QMLE: a Methodology for Statistical Inference of Service Demands from Queueing Data

WEIKUN WANG, Department of Computing, Imperial College London, UK
GIULIANO CASALE, Department of Computing, Imperial College London, UK
AJAY KATTEPUR, PERC, TCS Research, Mumbai, India
MANOJ K. NAMBIAR, PERC, TCS Research, Mumbai, India

Estimating the demands placed by services on physical resources is an essential step for the definition of performance models. For example, scalability analysis relies on these parameters to predict queueing delays under increasing loads. In this paper, we investigate maximum likelihood (ML) estimators for demands at load-independent and load-dependent resources in systems with parallelism constraints. We define a likelihood function based on state measurements and derive necessary conditions for its maximization. We then obtain novel estimators that accurately and inexpensively obtain service demands using only aggregate state data. With our approach, and also thanks to approximation methods for computing marginal and joint distributions for the load-dependent case, confidence intervals can be rigorously derived, explicitly taking into account both topology and concurrency levels of the services. Our estimators and their confidence intervals are validated against simulations and real system measurements for two multi-tier applications, showing high accuracy also in the presence of load-dependent resources.

CCS Concepts: • **Mathematics of computing** → **Maximum likelihood estimation**;

Additional Key Words and Phrases: Estimation, service demand, maximum likelihood, queueing networks

ACM Reference Format:
Weikun Wang, Giuliano Casale, Ajay Kattepur, and Manoj K. Nambiar. 2000. QMLE: a Methodology for Statistical Inference of Service Demands from Queueing Data. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 0, 0, Article 00 (2000), 30 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Guaranteeing Quality-of-Service (QoS) is an important concern for cloud service providers in order to meet customer expectations. Analytical models based on queueing theory can be used for performance analysis, resource allocation, and performance prediction and therefore can support engineers in coping with these challenges [23]. Closed queueing networks, in particular, are often used for software systems [23] since real applications are layered and

Authors' addresses: Weikun Wang, Department of Computing, Imperial College London, UK, weikun.wang11@imperial.ac.uk; Giuliano Casale, Department of Computing, Imperial College London, UK, g.casale@imperial.ac.uk; Ajay Kattepur, PERC, TCS Research, Mumbai, India, ajay.kattepur@tcs.com; Manoj K. Nambiar, PERC, TCS Research, Mumbai, India, m.nambiar@tcs.com.

2376-3639/2000/0-ART00 $15.00
https://doi.org/0000001.0000001

thus operate under pooling constraints that limit the maximum parallelism level at each layer and the underpinning hardware resources. For instance, Layered Queueing Networks (LQN), which describe the nested structure of softwares, are based on closed queueing networks and have applications in a wide range of distributed computer systems [12]. Even though solution methods for such models have been systematically investigated, the inference of their parameters from real measurements is often difficult, but still essential to obtain accurate performance predictions [37]. Among the required input parameters, the *service demand*, i.e., the cumulative time a request seizes from a resource excluding contention overheads, is particularly challenging to estimate. Since demands need to exclude contention overheads, they can require fine-grained application profiling that typically introduces monitoring overheads in the system. Statistical inference addresses this issue by determining demand estimates from high-level measurements of throughputs, resource utilizations, queue-length sizes, or request response times [6, 17, 21, 22, 27, 29, 43] .

Linear regression methods that use utilization and request throughputs have been traditionally used for this purpose, however these methods are known to suffer from a variety of issues [17], such as multi-collinearity that leads to biased estimates [15]. Recently, the idea of exploiting response time measurements has also been investigated [17, 29, 43]. Nonetheless, collecting response times may pose unacceptably large overheads in production applications, in particular if one needs to track multiple requests across tiers. Moreover, response-time based estimation is sensitive to the scheduling policies used at the resources, thus requiring policy-specific inference techniques [17].

In this paper, we instead investigate a state-based approach to obtain demand estimates. We assume that the software service has a finite threading limit and runs on a set of parallel or distributed resources. Stemming from this assumption, we abstract the application using a closed queueing network model where we represent the state as the number of concurrently executing jobs at each resource. Since state samples, i.e., measurements of the number of executing requests at each resource, are relatively easy to collect in real systems, we investigate the service demand estimation problem from state measurements. Differently from other measures such as response times, in time-shared resources, such as CPUs, queue-length samples are weakly sensitive to scheduling policies and to the moments of the service demand higher than the mean, as a consequence of the insensitivity properties of the $M/GI/1$-PS queue. Therefore, state based demand estimation appears more widely applicable than response time based estimation.

In order to infer demands from state samples, we use the equilibrium distribution of product-form closed queueing networks to formulate the estimation as a likelihood maximization problem. We then develop analytical conditions required for likelihood maximization. Such conditions indicate that only aggregate state data is required to obtain demands, either mean queue-length samples or samples of marginal state probabilities, depending on the resource type. Stemming from this result, we propose a tight closed-form approximation for the maximum likelihood estimator. Moreover we derive efficient expressions for the Hessian matrix of the likelihood function and use them to obtain the confidence intervals associated to the estimated demands.

Although most existing demand estimation methods are limited to load-independent models, i.e., models where the service demand remains constant irrespectively of the number of requests running at a resource, our conditions to identify maximum likelihood estimators readily extend to load-dependent models, where demands are functions of the queue-length size. Real-world systems typically exhibit a load dependent behavior, often as a result of job parallelism. Therefore the ability to estimate how a request demand varies with the

load is essential for predicting performance of real applications. We illustrate efficiency and accuracy of the proposed estimators using data from simulation and experiments on two multi-tier applications, Apache OFBiz and JPetStore.

This paper generalizes our preliminary work in [41], adding in particular an approximate method for computing marginal probabilities and second-order joint probabilities in load-dependent closed queueing networks, derived from the Distribution Analysis by Chain (DAC) algorithm [11], that improves the computational efficiency of load-dependent estimation, often by orders of magnitude. We also extend the validation in [41] with a novel case study on Apache OFBiz.

The rest of the paper is organized as follows. Section 2 reviews previous work on service demand estimation. Section 3 gives technical background. A motivating example is given in Section 4. Section 5 and 6 present the proposed demand estimation algorithms for load-independent and load-dependent models, respectively. Validations and experimental case studies are given in sections 7 and 8. A conclusion is given in Section 9.

2 RELATED WORK

Existing work on resource demand estimation often focuses on statistical inference from CPU utilization, throughputs and response time measurements. In particular, methods for regressing CPU utilization and throughput to obtain service demand have gained much attention [27, 32, 33, 43]. Later [6] has proposed an approach for robust demand estimation, based on a Least Trimmed Squares regression technique. However regression methods suffer from known problems, such as multi-collinearity [15] that can lead to biased estimates. To overcome this shortcoming, several algorithms have been proposed. Kalman filters [39, 42, 44] have proven effective in parameter tracking and offer theoretical guarantees under certain assumptions, but they do not typically achieve the lowest mean demand estimation errors among existing estimation methods [37]. Other methods including clustering [8, 9], pattern recognition [13, 16], independent component analysis [36] have also been used in multiclass analysis of service demands. However, these methods require CPU utilization measurements that are not always available at the granularity of each individual service class and that can be difficult to measure accurately inside virtual machines. Moreover, several studies indicate that utilization-based demand estimation can incur fairly large errors [17, 37]. Compared to these algorithms, our proposed methods rely on queue-length samples, which can be collected by monitoring the number of running worker threads in a system and the class of the running requests.

Besides CPU utilization, response times have also been used for estimating service demands. The work in [21] defines a quadratic programming using end-to-end response time and CPU utilization together with request throughputs. The methods introduced in [17] and [34] also employ response time values, but they only apply to FCFS servers. The works in [2, 22] focus on estimating demands for some load-independent queueing models through optimization programs that use response time data. Recent work [29] also proposes new algorithms based on regression and maximum likelihood methods for response time data. However, collecting response data may pose additional overhead to the system compared to queue-length monitoring, since both arrivals and departures need to be continuously tracked and response time based fitting algorithms are more sensitive to the scheduling policy.

State samples have also been exploited in previous work for demand estimation. The study in [40] uses Gibbs sampling and Bayesian estimation methods to obtain service demands from queue-length data. The proposed algorithm, called GQL, also allows for prior information in the estimation. However, GQL is computationally expensive, with running times often

exceeding tens of minutes or even hours on large models. This makes the technique difficult to apply to online systems, besides being much slower than prior art methods. The authors in [38] have developed an algorithm based on Bayesian inference, which has been shown to be robust to missing data. A Ornstein-Uhlenbeck diffusion is used for demand estimation in [34] also using queue-length samples. Compared to the present work, the methods in [38] and [34] are limited to open models, while we here focus on closed models, which therefore can take into account thread parallelism limits commonly in place in complex software.

As mentioned, our method offers confidence intervals on demand estimates. Such intervals can be readily obtained in inference methods based on linear regression, which provide confidence intervals under assumptions on the distribution of the residuals [27, 32, 33, 43]. However, since these estimators are applied to a single resource, they cannot take into account the covariance between measurements arising due to the inter-dependencies between resources, which are instead taken into account by our confidence intervals.

Recent work in [1] proposes a method that can estimate online demands in single-class models using profiling of the cumulative CPU execution time. Compared to this work, our technique applies also to multiclass systems. A method for obtaining confidence interval is given in [20], however this is also limited to single-class models only and does not apply to load-dependent networks. [14] propose a scalable approach to estimate resource demand through linear programming, which can take into account workload mixes. This method has not been yet generalized to load-dependent networks, which are instead addressed in the present work. The work in [18] is to our knowledge the first attempt to estimate the demand in a load-dependent queueing network, however it only applies to open systems and requires a-priori knowledge of a parametric expression for the load-dependent function. Instead, the estimation technique presented in this paper applies to closed models. Lastly, [26] proposes a method to model service demands a s function of the workload by collecting measurements with single user tests. However, this method is limited to CPUs and it is not always possible to run single user tests in production systems.

3 REFERENCE MODEL

In this section we introduce the reference model used for estimation. A summary of the main notation is given in Table 1. We consider distributed services composed by a network of parallel or distributed resources, e.g., parallel cores, web servers. We assume that the system may be modelled as a product-form closed queueing network [3]. These are closed networks of $-/GI/1$-PS, $-/M/1$-FCFS, or $-/GI/\infty$ (i.e., pure delay) stations, with multiple service classes, probabilistic routing and a finite population of jobs for each class.

