A Fixed-Point Algorithm for Closed Queueing Networks

Ramin Sadre, Boudewijn R. Haverkort, Patrick Reinelt

University of Twente

Dept. Electrical Engineering, Mathematics and Computer Science

P.O. Box 217, 7500 AE Enschede, the Netherlands

`r.sadre@cs.utwente.nl`, `brh@cs.utwente.nl`

Abstract. In this paper we propose a new efficient iterative scheme for solving closed queueing networks with phase-type service time distributions. The method is especially efficient and accurate in case of large numbers of nodes and large customer populations. We present the method, put it in perspective, and validate it through a large number of test scenarios. In most cases, the method provides accuracies within 5% relative error (in comparison to discrete-event simulation).

1 Introduction

Queueing networks (QNs) have been used widely since the early 1970's for the analysis of performance problems in computer and communication systems. For many classes of queueing networks elegant and efficient solution methods exist.

In case the QNs under study are open ("OQNs") and contain queueing stations with infinite capacity, i.e., when the number of customers is not *a priori* restricted, product-form results exist, such as those for Jackson networks [17]. A disadvantage of these results is that they are only valid under a number of restrictions: the service times need to be exponentially distributed when combined with FCFS scheduling, the stations have unbounded buffer capacity, and all arrival processes are Poissonian. These restrictions have led researchers to search for extensions and approximations.

Queueing network models with either finite customer number or with finite buffers, and, hence, with customer losses, can be analyzed via the numerical solution of the underlying CTMC. However, this method is sensitive to the well-known phenomenon called state-space explosion. One way to handle this problem for open queueing networks is a *decomposition approach*. It has been motivated by the approximate solution method of large open queueing networks with infinite-buffer stations and FCFS scheduling, as proposed by Kühn [19] and later extended by Whitt [31, 32]. The decomposition is done at queueing station level, i.e., the queueing stations are analyzed as separate models. These methods have been extended and refined lately in the context of the tool FiFiQueues. During the analysis, traffic descriptors are "exchanged between the stations", thus representing the streams of jobs flowing between them. We will elaborate on this in Section 2.

In case the QNs under study are closed (“CQNs”), i.e., when a finite fixed population of customers is present in the network, and when some other restrictions apply, Gordon and Newell first described a product-form for closed queueing networks [12], which was later extended by Baskett *et al.* to the now well-known class of BCMP networks [2]. Buzen developed an elegant solution strategy to compute the normalizing constant [8], and later, using the arrival theorem, Reiser and Lavenberg developed the now-widely used mean-value analysis approach [24]. Various extensions to these algorithms and model class have been developed, cf. textbooks like [10]. Apart from a number of modeling restrictions, such as negative exponential service times in combination with FCFS scheduling, all of the developed algorithms suffer from increasing (above linear) complexity when the number of stations, the number of customers, or the number of model classes (or routing chains) grow.

It is for the above reasons, that we have sought to come up with an alternative method for analyzing large closed queueing networks. Although little work has been reported on this so far, we found some of our inspiration in the fixed-point approach developed by Bolch *et al.* [6] (as also described in [16, Chapter 11.5]). Our approach consists of elevating the fixed-point algorithms that have been developed and successfully applied for open queueing networks to closed queueing networks. In doing so, we have encountered a number of problems, that we, however, have been able to deal with, after having experimented with the new method. In comparison to other approaches, our work is more generally applicable, and also less costly than previously reported approaches. We will discuss related work in a separate section.

The rest of this paper is structured in the following way. Section 2 is devoted to a fixed-point method for open queueing networks, as this approach forms the basis of our new method for closed queueing networks, that is described in Section 3. After that, we report experimental results on a variety of networks in Section 4. Section 5 presents directly related work, whereas Section 6 concludes the paper.

2 Fixed-point analysis of OQNs

Fixed-point iteration methods have been employed successfully to evaluate large open queueing networks with non-Poissonian arrivals and non-exponential service time distributions, with or without job losses (bounded buffers). The idea has been to compute, iteratively, the traffic arriving at each queueing station in such a queueing network, such that individual queueing stations can, in essence, be analyzed in isolation [13–15, 30]. The main algorithm is outlined in Figure 1. The traffic from station i to station j in the queueing network is described by a traffic descriptor $desc_{i,j}$. Note that we do, at this point, not make the form of this traffic descriptor explicit; in practice, it will contain such quantities as the traffic rate and, possibly, the variance. The external traffic arriving at a station j is denoted as $desc_{ext,j}$. In each step, a new set of traffic descriptors $desc^{(l)} = \{desc_{i,j}^{(l)} | i, j\}$ is computed. The algorithm stops when the

```

1 initialize all traffic descriptors  $desc_{i,j}^{(0)}$ :
2   set  $desc_{i,j}^{(0)}$  to the null value if  $i \neq ext$ 
3   set  $desc_{i,j}^{(0)}$  to the specified value if  $i = ext$ 
4  $l := 0$ 
5 do
6    $l := l + 1$ 
7   analyze each queueing station  $i$ 
8   and compute  $desc_{i,j}^{(l)}$  for all nodes  $j$ 
9 while  $dist(desc^{(l)}, desc^{(l-1)}) > \varepsilon$ 

```

Fig. 1. Decomposition-based analysis procedure for open queueing networks

distance $dist(desc^{(l-1)}, desc^{(l)})$ ($l \geq 1$) between two successive sets of descriptors is smaller or equal than a given threshold ε . Descriptors set to the *null* value in line 2 are ignored in line 7; the *null* value indicates that only information about the external arriving traffic (line 3) is available when the algorithm starts. In general, it is not known whether a fixed point is unique or can/will be found. However, in our experiments with the FiFiQueues network analyzer the algorithm always terminated; furthermore, in [25] the existence of a fixed-point is proven.

