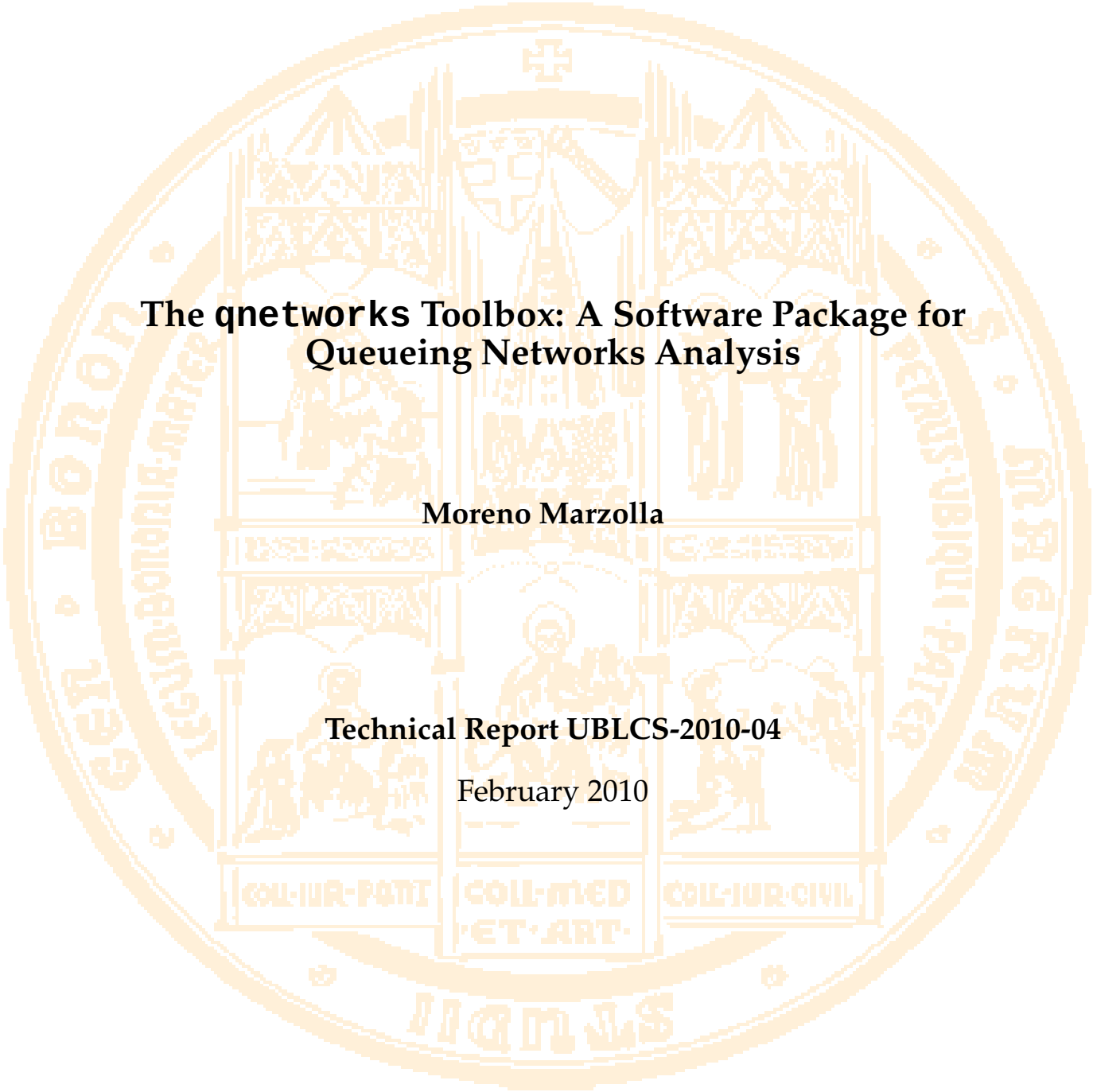

The `qnetworks` Toolbox: A Software Package for Queueing Networks Analysis

Moreno Marzolla

Technical Report UBLCS-2010-04

February 2010

Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna (Italy)

The University of Bologna Department of Computer Science Research Technical Reports are available in PDF and gzipped PostScript formats via anonymous FTP from the area `ftp.cs.unibo.it:/pub/TR/UBLCS` or via WWW at URL `http://www.cs.unibo.it/`. Plain-text abstracts organized by year are available in the directory `ABSTRACTS`.

Recent Titles from the UBLCS Technical Report Series

2008-16 *Stochastic Semantics in the Presence of Structural Congruence: Reduction Semantics for Stochastic Pi-Calculus*, Bravetti, M., July 2008.

2008-17 *Measures of conflict and power in strategic settings*, Rossi, G., October 2008.

2008-18 *Lebesgue's Dominated Convergence Theorem in Bishop's Style*, Sacerdoti Coen, C., Zoli, E., November 2008.

2009-01 *A Note on Basic Implication*, Guidi, F., January 2009.

2009-02 *Algorithms for network design and routing problems (Ph.D. Thesis)*, Bartolini, E., February 2009.

2009-03 *Design and Performance Evaluation of Network on-Chip Communication Protocols and Architectures (Ph.D. Thesis)*, Concer, N., February 2009.

2009-04 *Kernel Methods for Tree Structured Data (Ph.D. Thesis)*, Da San Martino, G., February 2009.

2009-05 *Expressiveness of Concurrent Languages (Ph.D. Thesis)*, di Giusto, C., February 2009.

2009-06 *EXAM-S: an Analysis tool for Multi-Domain Policy Sets (Ph.D. Thesis)*, Ferrini, R., February 2009.

2009-07 *Self-Organizing Mechanisms for Task Allocation in a Knowledge-Based Economy (Ph.D. Thesis)*, Marcozzi, A., February 2009.

