Reduction of Closed Queueing Networks for Efficient Simulation

JOHN F. SHORTLE, BRIAN L. MARK, and DONALD GROSS
George Mason University

This article gives several methods for approximating a closed queueing network with a smaller one. The objective is to reduce the simulation time of the network. We consider Jackson-like networks with Markovian routing and with general service distributions. The basic idea is to first divide the network into two parts—the core nodes of interest and the remaining nodes. We suppose that only metrics at the core nodes are of interest. The remaining nodes are collapsed into a reduced set of nodes, in an effort to approximate the flows into and out of the set of core nodes. The core nodes and their interactions are preserved in the reduced network. We test the network reductions for accuracy and speed. By randomly generating sample networks, we test the reductions on a large variety of test networks, rather than on a few specific cases. The main conclusion is that the reductions work well when the squared coefficients of variation of the service distributions are not all small (that is, the network is not close to being deterministic) and for nodes where the utilization is not too high or too low.

Categories and Subject Descriptors: I.6.5 [**Simulation and Modeling**]: Model Development

General Terms: Algorithms

Additional Key Words and Phrases: Queueing networks, network decomposition

ACM Reference Format:

Shortle, J. F., Mark, B. L., and Gross, D. 2009. Reduction of closed queueing networks for efficient simulation. ACM Trans. Model. Comput. Simul., 19, 3, Article 10 (June 2009), 22 pages.
DOI = 10.1145/1540530.1540531 http://doi.acm.org/10.1145/1540530.1540531

1. INTRODUCTION

The objective of this article is to develop algorithms for reducing the size of large queueing networks, for the purposes of more efficient network simulation. In many queueing networks, performance metrics are of interest only at

This work has been supported in part by the National Science Foundation under Grant IIS-0325074.
Authors' addresses: J. F. Shortle, D. Gross, Systems Engineering and Operations Research, George Mason University, 4400 University Dr. MS 4A6, Fairfax, VA 22030; email: {jshortle,dgross1}@gmu.edu; B. L. Mark, Electrical and Computer Engineering, George Mason University, 4400 University Dr. MS 1G5, Fairfax, VA 22030; email: bmark@gmu.edu.

DOI 10.1145/1540530.1540531 http://doi.acm.org/10.1145/1540530.1540531

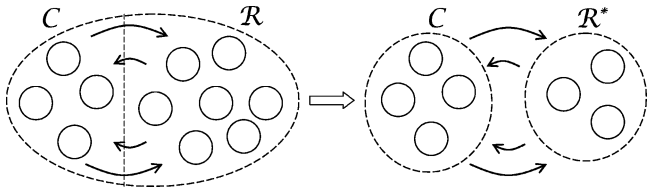

Fig. 1. Basic network reduction approach.

a small number of nodes in the network. The other nodes are of interest only to the extent that they influence the performance metrics at the core nodes. The basic approach is to divide a queueing network into two subnetworks: $\mathcal{C}$ and $\mathcal{R}$ (Figure 1). Subnetwork $\mathcal{C}$ contains the core nodes of interest, and subnetwork $\mathcal{R}$ contains the remaining nodes. The goal is to approximate $\mathcal{R}$ with a smaller subnetwork $\mathcal{R}^*$, such that simulation of $\mathcal{C}$ with $\mathcal{R}^*$ approximates the simulation of $\mathcal{C}$ with $\mathcal{R}$. The performance metrics of $\mathcal{C}$ should be similar in both networks, but the performance metrics in $\mathcal{R}^*$ and $\mathcal{R}$ do not necessarily need to correspond.

The key idea of this article is to match the first two moments of the flows from $\mathcal{R}^*$ to $\mathcal{C}$ with those from $\mathcal{R}$ to $\mathcal{C}$. That is, the service distributions and routing probabilities of the reduced network are chosen so that the first and second moments of the arrival processes from $\mathcal{R}^*$ to $\mathcal{C}$ match (are approximations to) the corresponding moments of the arrival processes from $\mathcal{R}$ to $\mathcal{C}$ in the original network. The main reduction we propose includes one node in $\mathcal{R}^*$ for each node in $\mathcal{C}$.

The motivation for this research is simulation of the National Airspace System (NAS). The NAS contains about 30 large hub airports (e.g., Atlanta), about 30 medium-sized hub airports (e.g., Cleveland), about 50 small hub airports (e.g., Colorado Springs), about 600 other airports with scheduled flights, and thousands of other public-use airports. On a typical weekday, there are about 5,000 commercial flights in the air at one time, plus many general aviation flights, unscheduled business flights, and so forth. Detailed full-scale simulations of the NAS can be quite time consuming. For example, Yousefi et al. [2003] simulated commercial and general aviation flights in the northeast United States using the Total Airport and Airspace Modeler (TAAM).[1] The simulation, running in a deterministic mode, took about 8 hours of computer time to generate 24 hours of simulated time. Thus, running TAAM multiple times in a stochastic mode is extremely time consuming. Methods are needed to simulate these large systems, with an appropriate balance of accuracy and speed.

This article gives algorithms for constructing reduced approximations of large Jackson-like networks with general service distributions. A high fidelity simulator like TAAM is quite complex, so we do not suggest that these Jackson-like networks can be used to replicate a model like TAAM. The purpose here is to test core ideas on these simpler networks and later to extend the research to more complex simulations.

[1]Preston Aviation Solutions, `http://www.preston.net/products/TAAM.htm`.

A unique feature of this research is that we test our network reductions on sample networks that are randomly generated. By automating this process, we can test the reductions on hundreds or thousands of networks, rather than on individual test cases, as is often done.

This article is organized as follows: Section 2 contains a brief summary of related research in queueing network approximations and large-scale network simulations. Section 3 gives preliminary definitions and specifies notation. Section 4 presents a reduction algorithm and gives an example application of the algorithm to a simple network. Section 5 applies the reduction algorithm to a large set of randomly generated networks. Section 6 considers the benefits gained in simulation.

2. RELATED LITERATURE

There is a wide body of literature on the approximation of queueing networks. This section highlights general approaches that are related to the approximations given in this article.

Decomposition Methods. These methods consist of decomposing a network into smaller subnetworks and then analyzing each subnetwork separately. Typically, the sub-networks consist of individual queues. Examples of single-queue decompositions are given in Gelenbe and Pujolle [1998], Kuehn [1979], Bitran and Tirupati [1988], Reiman [1990], and Whitt [1983b, 1983a]. An example of a multi-queue decomposition is given in Dai et al. [1994]. A popular approach is a two-moment parametric decomposition approximation implemented in the Queueing Network Analyzer (QNA) [Whitt 1983b, 1983a]. In this approach, a network is approximated as a set of individual isolated $GI/G/1$ queues. The method solves a series of balance equations to approximately match the first and second moments of the arrival and departure processes at each queue. Then, assuming the arrival process to each queue is a renewal process, performance metrics at each queue are computed using approximation formulas for the $GI/G/1$ queue. The fundamental assumption is that the superposition of interdeparture processes from upstream nodes flowing to a downstream node is approximately a renewal process (e.g., Whitt [1982]). Many variations based on this two-moment decomposition approximation have also been proposed (e.g., Suresh and Whitt [1990] and Whitt [1995]).

Aggregation Methods. These methods simplify a network by aggregating subnetworks into individual nodes. The characteristics of the individual nodes are determined from the local, isolated behavior of the subnetworks. The global system behavior is determined by analyzing the network of aggregated nodes. In other words, these methods analyze the interaction of global variables between subnetworks separately from the interaction of local variables within subnetworks. An example of an exact aggregation method is Norton's theorem [Chandy and Georganas 1975; Chandy et al. 1975]. Consider an M-node closed Jackson network that is divided into two subnetworks: $\mathcal{C} = \{M\}$ and $\mathcal{R} = \{1, \ldots, M-1\}$. Norton's theorem says that $\mathcal{R}$ can be aggregated into a

single node, R_{eq}, having a state-dependent service rate. This aggregation also works when $\mathcal{C}$ contains more than one node, provided the network satisfies a particular routing structure [Boucherie and van Dijk 1993]. Further generalizations exist for multiple customer classes and state-dependent routing [Boucherie and van Dijk 1993; Boucherie and Stewart 1998]. When an aggregation method provides exact results, the network is said to be *decomposable*. Curtois [1975, 1977] has given conditions under which aggregation methods provide approximately accurate results (see also Bolch et al. [2006]). Such systems are said to be *quasi-decomposable*.