The approach as described above, was developed in the mid 1990's [13–15, 30], essentially as an extension of Whitt's QNA approach [31] by replacing the core of his analysis: the analysis of the queueing stations themselves (the “service operation”). Unlike QNA, this new approach (called QNAUT) does not use the descriptor of the arrival traffic directly to compute the departure traffic descriptor, but assumes that the arrival traffic descriptor can be used to construct a phase-type (PH) renewal process which approximates the “real” underlying arrival process. This allows for the inclusion of finite-buffer queueing stations as well as for the analysis of the queueing stations by matrix-geometric and general Markovian techniques, instead of the approximations used originally in QNA.

Around the turn of the century, we extended the QNAUT-approach, in that we removed a few approximate steps and enhanced the model class [26, 28, 27]. This approach, as well as the analysis tool developed from it, is named *FiFiQueues* (Fixpoint-based analysis of networks with Finite Queues). In FiFiQueues an open queueing network model is specified by the following parameters:

1. The number of queueing stations n .
2. The description of each queueing station. The queueing stations can have finite or infinite capacity and are analyzed as PH|PH|1($|K|$) queues. The service processes can be arbitrary phase-type renewal processes. A PH|PH|1($|K|$) queue is analyzed by means of the CTMC underlying the corresponding Quasi-Birth-and-Death process.
3. A routing matrix $\mathbf{R} = (r_{i,j})$ of size $n \times n$ for the Markovian routing where $r_{i,j}$ specifies the routing probability from station i to station j .
4. The descriptors of the external arrival traffic for each station.

Fig. 2. CQN and the corresponding cut OQN

As in QNA, the external arrival processes as well as the inter-node traffic streams are described by the first and second moment of the inter-arrival times. The traffic descriptor $\langle \lambda, c_a^2 \rangle$ contains the arrival rate λ and the squared coefficient of variation c_a^2 of the inter-arrival time distribution. In order to obtain the arrival process for a PH|PH|1($|K$) station, a PH renewal process has to be fitted to the arrival traffic descriptor $\langle \lambda, c_a^2 \rangle$. Traffic descriptors with $c_a^2 \leq 1$ are mapped to modified Erlang-distributions. In case $c_a^2 > 1$, a hyper-exponential distribution with two phases and so-called balanced means is used. In the following sections, we use the same fitting procedure for the service processes, too, i.e., we specify a service process by the service rate μ and the squared coefficient of variation c_s^2 of the service time distribution.

Finally, FiFiQueues comprises two post-processing steps that are performed after the fixed-point iteration. They allow for the computation of additional performance measures and yield (i) node-specific results, e.g., the mean queue length $E[N_i]$ for each station i , and (ii) network-wide results, e.g., the total network throughput.

3 Fixed-point analysis of CQNs

We first describe in general terms an iterative approach for CQNs in Section 3.1. Before we make this approach more specific, we discuss the issue of bottleneck identification and its impact on performance measures in CQNs in Section 3.2. We then proceed with our actual algorithm in Section 3.3 and discuss complexity issues in Section 3.4.

3.1 General procedure

The decomposition approach for OQNs cannot be directly applied to CQNs because the bounded number of customers in a closed system prevents an intuitive decomposition. Hence, we transform a CQN into an OQN by cutting one of its connections. This is shown for an example network in Figure 2. For this OQN we have to find an external arrival traffic descriptor arr such that

1. the external arrival descriptor arr is equal to the (resulting) descriptor dep of the traffic that leaves the network;
2. the number of jobs in the network is equal to the fixed population q of the CQN.

```

1 cut CQN to obtain OQN
2 initialize arr
3 loop
4   analyze OQN and obtain departure dep
5   if  $err(arr, dep) > \delta_1$  or  $err'(\sum_{i=1}^n E[N_i], q) > \delta_2$  then
6     choose new arr based on the analysis results
7   else
8     stop iteration
9   endif
10 endloop

```

Fig. 3. Iterative procedure to solve CQNs

We aim to find *arr* by applying the iteration procedure shown in Figure 3 to the CQN. The functions *err* and *err'* are appropriate error functions and δ_1 resp. δ_2 the corresponding error bounds. To implement this procedure we have to address three issues:

1. the location of the cut in order to obtain an open network (line 1);
2. the analysis of the open queueing network (line 4);
3. the computation of a new arrival descriptor inside the iteration (line 6).