In performance engineering, the $-/GI/1$-PS queueing system is a common abstraction of resources that operate under time-sharing scheduling, such as CPUs. A network of such queues can be used either to model contention within each core of a single computer system, in which case a $-/GI/1$-PS queue may represent a single core, or to describe distributed systems, where instead each $-/GI/1$-PS queue represents a server. In the latter case, to model that a single server features multiple cores, one may either approximately use a load-independent model with a rate proportional to the number of cores in the server, or model the queue service process as load-dependent, as we show in our case studies.

In real-world applications, either running on a single machine or adopting a distributed architecture, it is common to implement admission control mechanisms to bound the number of jobs in execution. This is done to avoid memory swapping and performance degradation arising due to an excessively large number of jobs contending the resources. In these situations, queue-lengths at the resources are upper bounded by the maximum parallelism level, and

closed queueing network models, rather than open models, will be better able to capture heavy-load scenarios where such a constraint becomes a factor limiting performance. This is done by setting the number of jobs in the model to match the parallelism constraint. In the case where the constraint does not differentiate between classes of services, the per-class job populations in the closed model are usually chosen to capture the average mix of job classes seen in the system.

We assume the queueing network model to have R classes of jobs, M queues, a think time of θ_{0j} for job class j, a service demand θ_{ij} at queue i for class j, and a population of N_j jobs of class j. The population vector is denoted by $\boldsymbol{N} = (N_j)$. Note that the service demand θ_{ij} represents the *cumulative* time that a class-j job spends at resource i in between service completions. Since closed queueing networks with different routing probabilities, but identical demands, have the same utilization and end-to-end performance metrics, it is sufficient to obtain the demands θ_{ij}, the think times, and the number of jobs within each class to parameterize a performance model solvable by mean-value analysis [31].

A station is either a delay node where jobs spend their think time or a queue. Indexes range in $1 \leq i, k \leq M, 1 \leq j, h \leq R$. In the presence of class switching, the model can be transformed into an equivalent queueing network where all jobs belonging to the same chain are placed into the same class [3]; thus we assume that jobs cannot switch class. When needed, we will explicit the dependence of the above metrics on the demand vector $\boldsymbol{\theta} = (\theta_{01}, \ldots, \theta_{MR})$.

Let n_{0j} be the total number of class j jobs in thinking state and let n_{ij} be the number of class-j jobs at station i. Define $n_i = \sum_{j=1}^{R} n_{ij}$ as the total number of jobs at station i. The probability of observing state $\boldsymbol{n} = (n_{01}, \ldots, n_{0R}, n_{11}, \ldots, n_{1R}, \ldots, n_{MR})$ at equilibrium is

$$\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi}, \boldsymbol{N}) = \frac{1}{g(\boldsymbol{\chi}, \boldsymbol{N})} \left(\prod_{j=1}^{R} \frac{\theta_{0j}^{n_{0j}}}{n_{0j}!} \right) \prod_{i=1}^{M} n_i! \prod_{j=1}^{R} \frac{\theta_{ij}^{n_{ij}}}{n_{ij}!} \prod_{u=1}^{n_i} \gamma_i(u), \tag{1}$$

where $\gamma_i(u)$ is the load-dependent function that scales the demand for station i when its queue-length is u, $\boldsymbol{\gamma} = (\gamma_i(u))$, $\boldsymbol{\chi} = (\boldsymbol{\theta}, \boldsymbol{\gamma})$, and $g(\boldsymbol{\chi}, \boldsymbol{N})$ is the normalizing constant ensuring that $\sum_{\boldsymbol{n} \in \mathcal{S}} \mathbb{P}(\boldsymbol{n}|\boldsymbol{\theta}) = 1$, being $\mathcal{S}$ the state space of a set of vectors $\boldsymbol{n}$, where $\sum_{i=0}^{M} n_{ij} = N_j, n_{ij} \geq 0$. The case $\gamma_i(u) = 1, 1 \leq u \leq n_i$, in which each demand is independent of the station queue-length state, is referred to as the *load-independent* case, whereas the general case is referred to as the *load-dependent* case. For ease of readability, in the following we will omit the population vector $\boldsymbol{N}$ unless required by the derivations.

In order to estimate demands, let us consider a set of independent[1] state samples $\boldsymbol{n}^l \in \boldsymbol{D}$, $\boldsymbol{D}$ being a dataset of empirical observations of vectors $\boldsymbol{n}^l$, $1 \leq l \leq L$. The problem of estimating the demands $\boldsymbol{\theta}$ and scaling factors $\boldsymbol{\gamma}$ may be formulated as obtaining a maximum likelihood (ML) estimator

$$\hat{\boldsymbol{\chi}} = (\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}}) = \arg\max_{\boldsymbol{\chi} \in \boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\chi}|\boldsymbol{D}) = \arg\max_{\boldsymbol{\chi} \in \boldsymbol{\Theta}} \prod_{l=1}^{L} \mathbb{P}(\boldsymbol{n}^l|\boldsymbol{\chi}) \tag{2}$$

where $\mathcal{L}(\cdot|\boldsymbol{D})$ is the likelihood function for the dataset $\boldsymbol{D}$, $\mathbb{P}(\cdot|\boldsymbol{\chi})$ is defined as in (1), and $\boldsymbol{\Theta}$ is the parameter space composed by the candidate demands $\boldsymbol{\theta}$ and scaling factors $\boldsymbol{\gamma}$.

[1] We have frequently observed in real web traces that the assumption of independence of state samples is typically acceptable for estimation purposes, with the autocorrelation function of samples quickly decreasing as the time spacing between samples increases, see [40] for a recent analysis of this problem.

Table 1. Summary of main notations used in this paper

R	the number of job classes
M	the number of queues
N_j	the number of class-j jobs
N	the total number of jobs
$\boldsymbol{N}$	population vector with entries N_j, $j = 1, \ldots, R$
θ_{0j}	the think time of class-j jobs
θ_{ij}	the service demand of class-j jobs at the ith station
$\boldsymbol{\theta}$	service demand matrix with elements θ_{ij} $i = 1, \ldots, M$, $j = 1, \ldots, R$
$\gamma_i(u)$	the load-dependent scaling factors
$\boldsymbol{\gamma}$	scaling factor matrix with elements $\gamma_i(u)$ $i = 1, \ldots, M$, $u = 1, \ldots, N$
n_{ij}	the number of class-j jobs at the ith station
n_i	the total number of jobs at the ith station
$\boldsymbol{D}$	dataset of empirical state samples
L	the number of samples in D
$\mathcal{L}(\cdot\|\boldsymbol{D})$	likelihood function conditional on the dataset $\boldsymbol{D}$
$n_{ij}^{(l)}$	the number of class-j jobs at the ith station of the lth observed sample
x_{ij}	the mean queue-length of class-j jobs
$\widetilde{x}_{ij}$	empirical mean queue-lengths over the dataset $\boldsymbol{D}$
$\boldsymbol{\Theta}$	the parameter space for $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$
$\mathbf{1}_j$	vector of zeros with a one in the jth position
$g(\boldsymbol{\chi}, \boldsymbol{N})$	the normalizing constant of the steady state probability
$g^{-i}(\boldsymbol{\chi}, \boldsymbol{N})$	the normalizing constant of the original model with the ith queue removed
$\mathcal{S}$	the state space of the model

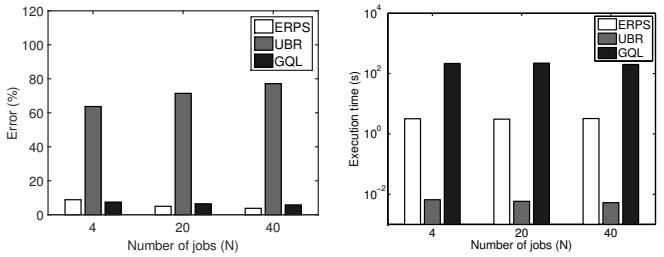
4 MOTIVATING EXAMPLE

We now give a motivating example that compares three state-of-the-art demand estimation algorithms and illustrates their limitations. We consider three methods that are representative of different paradigms for demand inference: utilization-based regression (UBR) (e.g., [43]), Gibbs Sampling for Queue Lengths (GQL) [40] and Extended Regression for Processor Sharing (ERPS) [29]. UBR is based on multivariate linear regression of CPU utilization against request throughput. GQL combines Bayesian estimation and Gibbs sampling to obtain service demands from queue length data; GQL is an iterative algorithm along each dimension of the demand vector and thus computationally expensive. ERPS is a regression-based method that relies on response time and arrival queue length measurements.

We generate random queueing models with $M = 2$ queues and $R = 4$ classes of requests and assume that the total number of users N varies in $\{4, 20, 40\}$, demands vary in $[0, 1]$. We generate 80 models by randomly choosing N_j and θ_{ij} using a uniform distribution.

The think time is assumed known and it is set in all random models to $\theta_{0j} = 1, \forall j$. All service processes are load independent. The required data for each algorithm is generated via simulation using the methodology described later in Section 7. Simulation is carried out until 5×10^5 service completions are observed in the system.

Figure 1 illustrates the mean relative absolute error and the execution time for the above algorithms. It can be noted that UBR shows a low estimation accuracy. Figure 1(a) instead



(a) Estimation Error (b) Execution time

Fig. 1. Comparison of estimation methods presented in prior work

shows that GQL and ERPS achieve good accuracy, but Figure 1(b) reveals that the execution time of GQL is fairly large. ERPS performs well in terms of both accuracy and execution time, nonetheless it requires response time data, which assumes the ability to track the state of individual requests across the distributed system.

Overall, the results in Figure 1 indicate that existing methods are either accurate or inexpensive, but seldom they feature both properties. In the rest of the paper we develop a novel family of estimators that are at the same time accurate and computationally efficient.As shown later in the experimental results in Section 7.1, which are carried out on very similar models, the accuracy of the proposed estimators is comparable to the one of GQL and ERPS, but a fraction of the time being a closed form expression.

5 LOAD-INDEPENDENT RESOURCES

In this section, we focus on demand estimation for load-independent models, thus we ignore scaling factors since $\gamma_i(u) = 1$, $\forall i, u$, and use the short-hand notation for the likelihood $\mathcal{L}(\boldsymbol{\theta}) \equiv \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{D})$. We thus focus on deriving estimators for the service demand vector $\boldsymbol{\theta}$ only and their confidence intervals. Thus, in this section the $\boldsymbol{\chi}$ vector is replaced by $\boldsymbol{\theta}$. Finally, we introduce a closed-form approximate formula that simplifies the task of obtaining an accurate estimate.