Heavy Traffic Methods. These methods are based on the principle that the queue length process of certain networks in heavy traffic can be approximated by reflected Brownian motion (RBM) (e.g., Reiman [1984]). First, one determines the RBM associated with the queueing network in question [Harrison and Nguyen 1990], and then one numerically calculates the stationary distribution of the RBM [Dai and Harrison 1992]. This is called the QNET method [Harrison and Nguyen 1990; Dai and Harrison 1993; Dai et al. 1997]. One problem is that the computational requirement grows significantly in the size of the network (though hybrid decompositions methods can mitigate this problem, as discussed in Dai et al. [1994]).

Large-Network Simulation. Much research on large-network simulation has been motivated by telecommunications and Internet applications. In this domain, network size can be extremely problematic. For example, Riley and Ammar [2002] conservatively estimate that it would take a year of CPU time to simulate 100 seconds of activity on the Internet (based on 2002 data). Many large-network simulations make use of Internet-specific protocols (e.g., Nicol et al. [2003] and Riley [2003]), and are not appropriate in other contexts. Also, fluid models (e.g., Liu et al. [1999, 2003]) work well to approximate high volume communication channels, but are not suitable in the aviation domain, where the volume of flights is not high enough to justify the fluid approximation. An overview of large-network simulation is given in Nicol et al. [2005]. Another common approach is parallelization (e.g., Cowie et al. [1999] and Rao and Wilsey [1999]). We do not consider parallelization methods here. This is because we improve simulation speed by reducing the size of the network. Thus, parallel methods can be applied equally well to the original and reduced networks.

This article uses elements of both aggregation and decomposition. The approximations use aggregation in the sense that we reduce a subnetwork, $\mathcal{R}$, to a smaller subnetwork, $\mathcal{R}^*$. Thus, the approximate network is smaller than the original network. The approximations use decomposition in the sense that the queueing parameters for $\mathcal{R}^*$ are determined using decomposition algorithms. Two key differences are:

—We convert the original network into two subnetworks of multiple queues, rather than into multiple subnetworks of individual queues. The purpose is to preserve key interactions within the core subnetwork, $\mathcal{C}$. Most decomposition

methods separate the network into individual queues.[2] Also, most aggregation methods reduce subnetworks to single queues.

—We simulate the final reduced network. In contrast, most aggregation and decomposition methods compute approximate results analytically.

In summary, the end result of our approximation is a reduced network that is analyzed using simulation, rather than through approximate numerical methods. Simulation is used so that the approximations can be extended to more complex network models in the future. For example, a departure schedule where customers leave at scheduled times (or later if delayed) can be modeled well with simulation but not with a Jackson network. This article specifically deals with Jackson-like networks, while future work will extend the application of the methods to more general models.

3. PRELIMINARIES AND NOTATION

This section defines notation and fundamental algorithms used throughout this article. We also give an example network to illustrate concepts.

Consider a closed Jackson-like network consisting of N nodes and M customers with general service times. Assume that each node i, has a single server with general service distribution G_i. (The methods in this article generalize to networks of multiserver queues, but we only test single-server cases here). Let S_i denote a random service time at node i. A customer who completes service at node i transitions to node j with probability $p_{i,j}$. All service times and transitions are independent. Further, divide the network into two sets of nodes, $\mathcal{C}$ and $\mathcal{R}$, where $\mathcal{C}$ contains the core nodes of interest and $\mathcal{R}$ contains the remaining nodes. Assume that $\mathcal{C}$ contains $N_C \leq N-2$ nodes. In the envisioned applications, we expect that $N_C \ll N$.

In the aviation context, the nodes might represent airports, runways, gates, and flight corridors. For example, during a single flight, an airplane departs its gate, enters a departure queue, departs, flies through several air sectors en route to the destination, lands on a runway, and then goes to a gate. Each of these phases can be considered a queue with a limited resource. For example, if an air sector is too crowded, controllers may delay an airplane from entering that sector by extending its flight path. Similarly, if there are not enough gates at an airport upon arrival, an airplane must wait until a gate becomes free. In a sector, the service distribution is typically defined as the unimpeded time it takes an airplane to traverse the sector. At a gate, the service time is the time it takes to unload and then reload the airplane with passengers. At a runway, the service time is the minimum required separation between aircraft. Transition probabilities in the network represent the frequencies with which airplanes transition from one node to another and could be determined by flight schedules. However, a key difference is that this article investigates networks

[2]Dai et al. [1994] give a decomposition method that divides a network into subnetworks of multiple queues. However, the groupings are based on similar traffic intensities. In this article, the groupings are predefined, possibly based on known interactions—for example, shuttle flights back and forth between a city pair—and may contain nodes with a diverse range of traffic intensities.

Table I. Network Notation

Inputs	N	Number of nodes in network
	N_C	Number of nodes in $\mathcal{C}$
	M	Number of customers in network
	$p_{i,j}$	Transition probability from i to j
	G_i	Service distribution at i
	S_i	Random service time at i
	μ_i	Service rate at i: $\mu_i \equiv 1/\mathrm{E}[S_i]$
	c_{si}^2	SCV of service distribution at i: $c_{si}^2 \equiv \mathrm{Var}[S_i]/\mathrm{E}^2[S_i]$
Inter-mediate Variables	λ_i	Throughput at i
	ρ_i	Utilization at i: $\rho_i \equiv \lambda_i/\mu_i$
	c_{aj}^2	SCV of inter-arrival times to j
	c_{di}^2	SCV of inter-departure times from i
	$c_{di,aj}^2$	SCV of inter-event times: departures from i arriving to j
Outputs	W_{qi}	Mean waiting time in queue at i

where node transitions are random, versus predetermined transitions governed by aircraft schedules.

Table I defines parameters associated with the network. The first set are considered inputs. The second set are internal traffic descriptors. These are not given explicitly, but must be computed through analysis of the network. They are used as intermediate variables in the process of obtaining the final results. The last set, W_{qi}, is considered outputs. These are the performance metrics that the practitioner is trying to estimate. In the table, SCV refers to the squared coefficient of variation (variance / mean^2).

The focus of this article is to efficiently estimate the performance metrics in Table I using simulation. However, to do this, we will make use of analytical approaches for approximating the intermediate variables. These approaches will be used in constructing reduced networks as approximations to the original networks (Section 4). Then, we will simulate the reduced and original networks and compare speed and accuracy (Section 6). The rest of this section describes analytical techniques used in this process.

If all service distributions are exponential, then network performance metrics can be computed exactly using well-known algorithms, such as Buzen's algorithm or mean value analysis (e.g., Gross and Harris [1998]).[3] If the service distributions are general, an approximate analytical method can be used, such as a parametric decomposition approximation, for example, Whitt [1983b, 1983a], Kim [2004].

The following algorithm describes a method to approximate λ_i, ρ_i, c_{aj}^2, c_{di}^2, and $c_{di,aj}^2$ (in Table I). This is a variation of the two-moment parametric approximation in Whitt [1983b, 1983a]. The purpose of approximating these metrics is not to obtain the metrics themselves, but to create a reduced network as an approximation to the original network. The reduced network is then simulated (Section 6), with the goal of estimating quality of service metrics like W_q for the core nodes in $\mathcal{C}$.

[3]To download Queueing Theory Software QTS, a computer implementation of these algorithms, see ftp://ftp.wiley.com/public/ sci_tech_med/queueing_theory/.

Algorithm A. Approximate λ_i, ρ_i, c_{aj}^2, c_{di}^2, and $c_{di,aj}^2$ (in Table I).

(1) Approximate λ_i and ρ_i: For the moment, assume that all service distributions are exponential (the network is a closed Jackson network). Calculate λ_i and ρ_i from the standard flow-balance equations using Buzen's algorithm or mean value analysis. For an open network, λ_i is exact for general service distributions because the flow-balance equations are independent of the service distributions. Therefore, this approximation for λ_i is accurate when the original closed network is well approximated by an open network.[4]

(2) Approximate c_{aj}^2: Solve the following system of equations for c_{aj}^2 (e.g., Kim [2004], Equation 6). The equations are linear in c_{aj}^2. These equations essentially represent a second moment flow balance between the nodes.

$$c_{aj}^2 = \frac{1}{\lambda_j} \sum_i \lambda_i p_{i,j} \left(p_{i,j}[\rho_i^2 c_{si}^2 + (1-\rho_i^2)c_{ai}^2] + 1 - p_{i,j}\right), \qquad j = 1, \ldots, N. \tag{1}$$

The equations can be derived from the following queueing, splitting, and superposition formulas (e.g., Whitt [1983b], Equations 38, 36, and 31[5]):

(a) Queueing:

$$c_{di}^2 = \rho_i^2 c_{si}^2 + (1-\rho_i^2)c_{ai}^2, \tag{2}$$

(b) Splitting:

$$c_{di,aj}^2 = p_{i,j} c_{di}^2 + (1 - p_{i,j}), \tag{3}$$

(c) Superposition:

$$c_{aj}^2 = \frac{1}{\lambda_j} \sum_i \lambda_i p_{i,j}\, c_{di,aj}^2. \tag{4}$$

These equations are approximate relationships. (3) is exact under Markovian routing of a renewal process.