These issues are discussed in detail in Section 3.3 but we can already make the following observations:

- (Back) blocking at the queues is not allowed if we analyze the open queueing network by a decomposition-based method. This would require that information about free queueing capacities is exchanged between queues, which is not supported by the decomposition approach for OQN in which individual stations are analyzed in isolation. Hence, we will assume in the following that all queues have infinite capacity.
- Although the sketched procedure looks very simple, its implementation is critical for complex network classes and traffic descriptors. It is yet unknown whether the iteration procedure always terminates and whether more than one correct solution exist for a given CQN. However, in our experiments (see below) it always terminated with satisfying results.
- The stopping condition $err'(\sum_{i=1}^n E[N_i], q) \leq \delta_2$ provides only an approximation to the original condition that the number of jobs in the CQN is *q*. Indeed, variations in the number of customers present due to the stochastic nature of the arrival and service processes causes the number of jobs in the OQN to vary around *q*, which is clearly not the case in a true closed QN.

3.2 Characteristics of the bottleneck

Before we present the implementation of the analysis procedure for CQNs in detail in Section 3.3, we discuss some important characteristics of the so-called

Fig. 4. Example Gordon-Newell QN

bottleneck in a CQN. We will use results from bottleneck analysis in the further development of our algorithm.

The (relative) throughput of the queueing stations in a CQN is limited by the bottleneck which can be determined by solving the (first-order) traffic equations [16]:

$$V_j = \sum_{i=1}^n V_i r_{i,j} = V_1 r_{1,j} + \sum_{i=2}^n V_i r_{i,j} = r_{1,j} + \sum_{i=2}^n V_i r_{i,j}, \quad \text{with } V_1 = 1,$$

where the so-called visit ratios $V_j = X_j/X_1$ express the throughput of station j relative to node 1. The ratio $D_i = V_i/\mu_i$, for each station i , is the so-called service demand (per passage) at station i ; the bottleneck is the node i with the highest value of D_i .

The bottleneck does not only influence the throughput of the queueing stations but also their queue length distribution. We illustrate this with the CQN shown in Figure 4. It is a Gordon-Newell queueing network (GNQN), i.e., all stations are of M|M|1-type. The figure shows the routing probabilities and the service rates of each node. A quick computation reveals that $D_1 = \frac{2}{3}$, $D_2 = \frac{14}{25}$, $D_3 = \frac{1}{2}$, $D_4 = \frac{1}{2}$ and $D_5 = 1$. Clearly, station 5 is the bottleneck. Given a large population, we can expect a large number of customers to reside in station 5, always, so that its utilization will approach 100%. A (discrete-event) simulation of the network with population $q = 50$ yields for each station the utilization ρ (note that $\rho_i = D_i/D_5 = D_i$), the mean $E[N]$ and the squared coefficient of variation c_N^2 of the queue length distribution. The results (with relative 95%-confidence intervals smaller than 3%) are shown in the column titled “sim” of Table 1. The fact that station 5 is a rather distinct bottleneck, leads to a very deterministic queue length distribution for that station (its c_N^2 is very close to 0), i.e., almost all of the time, almost all jobs are waiting in the bottleneck queue.

3.3 CQN analysis with FiFiQueues

We now describe how the general iteration scheme for CQNs can be “implemented” using FiFiQueues (see Section 2) as analysis method for the generated OQNs. We have called the resulting analysis method *FiFiQueues Non-Blocking*

node		decomp	sim	relerr	node		decomp	sim	relerr
1	ρ	0.67	0.67	0.0%	4	ρ	0.50	0.50	0.0%
	$E[N]$	2.00	2.00	0.0%		$E[N]$	1.00	1.00	0.0%
	c_N^2	1.50	1.51	-0.7%		c_N^2	2.00	1.96	2.0%
2	ρ	0.56	0.56	0.0%	5	ρ	1.00	1.00	0.0%
	$E[N]$	1.27	1.27	0.0%		$E[N]$	44.7	44.7	0.0%
	c_N^2	1.79	1.80	0.6%		c_N^2	0.02	0.01	100%
3	ρ	0.50	0.50	0.0%					
	$E[N]$	1.00	1.00	0.0%					
	c_N^2	2.00	2.03	-1.5%					

Table 1. Numerical results for the example GNQN ($q = 50$)

```

1 Determine bottleneck node  $b$  of closed network
2 Cut connection to  $b$  and obtain open network
3 Limit capacity of  $b$  to  $q$ 
4  $\lambda_{arr,low} := 0$  ;  $\lambda_{arr,high} := h$ 
5  $c_{dep}^2 := 1$ 
6 do
7    $\lambda_{arr} := \frac{1}{2} \cdot (\lambda_{arr,high} + \lambda_{arr,low})$  ;  $c_{arr}^2 := c_{dep}^2$ 
8   call FiFiQueues to obtain dept. descriptor ( $\lambda_{dep}, c_{dep}^2$ )
9   if  $\sum_{i=1}^n E[N_i] > q$  or network is unstable then
10      $\lambda_{arr,high} := \lambda_{arr}$ 
11   else
12      $\lambda_{arr,low} := \lambda_{arr}$ 
13   endif
14 while  $err(\lambda_{arr,low}, \lambda_{arr,high}) > \delta_1$  or  $err'(\sum_{i=1}^n E[N_i], q) > \delta_2$ 

```

Fig. 5. Analysis procedure for CQNs based on FiFiQueues

Closed (FiFiQueues-NBC) [23]. Its model class is the model class of the original FiFiQueues adapted to CQNs, that is, without external arrivals and departures.