5.1 Necessary conditions

We begin by assuming that the parameter space $\boldsymbol{\Theta}$ is a compact set and that the think times θ_{0j} are known for each class j and strictly positive. Under the above conditions, it is simple to show that the likelihood function is continuous and that a ML estimator exists [28]. We also assume that $\boldsymbol{\Theta}$ is large enough for the true value of the demand estimate $\boldsymbol{\theta}^*$ to be a point in the interior of this set. A consequence of this assumption is that our results do not cover the estimation of demands with value $\theta^*_{ij} = 0$. This is equivalent to say that we assume a-priori knowledge of what classes of jobs can visit a given resource, which seems a realistic assumption in many practical situations; for classes that do not visit a resource the analyst can simply set their service demand to zero.

Let $x_{ij}(\hat{\boldsymbol{\theta}}) \equiv x_{ij}(\hat{\boldsymbol{\theta}})$ be the mean queue-length of class-j jobs at node i, as predicted by a product-form queueing network model parameterized with demands $\hat{\boldsymbol{\theta}}$ and population $\boldsymbol{N}$. Under the above assumptions, we can give the following characterization of the ML estimator in (2).

THEOREM 1. *Given a dataset $\boldsymbol{D}$, a necessary condition for an interior point of $\boldsymbol{\Theta}$ to be an ML estimator $\hat{\boldsymbol{\theta}}$ of the service demand vector $\boldsymbol{\theta}$ is that*

$$x_{ij}(\hat{\boldsymbol{\theta}}) = \widetilde{x}_{ij}, \qquad \forall i, j,$$

where $\widetilde{x}_{ij} = \widetilde{x}_{ij}(\boldsymbol{D}) = \sum_{l=1}^{L} n_{ij}^{(l)}/L$ are the empirical mean queue-lengths calculated over the dataset $\boldsymbol{D}$.

PROOF. A sufficient condition for existence of a ML estimator is that the parameter space is compact and the likelihood function continuous. We have assumed $\boldsymbol{\Theta}$ to be compact and since the normalising constant is continuous in $\boldsymbol{\theta}$ and $g(\boldsymbol{\theta}) \neq 0$ since $\theta_{0j} > 0$, $\forall j$, then $\mathcal{L}(\boldsymbol{\theta})$ is also continuous in $\boldsymbol{\theta}$.

Given existence, we now determine the ML estimator. Consider the following relationship proved in [10]

$$\frac{\partial g(\boldsymbol{\theta})}{\partial \theta_{ij}} = \frac{x_{ij}(\boldsymbol{\theta})}{\theta_{ij}} g(\boldsymbol{\theta}) \tag{3}$$

We now differentiate the log-likelihood with respect to θ_{ij}

$$\begin{aligned} \frac{\partial \log \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_{ij}} =& \frac{\partial}{d\theta_{ij}} \left(\sum_{l=1}^{L} \sum_{i=1}^{M} \log n_i^{(l)}! + \sum_{r=1}^{R} n_{ir}^{(l)} \log \theta_{ir} - \log n_{ir}^{(l)}! - \log g(\boldsymbol{\theta}) \right) \\ =& \frac{n_{ij}^{(l)}}{\theta_{ij}} - \frac{1}{g(\boldsymbol{\theta})} \frac{\partial g(\boldsymbol{\theta})}{\partial \theta_{ij}} \end{aligned} \tag{4}$$

Plugging (3) in the expression, we find the expression

$$\frac{\partial \log \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_{ij}} = \sum_{l=1}^{L} \left(\frac{n_{ij}^{(l)}}{\theta_{ij}} - \frac{x_{ij}(\boldsymbol{\theta})}{\theta_{ij}} \right) = L \frac{\widetilde{x}_{ij} - x_{ij}(\boldsymbol{\theta})}{\theta_{ij}} \tag{5}$$

A stationarity point needs thus to satisfy $x_{ij}(\hat{\boldsymbol{\theta}}) = \widetilde{x}_{ij}$.

□

Note that the predicted mean queue lengths $x_{ij}(\hat{\boldsymbol{\theta}})$ can be computed using the MVA algorithm [31].

The main contribution of Theorem 1 is to provide a theoretical case for estimating demands in load-independent models by simply matching theoretical predictions of mean queue-lengths to the observed mean values in the real system. According to Theorem 1 this does not introduce a bias. Furthermore, the theorem states that the demand estimation depends only on the *mean* queue-length, even though the maximum-likelihood function depends on the measured states of the distributed system. That is, if one can find a vector $\boldsymbol{\theta}$ that generates by the MVA algorithm mean queue-length predictions that are identical to the observed values, then this vector will satisfy the necessary condition to be a ML estimator.

Since mean queue-lengths uniquely determine throughputs, response times and utilizations [31], even if Theorem 1 is only necessary for $\hat{\boldsymbol{\theta}}$ to be the ML estimator, the condition of the theorem ensures that $\hat{\boldsymbol{\theta}}$ will achieve correct mean performance predictions that reproduce the measurements. Hence, while in principle several vectors $\boldsymbol{\theta}$ may satisfy the same necessary condition, any of these appears a suitable choice for software performance prediction purposes. In order to prove that Theorem 1 gives a sufficient condition, one would need to show that a given set of queue-length values $x_{ij}(\boldsymbol{\theta})$ can be obtained by a single vector $\boldsymbol{\theta}$. This appears intuitive, but we are not aware of a formal characterization of this property

for product-form models, presumably due to the non-linearity of the MVA equations that prevents this derivation also in our case.

In order to provide a deeper analysis of the demand vectors obtained from (1), we derive the expression of the Hessian matrix for the underpinning closed queueing network that can be used to verify that a candidate vector is indeed a local maximum for (2). We denote by $\boldsymbol{H}(\hat{\boldsymbol{\theta}})$ the Hessian matrix of the log-likelihood and use the following shorthand notation

$$\boldsymbol{H}(\hat{\boldsymbol{\theta}})_{ij,kh} = \left.\frac{\partial^2 \log \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_{ij} \partial \theta_{kh}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$$

We now give the main result for the Hessian matrix. Let $x_{kh}^{+k}(\hat{\boldsymbol{\theta}}, \boldsymbol{N} - \mathbf{1}_j)$ be the mean queue length in a model obtained by adding an identical replica of queue k to the closed network under study and by removing a job of class j from the population vector. We take the convention that $x_{kh}^{+k}(\hat{\boldsymbol{\theta}}, \boldsymbol{N} - \mathbf{1}_j)$ also includes the queue-length of the replica station, whereas $x_{kh}^{+i}(\hat{\boldsymbol{\theta}}, \boldsymbol{N} - \mathbf{1}_j)$ for $k \neq i$ does not.

THEOREM 2. *The Hessian matrix of* $\log \mathcal{L}(\boldsymbol{\theta})$ *evaluated at* $\hat{\boldsymbol{\theta}}$ *is a* $MR \times MR$ *matrix with elements*

$$\boldsymbol{H}(\hat{\boldsymbol{\theta}})_{ij,kh} = \begin{cases} \dfrac{L}{\hat{\theta}_{kh}^2}(x_{kh}(\hat{\boldsymbol{\theta}})(x_{kh}(\hat{\boldsymbol{\theta}}) - x_{kh}^{+k}(\hat{\boldsymbol{\theta}}, \boldsymbol{N} - 1_j)) - \widetilde{x}_{kh}), & i = k, j = h \\ \dfrac{L x_{ij}(\hat{\boldsymbol{\theta}})}{\hat{\theta}_{ij}\hat{\theta}_{kh}}(x_{kh}(\hat{\boldsymbol{\theta}}) - x_{kh}^{+i}(\hat{\boldsymbol{\theta}}, \boldsymbol{N} - \mathbf{1}_j)), & \textit{otherwise} \end{cases}$$

where $\boldsymbol{N} = (N_1, \ldots, N_R)$ *and* L *is the total number of samples.*

PROOF. From (5) we have

$$\frac{\partial \log \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_{ij}} = \frac{1}{\mathcal{L}(\boldsymbol{\theta})}\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_{ij}} = L\frac{\widetilde{x}_{ij} - x_{ij}(\boldsymbol{\theta})}{\theta_{ij}} \tag{6}$$

We now note that $x_{ij}(\boldsymbol{\theta}) = \theta_{ij} g^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j)/g(\boldsymbol{\theta})$, where 1_j is a vector of zeros but a one in the j-th position and $g^{+i}(\cdot)$ refers to a model with an additional station identical to queue i in the original model (i.e., with the same demands as node i) [19, eq. (12)].

If $i = k$ and $j = h$, we then have

$$\begin{aligned} \frac{\partial x_{ij}(\boldsymbol{\theta})}{\partial \theta_{ij}} &= \frac{g^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - 1_j)}{g(\boldsymbol{\theta})} - \frac{\theta_{ij} g^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j) x_{ij}(\boldsymbol{\theta})}{\theta_{ij} g(\boldsymbol{\theta})} + \frac{\theta_{ij}}{g(\boldsymbol{\theta})}\frac{x_{ij}^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j)}{\theta_{ij}} g^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j) \\ &= \frac{x_{ij}(\boldsymbol{\theta})}{\theta_{ij}}(1 + x_{ij}^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j) - x_{ij}(\boldsymbol{\theta})) \end{aligned} \tag{7}$$

For the other cases, we find with similar passages

$$\begin{aligned} \frac{\partial x_{ij}(\boldsymbol{\theta})}{\partial \theta_{kh}} &= \frac{\theta_{ij}}{g(\boldsymbol{\theta})}\frac{x_{kh}^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j)}{\theta_{kh}} g^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j) - \frac{\theta_{ij} g^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j) x_{kh}(\boldsymbol{\theta})}{\theta_{kh} g(\boldsymbol{\theta})} \\ &= \frac{x_{ij}(\boldsymbol{\theta})}{\theta_{kh}}(x_{kh}^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j) - x_{kh}(\boldsymbol{\theta})) \end{aligned} \tag{8}$$

Combining (7) and (8) with (5) and the definition of the Hessian matrix, the diagonal elements of the Hessian are

$$\begin{aligned}\boldsymbol{H}(\boldsymbol{\theta})_{ij,ij} &= \frac{\partial^2 \log \mathcal{L}(\boldsymbol{\theta})}{\partial^2 \theta_{ij}} = L\frac{x_{ij}(\boldsymbol{\theta}) - \widetilde{x}_{ij}}{\theta_{ij}^2} - \frac{L}{\theta_{ij}}\frac{\partial x_{ij}(\boldsymbol{\theta})}{\partial \theta_{ij}} \\ &= L\frac{x_{ij}(\boldsymbol{\theta})(x_{ij}(\boldsymbol{\theta}) - x_{ij}^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j)) - \widetilde{x}_{ij}}{\theta_{ij}^2}\end{aligned}$$

The non-diagonal elements are

$$\begin{aligned}\boldsymbol{H}(\boldsymbol{\theta})_{ij,kh} &= \frac{\partial^2 \log \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_{ij}\partial \theta_{kh}} = -\frac{L}{\theta_{ij}}\frac{\partial x_{ij}(\boldsymbol{\theta})}{\partial \theta_{kh}} \\ &= \frac{L x_{ij}(\boldsymbol{\theta})}{\theta_{ij}\theta_{kh}}(x_{kh}(\boldsymbol{\theta}) - x_{kh}^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j))\end{aligned}$$

Setting $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ completes the proof. □

Using the Hessian matrix at $\hat{\boldsymbol{\theta}}$, we can easily check if the generated estimate is a local minimum, a local maximum or a saddle point. In particular, if $\boldsymbol{H}(\hat{\boldsymbol{\theta}})$ is invertible and positive definite, i.e. all its eigenvalues are positive, then $\hat{\boldsymbol{\theta}}$ is a local minimum. If $\boldsymbol{H}(\hat{\boldsymbol{\theta}})$ is negative definite, i.e. all its eigenvalues are negative, then $\hat{\boldsymbol{\theta}}$ is a point of local maximum and therefore a local maximum likelihood estimator.