2009-08 *3-Dimensional Protein Reconstruction from Contact Maps: Complexity and Experimental Results (Ph.D. Thesis)*, Medri, F., February 2009.

2009-09 *A core calculus for the analysis and implementation of biologically inspired languages (Ph.D. Thesis)*, Versari, C., February 2009.

2009-10 *Probabilistic Data Integration*, Magnani, M., Montesi, D., March 2009.

2009-11 *Equilibrium Selection via Strategy Restriction in Multi-Stage Congestion Games for Real-time Streaming*, Rossi, G., Ferretti, S., D'Angelo, G., April 2009.

2009-12 *Natural deduction environment for Matita*, C. Sacerdoti Coen, E. Tassi, June 2009.

2009-13 *Hints in Unification*, Asperti, A., Ricciotti, W., Sacerdoti Coen, C., Tassi, E., June 2009.

2009-14 *A New Type for Tactics*, Asperti, A., Ricciotti, W., Sacerdoti Coen, C., Tassi, E., June 2009.

2009-15 *The k-Lattice: Decidability Boundaries for Qualitative Analysis in Biological Languages*, Delzanno, G., Di Giusto, C., Gabbrielli, M., Laneve, C., Zavattaro, G., June 2009.

2009-16 *Landau's "Grundlagen der Analysis" from Automath to lambda-delta*, Guidi, F., September 2009.

2010-01 *Fast overlapping of protein contact maps by alignment of eigenvectors*, Di Lena, P., Fariselli, P., Margara, L., Vassura, M., Casadio, R., January 2010.

2010-02 *Optimized Training of Support Vector Machines on the Cell Processor*, Marzolla, M., February 2010.

2010-03 *Modeling Self-Organizing, Faulty Peer-to-Peer Systems as Complex Networks* Ferretti, S., February 2010.

The `qnetworks` Toolbox: A Software Package for Queueing Networks Analysis

Moreno Marzolla[1]

Technical Report UBLCS-2010-04

February 2010

Abstract

Queueing Networks (QNs) are a useful performance modelling notation. They can be used to describe many kinds of systems, and efficient solution techniques have been developed for some classes of QN models. Despite the fact that QNs have been extensively studied, very few software packages for QN analysis are available today. In this paper we describe the qnetworks *toolbox, a free software package for QN analysis implemented in GNU Octave.* qnetworks *provides implementations of solution algorithms for single-station queueing systems as well as for product- and some non product-form QN models. Exact, approximate and bound analysis can be performed. Additional utility functions and algorithms for Markov Chains analysis are also included. The* qnetworks *package is available as free software, allowing users to study, modify and extend the code. This makes* qnetworks *a viable teaching tool.*

1. Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, I-40127 Bologna, Italy, Email: `marzolla@cs.unibo.it`

1 Introduction

QNs are a very powerful modelling notation; they can be applied to many different domains, including computer networks, supply chain analysis, software systems, street traffic and others [21]. QNs has been subject to extensive studies, and a vast literature of solution algorithms exists. QN models can be evaluated either by simulation, or using analytical and numerical techniques. Simulation has the advantage of being able to evaluate any kind of system, including extended QN models for which other solution techniques are either not available, or only produce approximate results. However, simulation can require significant time to accurately evaluate complex models, and the computed results are only given as confidence intervals. Furthermore, evaluation of the same model with different parameters (the so-called "what-if" analysis) is computationally costly as it involves a large number of simulation runs.

There is a vast literature on numerical solution techniques for QN models (see [5] and references therein); despite this, there is a shortage of software tools implementing these algorithms. This is particularly unfortunate for many reasons: people keep reimplementing the same old algorithms over and over again, which is error prone and time consuming. This is especially true since some QN algorithms can be tricky to implement correctly. Effort put on implementing old algorithms could be better spent solving interesting modelling problems, or developing new algorithms.

In this paper we present qnetworks, a QN analysis package written in GNU Octave. GNU Octave [10] is an interpreted language for numerical computations very similar to MATLAB[2][15], to which it is mostly compatible. qnetworks provides a set of functions for analyzing product-form (PF) as well as some non PF QN models; bound computation, evaluation of single-station queueing systems and Markov Chains analysis are also possible. qnetworks is free software: users can inspect, modify and redistribute the code, which makes qnetworks a viable teaching tool.

qnetworks is not an integrated modelling tool, like JMT [4] or RESQ [18]. Rather, it is a library of functions which can be used as building blocks for analyzing QN models. The Octave interactive environment provides the "glue" which allows complex models to be quickly analyzed, enabling a greater degree of flexibility which is usually not provided by rigid integrated modelling environments. Models can be defined and solved programmatically, so that fully automated batch analysis can be easily implemented. However, a significant understanding of QN modelling is necessary in order to use qnetworks. For this reason, qnetworks is not appropriate for the casual user, for which a less flexible but more user-friendly tool would be advisable.

Different usage scenarios for qnetworks can be identified:

- **Incremental model development**: qnetworks and GNU Octave are an ideal platform for rapid prototyping and iterative refinement of QN models. Models can be defined and analyzed quickly using the function provided by qnetworks. The Octave language provides very convenient features for vector manipulations which allow models to be defined concisely.

- **Modelling environment**: large and complex performance studies can be performed, as models involving repetitive or embedded structures can be easily defined. Ad-hoc solution techniques can be realized on top of the available functions. As a specific example, we show later in this paper how hierarchical modelling with flow-equivalent service centers can be done with qnetworks, even if no facility to perform such kind of analysis is provided by the package.

- **Queueing Network research**: new QN analysis algorithms can be implemented inside qnetworks and tested against existing ones. Contributions to the qnetworks package are highly welcome. For many QN algorithms described in the literature, no implementation is readily available. We hope that qnetworks will encourage researchers and practitioners

2. MATLAB is a trademark of The MathWorks Inc.

to provide implementations of their own algorithms, so that others can use and improve them.

