + a) = \boldsymbol{r}' - \boldsymbol{e}_{ij}$ or $\boldsymbol{R}(\tau_l + a) = \boldsymbol{r} \preceq \boldsymbol{R}'(\tau_l + a) = \boldsymbol{r}' - \boldsymbol{e}_{i,s_i+1}$. The partial order is preserved.

Above all the theorem is proved. □

A.2. Proof of Theorem 3.2

Proof. Suppose the current time is τ_l, we only need to show that if $\underline{\boldsymbol{R}}(\tau_l) \preceq \boldsymbol{R}(\tau_l) \preceq \bar{\boldsymbol{R}}(\tau_l)$ then $\underline{\boldsymbol{R}}(\tau_l + a) \preceq \boldsymbol{R}(\tau_l + a) \preceq \bar{\boldsymbol{R}}(\tau_l + a)$ for $a = T_l^{\min}$ since all chains do not change for $a \in (0, T_l^{\min})$.

We can see that the upper-bound chain updating function $\bar{\phi}$ is similar to the updating function ϕ for $\boldsymbol{R}(t)$. The only difference is that at time $\tau_{l+1} = \tau_l + T_l^{\min}$ when $T_l^{\min} = \xi_0^{(l)}$ and $\bar{R}_{i,s_i+1}(\tau_l) = N$ we use $\bar{\oplus}$ instead of $\oplus$, i.e. $\bar{\boldsymbol{R}}(\tau_{l+1}) = \bar{\boldsymbol{R}}(\tau_l) \bar{\oplus} \mathbf{1}$. When $\bar{R}_{i,s_i+1}(\tau_l) = N$, the chains $\boldsymbol{R}(t)$ may be either such that $R_{i,s_i+1}(\tau_l) = N$ or $R_{i,s_i+1}(\tau_l) < N$. (a) If $T_l^{\min} = \xi_0^{(l)}$ and $R_{i,s_i+1}(\tau_l) = N$ for some i then $\boldsymbol{R}(\tau_{l+1}) = \boldsymbol{R}(\tau_l) \preceq \bar{\boldsymbol{R}}(\tau_l) \preceq \bar{\boldsymbol{R}}(\tau_{l+1}) = \bar{\boldsymbol{R}}(\tau_l) \bar{\oplus} \mathbf{1}$. The partial order is preserved. (b) If $T_l^{\min} = \xi_0^{(l)}$ and $R_{i,s_i+1}(\tau_l) < N$ for all i then $\boldsymbol{R}(\tau_{l+1}) = \boldsymbol{R}(\tau_l) \oplus \mathbf{1} \preceq \bar{\boldsymbol{R}}(\tau_{l+1}) = \bar{\boldsymbol{R}}(\tau_l) \bar{\oplus} \mathbf{1}$ since $R_{i,s_i+1}(\tau_l) + 1 \leq \min\{N, \bar{R}_{i,s_i+1}(\tau_l) + 1\}$. Therefore the chain $\boldsymbol{R}(t)$ is always bounded by $\bar{\boldsymbol{R}}(t)$.

The lower-bound chain updating function $\underline{\phi}$ is also similar to ϕ. The only difference is that if $T_l^{\min} = \xi_0^{(l)}$ and $\bar{R}_{i,s_i+1}(\tau_l) = N$ we keep $\underline{\boldsymbol{R}}(\tau_{l+1})$ unchanged, i.e. $\underline{\boldsymbol{R}}(\tau_{l+1}) = \underline{\boldsymbol{R}}(\tau_l)$. On the other hand, if a new job arrives then we must have $\boldsymbol{R}(\tau_{l+1}) = \boldsymbol{R}(\tau_l) \oplus \mathbf{1}$ or $\boldsymbol{R}(\tau_{l+1}) = \boldsymbol{R}(\tau_l)$. Therefore $\underline{\boldsymbol{R}}(\tau_{l+1}) = \underline{\boldsymbol{R}}(\tau_l) \preceq \boldsymbol{R}(\tau_l) \preceq \boldsymbol{R}(\tau_{l+1})$. Therefore the chain $\boldsymbol{R}(t)$ is always bounded below by $\underline{\boldsymbol{R}}(t)$. □