We also remark that Theorem 1 does not specify how one can find the demand vector $\hat{\boldsymbol{\theta}}$, since the expression of $\hat{\boldsymbol{\theta}}$ is given there in implicit form. An explicit approximation for $\hat{\boldsymbol{\theta}}$ is developed later in Section 5.3.

5.2 Exact confidence intervals

In this section, we assume that the vector $\hat{\boldsymbol{\theta}}$ in Theorem 1 has been obtained and we give a characterization of the resulting confidence intervals. As shown in the proof of the next theorem, this result follows from the fact that we have found that the Fisher information matrix $\boldsymbol{I}$ can be explicitly computed for a closed queueing network, since this is simply the negative Hessian matrix at $\hat{\boldsymbol{\theta}}$. For the confidence interval, we assume the critical value c is given, which determines the confidence level (e.g., $c = 1.96$ is used to obtain 95% confidence level).

Corollary 1. *Assume that $\hat{\boldsymbol{\theta}}$ satisfies the standard regularity conditions for asymptotic normality. The confidence interval for the ML estimator is then given by*

$$\hat{\theta}_{ij} \pm c\sqrt{(\boldsymbol{I}(\hat{\boldsymbol{\theta}})^{-1})_{ij,ij}}$$

where $\boldsymbol{I}(\hat{\boldsymbol{\theta}})$ is the negative Hessian matrix, i.e. $\boldsymbol{I}(\hat{\boldsymbol{\theta}}) = -\boldsymbol{H}(\hat{\boldsymbol{\theta}})$.

Proof. The distribution of the maximum likelihood estimator $\boldsymbol{\theta}$ is asymptotically normal with mean $\hat{\boldsymbol{\theta}}$ and the covariance matrix being approximated by the inverse of the Fisher Information matrix $\boldsymbol{I}(\hat{\boldsymbol{\theta}})$ [28]. The resulting confidence interval for $\hat{\theta}_{ij}$ is then

$$\hat{\theta}_{ij} \pm c\sqrt{\boldsymbol{I}(\hat{\boldsymbol{\theta}})^{-1}_{ij,ij}}$$

where c is the appropriate critical value (e.g. 1.96 for 95% confidence level). The Fisher Information for unknown parameters $\boldsymbol{\theta}$ is a matrix $\boldsymbol{I}(\boldsymbol{\theta})$ defined by elements [28, Chap. 9]

$$\boldsymbol{I}(\boldsymbol{\theta})_{ij,kh} = -\mathbb{E}\left[\frac{\partial^2 \log \mathcal{L}(\boldsymbol{\theta})}{\partial\theta_{ij}\partial\theta_{kh}}\right]$$

At the maximum likelihood estimator $\hat{\boldsymbol{\theta}}$, the latter simplifies to [28, Chap. 2]

$$\boldsymbol{I}(\hat{\boldsymbol{\theta}})_{ij,kh} = -\frac{\partial^2 \log \mathcal{L}(\boldsymbol{\theta})}{\partial\theta_{ij}\partial\theta_{kh}}\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = -\boldsymbol{H}(\hat{\boldsymbol{\theta}})_{ij,kh}$$

The Hessian matrix provided in Theorem 2 thus provide the formulas required to compute the confidence intervals. □

The above expression for the confidence intervals can assist in evaluating the ML estimation accuracy. The main result is that, similarly to the ML estimator, also confidence intervals can be computed using the standard MVA algorithm, since this involves evaluating models where some queues are replicated, i.e., where we add new stations having identical demands. As we show later in this paper, confidence interval calculation is more complex in load-dependent models.

5.3 Approximate Closed-Form Expression

We now turn our attention to computing an estimator $\hat{\boldsymbol{\theta}}$ that satisfies the necessary conditions of Theorem 1. One possibility is to apply search method such as numerical optimization or fixed point iteration to find a vector $\hat{\boldsymbol{\theta}}$ that matches the empirical mean queue-lengths. However, since the expression of x_{ij} in the theorem is non-linear, this turns out to be expensive in the case of numerical optimization. Moreover, we have not been able to find fixed point iteration schemes that were converging on all instances.

To cope with the above problems, we develop in this section an approximation of $\hat{\boldsymbol{\theta}}$ using the Bard-Schweitzer (BS) algorithm [35]. This is an iterative approximation that seeks for a fixed-point solution to the closed queueing network by iterating on the values of the mean queue-lengths. By doing so, this method has computational costs independent of the number of jobs in the model and typically converges to good estimates in a few iterations. Analytical expressions are given later in the proof of Theorem 3.

Our idea is to relax the necessary condition of Theorem 1 by requiring that the mean queue-length $x_{ij}(\hat{\boldsymbol{\theta}})$ is computed not by exact methods, but by the BS approximate mean-value analysis, which provides a simple non-recursive form for $x_{ij}(\hat{\boldsymbol{\theta}})$. The BS approximation is rather accurate for multiclass models and therefore the error of the approximation of the necessary condition for the ML estimator tends to be small.

THEOREM 3. *Assume* $\sum_{k=1}^{M} \widetilde{x}_{kj} \neq N_j$, $\forall j$. *Let* $\boldsymbol{\theta}^{bs}$ *be an interior point of* $\boldsymbol{\Theta}$ *and assume that* $x_{ij}^{bs}(\boldsymbol{\theta}^{bs}) = \widetilde{x}_{ij}$ *for some vector* $\boldsymbol{\theta}^{bs}$, *where* $x_{ij}^{bs}(\cdot)$ *is the Bard-Schweitzer approximation of* $x_{ij}(\cdot)$. *Then*

$$\theta_{ij}^{bs} = \frac{\widetilde{x}_{ij}}{(N_j - \sum_{k=1}^{M} \widetilde{x}_{kj})} \frac{\theta_{0j}}{(1 + \sum_{h=1}^{R} \widetilde{x}_{ih} - \widetilde{x}_{ij}/N_j)} \tag{9}$$

PROOF. By the Arrival Theorem [31] we have:

$$\theta_{ij} = \frac{x_{ij}(\boldsymbol{\theta})}{T_j(\boldsymbol{\theta})(1 + A_{ij}(\boldsymbol{\theta}))}$$

where

$$T_j(\boldsymbol{\theta}) = \frac{N_j}{\theta_{0j} + \sum_{i=1}^{M} \theta_{ij}(1 + A_{ij}(\boldsymbol{\theta}))}$$

is the mean class-j throughput and $A_{ij} = \sum_{r=1}^{R} x_{ir}(\boldsymbol{\theta}, \boldsymbol{N} - \mathbf{1}_j)$ is the mean queue-length seen upon arrival to queue i by a job of class j. Since by Little's law $T_j(\boldsymbol{\theta})\theta_{0j} + \sum_{i=1}^{M} x_{ij}(\boldsymbol{\theta}) = N_j$, we get

$$\theta_{ij} = \frac{x_{ij}(\boldsymbol{\theta})}{(N_j - \sum_{i=1}^{M} x_{ij}(\boldsymbol{\theta}))} \frac{\theta_{0j}}{(1 + A_{ij}(\boldsymbol{\theta}))} \tag{10}$$

Direct substitution can be used to check that this expression holds also for the BS solution, which is based on the approximation $A_{ij} \approx \sum_{r=1}^{R} x_{ir}^{bs}(\boldsymbol{\theta}) - x_{ij}^{bs}(\boldsymbol{\theta})/N_j$. Plugging the last expression in (10) and simplifying we get the approximation

$$\theta_{ij}^{bs} = \frac{x_{ij}^{bs}(\boldsymbol{\theta}^{bs})}{(N_j - \sum_{i=1}^{M} x_{ij}^{bs}(\boldsymbol{\theta}^{bs}))} \frac{\theta_{0j}}{(1 + x_i^{bs}(\boldsymbol{\theta}^{bs}) - x_{ij}^{bs}(\boldsymbol{\theta}^{bs})/N_j)}$$

For the BS estimator, the last expression becomes (9) since $x_{ij}^{bs}(\boldsymbol{\theta}^{bs}) = \widetilde{x}_{ij}$. □

Theorem 3 is a closed-form formula that can be readily computed using the empirical mean queue lengths. This makes it suitable for online use. Note though that the condition $\sum_{k=1}^{M} \widetilde{x}_{kj} \neq N_j$ requires that the observation period is long enough to observe some jobs j in thinking state, according to their class think time θ_{0j}, for all classes j. For ease of reference, throughout we refer to the demand vector $\boldsymbol{\theta}$ obtained with (9) as the QMLE demand estimator.

5.4 Estimation Methodology

The methodology to apply the results given in the previous sections may be summarized as follows. One starts by collecting a number of state samples from a running version of the system, and first determines the empirical mean queue-lengths $\widetilde{x}_{ij}$. Using these values, Theorem 3 will then return the QMLE demand estimates. Note that the QMLE method does not require samples of the joint states of multiple stations, but rather of mean queue-lengths for each class. The latter can be collected through periodic sampling of a station only. The degree of overhead implied by monitoring data collection is usually tunable, depending on the frequency of collection of the samples and the monitoring tool used.