(3) Approximate c_{di}^2 and $c_{di,aj}^2$: Use (2) and (3).

(4) Approximate the following quantities using values for λ_i and $c_{di,aj}^2$ obtained previously.

(a) The throughput from $\mathcal{R}$ to $\mathcal{C}$ is

$$\lambda_{\mathcal{R}} \equiv \sum_{i \in \mathcal{R}} \sum_{j \in \mathcal{C}} \lambda_i p_{i,j}. \tag{5}$$

(b) The routing probabilities from a node i, to any node in $\mathcal{R}$ (or $\mathcal{C}$) are

$$p_{i,\mathcal{R}} \equiv \sum_{j \in \mathcal{R}} p_{i,j}, \qquad p_{i,\mathcal{C}} \equiv \sum_{j \in \mathcal{C}} p_{i,j}. \tag{6}$$

Thus, $p_{i,\mathcal{R}} + p_{i,\mathcal{C}} = 1$, for any node i.

(c) The aggregate routing probabilities from $\mathcal{R}$ to a node $j \in \mathcal{C}$ are

$$p_{\mathcal{R},j} \equiv \frac{1}{\lambda_{\mathcal{R}}} \sum_{i \in \mathcal{R}} \lambda_i p_{i,j}. \tag{7}$$

Thus, $\sum_{j \in \mathcal{C}} p_{\mathcal{R},j} = 1$.

[4]A similar approach adapting open approximations to closed networks is the fixed population mean (FPM) method [Whitt 1984]. In this approach, the closed network is first converted to an open network by changing one node to a source/sink. The system is then analyzed as an open network using a parametric decomposition, where the external throughput, λ, is chosen so that the expected number of customers in the network equals M. Both methods rely on the assumption that the closed network is well approximated by an open network.

[5]QNA uses a modified version of (2) and (4).

(d) The (approximate) SCV of the arrival process to a node $j \in \mathcal{C}$, due to flows only from $\mathcal{R}$, is

$$c^2_{a\mathcal{R},j} \equiv \frac{1}{\lambda_{\mathcal{R}} \cdot p_{\mathcal{R},j}} \sum_{i \in \mathcal{R}} \lambda_i p_{i,j} \, c^2_{di,aj}. \tag{8}$$

This is the superposition formula (4), where the summation is taken only over nodes in $\mathcal{R}$; furthermore the normalizing constant $(\lambda_{\mathcal{R}} \cdot p_{\mathcal{R},j})$ represents the combined flow from $\mathcal{R}$ to j, rather than the total flow (λ_j) through j.

This algorithm can also be extended to multiserver queues. The splitting equation (3) and superposition equation (4) remain the same, but the queueing equation (2) is changed to account for multiple servers (e.g., Whitt [1983b], Equation 39). The combination of these equations leads to an appropriately modified version of (1).

4. NETWORK REDUCTIONS

This section presents two network reductions. The first reduction involves collapsing all nodes in $\mathcal{R}$ into a single node $\mathcal{R}^*$, as shown in Figure 2. We define the routing probabilities and service rates in the reduced network to achieve approximately the same throughput in $\mathcal{C}$ for the original and reduced networks. This is specified precisely below. For notation, we use a hat (ˆ) to denote parameters in the reduced network.

Reduction 1.

(1) Collapse all nodes in $\mathcal{R}$ to a single node $\mathcal{R}^*$ (see Figure 2).

(2) Routing probabilities in the reduced network are:

$$\begin{aligned} \hat{p}_{i,j} &= p_{i,j}, \qquad i \in \mathcal{C}, j \in \mathcal{C}, \\ \hat{p}_{i,\mathcal{R}^*} &= p_{i,\mathcal{R}}, \qquad i \in \mathcal{C}, \text{ see (6)}, \\ \hat{p}_{\mathcal{R}^*,j} &= p_{\mathcal{R},j}, \qquad j \in \mathcal{C}, \text{ see (7)}. \end{aligned}$$

(3) Service rates in the reduced network are:

$$\hat{\mu}_i = \mu_i, \qquad i \in \mathcal{C}.$$

For the node $\mathcal{R}^*$, $\hat{\mu}_{\mathcal{R}^*}$ is chosen so that $\hat{\lambda}_{\mathcal{R}^*} = \lambda_{\mathcal{R}}$; see (5). (This can be done, for example, using a binary search on $\hat{\mu}_{\mathcal{R}^*}$. Given a value for $\hat{\mu}_{\mathcal{R}^*}$, $\hat{\lambda}_{\mathcal{R}^*}$ is determined by applying MVA to the reduced network. A binary search is applied by varying $\hat{\mu}_{\mathcal{R}^*}$ until the resulting $\lambda_{\mathcal{R}^*}$ equals $\lambda_{\mathcal{R}}$. Initial low and high values for $\hat{\mu}_{\mathcal{R}^*}$ can be $\lambda_{\mathcal{R}}$ and $2\lambda_{\mathcal{R}}$.) The service distribution family for node $\mathcal{R}^*$ is set to the service distribution family corresponding to the node $j \in \mathcal{R}$ with maximum throughput to $\mathcal{C}$ (that is, the j that maximizes $\sum_{i \in C} \lambda_j p_{j,i}$).

Typically, the value for $\hat{\mu}_{\mathcal{R}^*}$ found in step 3 is just slightly larger than $\lambda_{\mathcal{R}}$. This is because if $\mathcal{R}$ is a large subnetwork, then there is a high probability that there is at least one customer in the single node $\mathcal{R}^*$. So $\hat{\rho}_{\mathcal{R}^*}$ is slightly less than 1. Thus, $\hat{\mu}_{\mathcal{R}^*} = \hat{\lambda}_{\mathcal{R}^*}/\hat{\rho}_{\mathcal{R}^*}$ is slightly greater than $\lambda_{\mathcal{R}}$ when $\hat{\lambda}_{\mathcal{R}^*} = \lambda_{\mathcal{R}}$.

Reduction 1 is similar to variations of Norton's theorem in which a group of nodes in a network are replaced by a single node—for example, Boucherie

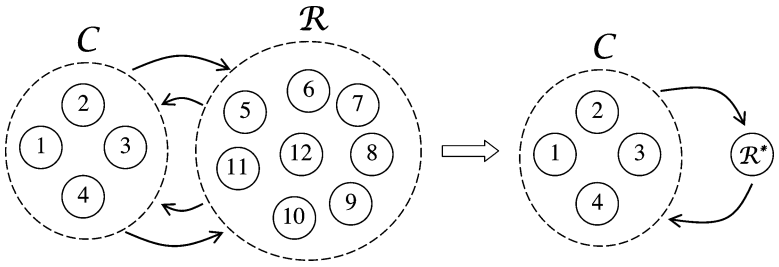

Fig. 2. Reduction 1.

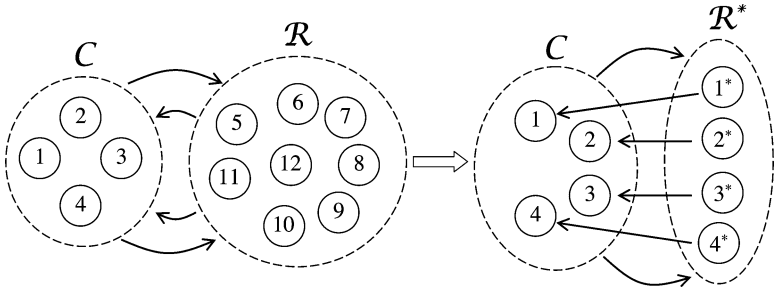

Fig. 3. Reduction 2.

and van Dijk [1993]. The method given in that article is exact; however, it only works under very specific conditions on the routing structure of the network. Essentially, the condition requires that there is only one node in $\mathcal{R}$ that is an entry point from $\mathcal{C}$ and only one node in $\mathcal{R}$ that is an exit point to $\mathcal{C}$. The precise conditions are slightly more general, but this is the basic idea. The conditions do not typically hold for the networks in this article, except in the case where $\mathcal{C}$ consists of a single node.