The analysis procedure for CQNs using FiFiQueues is shown in Figure 5. The outer iteration uses an interval splitting technique to determine an appropriate value λ_{arr} . The algorithm is based on two assumptions.

First, we assume that the number of jobs in the network q can be reached by an interval splitting method for the arrival rate λ_{arr} . The argument is similar to the one used in the functional approximation approach for closed BCMP networks, cf. [6]. The initial value h in line 4 has to be set to an appropriate large value (a too large initial value only slows down the convergence — overloaded networks are avoided by the test in line 9). Note that we do not need to test λ_{arr} and λ_{dep} for equality since this is always fulfilled in networks without losses.

The second assumption concerns the squared coefficient of variation c^2 . We have observed in the past that large queueing networks tend to “emboss” a network specific value for c^2 to the traffic stream. This means that the c^2 value of a traffic stream seems to depend only on the service processes and not on the c^2 value of the external arrival streams, whenever the traffic passes through a

sufficiently large number of queueing stations, provided that the utilization of the queueing stations is reasonably high. This is the reason why we have chosen an arbitrary initial value for c_{dep}^2 in line 5 and simply assign c_{dep}^2 to c_{arr}^2 in line 7.

The lines 1–3 of the algorithm are due to our observations in Section 3.2 concerning the bottleneck. In order to approach the situation in which there is a deterministic queue length distribution at the bottleneck station, we proceed the following way. We cut the CQN directly in front of the bottleneck (lines 1–2) and transform the bottleneck station into a queueing station with finite capacity q (line 3). When the bottleneck station experiences a high load and, hence, most of the jobs are waiting in the queue of the bottleneck node, this finite capacity limits the maximum number of jobs in the network and leads to a more deterministic queue length distribution at the bottleneck. Our experiments have shown that we can select an arbitrary connection to the bottleneck for the cut if more than one connection exists. Similarly, if more than one bottleneck exists, an arbitrary one is selected as finite capacity station.

Note that the initial value h of $\lambda_{arr,high}$ (line 4) must be sufficiently high in order to obtain a load of 100% at the bottleneck station. If the bottleneck has only one incoming edge, h must be at least twice the service rate of the bottleneck due to the factor of $\frac{1}{2}$ in line 7. Our experiments suggest to use a slightly larger factor of 2.5 in order to compensate for the losses at the bottleneck station.

The numerical results for the Gordon-Newell queueing network shown in Figure 4 with $q = 50$ are displayed in the column labeled “decomp” in Table 1. The right column titled “relerr” gives the error between the decomposition approach and the simulation, relative to the latter. Note that the large relative error of node 5’s c_N^2 is caused by the fact that the absolute numbers themselves are very small. The other relative errors are within the 95%-confidence intervals of the simulation.

3.4 Complexity

The proposed iterative CQN algorithm consists of two iterations of which the step count is usually not known in advance. The inner iteration is part of the FiFiQueues algorithm for OQNs. In each inner iteration all queueing stations are analyzed. Note that only the bottleneck station is modeled as a finite queueing station (of size q) and, hence, the time complexity of its analysis depends on the population q . Concerning the outer iteration, we have observed that there is no direct dependency on the population q (see Section 4.3 for a detailed example). Our experiments have shown that even for complex networks with large populations, the required number of inner and outer iterations usually stays below 15, resp. 30.

In addition to the iterations, the algorithm has to identify the bottleneck of the network. The solution of the system of traffic equations has a time complexity of $O(n^3)$ if a direct solution method like Gaussian elimination is employed, but reduces to $O(c \cdot n)$ in practice when sparse storage and an iterative solver such as Gauss-Seidel are used (where c is the average number of outgoing connections per station).

Fig. 6. Cyclic three-queue CQN

One distinct bottleneck: $\mu_1 = \mu_3 = 1, \mu_2 = 0.5$					One bottleneck: $\mu_1 = 1, \mu_2 = 2, \mu_3 = 1.1$				
node		decomp	sim	relerr	node		decomp	sim	relerr
1	ρ	0.5	0.5	0.0%	1	ρ	0.95	0.95	0.0%
	$E[N]$	1.5	1.55	-3.2%		$E[N]$	11.90	11.17	6.5%
2	ρ	1.0	1.0	0.0%	2	ρ	0.48	0.47	2.1%
	$E[N]$	17.0	17.0	0.0%		$E[N]$	1.34	1.32	1.5%
3	ρ	0.5	0.5	0.0%	3	ρ	0.84	0.86	-2.3%
	$E[N]$	1.5	1.49	0.7%		$E[N]$	7.76	7.51	3.3%