Confidence intervals can be generated using (1) parameterized with the QMLE estimates. Alternatively, a slightly more precise estimate may be obtained by performing a numerical search over the $\hat{\boldsymbol{\theta}}$ vectors up to matching the necessary condition in Theorem 3.

6 LOAD-DEPENDENT RESOURCES

In this section, we illustrate how the previous results generalize to the load-dependent case. Here the problem is more complex since one needs to estimate not just the demands θ_{ij}, but also the scaling factors $\gamma_i(u)$, which together define the mean demand $\theta_{ij}(u) = \theta_{ij}\gamma_i(u)$ for station i when it has u enqueued jobs. Recall that we have denoted by $\boldsymbol{\gamma}$ the vector collecting the scaling factors $\gamma_i(u)$, $\forall i, u$. Here we present the ML estimates for the service demand vector $\boldsymbol{\theta}$ as well as for the scaling function $\boldsymbol{\gamma}$, which both appear in the likelihood (2). We then introduce a technique to identify the initial points that help in efficiently searching for the optimal estimates.

6.1 Necessary conditions

We take similar assumptions for the parameters set as for the load-independent case, with the main difference being that the scaling factors $\gamma_i(u)$ are unknown. We assume these terms $\gamma_i(u)$ to be bounded and, without loss of generality, we take $\gamma_i(1) = 1$ so that $\theta_{ij}(1) = \theta_{ij}$. These conditions guarantee existence of the estimators [28].

THEOREM 4. *Given a dataset $\boldsymbol{D}$, a necessary condition for a point $\hat{\boldsymbol{\chi}} = (\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}})$ in the interior point of $\boldsymbol{\Theta}$ to be a ML estimator of demands and scaling factors is that*

$$x_{ij}(\hat{\boldsymbol{\chi}}) = \widetilde{x}_{ij}, \qquad \forall i, j,$$

and

$$\mathbb{P}(n_k = v|\hat{\boldsymbol{\chi}}) = \mathbb{P}(\widetilde{n}_k = v), \qquad \forall k, v$$

where $\widetilde{x}_{ij} \equiv \widetilde{x}_{ij}(\boldsymbol{D})$ and $\mathbb{P}(\widetilde{n}_k = v) \equiv \mathbb{P}(\widetilde{n}_k = v|\boldsymbol{D})$ are respectively the empirical mean queue-length and empirical marginal queue-length probabilities obtained from the dataset $\boldsymbol{D}$.

PROOF. Similar to the proof of Theorem 1, it is not difficult to verify the existence of the ML estimator. From [10], the relationship in (3) still holds for the load-dependent queueing network. Define $\boldsymbol{\chi} = (\boldsymbol{\theta}, \boldsymbol{\gamma})$, therefore considering θ_{ij}, we have

$$\frac{\partial \log \mathcal{L}(\boldsymbol{\chi})}{\partial \theta_{ij}} = \sum_{l=1}^{L} \left(\frac{n_{ij}^{(l)}}{\theta_{ij}} - \frac{x_{ij}(\boldsymbol{\chi})}{\theta_{ij}} \right) = L \frac{\widetilde{x}_{ij} - x_{ij}(\boldsymbol{\chi})}{\theta_{ij}} \tag{11}$$

which implies the condition $x_{ij}(\hat{\boldsymbol{\chi}}) = \widetilde{x}_{ij}$. For $\gamma_k(v)$, we have

$$\frac{\partial \log \mathcal{L}(\boldsymbol{\chi})}{\partial \gamma_k(v)} = \frac{L\mathbb{P}(\widetilde{n}_k \geq v)}{\gamma_k(v)} - \frac{L}{g(\boldsymbol{\chi})} \frac{\partial g(\boldsymbol{\chi})}{\partial \gamma_k(v)} = L \frac{\mathbb{P}(\widetilde{n}_k \geq v) - \mathbb{P}(n_k \geq v|\boldsymbol{\chi})}{\gamma_k(v)} \tag{12}$$

with $\mathbb{P}(\widetilde{n}_k \geq v) \equiv \mathbb{P}(\widetilde{n}_k \geq v)$. The stationarity point $\hat{\boldsymbol{\chi}}$ requires $\mathbb{P}(\widetilde{n}_k \geq v) = \mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}})$, $\forall k, v$, that further simplifies to $\mathbb{P}(\widetilde{n}_k = v) = \mathbb{P}(n_k = v|\hat{\boldsymbol{\chi}}), \forall k, v$. □

Let $\hat{\boldsymbol{\chi}} = (\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}})$, we now generalize the Hessian matrix notation to the load-dependent case. Let us first observe that we can partition the Hessian matrix as

$$\boldsymbol{H}(\hat{\boldsymbol{\chi}}) = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}$$

where

$$\boldsymbol{A}_{ij,i'j'} = \left. \frac{\partial^2 \log \mathcal{L}(\boldsymbol{\chi})}{\partial \theta_{ij} \partial \theta_{i'j'}} \right|_{\boldsymbol{\chi}=\hat{\boldsymbol{\chi}}} \boldsymbol{B}_{ij,kv} = \left. \frac{\partial^2 \log \mathcal{L}(\boldsymbol{\chi})}{\partial \theta_{ij} \partial \gamma_k(v)} \right|_{\boldsymbol{\chi}=\hat{\boldsymbol{\chi}}} \boldsymbol{C}_{kv,k'v'} = \left. \frac{\partial^2 \log \mathcal{L}(\boldsymbol{\chi})}{\partial \gamma_k(v) \partial \gamma_{k'}(v')} \right|_{\boldsymbol{\chi}=\hat{\boldsymbol{\chi}}}$$

where $i, i', k, k' = 1, \ldots, M; j, j' = 1, \ldots, R; v, v' = 1, \ldots, \sum_j N_j$. The expression of the above partial derivatives is derived in the next theorem.

THEOREM 5. *The Hessian matrix of $\mathcal{L}(\boldsymbol{\chi})$ evaluated at $\hat{\boldsymbol{\chi}} = (\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}})$ is a $M(R+N) \times M(R+N)$ matrix with elements*

$$\boldsymbol{A}_{ij,i'j'} = \begin{cases} \frac{L}{\hat{\theta}_{ij}^2}(x_{ij}(\hat{\boldsymbol{\chi}})^2 - E[n_{ij}^2|\hat{\boldsymbol{\chi}}] + x_{ij}(\hat{\boldsymbol{\chi}}) - \widetilde{x}_{ij}) & \text{if } i = i', j = j' \\ \frac{L(x_{ij}(\hat{\boldsymbol{\chi}})x_{i'j'}(\hat{\boldsymbol{\chi}}) - E[n_{ij}n_{i'j'}|\hat{\boldsymbol{\chi}}])}{\hat{\theta}_{ij}\hat{\theta}_{i'j'}} & \text{otherwise} \end{cases}$$

$$\boldsymbol{B}_{ij,kv} = L\frac{x_{ij}(\hat{\boldsymbol{\chi}})\mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}}) - E[n_{ij}|\hat{\boldsymbol{\chi}}, n_k \geq v]}{\hat{\theta}_{ij}\hat{\gamma}_k(v)}$$

$$\boldsymbol{C}_{kv,k'v'} = \begin{cases} L\frac{\mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}})^2 - \mathbb{P}(\widetilde{n}_k \geq v)}{\hat{\gamma}_k(v)^2} & \text{if } k = k', v = v' \\ \frac{L(\mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}})\mathbb{P}(n_{k'} \geq v'|\hat{\boldsymbol{\chi}}) - \mathbb{P}(n_k \geq v, n_{k'} \geq v'|\hat{\boldsymbol{\chi}}))}{\hat{\gamma}_k(v)\hat{\gamma}_{k'}(v')} & \text{otherwise} \end{cases}$$

PROOF. The following computes the partial derivative of $x_{ij}(\boldsymbol{\chi})$ with respect to $\theta_{i'j'}$. If $i = i', j = j'$, we have

$$\begin{aligned} \frac{\partial x_{ij}(\boldsymbol{\chi})}{\partial \theta_{i'j'}} &= \frac{\partial \sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\partial \theta_{ij}} \\ &= \frac{\sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})(n_{ij} - x_{ij}(\boldsymbol{\chi}))}{\theta_{ij}} \\ &= \frac{\sum_{\boldsymbol{n}\in S} n_{ij}^2\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi}) - x_{ij}(\boldsymbol{\chi})\sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\theta_{ij}} \\ &= \frac{E[n_{ij}^2|\boldsymbol{\chi}] - x_{ij}^2(\boldsymbol{\chi})}{\theta_{ij}} \end{aligned} \tag{13}$$

Otherwise, we have with a similar derivation

$$\begin{aligned} \frac{\partial x_{ij}(\boldsymbol{\chi})}{\partial \theta_{i'j'}} &= \frac{\partial \sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\partial \theta_{i'j'}} \\ &= \frac{\sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})(n_{i'j'} - x_{i'j'}(\boldsymbol{\chi}))}{\theta_{i'j'}} \\ &= \frac{\sum_{\boldsymbol{n}\in S} n_{ij}n_{i'j'}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi}) - x_{i'j'}(\boldsymbol{\chi})\sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\theta_{i'j'}} \\ &= \frac{E[n_{ij}n_{i'j'}|\boldsymbol{\chi}] - x_{ij}(\boldsymbol{\chi})x_{i'j'}(\boldsymbol{\chi})}{\theta_{i'j'}} \end{aligned}$$

Substituting (13) and the last result into the derivative of (11) with respect to $\theta_{i'j'}$, we get

$$\boldsymbol{A}_{ij,ij} = L\frac{x_{ij}(\boldsymbol{\chi}) - \widetilde{x}_{ij}}{\theta_{ij}^2} - \frac{L}{\theta_{ij}}\frac{\partial x_{ij}(\boldsymbol{\chi})}{\theta_{ij}} = \frac{L}{\theta_{ij}^2}\left(x_{ij}^2(\boldsymbol{\chi}) - E[n_{ij}^2|\boldsymbol{\chi}] + x_{ij}(\boldsymbol{\chi}) - \widetilde{x}_{ij}\right)$$

and

$$\boldsymbol{A}_{ij,i'j'} = \frac{\partial^2 \log \mathcal{L}(\boldsymbol{\chi})}{\partial \theta_{ij}\partial \theta_{i'j'}} - \frac{L}{\theta_{ij}}\frac{\partial x_{ij}(\boldsymbol{\chi})}{\theta_{i'j'}} = L\frac{x_{ij}(\boldsymbol{\chi})x_{i'j'}(\boldsymbol{\chi}) - E[n_{ij}n_{i'j'}|\boldsymbol{\chi}]}{\theta_{ij}\theta_{i'j'}}$$