Figure 3 shows the basic idea for our second reduction. Each node i in $\mathcal{C}$ has a corresponding node i^* in $\mathcal{R}^*$. Node i^* controls the flow of customers into i. By choosing an appropriate service rate for i^*, we can approximately match the first two moments of the arrival stream into node i, with respect to the original network. In contrast, Reduction 1 matches the average flows into each node, but does not have enough degrees of freedom to match the second moment of flows to each node in $\mathcal{C}$.

Reduction 2.

(1) Collapse all nodes in $\mathcal{R}$ to a subnetwork, $\mathcal{R}^*$, consisting of N_C nodes (see Figure 3).
(2) Routing probabilities in the reduced network are:

$$\begin{aligned}
\hat{p}_{i,j} &= p_{i,j}, && i \in \mathcal{C}, j \in \mathcal{C},\\
\hat{p}_{i^*,j^*} &= 0, && i^* \in \mathcal{R}^*, j^* \in \mathcal{R}^*,\\
\hat{p}_{i,j^*} &= p_{i,\mathcal{R}} \cdot p_{\mathcal{R},j}, && i \in \mathcal{C}, j^* \in \mathcal{R}^*, \quad \text{see (6) and (7)},\\
\hat{p}_{i^*,j} &= \delta_{i,j}, && i^* \in \mathcal{R}^*, j \in \mathcal{C},
\end{aligned}$$

where $\delta_{i,j} = 1$ if $i = j$, 0 otherwise. These equations completely specify the routing probability matrix $\hat{\mathbf{P}} = [\hat{p}_{i,j}]$ of the reduced network.

(3) The service distributions in $\mathcal{C}$ are the same in both networks. In particular, the first two moments are equal, so

$$\hat{\mu}_i = \mu_i, \qquad i \in \mathcal{C},$$
$$\hat{c}^2_{si} = c^2_{si}, \qquad i \in \mathcal{C}.$$

(4) For $i^* \in \mathcal{R}^*$, set:

$$\hat{c}^2_{si^*} = c^2_{a\mathcal{R},i}, \quad \text{see (8)}.$$

To determine $\hat{\mu}_{i^*}$, let $\hat{\mu}_{i^*} = b \cdot \lambda_{\mathcal{R}} \cdot p_{\mathcal{R},i}$, for each i, where b is a scaling constant. Choose b so that $\hat{\lambda}_{\mathcal{R}^*} = \lambda_{\mathcal{R}}$ (that is, to match the throughput out of $\mathcal{R}^*$ in the reduced network with the throughput out of $\mathcal{R}$ in the original network). The appropriate value of b can be found, for example, using a binary search with initial high and low values of 1 and 2. The service distribution family for node i^* is set to the service distribution family corresponding to the node $j \in \mathcal{R}$ with maximum throughput to i (that is, the j that maximizes $\lambda_j p_{j,i}$ for fixed i). (In this article, we consider service distributions from the Weibull, lognormal, and gamma families.)

The choice for $\hat{\mu}_{i^*}$ is motivated in a similar manner as in Reduction 1. Assuming that $\hat{\rho}_{i^*}$ is slightly less than 1, then $\hat{\mu}_{i^*} = \hat{\lambda}_{i^*}/\hat{\rho}_{i^*}$ is slightly greater than $\hat{\lambda}_{i^*}$. Ideally, we would like $\hat{\lambda}_{i^*} = \lambda_{\mathcal{R}} \cdot p_{\mathcal{R},i}$, which is the throughput from $\mathcal{R}$ to i in the original network. Thus, $\hat{\mu}_{i^*}$ should be slightly greater than $\lambda_{\mathcal{R}} \cdot p_{\mathcal{R},i}$. The value b in step 4 allows us to adjust $\hat{\mu}_{i^*}$ slightly upward.

Similarly, the choice for $\hat{c}^2_{si^*}$ gives an approximate match of the SCV of the arrival process from $\mathcal{R}^*$ to i in the reduced network to the SCV of the arrival process from $\mathcal{R}$ to i in the original network. This is because when $\hat{\rho}_{i^*} \approx 1$, the SCV of the departure process from i^* to i (or equivalently from $\mathcal{R}^*$ to i) is approximately the SCV of the service distribution of i^*. More specifically, in the reduced network,

$$\hat{c}^2_{a\mathcal{R}^*,i} = \frac{1}{\hat{\lambda}_{\mathcal{R}^*} \cdot \hat{p}_{\mathcal{R}^*,i}} \left(\hat{\lambda}_{i^*} \cdot \hat{c}^2_{di^*,ai}\right) = \frac{\hat{\lambda}_{i^*} \cdot \hat{c}^2_{di^*}}{\hat{\lambda}_{\mathcal{R}^*} \cdot \hat{p}_{\mathcal{R}^*,i}} \approx \frac{\hat{\lambda}_{i^*} \cdot \hat{c}^2_{si^*}}{\hat{\lambda}_{\mathcal{R}^*} \cdot \hat{p}_{\mathcal{R}^*,i}} = \hat{c}^2_{si^*} = c^2_{a\mathcal{R},i}.$$

The first equality is (8) applied to the reduced network, with the summation applied to nodes in $\mathcal{R}^*$, and using $\hat{p}_{j^*,i} = \delta_{j,i}$. The next equality follows since $\hat{c}^2_{di^*,ai} = \hat{c}^2_{di^*}$ from (3), since $\hat{p}_{i^*,i} = 1$. The next approximation follows since $\hat{c}^2_{di^*} \approx \hat{c}^2_{si^*}$ from (2), assuming that $\hat{\rho}_{i^*} \approx 1$. The next equality follows since $\hat{p}_{\mathcal{R}^*,i} = \hat{\lambda}_{i^*}/\hat{\lambda}_{\mathcal{R}^*}$, by definition, using (7) applied to the reduced network. The last equality substitutes the assigned value $\hat{c}^2_{si^*} \equiv c^2_{a\mathcal{R},i}$ from Reduction 2, step 4.

We note that both reductions require solving a Jackson network exactly. Thus, there is no benefit in using the reductions to analyze a Jackson network (where the service distributions are exponential). The benefit occurs when the service distributions are general and analytical solutions are not available. In this case, the simulation time for the reduced network is less than the simulation time for the original network. The analytical computations required in these reductions is not typically problematic. For example, we have used MVA

to solve a 500-node closed Jackson network in about 23 seconds on a 2 GHz PC. While the reductions require the analytical solution of a possibly large Jackson network, they eliminate the requirement to simulate a large network.

For the purpose of reproducibility, the supplement for this article [Shortle et al. 2009] gives a small example network and shows the results of the associated network reductions.

5. EXPERIMENTAL RESULTS USING ANALYTICAL METHODS

In this section, we test Reduction 1 on a large number of networks. To do this, we create an automated process that generates random networks and applies the reduction to each network.

While there are many ways to create random networks, we seek a method yielding something similar in structure to the existing national network of airports. In particular, we want networks that have a relatively small number of hub nodes and a relatively large number of feeder nodes. We can generate such networks based on the theory of *scale-free networks* (for an introduction to these networks, see Barabási and Bonabeau [2003]).

Scale-free networks arise from the following two principles describing how a network grows over time (e.g., Albert and Barabási [2002]):

(1) *Growth*. The network grows by the successive addition of nodes to the network. (This is different than starting with a fixed number of nodes and then growing the network by the successive addition of edges.) Edges are added to the network only when connecting a new node to existing nodes.
(2) *Preferential attachment*. When adding a new node to the network, it is preferable to connect it node to nodes that already connect to a large number of existing nodes.

We apply these principles using the following algorithm (adapted from Albert and Barabási [2002]):

Algorithm B.

Inputs: N (number of nodes) and k ($1 \leq k \leq N-1$), a connectivity factor controlling the sparseness of the network (higher values of k yield more edges in the network). Output: A network of N nodes and their connecting edges. Edges are undirected.

(1) Initialization. Start with an initial network of k nodes, where each node is connected via an edge to every other node except itself.
(2) For $i = k+1, k+2, \ldots, N$:
    (a) Growth. Add a new node i to the network. Node i is connected to the existing network as follows:
    (b) Preferential attachment. Randomly connect the new node to k nodes from the existing set of nodes in the network. The probability of connecting to a given node is proportional to the degree of that node.

Algorithm B generates a set of nodes and edges, but does not specify transition probabilities or service rates, nor does it apply the approximations described previously. The following algorithm describes a complete procedure for doing this.

Algorithm C.

Inputs: N (number of nodes), k (connectivity parameter), M (number of customers in network), B (maximum number of nodes in $\mathcal{C}$).