Three bottlenecks: $\mu_1 = \mu_2 = \mu_3 = 1$					Two bottlenecks: $\mu_1 = \mu_3 = 1, \mu_2 = 2$				
node		decomp	sim	relerr	node		decomp	sim	relerr
1	ρ	0.81	0.85	-4.7%	1	ρ	0.88	0.91	-3.3%
	$E[N]$	5.98	6.64	-9.9%		$E[N]$	8.25	9.36	-11.9%
2	ρ	0.86	0.85	1.2%	2	ρ	0.44	0.45	-2.2%
	$E[N]$	7.45	6.66	11.9%		$E[N]$	1.12	1.22	-8.2%
3	ρ	0.83	0.85	-2.4%	3	ρ	0.93	0.91	-1.1%
	$E[N]$	6.57	6.69	-1.8%		$E[N]$	10.63	9.42	12.8%

Table 2. Results for cyclic three-queue CQN for different rates μ_i and $q = 20$

4 Validation

In this section we examine the performance of the new decomposition-based method for CQNs, using four typical examples: a cyclic CQN (Section 4.1), two CQNs with merging and splitting of traffic streams (Section 4.2) and a more general complex CQN (Section 4.3).

4.1 A cyclic three-queue CQN

The first model is a simple CQN that consists of three queues in series as shown in Figure 6. All service times are hyper-exponentially distributed with $c_{service}^2 = 2$. This network does not require any traffic merging or splitting, so that the corresponding open network can be analyzed by FiFiQueues almost without any error.

Table 2 gives the results of the decomposition method in comparison to simulation for three different service rates. The population size was set to 20. The last column gives the relative errors. All relative 95%-confidence intervals of the simulation were below 1%.

$q = 5$					$q = 10$				
node		decomp	sim	relerr	node		decomp	sim	relerr
1	ρ	0.41	0.44	-6.8%	1	ρ	0.48	0.49	-2.0%
	E[N]	0.84	0.93	-9.7%		E[N]	1.28	1.35	-5.2%
2	ρ	0.89	0.89	0.0%	2	ρ	0.97	0.97	0.0%
	E[N]	3.27	3.14	4.1%		E[N]	7.42	7.34	1.1%
3	ρ	0.43	0.44	-2.3%	3	ρ	0.48	0.49	-2.0%
	E[N]	0.89	0.92	-3.3%		E[N]	1.30	1.31	-0.8%

$q = 30$					$q = 60$				
node		decomp	sim	relerr	node		decomp	sim	relerr
1	ρ	0.5	0.5	0.0%	1	ρ	0.5	0.5	0.0%
	E[N]	1.50	1.57	-4.5%		E[N]	1.50	1.57	-4.5%
2	ρ	1.0	1.0	0.0%	2	ρ	1.0	1.0	0.0%
	E[N]	27.0	26.9	0.4%		E[N]	57.0	56.9	0.2%
3	ρ	0.5	0.5	0.0%	3	ρ	0.5	0.5	0.0%
	E[N]	1.50	1.50	0.0%		E[N]	1.51	1.50	0.7%

Table 3. Results for cyclic three-queue CQN for various population sizes

Table 2 shows that the algorithm does best when one distinct bottleneck is present in the network, i.e., in case $\mu_1 = \mu_3, \mu_2 = 0.5$. Then our “trick” with the finite queue provides very good results. Even when two stations have similar service rates ($\mu_1 = 1, \mu_2 = 2, \mu_3 = 1.1$), still good results are obtained. The errors are, however, slightly larger in cases where more than one bottleneck exist. Since the algorithm can select only one node as bottleneck it is not able to distribute the jobs evenly over all nodes in case all service rates are equal ($\mu_1 = \mu_2 = \mu_3 = 1$). The worst (but still okay!) results are obtained when the network consists of two bottlenecks and one fast service station ($\mu_1 = \mu_3 = 1, \mu_2 = 2$); again, the algorithm can select only one node as bottleneck which results in different average queue lengths for node 1 and node 3 whereas the simulation indicates that both queue lengths should be equal.

The next experiment uses the same queueing network but this time $\mu_2 = 0.5$, $\mu_1 = \mu_3 = 1$, and the population is varied between 5 and 60. The results are shown in Table 3. As can be seen, the relative errors are larger for small population sizes. Similar results have been obtained for other CQNs. The explanation for this behavior is that the small number of jobs in the CQN causes correlations between the queue lengths. This fact contradicts with FiFiQueues’ assumptions about the network, hence, slightly worse results are obtained.

4.2 CQNs with merging and splitting

With these two CQNs we specifically evaluate how well our new algorithm handles queueing network topologies in which traffic streams are merged and split. The two networks and the obtained results for $q = 20$ are shown in Figure 7 (CQN 1), respectively Figure 8 (CQN 2). Table 4 shows the results for CQN 2

Fig. 7. CQN 1 with merging and splitting

Fig. 8. CQN 2 with merging and splitting

when the negative-exponential service time distribution of node 3 has been replaced by a hyper-exponential distribution with $c^2 = 10$.