Since

$$\begin{aligned}\frac{\partial \mathbb{P}(n_k \geq v|\boldsymbol{\chi})}{\partial \theta_{ij}} &= \sum_{\boldsymbol{n}\in S, n_k \geq v} \frac{\partial \mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\partial \theta_{ij}} \\ &= \frac{1}{\theta_{ij}} \sum_{\boldsymbol{n}\in S, n_k \geq v} \mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})(n_{ij} - x_{ij}(\boldsymbol{\chi})) \\ &= \frac{1}{\theta_{ij}} \big(E[n_{ij}|\boldsymbol{\chi}, n_k \geq v] - x_{ij}(\boldsymbol{\chi})\mathbb{P}(n_k \geq v|\boldsymbol{\chi})\big)\end{aligned}$$

we obtain using (12) that

$$\boldsymbol{B}_{ij,kv} = L\frac{x_{ij}(\boldsymbol{\chi})\mathbb{P}(n_k \geq v|\boldsymbol{\chi}) - E[n_{ij}|\boldsymbol{\chi}, n_k \geq v]}{\chi_{ij}\gamma_k(v)}$$

For the last term in the Hessian, note that if $k = k', v = v'$

$$\begin{aligned}\frac{\partial \mathbb{P}(n_k \geq v|\boldsymbol{\chi})}{\partial \gamma_k(v)} &= \sum_{\boldsymbol{n}\in S, n_k \geq v} \frac{\partial \mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\partial \gamma_k(v)} \\ &= \sum_{\boldsymbol{n}\in S, n_k \geq v} \frac{\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\gamma_k(v)} - \frac{\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{g(\boldsymbol{\chi})}\frac{\partial g(\boldsymbol{\chi})}{\partial \gamma_k(v)} \\ &= \sum_{\boldsymbol{n}\in S, n_k \geq v} \frac{1}{\gamma_k(v)}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})(1 - \mathbb{P}(n_k \geq v|\boldsymbol{\chi})) \\ &= \frac{1 - \mathbb{P}(n_k \geq v|\boldsymbol{\chi})}{\gamma_k(v)}\mathbb{P}(n_k \geq v|\boldsymbol{\chi})\end{aligned}$$

and otherwise with similar passages

$$\begin{aligned}\frac{\partial \mathbb{P}(n_k \geq v|\boldsymbol{\chi})}{\partial \gamma_{k'}(v')} &= \sum_{\boldsymbol{n}\in S, n_k \geq v} \frac{\partial \mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\partial \gamma_{k'}(v')} \\ &= \sum_{\boldsymbol{n}\in S, n_k \geq v, n_{k'} \geq v'} \frac{\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\gamma_{k'}(v')} - \sum_{\boldsymbol{n}\in S, n_k \geq v} \frac{\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{g(\boldsymbol{\chi})}\frac{\partial g(\boldsymbol{\chi})}{\gamma_{k'}(v')} \\ &= \frac{\mathbb{P}(n_k \geq v, n_{k'} \geq v'|\boldsymbol{\chi})}{\gamma_{k'}(v')} - \sum_{\boldsymbol{n}\in S, n_k \geq v} \frac{\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})\mathbb{P}(n_{k'} \geq v'|\boldsymbol{\chi})}{\gamma_{k'}(v')} \\ &= \frac{\mathbb{P}(n_k \geq v, n_{k'} \geq v'|\boldsymbol{\chi}) - \mathbb{P}(n_k \geq v|\boldsymbol{\chi})\mathbb{P}(n_{k'} \geq v'|\boldsymbol{\chi})}{\gamma_{k'}(v')}\end{aligned}$$

Therefore, we have

$$\begin{aligned}\boldsymbol{C}_{kv,kv} &= L\,\frac{\mathbb{P}(n_k \geq v|\boldsymbol{\chi}) - \mathbb{P}(\widetilde{n}_k \geq v)}{\gamma_k(v)^2} - L\,\mathbb{P}(n_k \geq v|\boldsymbol{\chi})\frac{1 - \mathbb{P}(n_k \geq v|\boldsymbol{\chi})}{\gamma_k(v)^2} \\ &= L\,\frac{\mathbb{P}(n_k \geq v|\boldsymbol{\chi})^2 - \mathbb{P}(\widetilde{n}_k \geq v)}{\gamma_k(v)^2}\end{aligned}$$

and

$$\boldsymbol{C}_{kv,k'v'} = L\,\frac{\mathbb{P}(n_k \geq v|\boldsymbol{\chi})\mathbb{P}(n_{k'} \geq v'|\boldsymbol{\chi}) - \mathbb{P}(n_k \geq v, n_{k'} \geq v'|\boldsymbol{\chi})}{\gamma_k(v)\gamma_{k'}(v')}$$

and the proof is completed by setting $\boldsymbol{\chi} = \hat{\boldsymbol{\chi}}$. □

The above results generalize the ones in Theorem 1 and Theorem 2 and similar considerations apply. We see in particular that for load-dependent models an optimization program can be formulated by minimizing the difference between theoretical and observed mean queue lengths and now also marginal probabilities. For example, the load-dependent MVA algorithm [3] can be used in an optimization program to find vectors $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\gamma}}$ that satisfy the mean queue-length necessary condition of Theorem 4. However, load-dependent MVA is known to be computationally expensive as the model size grows, having $O(MRN\prod_{r=1}^{R} N_r)$ time and $O(MN\sqrt{\prod_{r=1}^{R} N_r})$ storage requirements[4]. Therefore, these methods experience early memory bottlenecks when the model size grows. This means that even for small models with a few queues, load-dependent MVA is difficult to use in an optimization program due to its large computational requirements. Moreover, efficient computation of the marginal probability $\mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}})$ is also required by Theorem 4. To alleviate this computational bottleneck, we define a method to locate a good initial point for the optimization program and an approximate evaluation method for load-dependent models.

6.2 Initialization heuristic

As introduced in Section 6.1, an optimization program can be formulated to obtain the ML estimates from Theorem 4. However, the heavy computational requirement of MVA restricts its application to large models. Therefore, here we develop an algorithm to alleviate this problem by identifying a good initial point for the optimization program.

Observe that the structure of (1) allows us to conveniently work with logarithms, we write

$$\begin{aligned}\log(\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})) = &\sum_{i=1}^{M}\sum_{j=1}^{R} n_{ij}\log(\theta_{ij}) + \sum_{u=1}^{n_i}\log(\gamma_i(u)) - \log(g(\boldsymbol{\chi})) + \sum_{j=1}^{R} n_{0j}\log(\theta_{0j}) \\ &+ \sum_{i=1}^{M}\log(n_i!) - \sum_{i=0}^{M}\sum_{j=1}^{R}\log(n_{ij}!)\end{aligned} \tag{14}$$

For each observed state $\widetilde{\boldsymbol{n}} \in \mathcal{S}'$, where $\mathcal{S}'$ is the observed state space, we instantiate (14) as

$$\log(\mathbb{P}(\widetilde{\boldsymbol{n}}|\boldsymbol{\chi})) = \sum_{i=1}^{M}\sum_{j=1}^{R}\widetilde{n}_{ij}\log(\theta_{ij}) + \sum_{u=1}^{\widetilde{n}_i}\log(\gamma_i(u)) - \log(g(\boldsymbol{\chi})) + I \tag{15}$$

where $I = \sum_{j=1}^{R}\widetilde{n}_{0j}\log(\theta_{0j}) + \sum_{i=1}^{M}\log(\widetilde{n}_i!) - \sum_{i=0}^{M}\sum_{j=1}^{R}\log(\widetilde{n}_{ij}!)$ is a constant.

If the number of observed states $\widetilde{\boldsymbol{n}} \in \mathcal{S}'$ is large enough, as commonly the case in practice, by treating $\log(\mathbb{P}(\widetilde{\boldsymbol{n}})|\boldsymbol{\chi})$ as a measured response variable and $\log(\theta_{ij})$, $\log(\gamma_i(u))$ and the normalizing constant $\log(g(\boldsymbol{\chi}))$ as unknown variables, we can estimate the latter using a multivariate least squares method. This therefore provides an initial guess for the demands $\boldsymbol{\theta}$ and scaling factors $\boldsymbol{\gamma}$, without the need for computing the most expensive term, i.e., the normalizing constant $g(\boldsymbol{\chi})$. The above approach can therefore assist in initializing the optimization programs used to find the ML estimator. As we show later in the validation, this initial point can substantially improve the optimization compared to the use of a random initial point.

6.3 Approximate analysis

Even though the initialization heuristic improves the execution time of the estimation algorithm, the latter still requires a high computational effort. This is because the load-dependent MVA algorithm is expensive for large models. To tackle this problem, we propose

here a novel method for the efficient calculation of the marginal probabilities and second-order joint probabilities in load-dependent closed queueing networks. Throughout this section, for ease of presentation we explicit the dependence of the probabilities on the vector $\boldsymbol{N}$ as our main results are recurrence relations on $\boldsymbol{N}$.

PROPOSITION 1. *Assume $\theta_{ij} > 0, \gamma_k(v) > 0$ and $\gamma_k(1) = 1$, $\forall k, j, i, v$. When $M = 1$, we have the following recurrence relations*

$$\mathbb{P}(n_i|\boldsymbol{\chi}, \boldsymbol{N}) = \frac{\theta_{0j}\mathbb{P}(n_i|\boldsymbol{\chi}, \boldsymbol{N} - 1_j) + n_i\theta_{ij}\gamma_i(n_i)\mathbb{P}(n_i - 1|\boldsymbol{\chi}, \boldsymbol{N} - 1_j)}{\theta_{0j} + \sum_{t=1}^{N} t\theta_{ij}\gamma_i(t)\mathbb{P}(n_i = t - 1|\boldsymbol{\chi}, \boldsymbol{N} - 1_j)}$$

$$\mathbb{P}(n_{ij}, n_i|\boldsymbol{\chi}, \boldsymbol{N}) = \frac{\theta_{0j}\mathbb{P}(n_{ij}, n_i|\boldsymbol{\chi}, \boldsymbol{N} - 1_j) + n_i\theta_{ij}\gamma_i(n_i)\mathbb{P}(n_{ij} - 1, n_i - 1|\boldsymbol{\chi}, \boldsymbol{N} - 1_j)}{\theta_{0j} + \sum_{t=1}^{N} t\theta_{ij}\gamma_i(t)\mathbb{P}(n_i = t - 1|\boldsymbol{\chi}, \boldsymbol{N} - 1_j)}$$

$$\mathbb{P}(n_{ij}, n_{ir}, n_i|\boldsymbol{\chi}, \boldsymbol{N}) = \frac{\theta_{0j}\mathbb{P}(n_{ij}, n_{ir}, n_i|\boldsymbol{\chi}, \boldsymbol{N} - 1_j) + n_i\theta_{ij}\gamma_i(n_i)\mathbb{P}(n_{ij} - 1, n_{ir}, n_i - 1|\boldsymbol{\chi}, \boldsymbol{N} - 1_j)}{\theta_{0j} + \sum_{t=1}^{N} t\theta_{ij}\gamma_i(t)\mathbb{P}(n_i = t - 1|\boldsymbol{\chi}, \boldsymbol{N} - 1_j)}$$

for all $\forall r \neq j$, $i \neq 0$, and $n_i \geq 1$.