- **Teaching**: qnetworks is suitable for introducing QN modelling concepts and solution techniques. Students can immediately get a visual feedback from the solution of QN models by using the graphing capabilities provided by GNU Octave. For example, all plots on this paper has been produced by GNU Octave after solving the appropriate model with qnetworks.

In order to partially support the above claims, most of this paper is structured in a tutorial style, showing how qnetworks can actually be used in simple modelling studies. In Section 2 we briefly review related works in the area of QN software. In Section 3 we introduce some basic concepts and definitions about QNs. In Section 4 we illustrate the features of qnetworks and the algorithms which have been implemented. In Section 5 we give some usage examples to demonstrate how qnetworks can be used in practice. Finally, Section 7 contains conclusions and future works.

2 Related works

Over the years, many software packages for the solution of QN models have been developed. The Research Queueing Package (RESQ) [18] developed at IBM Research was one of the first very successful QN analysis packages. It provided a modelling language for describing extended QN models, which then could be solved by either analytical of simulation techniques. A graphical user interface (called RESQME [8]) was developed in order to facilitate the model definition process. A similar tool was QNAP2 [22], which provided different solution methods (analytical or simulation-based) for analyzing product and non-product form QNs. Networks are described using a textual notation; the QNAP2 tool itself was written in FORTRAN 77.

Unfortunately, both QNAP2 and RESQ are no longer available. Among the tools which are still available and in use are SHARPE, PDQ and JMT. The Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE) [17] is an integrated package for describing and analyzing hierarchical stochastic models, including QN, fault trees, reliability models and so on. Pretty Damn Quick (PDQ)[3] is a QN software package providing bindings for multiple languages (including Java, PHP, Perl, Python and C). This package implements the exact and approximate Mean Value Analysis (MVA) algorithm for closed QNs. Finally, the Java Modelling Tools (JMT)[4] [4] is a recent free software tool for construction and evaluation of QN models. JMT is developed by the Performance Evaluation Lab of the Politecnico di Milano, Italy. JMT is written in Java, which implies that it is highly portable across different execution environments. JMT is mostly devoted to simulation-based analysis of extended QNs, although it includes the MVA algorithm for single as well as multiclass networks.

Additional queueing theory software are listed at:

`http://web2.uwindsor.ca/math/hlynka/qsoft.html`.

It should be observed that most of the tools listed there (many hyperlinks are broken) are of very limited scope and/or obsolete.

3 Queueing Networks

QN models are used to describe systems consisting of a collection of resources and a population of requests (or jobs) which circulate demanding service from the resources. Each resource consists of a *service center*, which is represented by a queue connected to a number of identical servers. A QN model contains a finite number K of service centers. In an *open* model there are infinite streams of requests originating outside the system, which arrive to center k with rate λ_k; requests

3. `http://www.perfdynamics.com/Tools/PDQcode.html`
4. `http://jmt.sourceforge.net/`

can leave the system from any node. In a *closed* model there is a fixed population of N requests which continuously circulate through the system. *Mixed* models are also possible, in which there are multiple classes of requests, some of which are open and other closed.

QN analysis for single-class networks usually involves computing the steady-state probabilities $\pi_k(i)$ that there are i requests at center k. A class of QN models is said to have *product-form solution* if the steady state solution of the network can be expressed as the product of factors describing the state of each individual node. The first class of Product-Form Queueing Networks (PFQNs) was identified by Jackson [12] which discovered that single-class, open networks with the following properties have PF solution:

- Each node of the network can have Poisson arrival from outside; a job can leave the network from any node. λ_k denotes the external arrival rate to node k. Arrival rates may depend on the number of requests at the receiving node.
- All service times are exponentially distributed, and service discipline at all nodes is First-Come First-Served (FCFS).
- The k-th node consists of $m_k \geq 1$ identical servers with average service time S_k. The service time S_k may depend on the number of requests at node k.

The result of Jackson has been later extended to closed networks by Gordon and Newell [11], and to open, closed and mixed networks with multiple request classes by Baskett, Chandy, Muntz and Palacios [3]. Specifically, BCMP networks satisfy the following properties:

- Service discipline at each node can be FCFS, Processor Sharing (PS), Infinite Server (IS) or Last-Came First-Served, Preemptive Resume (LCFS-PR).
- Service times for FCFS nodes must be exponentially distributed and class-independent. Service times for the other kind of nodes must have rational Laplace transform, and can in general be class-dependent. The service time S_{ck} of class c requests at service center k might depend on the number of requests at that center.
- External arrivals to node k (if any) must be a Poisson process. λ_{ck} denotes the class c arrival rate at service center k. For closed classes, N_c is the number of class c requests.
- A class r customer completing service at queue i will either move to queue j as a class s request with probability P_{risj}, or leave the system with probability $1 - \sum_{j,s} P_{risj}$ which can be nonzero for some subset of queues.

Additional network types have been shown to have PF solution as well. PFQN are of particular interest because they have efficient solution algorithms; furthermore, despite their limitations (as stated above) PFQN are general enough to be useful for modelling large classes of actual systems. Unfortunately, there are many situations which can be encountered in modern systems which can only be represented with extended QN models which do not have PF solution. For example, fork-join parallelism, simultaneous resource possession, non-exponential service times and blocking due to finite capacity queues lead to networks which in general do not have PF solution. In some cases, approximate analysis is possible (the approach of flow-equivalent centers illustrated in Sec. 5.4 is widely used); otherwise the network can be handled through discrete-event simulation.