(1) Generate a random network (nodes and edges) using Algorithm B.
(2) For each node i, assign transition probabilities (from i) uniformly to those nodes connected to i. Specifically, let d_i be the degree of node i. Let $b_1, b_2, \ldots, b_{d_i}$ be the indices of the nodes connected to i. Randomly generate $(d_i - 1)$ independently identically distributed uniform variables on [0, 1]. Sort these random variables and call them $a_1 < a_2 < \cdots < a_{d_i-1}$. Define $a_0 \equiv 0$ and $a_{d_i} \equiv 1$. Let the transition probability from i to b_j be $a_j - a_{j-1}$.
(3) Specify the service distribution at each node. (Details on this step will be discussed later.)
(4) Choose the nodes in $\mathcal{C}$. First determine N_C, the number of nodes in $\mathcal{C}$, drawn uniformly from $\{1, 2, \ldots, B \leq N-2\}$. Then randomly choose N_C nodes, where each node has an equal probability of being selected.
(5) Create a reduced network.
(6) Compare performance metrics between the original and reduced networks.

By repeating Algorithm C many times, we can test the performance of the reductions over a large number of networks. In the following examples, we assume that all nodes have exponential service and we apply Reduction 1. Under this setup, it is possible to test the accuracy of the network reductions exactly, using MVA.

Effects of Server Utilization. Figure 4 shows an example in which we generate 500 random networks. The original networks all have $N = 12$ nodes, $M = 120$ customers, and service rates $\mu_i = 1$. The connectivity factor is $k = 5$ and the maximum number of core nodes is $B = 10$. Each point in the figure represents one node in $\mathcal{C}$ from one of the 500 randomly generated networks.[6] The y-axis specifies the relative error in W_q for a particular node ($|\hat{W}_q - W_q|/W_q$). The x-axis specifies the traffic intensity, ρ, of that node in the original network. For low values of ρ, the relative error is quite small. However, the relative error increases substantially when ρ is close to 1.

Effect of Number of Customers. Figure 5 shows a similar set of experiments in which we vary the number of customers in the network. The input parameters are $N = 30$, $k = 2$, $B = 28$, and service rates $\mu_i = 1$. We vary the number of customers: $M = 10$, $M = 100$, $M = 1000$. For each value of M, we generate 75 random networks. Although the figure contains a large number of points, one can see roughly three bands, corresponding to the cases $M = 10$, 100, and 1000. As the number of customers increases, the accuracy of the approximation improves.

This can be explained as follows:

[6]For a given network, the number of nodes in $\mathcal{C}$ is randomly chosen from $\{1, 2, \ldots, 10\}$, with an average value of 5.5. Thus, the figure contains approximately $500 \cdot 5.5 = 2750$ points.

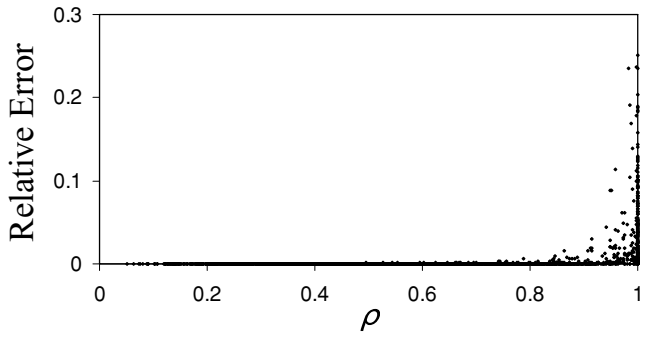

Fig. 4. Sample results for Reduction 1.

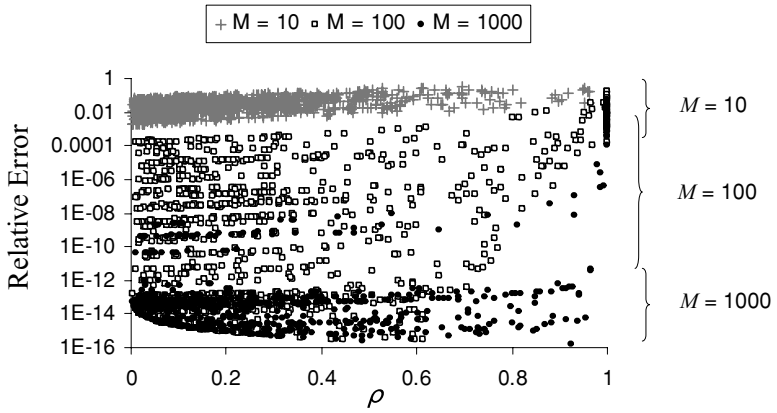

Fig. 5. Effects of varying the number of customers.

As the number of customers increases, the closed network behaves more like an open network, because the node with the highest congestion effectively acts like an infinite source and sink for the remaining nodes. For open Jackson networks, the performance metrics at a node i depend only on the service rate, μ_i, and the throughput, λ_i, which are approximately the same for the original and reduced networks.

Effect of Core Size. Figure 6 shows the dependence of relative error on the number of nodes in $\mathcal{C}$. In this example, we generate 500 networks with parameters $N = 30$, $B = 28$, $k = 5$, and $M = 200$. For a given node, the x-axis is the number of nodes, N_C, in $\mathcal{C}$, for a particular node within one of the 500 networks. The figure shows results on both a standard scale and a logarithmic scale. The logarithmic scale (on the right) does not reveal any noticeable trend in accuracy as a function of N_C. On the other hand, the linear scale shows improving accuracy as N_C increases (at least for $N_C \geq 6$ or so).[7] In other words, the worst-case

[7]Each point in the figure corresponds to one node in $\mathcal{C}$ in one of the 500 networks. Thus, networks with smaller N_C yield fewer points in the figure. The fact that the lowest N_C do not yield the worst-case accuracy values is probably an artifact of not enough data points in this region.

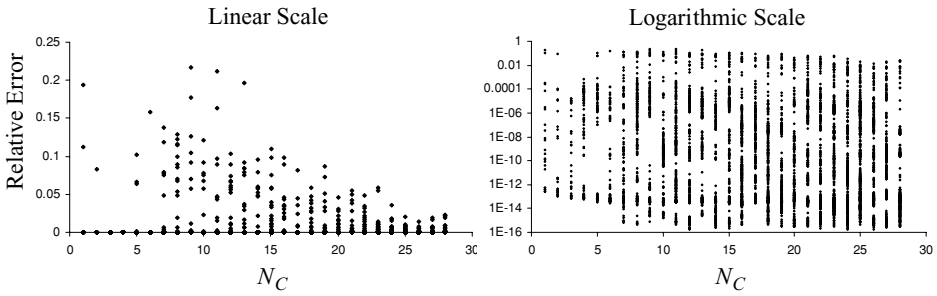

Fig. 6. Network reduction as a function of N_C, the number of nodes in $\mathcal{C}$, using Reduction 1.

nodes are improving as N_C increases, but the typical nodes (that is, near the median in accuracy) do not show a noticeable trend. We expect some improvement for larger N_C, since the approximation eliminates fewer nodes from the original network. However, this trend is only visible for the extreme points.

6. EXPERIMENTAL RESULTS USING SIMULATION

This section considers queueing networks with general service distributions. Since exact analytical solutions are not available, we use simulation to assess the accuracy of the network reductions. As in the previous section, we generate a set of random networks, apply a reduction to each network, and then assess the results collectively. We generate and analyze each network using Algorithm C, but we now use simulation in step 6, rather than exact formulas.

Comparison of Reductions 1 and 2. Figure 7 shows a comparison of Reductions 1 and 2. For each graph in the figure, we generate 200 random networks with parameters $N = 12$, $M = 120$, $k = 2$, and $B = 5$. The graphs are differentiated by the SCV of the service distribution within each network. The service distribution at each node is a Weibull distribution with a mean of 1 and SCV = 1.0, 0.5, 0.25, or 0.111, as specified. The y-axis is $|\hat{W}_q - W_q|/W_q$, where the expected queue waiting times are estimated using simulation. For each network, we run 20 simulation replications, where each replication has a run length of 50,000 time units and a warm-up length of 1,000 time units. The lengths of 95% confidence intervals for W_q and $\hat{W}_q$ are less than 0.01 (1% of the average service time) for node utilizations less than 0.8, though the confidence intervals can grow significantly as the utilization approaches 1.

The figure shows two trends. First, the accuracy degrades as the service distribution SCV becomes smaller, or as the service distribution becomes more deterministic. Second, Reduction 2 generally performs better than Reduction 1, particularly for smaller SCV values in the service distribution.