These examples illustrate that the algorithm for CQNs can only be as good as the underlying method for the open networks. Although q is not very small here, the errors are larger than in the case of three queues in series (see previous section) because FiFiQueues employs approximations to perform the traffic merging and splitting. Still, we judge these results very good.

4.3 A larger CQN

We finally consider a larger and more complex CQN, as shown in Figure 9. The evaluation results for populations q between 5 and 60 can be found in Table 5. As observed before, the relative errors are largest for the smallest populations.

In general, it is worth to emphasize the fact that our new algorithm provides the best results for large populations. These are exactly the most interesting cases, as for these cases the overall underlying continuous-time Markov chain

node		decomp	sim	relerr
1	$E[N]$	6.03	6.62	-8.9%
2	$E[N]$	5.08	5.39	-5.8%
3	$E[N]$	8.89	7.99	11.3%

Table 4. Results for CQN 2 with $c^2 = 10$ at the third node

Fig. 9. A larger CQN

(CTMC) would be the largest as well. The number of states NoS of a CTMC underlying a Gordon-Newell network is given by $NoS = \binom{n+q-1}{n-1}$, where n is the number of queueing stations and q is the population size [16]. For networks with phase-type service time distributions, the number of states for large q is approximately given by $NoS \approx \binom{n+q-1}{n-1} \cdot \prod_{i=1}^n m_i$, where m_i is the number of phases of the service time distribution of station i . Hence, the underlying CTMC of the CQN of Figure 9 with $n = 6$ and $q = 30$ would comprise approximately $2 \cdot 10^8$ states, whereas the largest CTMC constructed by FiFiQueues during the analysis of the same network has around 240 states only.

We finally comment on the convergence behavior of our new algorithm. For that purpose, Figure 10 shows for $q = 30$ how the algorithm modifies the arrival rate for the open network in each (outer) iteration step in order to reach the preset number of jobs. The interval splitting algorithm first lowers the arrival rate to a fourth of the initial value, then the arrival rate is slowly increased (until iteration 6). In this example the stopping criterion is met after 17 steps, however, we see that a good approximation is already reached after about 10 steps. The “dip” in the curves can easily be explained. The algorithm starts with a value for $\lambda \approx 1.22$, which clearly is too high. This value is then averaged with a value 0, leading to the second value of approximately 0.62. Again this value is too large, leading to the third value slightly above 0.3 (note: the left Y-axis starts at 0.3). Then the value for the arrival rate regains itself to a value around 0.55. The clear dip, hence, is an artifact of the interval splitting method; a more advanced method could probably avoid it. In total, our implementation takes three seconds to analyze the network for $q = 30$.

Finally, Figure 11 shows the number of jobs as function of the iteration step count, for four different populations. No direct dependency between the population and the number of required iterations can be observed. We again see a clear dip in the curves, for which the explanation as above holds as well.

$q = 5$					$q = 10$				
node		decomp	sim	relerr	node		decomp	sim	relerr
1	ρ	0.65	0.69	-5.8%	1	ρ	0.82	0.85	-3.5%
	$E[N]$	1.28	1.27	0.8%		$E[N]$	2.69	2.71	-0.7%
2	ρ	0.34	0.36	-5.6%	2	ρ	0.42	0.44	-4.5%
	$E[N]$	0.54	0.57	-5.3%		$E[N]$	0.84	0.87	-3.4%
3	ρ	0.33	0.36	-8.3%	3	ρ	0.42	0.44	-4.5%
	$E[N]$	0.56	0.59	-5.1%		$E[N]$	0.90	0.94	-4.2%
4	ρ	0.71	0.74	-4.1%	4	ρ	0.88	0.91	-3.2%
	$E[N]$	1.65	1.57	5.1%		$E[N]$	4.14	3.98	4.0%
5	ρ	0.34	0.36	-5.6%	5	ρ	0.43	0.44	-2.3%
	$E[N]$	0.54	0.56	-3.6%		$E[N]$	0.82	0.85	-3.5%
6	ρ	0.30	0.33	-9.1%	6	ρ	0.38	0.40	-5.0%
	$E[N]$	0.43	0.45	-4.4%		$E[N]$	0.62	0.65	-4.6%

$q = 30$					$q = 60$				
node		decomp	sim	relerr	node		decomp	sim	relerr
1	ρ	0.93	0.93	0.0%	1	ρ	0.94	0.94	0.0%
	$E[N]$	6.87	7.11	-3.4%		$E[N]$	8.47	8.32	1.8%
2	ρ	0.48	0.49	-2.0%	2	ρ	0.49	0.49	0.0%
	$E[N]$	1.07	1.10	-2.7%		$E[N]$	1.10	1.10	0.0%
3	ρ	0.48	0.49	-2.0%	3	ρ	0.49	0.49	0.0%
	$E[N]$	1.18	1.21	-2.5%		$E[N]$	1.22	1.23	-0.8%
4	ρ	0.99	1.00	-1.0%	4	ρ	1.00	1.00	0.0%
	$E[N]$	19.10	18.76	-1.8%		$E[N]$	47.36	47.48	-0.3%
5	ρ	0.48	0.47	-2.1%	5	ρ	0.49	0.49	0.0%
	$E[N]$	1.04	1.05	-1.0%		$E[N]$	1.07	1.07	0.0%
6	ρ	0.43	0.44	-2.3%	6	ρ	0.44	0.44	0.0%
	$E[N]$	0.77	0.77	0.0%		$E[N]$	0.79	0.79	0.0%

Table 5. Results for the larger CQN for various population sizes

5 Related work

Over the last decades, several other proposals to solve general closed queueing networks have been proposed. We discuss these below and indicate how these methods differ from ours.