4 Overview of qnetworks

qnetworks is a collection of numerical algorithms written in GNU Octave for exact or approximate solution of single- and multiclass QN models; open, closed or mixed networks are supported. GNU Octave has been chosen for different reasons. It is free software, available on multiple operating systems, including Windows, MacOSX and most Unix variants. Furthermore,

Table 1. Some of the functions provided by the qnetworks package

Function Name	Supported network type			
	Open	Closed	Single	Multi
qnopensingle()	√		√	
qnopenmulti()	√			√
qnconvolution()		√	√	
qnconvolutionld()		√	√	
qnclosedsinglemva()		√	√	
qnclosedsinglemvald()		√	√	
qnclosedmultimva()		√		√
qnclosedmultimvaapprox()		√		√
qnmix()	√	√		√
qnsolve()	√	√	√	√
qnmvablo()		√	√	
qnopenab()	√		√	
qnclosedab()		√	√	
qnopenbsb()	√		√	
qnclosedbsb()		√	√	
qnclosedgb()		√	√	

GNU Octave is mostly compatible with MATLAB, a language for numerical computations which is widely used in the research and industrial community. Thus, many students, researchers or practitioners interested in the numerical analysis of QN models will likely be already familiar with GNU Octave or MATLAB.

Technically, the qnetworks package includes a set of *m-scripts*; an m-script is a program specified in Octane interpreted language. While m-scripts are slower than compiled code, they allow maximum portability as they can be executed on any platform where the Octave interpreter has been ported. It should be observed that in most practical cases execution times of the algorithms in qnetworks are acceptable, so there is currently no need to rewrite the functions as compilable C/C++ code.

Table 1 lists the most important functions provided by the qnetworks package; documentation for each function can be accessed with the Octave **help** command, and is also included in the package documentation.

4.1 Single-station queueing systems

qnetworks provides functions for analyzing several types of single-station queueing systems [13, 5]: $M/M/m$[5], $M/M/m/k$, $M/M/\infty$, and asymmetric $M/M/m$ (this system contains m service centers with possibly different service rates), $M/G/1$ (general service time distribution) and $M/H_m/1$ (Hyperexponential service time distribution). For each kind of system, the following performance measures can be computed: utilization U, mean response time R, average number of requests in the system Q and throughput X.

4.2 Algorithms for product-form networks

qnetworks provides functions for analyzing PFQNs. For open networks, the qnopensingle() and qnopenmulti() functions apply to networks with single or multiple customer classes, respectively. These functions implement the well known equations for Jackson networks, and the extensions for BCMP open multiclass networks [14, 5].

For PF closed networks, exact as well as approximate algorithms are provided. For single-class closed networks, the MVA [16] and convolution [6] algorithms are implemented by the qnclosedsinglemva() and qnconvolution() functions respectively. Both support FCFS, LCFS-PR,

5. We use the standard Kendall's notation $A/B/C/K$, where A denotes the arrival process (M=Poisson), B denotes the service time distribution (M=exponential), C is the number of servers, K is the capacity of the system

PS and IS nodes; single and multiple server FCFS nodes are supported as well. qnclosedsinglemvald() and qnconvolutionld() implement the MVA and convolution algorithms for networks with general load-dependent service centers. We provide separate functions for networks with and without general load-dependent service centers because the former have a higher computational cost and require more memory, and at the same time are not frequently used. So we provide efficient implementations for the common case of networks without general load-dependent centers, while still allowing users to handle the general case using different functions.

For PF multiclass closed networks we implemented the multiclass MVA algorithm in the qnclosedmultimva() function. For networks with K service centers, C customer classes and population vector $(N_1, N_2, \ldots N_C)$, the multiclass MVA algorithm requires time $O\left(CK\prod_{i=1}^{C}(N_i+1)\right)$ and space $O\left(K\prod_{i=1}^{C}(N_i+1)\right)$. Due to its time and space complexity, the multiclass MVA algorithm is appropriate only for networks with small population and a limited number of customer classes. For larger networks, approximations based on the MVA have been proposed in the literature. qnetworks provides the Bard and Schweitzer approximation [19, 14] in function qnclosedmultimvaapprox().

Mixed multiclass PFQNs [3] are handled by the qnmix() function. In mixed networks, both open and closed classes of customers can be present at the same time; each class has its own routing probabilities. The qnmix() function does not currently allow class switching, nor supports general load-dependent queueing centers.

Finally, the higher-level function qnsolve() can be used as a single front-end to the algorithms described above. This function uses a less efficient, but more flexible representation of the network to be evaluated, and delegates the actual analysis to the appropriate solution algorithm (if available) for the particular kind of network.

4.3 Algorithms for non product-form networks

qnetworks includes algorithms for handling closed single-class networks with blocking. In blocking QNs, queues have a fixed capacity: a request joining a full queue will block until a slot in the destination node becomes available. Different blocking strategies have been investigated in the literature (see [2] for a review). The qnmvablo() function implements the MVABLO algorithm [1]. MVABLO is based on an extension of MVA, and computes approximate solutions for closed, single-class networks with Blocking After Service (BAS) blocking. According to the BAS discipline, a request joining a full queue blocks the source server until a slot is available at the destination.

Networks with blocking can also be analyzed with the qnmarkov() function. This function supports either open or closed, single-class networks where all queues have fixed capacity. Exact performance measures are derived by explicit construction of the underlying Markov Chain. This approach is appropriate for small networks only, due to the exponential growth of the size of the Markov Chain as the network increases.

4.4 Bound Analysis

It is often useful to compute bounds for the system throughput X or response time R. Performance bounds can be obtained very quickly, and can be useful for many performance studies such as problems involving on-line performance tuning of systems. qnetworks implements three different algorithms for computing performance bounds: Asymptotic Boundss (ABs) [9] for open and closed networks (functions qnopenab() and qnclosedab() respectively), Balanced System Boundss (BSBs) [23] for open and closed networks (functions qnopenbsb() and qnclosedbsb() respectively) networks, and Geometric Boundss (GBs) [7] for closed networks (function qnclosedgb()).