Table II shows the median and average simulation speed-up for the 200 networks (where speed-up is defined to be the time to simulate the original network divided by the time to simulate the reduced network). Reduction 1 is slightly faster than Reduction 2. This is expected because the reduction involves fewer nodes. Also, there is no noticeable trend in the speed-up factor as a function of the SCV of the service distribution. Finally, the average speed-up

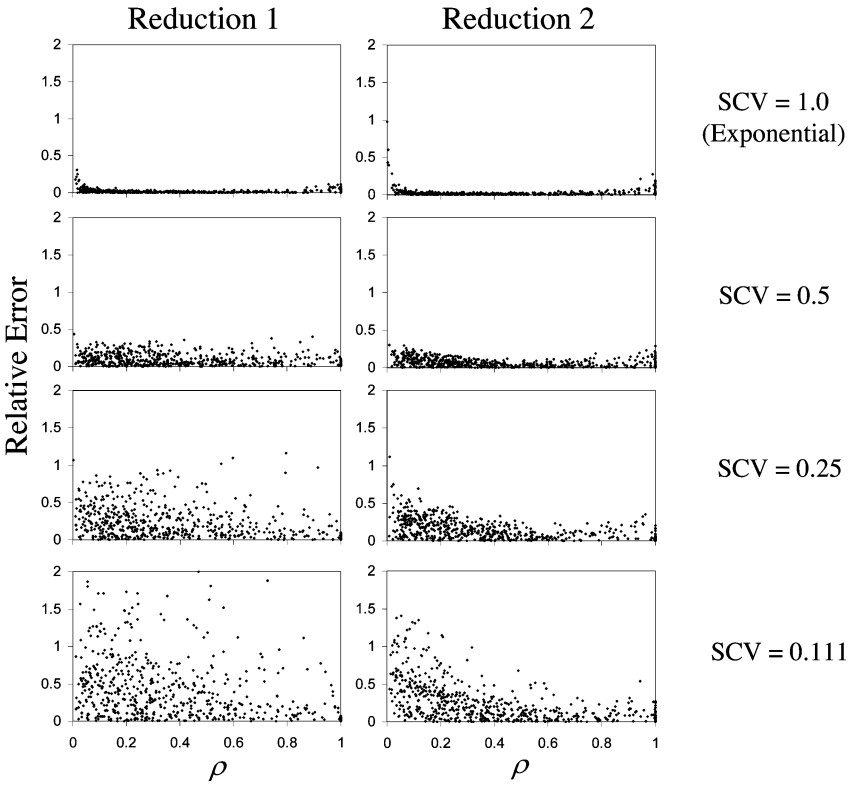

Fig. 7. Effects of varying the squared coefficient of variation for service distributions.

Table II. Simulation Speed-Up for Sample Networks

	Reduction 1		Reduction 2	
SCV	Median	Average	Median	Average
1.0	2.8	6.8	2.2	5.2
0.5	2.9	6.3	2.5	4.9
0.25	2.8	8.2	2.6	4.5
0.111	3.1	7.2	2.5	6.7

is much larger than the median. This is due to a small number of networks with extremely large speed-up factors. The remaining experiments all involve Reduction 2.

Other Service Distributions. The previous example assumed Weibull service distributions for all nodes. This example (Figure 8) shows the effect of varying the family of service distributions. There are four sets of experiments. In the first set, all service distributions are chosen from the Weibull family, as before. The second and third sets are similar, but with gamma and lognormal families replacing the Weibull family. In the last set, the service distribution for each node is randomly chosen from the Weibull, gamma, and lognormal families, with equal probability. In addition, the mean and SCV of the service distribution for each node is randomly chosen according to: $1/\mu_i \sim \text{Unif}[0.5, 2]$,

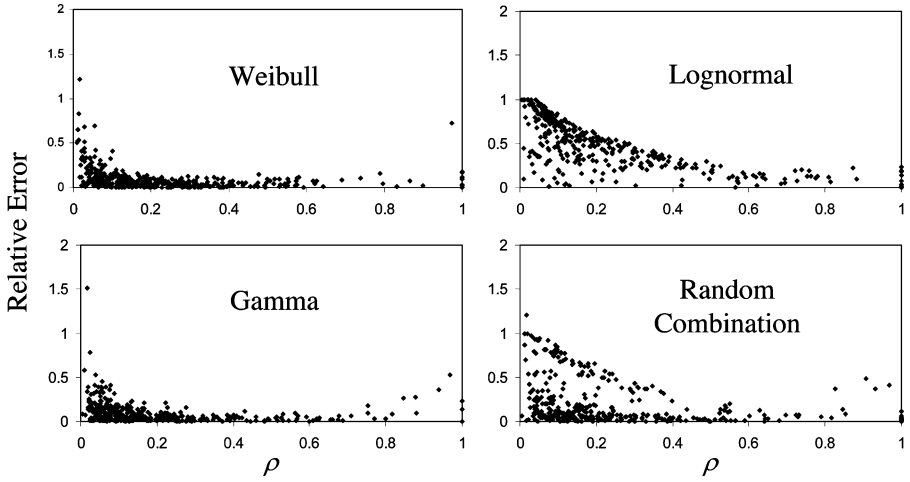

Fig. 8. Effects of varying the family of service distributions.

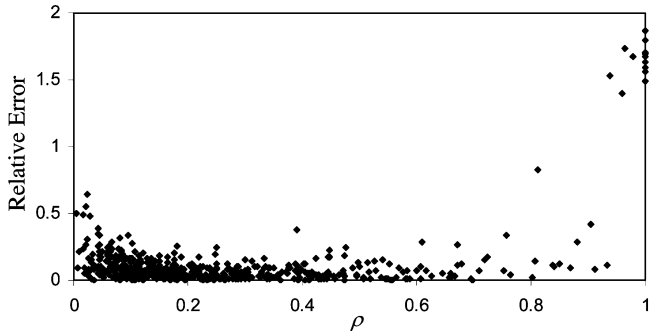

Fig. 9. Service distributions with SCVs greater than 1.

$c_{si}^2 \sim$ Unif[0.5, 0.75]. The other parameters are: 70 networks in each graph, $N = 50$, $M = 500$, $k = 3$, $B = 10$, simulation time 100,000, warmup time 10,000. The network reductions with the Weibull and gamma families show similar results. The lognormal family results in worse accuracy. This may be because the lognormal is a heavy-tailed distribution. The mixed case inherits a mixture of results from the specialized cases.

High SCV. Figure 9 shows an example in which the service SCVs are greater than 1. For each node, the service SCV is randomly chosen between 1 and 2.5 ($c_{si}^2 \sim$ Unif[1, 2.5]). The other parameters are: 100 networks, $N = 30$, $M = 200$, $k = 3$, $B = 10$, service family is Weibull, $1/\mu_i \sim$ Unif[0.5, 2], simulation time 100,000, warmup time 10,000. The figure shows that although the results are not as good as with exponential service, the accuracy does not suffer from the same kinds of problems as networks that have very low SCVs.

Variable SCV. Figure 10 shows an example in which the service SCV is randomly chosen over a wide range ($c_{si}^2 \sim$ Unif[.05, 1]). In comparison with

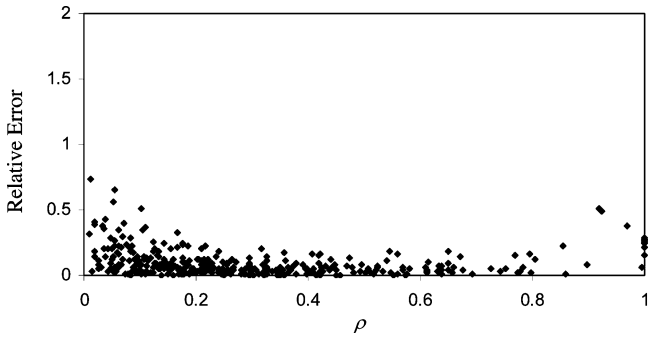

Fig. 10. Service distributions with SCVs between .05 and 1.

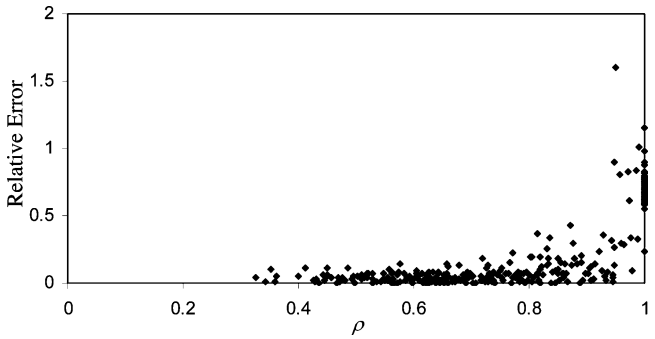

Fig. 11. Core nodes in $\mathcal{C}$ have high utilizations.