Of course, the simplest way to approximate the type of CQN we address is by just ignoring the second moment and do as if the service times follow a negative exponential distribution. Although good results have been reported for the overall network throughput with this approach (cf. [7, Chapter 10.1.4], esp. in case of squared coefficients of variation below 1 and large populations), in general, one cannot say that this approach yields good results for per-queue performance measures.

Kouvatsos and Xenios [18] have proposed a method for the analysis of arbitrary queueing networks with multiple servers and repetitive-service blocking using the Maximum Entropy Method (MEM). The idea of MEM is to find the

Fig. 10. Arrival rate (left Y -axis) and number of jobs (right Y -axis) in the CQN as function of the step number ($q = 30$)

Fig. 11. Number of jobs in the CQN as function of the step number, for four different populations

solution of the model that maximizes the entropy of the system under the condition that only the information given by the model specification is used. The analyzed network may be open or closed and consists of n finite multiple-server queues of type $GE|GE|m|K'; K$ where jobs can only leave the queue if the number of jobs in the queueing station is larger than K' . The complexity of the method is quite high. The solution algorithm consists of two stages that use iterative procedures. Stage 1 has a time order of about $O(c_1 \cdot n^6)$ in case all queues may block, i.e., their queueing capacity is smaller than the population size q . The complexity of stage 2 is $O(c_2 \cdot n^2 q^2)$, where c_1 and c_2 are the numbers of iterations in the successive stages.

The method put forward by Marie [20] also describes an approximation procedure for closed queueing networks with FCFS service stations and service time distributions described via the first two moments. In the original paper, only two small-scale examples have been presented. It appears that “Marie’s method” is especially suitable for small models, with multiple-server stations, a class our method does not aim at. Instead, we aim at larger models with single-server nodes.

Dallery *et al.* report on a number of variations and extensions of Marie’s method. In particular, [11] presents an alternative way (“operational analysis”) to derive a number of well-known results, among others, Marie’s method. [5] addresses a multiclass extension of Marie’s work, however, the use of non-exponential services is not specifically addressed. [4] unifies the method of Marie and another decomposition/aggregation-based method in the sense that they are both variants of the same (higher-level) principle of “summarizing” the environment of a single server via load-dependent arrival and service rates. Finally, [3] extends Marie’s work in the sense that population constraints are posed over subnetworks.

Many other methods have been developed for the analysis of some special CQNs containing finite queues. They only support very restricted network topologies, like two-queue tandem networks, etc., or are restricted to the BCMP

model class. We refer to [22] for an overview paper, as well as to the citations in [1]. Furthermore, approximate mean-value algorithms like the Bard-Schweizer [29] or the SCAT algorithm [21] do not apply, as our starting point is *not* a product-form queueing network. The decomposition methods proposed for stochastic Petri nets, e.g. [9], do not apply here, as they rely on the solution of non-structured sub-CTMCs, and do refer to a completely different model class.

6 Summary and conclusions

In this paper we have proposed a new and efficient decomposition-based method for the analysis of closed queueing networks. It is especially attractive because it is based on existing analysis methods for open queueing networks. A variety of evaluations, based on an implementation in the context of FiFiQueues, shows that the method is able to provide accurate results for a broad class of CQNs. Additionally, the method is very fast even for larger networks with large populations. However, the experiments have also shown that the method is less accurate when the CQN contains more than one bottleneck, which can be the case, for example, in load-balanced systems.

Naturally, our new method for CQN can only be as good as the method employed for the analysis of the employed underlying OQNs. Although we are quite satisfied with the performance of FiFiQueues for OQNs, improvements can still be made, e.g., one could think of using more sophisticated traffic descriptors like MAPs (Markovian arrival processes) than the two-moments descriptors of FiFiQueues. More research has to be done in this area, but it is to be expected that this requires a much more complex procedure for the estimation of the traffic descriptor than the one employed here; some recent research results in this field can be found in [25].