4.5 Validation

Almost all the functions provided by the qnetworks package include unit tests embedded inside the m-files. The tests can be invoked using Octave test function; it is also possible to run all tests

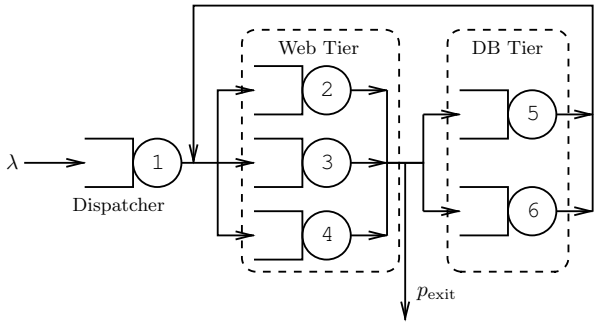

Figure 1. Open model of a two-tier E-commerce site; arrows denote nonzero flows

with a single command, which is particularly useful for checking the whole source distribution before releasing a new version.

As for many numerical softwares, testing QN packages can be nontrivial [20]. When possible, testing is done by computing results on reference networks for which correct values are known (e.g., from the literature). When exact solutions are not known, results can still be validated by computing them with different algorithms. For example, the MVA and convolution algorithms can be applied to the same network, and they must provide the same results (apart for a small deviation due to numerical inaccuracies). As another example, a $M/M/1/K$ queue is a special case of an $M/M/m/K$ queue with $m = 1$ servers. Thus, in this case the performance results provided by the qnmmmk() and qnmm1k must be the same. Finally, even when results cannot be directly compared, consistency checks can nevertheless be done. For example, the bounds on the system throughput computed by the AB or BSB equations must include the exact result provided by the MVA algorithm. Thus, it is possible to cross-check the qnclosedab(), qnclosedbsb() and qnclosedmva() functions.

5 Examples

In this section we present some usage examples of the qnetworks package.

5.1 Open network

As a first example, let us consider a simple model of a two-tier E-commerce site. The model is shown in Fig. 1 and consists of six FCFS service centers. Center 1 is the *dispatcher*, and is responsible for routing incoming requests to one of the Web servers (centers 2–4); we assume random routing with equal probability. After being processed by one of the Web servers, each request may leave the system with probability p_{exit}, or be forwarded to one of the Database servers (centers 5, 6).

We assume average service times $S_1 = 0.5$ at the dispatcher, $S_2 = S_3 = S_4 = 0.8$ at the Web servers and $S_5 = S_6 = 1.8$ at the Database servers; we set the arrival rate at center 1 as $\lambda_1 = 0.1$ requests/s and exit probability $p_{\text{exit}} = 0.5$. The transition probability matrix P is:

$$P = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \end{pmatrix}$$

This model can be defined with the following GNU Octave code:

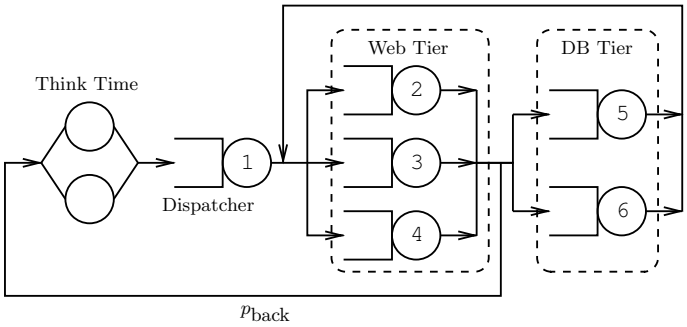

Figure 2. Closed model of a two-tier E-commerce site

```
p_exit = 0.5; # exit probability
i = 2:4; # indexes of Web servers
j = 5:6; # indexes of DB servers
P = zeros(6,6);
P(1,i) = 1/3;
P(i,j) = (1−p_exit)/2;
P(j,i) = 1/3;
S = [0.5 0.8 0.8 0.8 1.8 1.8];
lambda = [0.1 0 0 0 0 0];
V = qnvisits(P,lambda);
```

Note the use of *array slicing* to define the matrix P: variables i and j are ranges, and the single instruction P(j,i)=1/3 sets $P_{ji} = 1/3$ for all $j \in \{5, 6\}$ and $i \in \{2, 3, 4\}$.

In the code above we compute the visit counts V_k to service center k using the qnvisits() function. The visit counts V_k satisfy the equality $V_k = \lambda_k + \sum_{j=1}^{K} V_j P_{jk}$. In the example above, we get $V_1 = 1$, $V_2 = V_3 = V_4 = 0.6\bar{6}$ and $V_5 = V_6 = 0.5$.

The network is a PFQN and can be evaluated using the qnopensingle() function, as follows:

```
[U R Q X] = qnopensingle(sum(lambda),S,V);
```

where **sum**(lambda) is the global arrival rate $\sum_k \lambda_k$. The resulting utilizations are $U_1 = 0.05$, $U_2 = U_3 = U_4 = 0.05\bar{3}$ and $U_5 = U_6 = 0.09$. It is also easy to compute the maximum arrival rate λ_{sat} which the system can sustain: it is well known that $\lambda_{\text{sat}} = 1/\max_k\{S_k V_k\}$, and can be computed by the GNU Octave expression lambda_sat=1/**max**(S.∗V), which produces $\lambda_{\text{sat}} = 1.1\bar{1}$ as result. The expression S.∗V computes the vector of element-by-element products of S and V.

5.2 Closed network

We show in Fig. 2 a closed model which is based on the open model of Fig. 1. We have a fixed population of $N = 20$ requests which circulate through the service centers. Each request spends an average delay $Z = 5$ outside the system between service cycles. Z is also known as *think time* and is represented by the IS node in Fig. 2.