Figure 7, the results are closest to the SCV = 0.5 case, not the SCV = 0.111 case. In other words, having all low service SCVs can lead to very poor results, but having some low service SCV's does not seem to lead to the same poor results. The other parameters for this example are: 100 networks, $N = 30$, $M = 400$, $k = 3$, $B = 5$, service distribution is Weibull, $1/\mu_i \sim$ Unif[0.5, 2], simulation time 100,000, warmup time 10,000.

Small $\mathcal{C}$. Figure 11 investigates cases where $\mathcal{C}$ is relatively small and contains high utilization nodes. The number of core nodes, N_C, is chosen randomly between 2 and 5, where the total number of nodes is $N = 50$. The actual nodes in $\mathcal{C}$ are chosen to be the nodes with the highest utilizations. The other experimental parameters are: 100 networks, $M = 300$, $k = 3$, service family is Weibull, $1/\mu_i \sim$ Unif[0.5, 2], $c_{si^2} \sim$ Unif[0.5, 1], simulation time 100,000, warmup time 10,000. The example shows that the performance is not significantly different than other results seen earlier. The median speedup for this example is 9.8. This speedup factor is higher than results seen in Figure 7. The reason is that this example has a smaller number of nodes in $\mathcal{C}$ and a higher number of nodes in $\mathcal{R}$, even though the nodes in $\mathcal{C}$ are more congested in this example.

In general, we have not observed any noticeable effect due to the bottleneck node (the node with the highest utilization) being located in $\mathcal{C}$ or in $\mathcal{R}$. The key predictor in accuracy at a given node seems to be the utilization at that node,

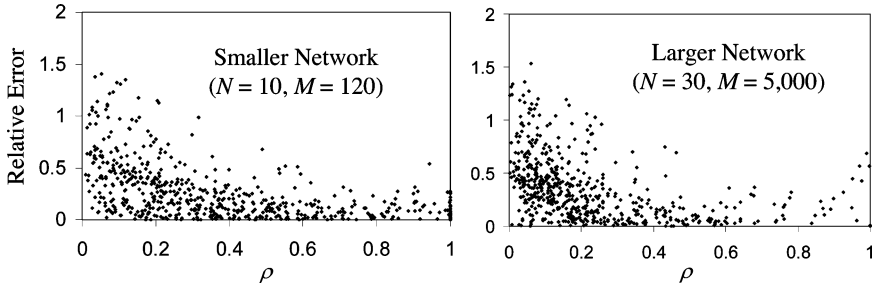

Fig. 12. Comparison of smaller and larger networks.

Table III. Median Relative Error for 200 Sample Networks

SCV	Reduction 2	Parametric Decomposition
1.0	0.01	0.02
0.5	0.06	0.11
0.25	0.12	0.23
0.111	0.17	0.42

regardless of whether the bottleneck node is in $\mathcal{C}$ or in $\mathcal{R}$. This can be partially observed in Figure 11, since the bottleneck node is in $\mathcal{C}$ for every network in this example, yet the results, as a function of ρ, are not much different than previous results, where the bottleneck node is usually not in $\mathcal{C}$.

Larger and Smaller Networks. This example compares larger and smaller networks. The larger networks have $N = 30$ nodes and $M = 5,000$ customers, while the smaller networks have $N = 10$ nodes and $M = 120$ customers. Common parameters are: 200 networks, $k = 2$, $B = 5$, Weibull service families, $\mu_i = 1$, 20 replications, simulation time 10,000,000, warmup time 100,000. Figure 12 shows a sample comparison, where the service SCV is 0.111. The figure reveals no discernable difference. Cases with other SCVs show essentially the same thing. (One exception is the case, SCV = 1.0, which is much more accurate for the larger networks, consistent with the analytical results given previously in Figure 5.) The median speedup for the larger networks is 6.8 compared to 2.5 for the smaller networks. As expected, the reductions are more efficient for larger networks. The lengths of 95% confidence intervals for W_q and $\hat{W}_q$ are less than 0.0001 for node utilizations less than 0.8 and less than 0.01 for node utilizations less than about 0.95.

Comparison with Parametric Decomposition. This example compares Reduction 2 with the parametric decomposition using (1)–(4). Although both methods use (1)–(4), a key difference is that Reduction 2 maintains the network interaction between nodes in the core set $\mathcal{C}$, whereas the decomposition approach models the nodes as isolated queues. Table III shows sample results comparing the median relative error using both methods. As expected, Reduction 2 is more accurate. However, the decomposition approximation does not require simulation, so it can be computed almost instantly.

The parameters used in this example are: 200 networks, $N = 12$, $M = 120$, $k = 2$, $B = 5$, Weibull service distributions, fixed c_{si}^2 as shown in the Table, 20 simulation replications, simulation time 50,000, warmup time 1,000.

The parametric decomposition method is implemented as follows:

The closed network is converted to an open network, as described in Whitt [1984]. The first-moment parameters, λ_j, are obtained by solving standard flow balance equations. The second-moment parameters, c_{aj}^2, are estimated by solving the linear system of equations in (1). Finally, the waiting time at an individual node j, is estimated using the derived values λ_j and c_{aj}^2, as input to standard approximation formulas for the $GI/G/1$ queue [Whitt 1983b].

Asymptotic Analysis. We now discuss the speedup magnitudes that are possible using these reductions. Both reductions try to replicate transitions from $\mathcal{C} \to \mathcal{C}$, $\mathcal{C} \to \mathcal{R}$, and $\mathcal{R} \to \mathcal{C}$, while eliminating transitions from $\mathcal{R} \to \mathcal{R}$. Assuming the simulation time is linear in the number of transitions, the speedup factor is roughly:

$$\frac{\text{Rate of all transitions in network}}{\text{Rate of transitions from } \mathcal{C} \to \mathcal{C}, \mathcal{C} \to \mathcal{R}, \text{ and } \mathcal{R} \to \mathcal{C}}.$$

If we also assume that the rate of transitions out of a node is proportional to the degree of the node, then the speedup factor is at least:

$$\frac{\text{Total degree of all nodes in the network}}{2 \times (\text{Total degree of nodes in } \mathcal{C})}.$$

(The denominator counts transitions from $\mathcal{C} \to \mathcal{R}$, $\mathcal{R} \to \mathcal{C}$, and double counts transitions from $\mathcal{C} \to \mathcal{C}$.)

To complete a rough asymptotic analysis, we assume that a network grows in the scale-free manner described in Section 5. Albert and Barabási [2002] have shown that the proportion, p_j, of nodes with degree j, approximately follows a power law, $p_j \approx Cj^{-\alpha}$, for some constant C. As an approximation, we replace the discrete power law with a continuous density function $f(x) \approx Cx^{-\alpha}$. In the worst case, the nodes in $\mathcal{C}$ are the nodes with the highest degree. Then for some value K, the fraction of nodes in $\mathcal{C}$ is:

$$\frac{N_C}{N} \approx \int_K^\infty Cx^{-\alpha} dx \propto K^{-(\alpha-1)}.$$

Thus, $K \propto (N/N_C)^{1/(\alpha-1)}$. The total degree of nodes in $\mathcal{C}$ is roughly:

$$\int_K^\infty xN(Cx^{-\alpha})\, dx \propto NK^{-(\alpha-2)} \propto N\left(\frac{N}{N_C}\right)^{-(\alpha-2)/(\alpha-1)}.$$

Since the total degree of all nodes in the network scales as N, the speedup factor scales as $(N/N_C)^{(\alpha-2)/(\alpha-1)}$. Albert and Barabási [2002] have found from simulation experiments that the parameter α is about 3, regardless of the connectivity factor used to grow the network. So if N_C is fixed, the speedup scales with $\sqrt{N}$. If $N_C/N = N^{-\beta}$, the speedup scales with $N^{\beta/2}$. More generally, the speedup may be greater, since we have made pessimistic assumptions throughout the analysis. Further, if the network grows in a manner that is not as

favorable towards high-degree connections, the speedup factor will be greater, since the network growth is more likely to add connections within $\mathcal{R}$.

7. CONCLUSIONS

This article gave several methods for reducing a closed queueing network to a smaller one. The networks considered in this article were closed Jackson-like networks with Markovian routing and general service distributions. The basic idea was to divide the network into two parts: the core nodes of interest, $\mathcal{C}$, and the remaining nodes, $\mathcal{R}$. One reduction collapsed all nodes in $\mathcal{R}$ into a single node. Another collapsed $\mathcal{R}$ into a small set of nodes $\mathcal{R}^*$. The parameters in $\mathcal{R}^*$ were chosen to approximately match input flows into $\mathcal{C}$, compared with the original network.