References

1. G. Balbo and G. Serazzi. Asymptotic analysis of multiclass closed queueing networks: Multiple bottlenecks. *Performance Evaluation*, 30:115–52, 1997.
2. F. Baskett, K.M. Chandy, R.R. Muntz, and F. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, 1975.
3. B. Baynat and Y. Dallery. Approximate techniques for general closed queueing networks with subnetworks having population constraints. *European Journal on Operations research*, 69:250–264, 1993.
4. B. Baynat and Y. Dallery. A unified view of product-form approximation techniques for general closed queueing networks. *Performance Evaluation*, 18(3):205–224, 1993.
5. B. Baynat and Y. Dallery. A product-form approximation method for general closed queueing networks with several classes of customers. *Performance Evaluation*, 24(3):165–188, 1996.
6. G. Bolch, G. Fleischmann, and R. Schreppel. Ein funktionales Konzept zur Analyse von Warteschlangennetzen und Optimierung von Leistungsgrößen. In *Messung*,

- Modellierung und Bewertung von Rechensystemen (MMB)*, *Proceedings*, volume 154, pages 327–342. Springer, 1987.
7. G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons, 1998.
 8. J.P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM*, 16(9):527–531, 1973.
 9. G. Ciardo and K.S. Trivedi. A decomposition approach for stochastic reward net models. *Performance Evaluation*, 18(3):37–59, 1993.
 10. A.E. Conway and N.D. Georganas. *Queueing Networks: Exact Computational Algorithms*. The MIT Press, 1989.
 11. Y. Dallery and X.-R. Cao. Operational analysis of stochastic closed queueing networks. *Performance Evaluation*, 14(1):43–61, 1992.
 12. W.J. Gordon and G.J. Newell. Closed queueing systems with exponential servers. *Operations Research*, 15:254–265, 1967.
 13. B. R. Haverkort. Approximate analysis of networks of PH|PH|1|K queues: Theory & tool support. In H. Beilner and F. Bause, editors, *MMB*, volume 977 of *Lecture Notes in Computer Science*, pages 239–253. Springer, 1995.
 14. B. R. Haverkort. QNAUT: Approximately analyzing networks of PH|PH|1|K queues. *Proceedings of the 1996 International Computer Performance and Dependability Symposium*, page 57, 1996.
 15. B. R. Haverkort. Approximate analysis of networks of PH|PH|1|K queues with customer losses: Test results. *Annals of Operations Research*, 79:271–291, 1998.
 16. B.R. Haverkort. *Performance of Computer Communication Systems—A Model-Based Approach*. John Wiley & Sons, 1998.
 17. J. R. Jackson. Networks of waiting lines. *Operations Research*, 5:518–521, 1957.
 18. D.D. Kouvatsos and N.P. Xenios. MEM for arbitrary queueing networks with multiple general servers and repetitive-service blocking. *Performance Evaluation*, 10:169–195, 1989.
 19. P. J. Kühn. Approximate analysis of general queueing networks by decomposition. *IEEE Transactions on Communications*, 27(1):113–126, 1979.
 20. R.A. Marie. An approximate analytical method for general queueing networks. *IEEE Transactions on Software Engineering*, 5(5):530–538, 1979.
 21. D. Neuse and K.M. Chandy. SCAT: A heuristic algorithm for queueing network models of computing systems. *ACM Performance Evaluation Review*, 10(3):59–79, 1981.
 22. R.O. Onvural. Survey of closed queueing networks with blocking. *ACM Computing Surveys*, 22(2):83–121, june 1990.
 23. P. Reinelt. Erweiterung des fixpunkt-basierten Analyseverfahrens von FiFiQueues auf geschlossene Warteschlangennetze. Diploma thesis, Distributed Systems group, RWTH Aachen, 2001.
 24. M. Reiser and S.S. Lavenberg. Mean value analysis of closed multichain queueing networks. *Journal of the ACM*, 22(4):313–322, 1980.
 25. R. Sadre. *Decomposition-Based Analysis of Queueing Networks*. PhD thesis, University of Twente, 2006.
 26. R. Sadre and B. R. Haverkort. FiFiQueues: fixed-point analysis of queueing networks with finite-buffer stations. In *MMB (Kurzvorträge)*, volume 99-16, pages 77–80. Universität Trier, 1999.
 27. R. Sadre, B. R. Haverkort, and A. Ost. An efficient and accurate decomposition method for open finite- and infinite-buffer queueing networks. In W. Stewart and B. Plateau, editors, *Proc. 3rd Int. Workshop on Numerical Solution of Markov Chains*, pages 1–20. Zaragoza University Press, 1999.

28. R. Sadre and B.R. Haverkort. FiFiQueues: fixed-point analysis of queueing networks with finite-buffer stations. In *Computer Performance Evaluation. Modelling Techniques and Tools: 11th International Conference, TOOLS 2000*, volume 1786 of *Lecture Notes in Computer Science*, pages 324–327. Springer, 2000.
29. P. Schweitzer. Approximate analysis of multichain closed queueing networks. In *Proceedings of the International Conference on Stochastic Control and Optimization*, 1979.
30. A. J. Weerstra. Using matrix-geometric methods to enhance the QNA method for solving large queueing networks. Diploma thesis, Department of Computer Science, University of Twente, 1994.
31. W. Whitt. The Queueing Network Analyzer. *The Bell System Technical Journal*, 62(9):2779–2815, 1983.
32. W. Whitt. Performance of The Queueing Network Analyzer. *The Bell System Technical Journal*, 62(9):2817–2843, 1983.