Again, we can define and solve the model with the following GNU Octave code:

```
p_back = 0.5; # back probability
i = 2:4; # range of Web servers
j = 5:6; # range of DB servers
P = zeros(6,6);
P(1,i) = 1/3;
P(i,j) = (1−p_back)/2;
P(i,1) = p_back;
P(j,i) = 1/3;
```

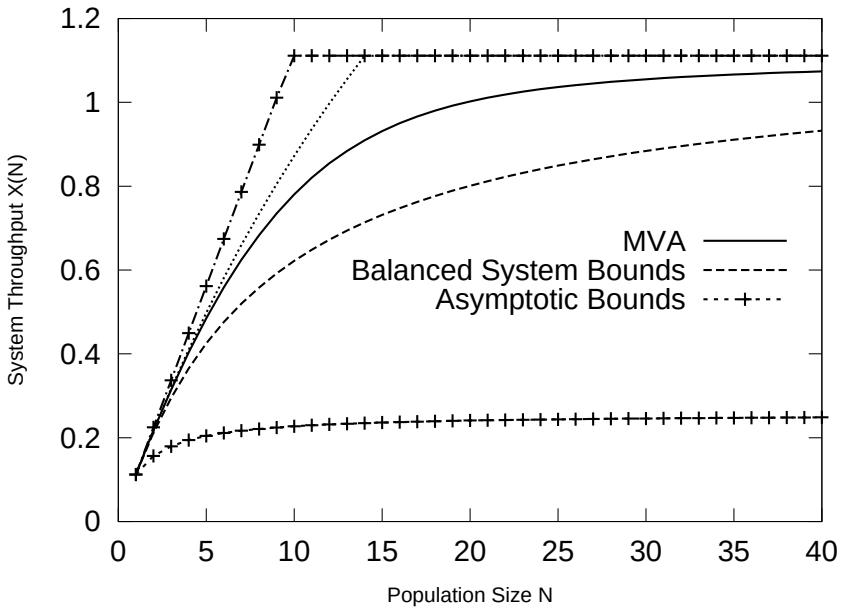

Figure 3. Comparison between AB, BSB and exact value of the throughput $X(N)$ of the closed network of Fig. 2 as a function of the population size N

```
S = [0.5  0.8  0.8  0.8  1.8  1.8];
V = qnvisits(P);
Z = 5; # Think Time
N = 20; # Population
m = ones(1,6); # m(k)=number of servers at center k
[U R Q X] = qnclosedsinglemva(N,S,V,m,Z);
```

The qnclosedsinglemva() function solves the given network using the MVA algorithm. The computed utilizations are $U_1 = 0.50112$, $U_2 = U_3 = U_4 = 0.53453$ and $U_5 = U_6 = 0.90202$.

5.3 Bottleneck Analysis

We can use the qnclosedab() and qnclosedbsb() functions to compute bounds on the system throughput X and response time R. If we consider the closed model defined above, bounds on the throughput can be computed as:

```
D = S.*V; # Service demands
[X_bsb_low X_bsb_up] = qnclosedbsb(N,D,Z); # Balanced System Bounds
[X_ab_low X_ab_up] = qnclosedab(N,D,Z); # Asymptotic Bounds
```

$D_k = S_k V_k$ is the service demand at center k. We plot in Fig. 3 the bounds and the exact throughput computed using MVA, for different values of the population size N.

5.4 Flow-equivalent centers

We now show how more complex analysis can be performed with qnetworks. Let us consider the closed model of Fig. 4(a). which is very similar to the one from Fig. 2 with the additional introduction of a capacity constraint: no more than M requests can be in the dashed region. Any request entering the fixed capacity region when M requests are already inside, must wait in a queue until a request leaves the region.

Models with capacity constraints have in general no PF solution. However, in this case it is possible to replace the fixed capacity region with a load-dependent service center [14], and solve the resulting model (which does have PF solution). More specifically, we proceed as follows:

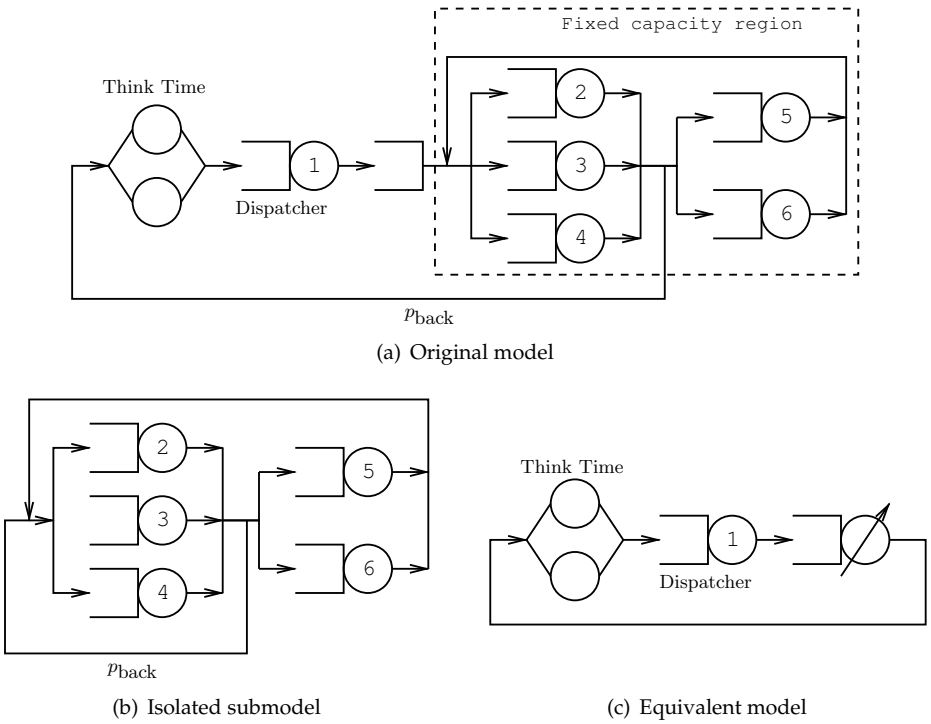

Figure 4. Closed model with capacity constraint

1. Define the complete model as in Section 5.2. Then, "short circuit" center 1 by setting its service time to zero (S(1)=0); we get the submodel in Fig. 4(b).