PROOF. Let $\boldsymbol{n} \in \mathcal{S}$ be a network state. The distribution analysis by chain (DAC) method [11, Eq. (20)] defines the following recurrence relation

$$\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi}, \boldsymbol{N}) = \frac{T_j}{N_j} \sum_{i=1}^{M} \theta_{ir} n_i \gamma_i(n_i) \mathbb{P}(\boldsymbol{n} - \boldsymbol{1}_{ij}|\boldsymbol{\chi}, \boldsymbol{N} - 1_j)$$

where $\boldsymbol{n} - \boldsymbol{1}_{ij}$ removes from $\boldsymbol{n}$ a job of class j in class i and by the load-dependent MVA algorithm [31]

$$T_j = \frac{N_j}{\theta_{0j} + \sum_{t=1}^{N} t\theta_{ij}\gamma_i(t)\mathbb{P}(n_i = t - 1|\boldsymbol{\chi}, \boldsymbol{N} - \boldsymbol{1}_j)}$$

is the throughput of class j. Let $\boldsymbol{n}_i = (n_{i1}, \ldots, n_{iR})$. When $M = 1$ it is $n_{0j} = N_j - n_{ij}$, $i = 1$, and since for the infinite server station $\gamma_0(n_0) = n_0$, the DAC recurrence relation specializes into

$$\mathbb{P}(\boldsymbol{n}_i|\boldsymbol{\chi}, \boldsymbol{N}) = \frac{T_j}{N_j}\theta_{0r}\mathbb{P}(\boldsymbol{n}_i|\boldsymbol{\chi}, \boldsymbol{N} - \boldsymbol{1}_j) + \frac{T_j}{N_j}\theta_{ir} n_i \gamma_i(n_i)\mathbb{P}(\boldsymbol{n}_i - \boldsymbol{1}_j|\boldsymbol{\chi}, \boldsymbol{N} - \boldsymbol{1}_j)$$

The expression of $\mathbb{P}(n_i|\boldsymbol{\chi}, \boldsymbol{N})$, $\mathbb{P}(n_{ij}, n_i|\boldsymbol{\chi}, \boldsymbol{N})$, and $\mathbb{P}(n_{ij}, n_{ir}, n_i|\boldsymbol{\chi}, \boldsymbol{N})$ follow directly from the last expression by summing over all possible values of the variables not appearing in the argument of these marginal and joint probabilities. □

Note that the probabilities used in the expressions in Proposition 1 are used in the calculation of the Hessian for load-dependent models, in particular for computing the terms $E[n_{ij}|\hat{\boldsymbol{\chi}}, n_k \geq v]$ and $E[n_{ij}n_{i'j'}|\hat{\boldsymbol{\chi}}]$. The terms in the expressions are zero whenever the marginal queue-length value is infeasible in the considered model, e.g., $n_{ij} + n_{ir} > n_i$. Even though the previous theorem does not allow to compute the case $n_i = 0$ directly, the corresponding values are readily obtained by requiring that probabilities sum to one. For example, one may set $\mathbb{P}(0|\boldsymbol{\chi}, \boldsymbol{N}) = \max(0, 1 - \sum_{n_i \geq 1} \mathbb{P}(n_i|\boldsymbol{\chi}, \boldsymbol{N}))$, where we use the maximum function for numerical robustness when $\sum_{n_i \geq 1} \mathbb{P}(n_i|\boldsymbol{\chi}, \boldsymbol{N})$ is close to 1.

Thanks to Proposition 1 we are able to compute the exact marginal probabilities using a number of steps equal to the total number of jobs in the network. However, the result only applies to networks with a single queue and a think time server ($M = 1$). In the general case with multiple stations ($M \geq 2$), one can either use DAC [11, Eq. (20)] or approximate

the marginal probabilities by the following aggregation technique. We compute the marginal probabilities for queue i by considering it in isolation in a closed network with think time $\theta^*_{0j} = \theta_{0j} + \sum_{k \neq i} W_{kj}$, where W_{kj} is the mean response time of class-j requests at station $k \neq i$, which can be efficiently computed by the BS approximation. Since the resulting model has a single queue, the marginal probabilities can be computed by applying Proposition 1.

With the above approximation, we can use Theorem 4 to numerically search for the demand estimators using the approximated marginal probabilities from Proposition 1. The mean queue length $x_{ij}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}})$ can be computed with the recently proposed QD-AMVA algorithm [7]. We refer to the resulting estimator as QMLE-LD.

6.4 Estimation Methodology and Confidence Intervals

The estimation methodology for load-dependent systems is similar to the load-independent case. Upon collecting measurements for the input dataset, the main difference compared to QMLE is that in addition to the empirical mean queue-lengths $\widetilde{x}_{ij}$ one needs to record also to estimate the marginal queue-length probabilities $\mathbb{P}(\widetilde{n}_k = v)$. Note that the latter are total value, not differentiated by class. However, similarly to QMLE, the collection of monitoring samples is sufficient to obtain the input data needed to parameterize the model. As such, there is no need for synchronous monitoring at all the resources of a distributed system. Overheads can normally be tuned by a suitable choice of the sampling frequency and of the monitoring tool. Both the QMLE-LD estimates and the matching of the necessary conditions in Theorem 4 now require to use a numerical search method, for which we recommend the initialization heuristic given in Section 6.2.

Confidence intervals may be again generated using the Hessian matrix in Theorem 5. The result given in Corollary 1 holds also for load-dependent systems as

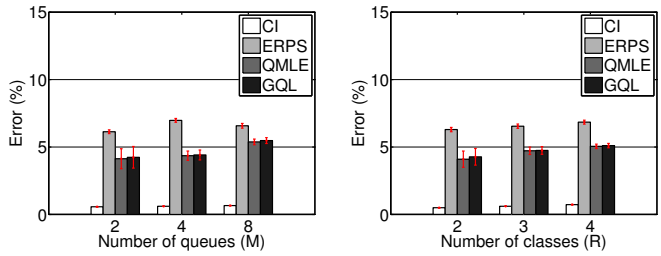
$$\hat{\theta}_{ij} \pm c\sqrt{(\boldsymbol{I}(\hat{\boldsymbol{\chi}})^{-1})_{ij,ij}}$$

$$\hat{\gamma}_k(v) \pm c\sqrt{(\boldsymbol{I}(\hat{\boldsymbol{\chi}})^{-1})_{kv,kv}}$$

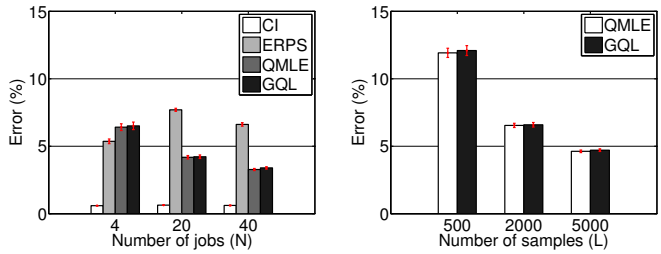
where the tuples (ij, ij) and (kv, kv) are here used to index the positions in the Hessian matrix of the diagonal elements corresponding to $\hat{\theta}_{ij}$ and to $\hat{\gamma}_k(v)$, respectively. In the formula, c is the desired critical value (e.g. 1.96 for 95% confidence level) and $\boldsymbol{I}(\hat{\boldsymbol{\chi}})$ is the negative Hessian matrix for the log-likelihood in the load-dependent case, i.e. $\boldsymbol{I}(\hat{\boldsymbol{\chi}}) = -\boldsymbol{H}(\hat{\boldsymbol{\chi}})$.

7 NUMERICAL VALIDATION

Following this method, we have evaluated the estimation algorithms using randomly generated queueing models. Service completions data are simulated from the underlying Markov Chain of a closed network, which is described in [3]; 5×10^5 service completions are simulated for each instance. We have then generated artificial measurements such as response time, CPU utilization, throughput and queue-length samples. In particular, to obtain independent queue-length samples, we have first computed the steady state probability from the simulation events. Then we have sampled from it by generating random numbers between 0 and 1 and determining which sample fits in the cumulative probability. This is also known as the inverse transform sampling. Our experiments have been run on a desktop machine with an Intel Core i7-2600 CPU, running at 3.4GHz with 16 GB of memory. We use the mean absolute percentage error as the evaluation criteria.



(a) Error sensitivity to M (b) Error sensitivity to R



(c) Error sensitivity to N (d) Error sensitivity to L

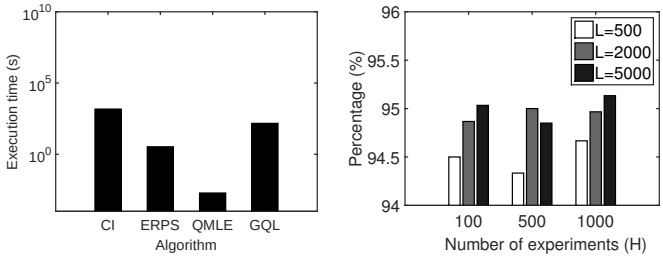
Fig. 2. Results for load-independent networks

7.1 Load-independent network

7.1.1 QMLE evaluation. We begin with evaluating the proposed algorithm for load-independent networks. We compare QMLE with several other existing demand estimation algorithms: CI [29], UBR [43], GQL [40] and ERPS [29]. UBR, ERPS and GQL have been already introduced in Section 4. CI requires the complete sample path of the requests for analysis. The input data for these algorithms is generated from the same simulation dataset, therefore they use 5×10^5 samples. In the case of GQL, the number of samples generated to estimate the service demand is 5000.