A.3. Proof of Lemma 4.1

Proof. If $r_{ij} > 0$ for $j = 1, \cdots, s_i$ then $r'_{ij} > 0$ for $j = 1, \cdots, s_i$ since $\boldsymbol{r} \preceq \boldsymbol{r}'$. Therefore $\boldsymbol{r}_i \uplus 1 \preceq \boldsymbol{r}' \uplus 1$ as both r_{i,s_i+1} and r'_{i,s_i+1} are added by 1.

If $r_{im} = 0$ for at least one m in $\{1, \cdots, s_i\}$ then $\boldsymbol{r}_i \uplus 1$ means that r_{ij}, $j = J(\boldsymbol{\eta}_i, \boldsymbol{r}_i)$ will be added by 1. On the other hand since $\boldsymbol{r} \preceq \boldsymbol{r}'$, we have that the upper one $\boldsymbol{r}'$ is such that $r'_{ij} = 0$ or $r'_{ij} > 0$. If $r'_{ij} = 0$ then $\boldsymbol{r}'_i \uplus 1$ based on rejection sampling also means that r'_{ij} will be added by 1. This implies $\boldsymbol{r}_i \uplus 1 \preceq \boldsymbol{r}' \uplus 1$. If $r'_{ij} > 0$ then $r'_{ih}, h \neq j$ will be added by 1 which also gives $\boldsymbol{r}_i \uplus 1 \preceq \boldsymbol{r}' \uplus 1$.

Above all, the Lemma is proved. □

A.4. Proof of Theorem 4.1

Proof. Suppose that the current time is τ_l. Following the proof of Theorem 3.2, we only need to show that upper-bound chain and lower-bound chain bound all chains when $a = T_l^{\min}$ and $T_l^{\min} = \xi_0^{(l)}$. This is because all other cases can be proved similarly as Theorem 3.2.

(1) If $a = T_l^{\min}$, $T_l^{\min} = \xi_0^{(l)}$ and $\bar{R}_{i,s_i+1}(\tau_l) < N$ for all i and $\underline{\boldsymbol{R}}(\tau_l) \preceq \boldsymbol{R}(\tau_l) \preceq \bar{\boldsymbol{R}}(\tau_l)$ then $\underline{\boldsymbol{R}}(\tau_l + a) = \underline{\boldsymbol{R}}(\tau_l) \uplus \mathbf{1} \preceq \boldsymbol{R}(\tau_l + a) = \boldsymbol{R}(\tau_l) \uplus \mathbf{1} \preceq \bar{\boldsymbol{R}}(\tau_l + a) = \bar{\boldsymbol{R}}(\tau_l) \uplus \mathbf{1}$, which is proved in Lemma 4.1.

(2) We also need to show that if $a = T_l^{\min}$, $T_l^{\min} = \xi_0^{(l)}$ and $\bar{R}_{i,s_i+1}(\tau_l) = N$ for some i and $\underline{\boldsymbol{R}}(\tau_l) \preceq \boldsymbol{R}(\tau_l) \preceq \bar{\boldsymbol{R}}(\tau_l)$ then

$$\underline{\boldsymbol{R}}(\tau_l + a) = \underline{\boldsymbol{R}}(\tau_l) \preceq \boldsymbol{R}(\tau_l + a) \preceq \bar{\boldsymbol{R}}(\tau_l + a) = \bar{\boldsymbol{R}}(\tau_l) \bar{\uplus} \mathbf{1}. \tag{24}$$