2. Solve the short-circuited submodel by computing the throughput $X_{\text{sub}}(n)$ along the removed node(s) as a function of the population size $n = 1, 2, \ldots M$. The computed value for $X_{\text{sub}}(n)$ can be used to derive the average service time $S_{\text{sub}}(n)$ of the flow-equivalent center which will replace the capacity constrained region. $S_{\text{sub}}(n)$ is defined as:

$$S_{\text{sub}}(n) = \begin{cases} 1/X_{\text{sub}}(n) & \text{if } 1 \leq n \leq M \\ 1/X_{\text{sub}}(M) & \text{if } M < n \leq N \end{cases}$$

   and can be computed with the following GNU Octave code:

```
Ssub = zeros(1,N); # Initialize to zero
M = 10; # Capacity constraint
for n=1:M
  [U R Q X] = qnclosedsinglemva(n,S,V);
  Ssub(n) = V(1)/X(1);
endfor
Ssub(M+1:N) = Ssub(M);
```

3. Build an equivalent model (see Fig. 4(c)) starting from the full model with the capacity constrained region replaced by a Flow-Equivalent Service Center (FESC). The service times for the FESC are those computed in the previous step. Let S_{kn} be the service time at center k when there are n requests; we have that $S_{1n} = 0.5$ and $S_{2n} = S_{\text{sub}}(n)$, for all n. The equivalent model is defined and solved with the following code:

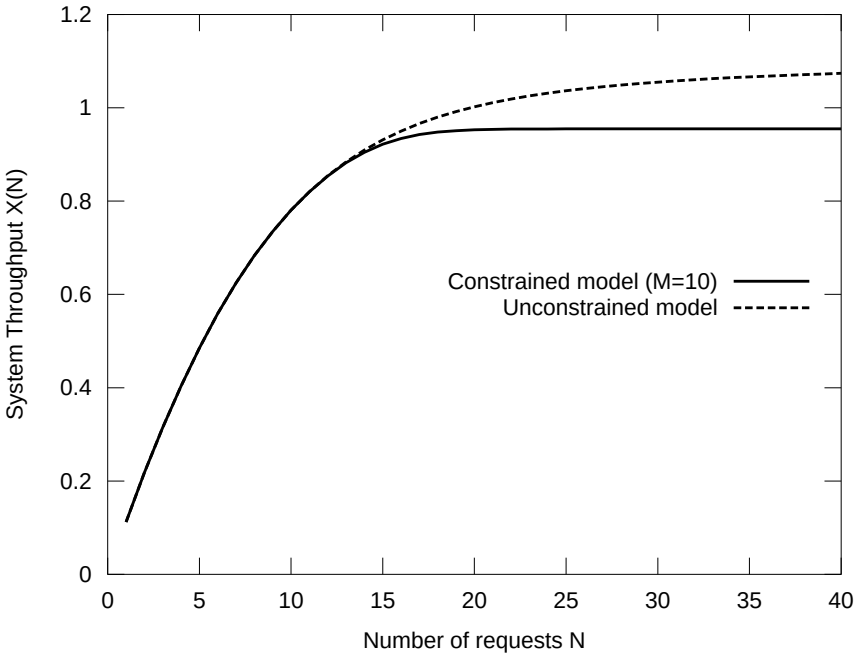

Figure 5. System Throughput $X(N)$ for the models of Fig. 2 and 4 as a function of the number of requests N.

```
S = [ 0.5*ones(1,N); Ssub ];
V = [1  1];
Z = 5;
[U R Q X] = qnclosedsinglemvald(N,S,V,Z);
```

By repeating the above for different values of the population size N we can produce the plot shown in Fig. 5. We show the system throughput $X(N)$ as a function of N. As expected, the system saturates shortly after the number of requests N exceeds the population constraint M.

6 Performance Considerations

We recall that qnetworks is entirely written as m-scripts running inside the GNU Octave interpreter. Despite this, performance of most of qnetworks functions are generally good. To give an example, we consider the qnclosedsinglemva() function implementing the MVA algorithm for single-class closed networks. These kind of networks are widely used in practice, so it is important to analyze them efficiently.

Fig. 6 illustrates the execution time of qnclosedsinglemva() for different values for the network size K and population N. The tests have been performed by creating a K server network, with random service times and visit counts. The tests have been performed on a Linux PC with an Intel Pentium 4 processor running at 2.4GHz with 1GB of RAM. We used GNU Octave version 3.2.3. For each combination of K and N, we consider the average execution time of 5 independent runs; for each run we build a new random network.

We observe that the largest network ($K = 2000$ servers and $N = 500$ requests) takes about half second to be analyzed on our machine. We also observe that for fixed K, the execution time increases linearly with the network size N. This is expected, as the computational complexity of MVA for single-class, load-independent service centers is $O(NK)$.