By randomly generating sample test networks, we tested the reductions on hundreds of networks, rather than on only a few specific cases. The networks were evaluated based on the relative accuracy in estimating the mean wait in queue W_q at each node. The following types of networks were tested in this article. Networks were generated randomly using scale-free algorithms, tending to yield networks with a small number of highly connected nodes and a large number of nodes with low degree. All nodes had a single server. Weibull, gamma, lognormal, and exponential service distributions were considered. Mean service rates varied from 0.5 to 2, and service SCVs varied from 0.05 to 2.5. Network sizes ranged from 12 to 50 nodes, with customer sizes ranging from 120 to 5,000. The experiments did not cover every combination of all variables, but attempted to provide a representative sample.

In the experiments conducted, the network reductions yielded poor approximations for nodes with utilizations near 1 or near 0 (the problems near 0 were partially related to the denominator of the relative error being small), or when the SCVs of all service distributions were low (less than 0.25), or when some of the service distributions were lognormal. The network reductions yielded modestly good results (relative errors less than about 20%) in other cases when service-distribution SCVs were not all less than 0.25, when service distributions were Weibull or gamma, and when node utilizations were moderate (between 0.2 and 0.8). Excellent results were achieved with exponential service, though such networks do not need simulation to analyze in the first place. Factors that did not influence the accuracy of the results were the size of the core set $\mathcal{C}$, the size of the overall network, or whether or not the bottleneck node was in $\mathcal{C}$ or $\mathcal{R}$.

The speedup was primarily related to the number of transitions within $\mathcal{R}$ that were eliminated by using the reduced network. Networks with large sets, $\mathcal{R}$, and small sets, $\mathcal{C}$, tended to have large speedup factors. All experiments confirmed a trend that the speedup is inversely related to the number of nodes in $\mathcal{C}$, for a fixed total number of nodes. Median speedups observed ranged from about 2 to 20.

There are multiple avenues for future work based on this research. One is to investigate more complicated subnetwork structures for $\mathcal{R}^*$ as a way to approximate network dynamics outside the core nodes. Another is to analyze networks

with more complex routing logic—for example, networks with scheduled departures. In such networks, simulation within the core set of nodes, $\mathcal{C}$, becomes more critical.

ACKNOWLEDGMENTS

The authors appreciate the work of the editors and referees in carefully reading the article and in making suggestions for improvements.

REFERENCES

ALBERT, R. AND BARABÁSI, A. L. 2002. Statistical mechanics of complex networks. *Rev. Mod. Physics 74,* 1, 47–97.

BARABÁSI, A. L. AND BONABEAU, E. 2003. Scale-free networks. *Sci. Amer. 288,* 5, 60–69.

BITRAN, G. AND TIRUPATI, D. 1988. Multiproduct queueing networks with deterministic routing: Decomposition approach and the notion of interference. *Manag. Sci. 34*, 75–100.

BOLCH, G., GREINER, S., DE MEER, H., AND TRIVEDI, K. S. 2006. *Queueing Networks and Markov Chains*. Wiley, NJ.

BOUCHERIE, R. J. AND STEWART, M. 1998. Norton's equivalent for batch routing queueing networks with independently routing customers. *Stochastic Models 14,* 5, 1091–1112.

BOUCHERIE, R. J. AND VAN DIJK, N. M. 1993. A generalization of Norton's theorem for queueing networks. *Queueing Syst. 13*, 251–289.

CHANDY, K. M. AND GEORGANAS, N. D. 1975. Decomposition and aggregation by class in closed queueing networks. *IEEE Trans. Softw. Eng. 19*, 36–42.

CHANDY, K. M., HERZOG, U., AND WOO, L. 1975. Parametric analysis of queueing networks. *IBM J. Res. Devel. 19*, 43–49.

COWIE, J., LIU, H., LIU, J., NICOL, D., AND OGIELSKI, A. 1999. Towards realistic million-node internet simulations. In *Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications*.

CURTOIS, P. 1975. Decomposability, instabilities and saturation in multi-programming systems. *Commun. ACM 18*, 371–377.

CURTOIS, P. 1977. *Decomposability Queueing and Computer Science Applications*. ACM monograph series, Academic Press, New York.

DAI, J. G. AND HARRISON, J. M. 1992. Reflected Brownian motion in an orthant: numerical methods for steady-state analysis. *Ann. Appl. Prob. 2*, 65–86.

DAI, J. G. AND HARRISON, J. M. 1993. The QNET method for two-moment analysis of closed manufacturing systems. *Ann. Appl. Prob. 3*, 968–1012.

DAI, J. G., NGUYEN, V., AND REIMAN, M. I. 1994. Sequential bottleneck decomposition: An approximation method for generalized Jackson networks. *Oper. Res. 42*, 119–136.

DAI, J. G., YEH, D. H., AND ZHOU, C. 1997. The QNET method for re-entrant queueing networks with priority disciplines. *Oper. Res. 45,* 4, 610–623.

GELENBE, E. AND PUJOLLE, G. 1998. *Introduction to Queueing Networks*, 2nd ed. Wiley, New York.

GROSS, D. AND HARRIS, C. 1998. *Fundamentals of Queueing Theory*, 3rd ed. Wiley, New York.

HARRISON, J. M. AND NGUYEN, V. 1990. The QNET method for two-moment analysis of open queueing networks. *Queue. Syst. Theory Appl. 6*, 1–32.

KIM, S. 2004. The heavy-traffic bottleneck phenomenon under splitting and superposition. *Europ. J. Oper. Res. 157*, 736–745.

KUEHN, P. J. 1979. Approximate analysis of general queueing networks by decomposition. *IEEE Trans. Commun. 27*, 113–126.

LIU, B., GUO, Y., KUROSE, J., TOWSLEY, D., AND GONG, W. 1999. Fluid simulation of large scale networks: Issues and tradeoffs. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*.

LIU, Y., PRESTI, F. L., MISRA, V., TOWSLEY, D., AND GU, Y. 2003. Fluid models and solutions for large-scale IP networks. In *Proceedings of ACM Sigmetrics*.

NICOL, D. M., LILJENSTAM, M., AND LIU, J. 2005. Advanced concepts in large-scale network simulation. In *Proceedings of the 2005 Winter Simulation Conference*. IEEE, Piscataway, NJ, 153–166.

NICOL, D. M., LIU, J., LILJENSTAM, M., AND YAN, G. 2003. Simulation of large-scale networks using SSF. In *Proceedings of the 2003 Winter Simulation Conference*. IEEE, Piscataway, NJ, 650–657.

RAO, D. M. AND WILSEY, P. A. 1999. Simulation of ultra-large communication networks. In *Proceedings of the International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*.

REIMAN, M. 1984. Open queueing networks in heavy traffic. *Math. Oper. Res. 9,* 3, 441–458.

REIMAN, M. I. 1990. Asymptotically exact decomposition approximations for open queueing networks. *Oper. Res. Lett. 9*, 363–370.

RILEY, G. AND AMMAR, M. H. 2002. Simulating large networks—how big is big enough? In *Proceedings of the First International Conference on Grand Challenges for Modeling and Simulation*.

RILEY, G. F. 2003. Large-scale network simulations with GTNetS. In *Proceedings of the 2003 Winter Simulation Conference*. IEEE, Piscataway, NJ, 676–684.

SHORTLE, J. F., MARK, B. L., AND GROSS, D. 2009. Supplement for Reduction of closed queueing networks for efficient simulation. *Trans. Model. Comput. Simul. 19*, 3. Online supplement.

SURESH, S. AND WHITT, W. 1990. The heavy-traffic bottleneck phenomenon in open queueing networks. *Oper. Res. Lett. 9*, 355–362.

WHITT, W. 1982. Approximating a point process by a renewal process, I: Two basic methods. *Oper. Res. 30*, 125–147.

WHITT, W. 1983a. Performance of the QNA. *Bell Syst. Tech. J. 62,* 9, 2816–2843.

WHITT, W. 1983b. The queueing network analyzer. *Bell Syst. Tech. J. 62,* 9, 2779–2815.

WHITT, W. 1984. Open and closed models for networks of queues. *AT&T Bell Lab. Tech. J. 63,* 9, 1911–1979.

WHITT, W. 1995. Variability functions for parametric-decomposition approximations of queueing networks. *Manag. Sci. 41*, 1704–1715.

YOUSEFI, A., DONOHUE, G. L., AND QURESHI, K. M. 2003. Investigation of en-route metrics for model validation and airspace design using the Total Airport and Airspace Modeler (TAAM). In *Proceedings of the 5th EUROCONTROL/FAA ATM R&D Conference*. Budapest, Hungary.

Received February 2007; revised March 2008, June 2008, October 2008; accepted October 2008