There are two cases to be considered. (a) If $a = T_l^{\min}$, $T_l^{\min} = \xi_0^{(l)}$ and $R_{i,s_i+1}(\tau_l) = N$ for some i then (24) holds since $\boldsymbol{R}(\tau_l + a) = \boldsymbol{R}(\tau_l)$ in such cases. (b) If $a = T_l^{\min}$, $T_l^{\min} = \xi_0^{(l)}$ and $R_{i,s_i+1}(\tau_l) < N$ for all i, then $\boldsymbol{R}(\tau_l + a) = \boldsymbol{R}(\tau_l) \uplus \mathbf{1}$. In such case (24) also holds since $\boldsymbol{R}(\tau_l + a) = \boldsymbol{R}(\tau_l) \uplus \mathbf{1} \preceq \bar{\boldsymbol{R}}(\tau_l + a) = \bar{\boldsymbol{R}}(\tau_l) \bar{\uplus} \mathbf{1}$.

Above all the theorem is proved. □

References

Baccelli F., Massey, W. A., & Towsley D. (1989). Acyclic fork-join queuing networks. *Journal of the ACM*, *36*(3), 615-642. http://dx.doi.org/10.1145/65950.65957

Balsoma, S., Donatietllo, L., & van Dijk, N. M. (1998). Bound performance models of heterogeneous parallel processing systems. *IEEE Trans. Parallel Distributed Systems*, *9*, 1041-1056. http://dx.doi.org/10.1109/71.730531

Cooper, R. B. (1976). Queues with ordered servers that work at different rates. *Opsearch*, *13*, 69-78.

Dai, H. (2008). Perfect sampling methods for random forests. *Advances in Applied Probability, 40*, 897-917. http://dx.doi.org/10.1239/aap/1222868191

Dai, H. (2011). Exact simulation for fork-join networks. *Advances in Applied Probability, 43*, 484-503. http://dx.doi.org/10.1239/aap/1308662489

Flatto, L., & Hahn, S. (1984). Two parallel queues created by arrivals with two demands I. *SIAM J. Appl. Math., 44*, 1041-1053. http://dx.doi.org/10.1137/0144074

Flatto, L. (1985). Two parallel queues created by arrivals with two demands II. *SIAM J. Appl. Math., 45*, 861-878. http://dx.doi.org/10.1137/0145052

Huber, M. (2004). Perfect sampling using bounding chains. *Annals of Applied Probability, 14*, 734-753. http://dx.doi.org/10.1214/105051604000000080

Kendall, W. S., & Moller, J. (2000). Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Advances in Applied Probability, 32*, 844-865. http://dx.doi.org/10.1239/aap/1013540247

Ko, S-S., & Serfozo, R. F. (2004). Response Times in M/M/s Fork-Join Networks. *Advances in Applied Probability, 36*, 854-871. http://dx.doi.org/10.1239/aap/1093962238

Kumar, B. K., & Madheswari, S. P. (2007). Transient Solution of and M/M/2 queue with heterogeneous servers subject to catastrophes. *Information and Management Sciences, 18*, 63-80.

Lebrecht, A. S., & Knottenbelt, W. J. (2007). Response time approximations in fork-join queues. *National Workshop Paper, 23rd Annual UK Performance Engineering Workshop (UKPEW).*

Nelson, R., & Tantawi, A. N. (1988). Approximate analysis of fork-join synchronization in parallel queues. *IEEE transactions on Software Engineering, 37*(6), 739-743.

Propp, J. G., & Wilson, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures Algorithms, 9*, 223-252. http://dx.doi.org/10.1002/(SICI)1098-2418(199608/09)9:1/2<223::AID-RSA14>3.0.CO;2-O

Raghavan, N. R. S., & Viswanadham, N. (2001). Generalized queueing network analysis of integrated supply chains. *International Journal of Production Research, 39,* 205-224. http://dx.doi.org/10.1080/00207540010003846

Ross, S. M. (2007). *Introduction to Probability Models*, Academic Press.