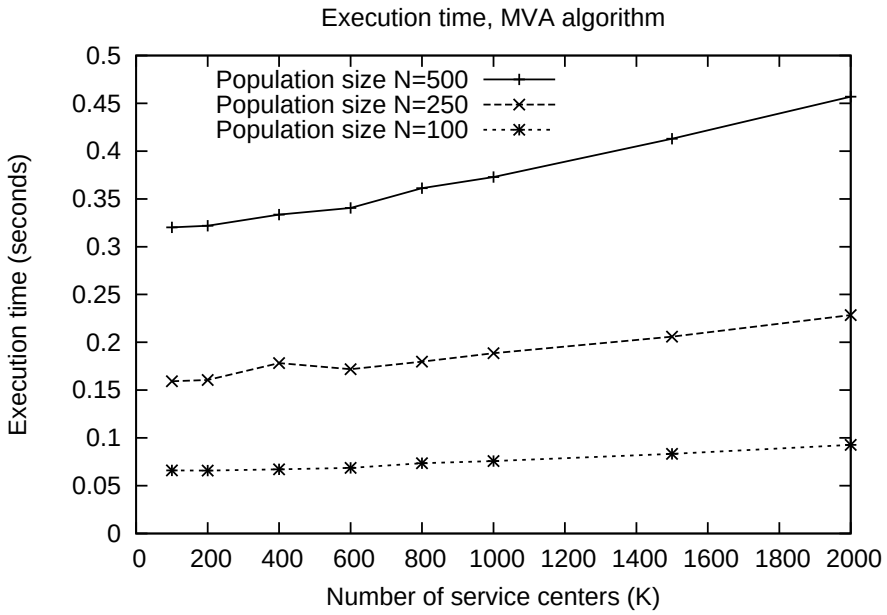

Figure 6. Execution time of the qnclosedsinglemva() function (in seconds, average of five measurements).

7 Conclusions

In this paper we described qnetworks, a QN analysis package for GNU Octave. After illustrating the main features of qnetworks, we gave some practical usage example showing how the Octave environment coupled with qnetworks can be used to solve QN models.

There is a vast literature of numerical solution algorithms for QNs, and we currently implemented some of the most important ones. We are extending qnetworks by including functions to evaluate additional types of non-exponential single-station queueing systems, as well as additional classes of QNs, including QNs with blocking as they have many practical applications.

qnetworks is available at `http://www.moreno.marzolla.name/software/qnetworks` and can be used, modified and distributed under the terms of the GNU General Public License (GPL) version 3.

References

[1] Ian F. Akyildiz. Mean value analysis for blocking queueing networks. *IEEE Transactions on Software Engineering*, 1(2):418–428, April 1988.

[2] V. Balsamo, S. De Nitto Personé and R. Onvural. *Analysis of Queueing Networks with Blocking*. Kluwer Academic Publishers, 2001.

[3] Forest Baskett, K. Mani Chandy, Richard R. Muntz, and Fernando G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2):248–260, 1975.

[4] Marco Bertoli, Giuliano Casale, and Giuseppe Serazzi. JMT: performance engineering tools for system modeling. *SIGMETRICS Perform. Eval. Rev.*, 36(4):10–15, 2009.

[5] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley, 1998.

[6] Jeffrey P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Comm. ACM*, 16(9):527–531, September 1973.

[7] G. Casale, R. R. Muntz, and G. Serazzi. Geometric bounds: a non-iterative analysis technique for closed queueing networks. *IEEE Transactions on Computers*, 57(6):780–794, June 2008.

[8] Kow C. Chang, Robert F. Gordon, Paul G. Loewner, and Edward A. MacNair. The Research Queuing Package Modeling Environment (RESQME). In Gerald W. Evans, Mansooreh Mollaghasemi, Edward C. Russell, and William E. Biles, editors, *Winter Simulation Conference*, pages 294–302. ACM Press, 1993.

[9] Peter J. Denning and Jeffrey P. Buzen. The operational analysis of queueing network models. *ACM Computing Surveys*, 10(3):225–261, September 1978.

[10] John W. Eaton. *GNU Octave Manual*. Network Theory Limited, 2002.

[11] William J. Gordon and Gordon F. Newell. Closed Queuing Systems with Exponential Servers. *Operations Research*, 15(2):254–265, 1967.

[12] James R. Jackson. Jobshop-like queueing systems. *Manage. Sci.*, 50(12 Supplement):1796–1802, 2004.

[13] Leonard Kleinrock. *Queueing Systems: Volume I–Theory*. Wiley Interscience, New York, 1975.

[14] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice Hall, 1984.

[15] *MATLAB*. The MathWorks Inc., Natick, Massachussets, 2003.

[16] M. Reiser and S. S. Lavenberg. Mean-value analysis of closed multichain queuing networks. *Journal of the ACM*, 27(2):313–322, April 1980.

[17] Robin Sahner, Kishor S. Trivedi, and Antonio Puliafito. *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*. Kluwer Academic Publishers, 1996.

[18] Charles H. Sauer, M. Reiser, and Edward A. MacNair. RESQ: a package for solution of generalized queueing networks. In *AFIPS National Computer Conference*, volume 46 of *AFIPS Conference Proceedings*, pages 977–986. AFIPS Press, 1977.

[19] P. Schweitzer. Approximate analysis of multiclass closed networks of queues. In *Proc. Int. Conf. on Stochastic Control and Optimization*, pages 25–29, June 1979.

[20] Herb Schwetman. Testing network-of-queues software. Technical Report CSD-TR-330, Purdue University, January 1 1980.

[21] Giuseppe Serazzi. Performance Evaluation Modelling with JMT: learning by examples. Technical Report 2008.09, Politecnico di Milano, 2008.

[22] Michel Véran and Dominique Potier. QNAP2: A portable environment for queueing systems modelling. Technical Report 314, Institut National de Recherche en Informatique et en Automatique, June 1984.

[23] J. Zahorjan, K. C. Sevcick, D. L. Eager, and B. I Galler. Balanced job bound analysis of queueing networks. *Comm. ACM*, 25(2):134–141, February 1982.