The parameters for the random models are $M \in \{2, 4, 8\}$, $R \in \{2, 3, 4\}$, $N = \sum_j N_j \in \{4, 20, 40\}, \theta_{0j} \in \{1, 5, 10\}$. For each generated model, 80 sub-models are defined by randomly generating N_j and θ_{ij} using the uniform distribution. Without loss of generality, demands are normalized so that $\sum_{j=1}^{R} \theta_{ij} = 1$. Here, we focus on the QMLE estimator in Theorem 3 since it is much more practical to compute than the exact one in Theorem 1, which requires nonlinear search.

Figure 2 presents a sensitivity analysis of the considered algorithms with respect to the number of jobs and state samples. Each bar represents the group of experiments for which one of the experimental parameters is the same. The error for UBR is around 100% and not included in the diagram for graphical reasons. From the figure, it can be noticed that CI is the most accurate method since it relies on the knowledge of the complete sample path. However, this method typically cannot be applied to production systems, where only sample measurements are available. The errors of QMLE and GQL are similar and around



(a) Execution time (b) Confidence interval (95%)

Fig. 3. Execution time and confidence interval validation

5%. This is remarkable since QMLE and GQL use the same input data, but GQL features a much more complex algorithm based on Gibbs sampling and iterative approximation of the normalising constant $g(\boldsymbol{\chi})$ of the state probabilities of the queueing network. ERPS is worse than QMLE in terms of accuracy, but generally still quite accurate. As expected, the accuracy of QMLE and GQL increases as the number of observed queue-length samples increases. However, with just 500 queue-length samples QMLE already achieves a small 10% error.

Figure 3(a) shows the execution times. QMLE is much faster than the other algorithms with an average execution time of 0.0002s against 1400s (CI), 3s (ERPS), and 148s (GQL).

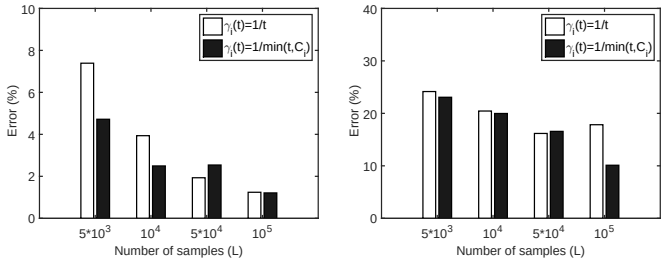
7.1.2 Confidence Interval validation. Validation on the confidence interval requires computing the maximum likelihood estimates of the service demand $\hat{\boldsymbol{\theta}}$. For this purpose, we have implemented a fixed point iteration method based on Theorem 1 to estimate $\hat{\boldsymbol{\theta}}$. The test is based on a queueing model with $M = 2, R = 3, N = 4$. Each test consists of H experiments with a queue-length dataset $\boldsymbol{D}$ of $L = \{500, 2000, 5000\}$ entries generated from the same model. Each experiment has a different queue-length dataset which is generated from the same simulation events. Confidence intervals are computed at 95% significance level.

Figure 3(b) shows the confidence interval validation result. H is set to $\{100, 500, 1000\}$. The vertical axis shows the percentage of the cases where the exact demand lies in the confidence interval of the estimated demand. For different L and H, results suggest that the confidence interval appears to be correct.

7.2 Load-dependent networks

7.2.1 Exact analysis. For the evaluation of load-dependent queueing networks, we have used the MATLAB *fmincon* solver to estimate $\hat{\boldsymbol{\theta}}$ and $\hat{\gamma}_i(t)$ based on Theorem 4. We consider the following scaling factors $\gamma_i(t)$: $\gamma_i(t) = 1/t$ and $\gamma_i(t) = 1/\min(t, C_i)$, where C_i is the number of servers at queueing node i. These two represent some typical load-dependent functions.

The random models generated here consider $M = 2$ queues, $R = 2$ classes, $N = 8$ jobs, think time $\theta_{0j} \in \{1, 5, 10\}$ and $C_i \in \{2, 3, 4\}$. This is a very small model, but we are limited in scalability by the cost of load-dependent MVA. We generate 8 sub-models for each combination of parameters and randomly generate the number of jobs and the demands using a uniform distribution. Figure 4 shows the result for different scaling factors. It is



(a) Error on demand $\boldsymbol{\theta}$ (b) Error on $\boldsymbol{\gamma}$

Fig. 4. Results for load-dependent networks

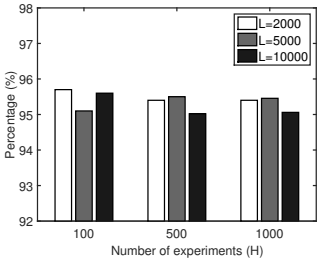


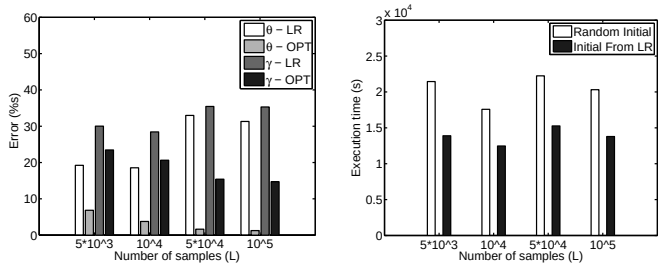
Fig. 5. Confidence interval validation (95%)

easy to observe that the error drops as the number of observed queue-length samples L increases since the average queue-length and marginal probability becomes more accurate. Given $L = 5000$ samples the error for the demands is already below 10% and the error on scaling factors around 20%.

7.2.2 Confidence interval validation. We here present the confidence interval validation based on Section 6.1. The exact demand is computed in the same way as in the previous section. The test is based on a queueing model with $M = 2$ queues, $R = 2$ classes, $N = 4$ jobs. We explicitly consider the multi-core load-dependent behavior here with $C_i = 2$. Each test consists of H experiments with a queue length dataset $\boldsymbol{D}$ of $L = \{2000, 5000, 10000\}$ entries generated from the same model. We use confidence intervals at 95% confidence level. The result is presented in Figure 5 and shows that the computed confidence interval appear correct, up to statistical noise.

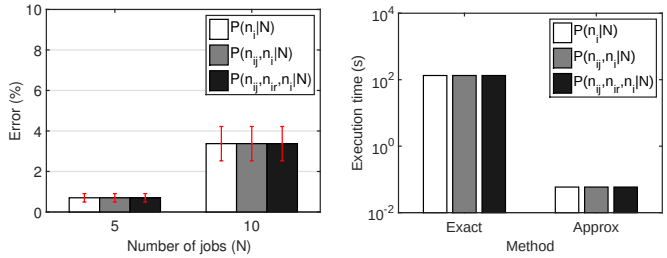
7.2.3 Initialization heuristic validation. Here, we present the evaluation for the proposed method in Section 6.2 to determine the initial point for the optimization program and the impact on the evaluation process.

The random queueing models are generated with $M = 2$, $R = 2$, $N = 4$, $C \in \{2, 3, 4\}$, $\theta_{0j} \in \{1, 5, 10\}$; 8 instances are randomly generated. Figure 6 shows results. In Figure 6(a), we demonstrate the error on the demands $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ comparing both the initial points (referred to as LR) and the estimates from the optimization program (referred to as OPT). Clearly the initial points already provide accurate estimates. Figure 6(b) compares the execution time of the optimization program with randomized initial points and with the initial points



(a) Error on $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ (b) Execution time

Fig. 6. Sensitivity of results to the initialization heuristic



(a) Error on N (b) Execution time

Fig. 7. Errors and execution times for approximating marginal probabilities

returned by the LR heuristic. It can be noticed that, using the heuristic, the execution time drops significantly. Note that the time to compute the initial points using LR is not included in the averages, but it is negligible since it is based on linear regression and on all instances it takes always less than 0.1s.

7.2.4 Approximating marginal probabilities. We now evaluate the approximation algorithm for the marginal probabilities proposed in Section 6.3 by comparing it against the direct numerical solution of the continuous-time Markov chain underlying the queueing network model. The experiment is based on queueing models with $M \in \{2, 3\}, R \in \{2, 3\}, N \in \{5, 10\}$. The load dependent function is considered to be the multi server function where the number of servers varies in $C \in \{2, 4\}$. 10 sub-models are defined by randomly generating the number of jobs and the demands from the uniform distribution. We restrict our study to small and medium-size models due to the computational demand of the exact algorithm. The error is computed as the L1-norm between the observed and the computed probabilities.

Figure 7 presents the evaluation result. It can be seen that the proposed algorithm is able to compute the marginal probabilities efficiently and with low approximation error. The execution time, in particular, is orders of magnitude less than for the exact CTMC-based computation.

7.2.5 Approximate load-dependent analysis. We now assess the application of approximate marginal probabilities to load-dependent demand estimation. We focus on load-dependence functions of the kind $\gamma_i(u) = 1/\min(u, C_i)$, which are used to represent multi-server node i with C_i servers. Mean queue lengths are computed using the QD-AMVA method [7], which provides a computationally efficient implementation of approximate MVA in the presence of load-dependent behaviour, but which cannot provide marginal probabilities.

The initial demand estimate is obtained using the *LR* heuristic. In order to smooth discontinuities in the minimum function, we approximate $\min(u, C_i)$ using a *softmin* function, i.e.,

$$\min(u, C_i) \approx \frac{u\mathrm{e}^{\alpha u} + C_i\mathrm{e}^{\alpha C_i}}{\mathrm{e}^{\alpha u} + \mathrm{e}^{\alpha C_i}} \tag{16}$$

that is exact in the limit $\alpha \to -\infty$; throughout we set $\alpha = -10$. The advantage of this representation is that it requires to estimate only C_i to uniquely define the $\gamma_i(u)$ values.

For validation, we then randomly generate queueing models with $M \in \{2, 4\}, R \in \{2, 3\}, N \in \{4, 10, 20\}, C \in \{2, 3, 4\}$. In total, we simulate 5×10^4 samples. Figure 8 presents the evaluation results. The figure shows the error incurred at the initial estimated obtained from the *LR* heuristic (LD-LR) and the result obtained after numerical search based on the approximate marginal probabilities (LD-APP). LD-APP largely improves accuracy compared to the LD-LR initial points. The error on the estimation of C_i is similar for both methods. In Figure 8(d), we also report the execution time of LD-LR and LD-APP. Due to the fast computation of linear regression, the initial points can be identified with negligible time. However, thanks to the approximation for marginal probabilities, also the numerical search converges in a few seconds.

8 CASE STUDIES

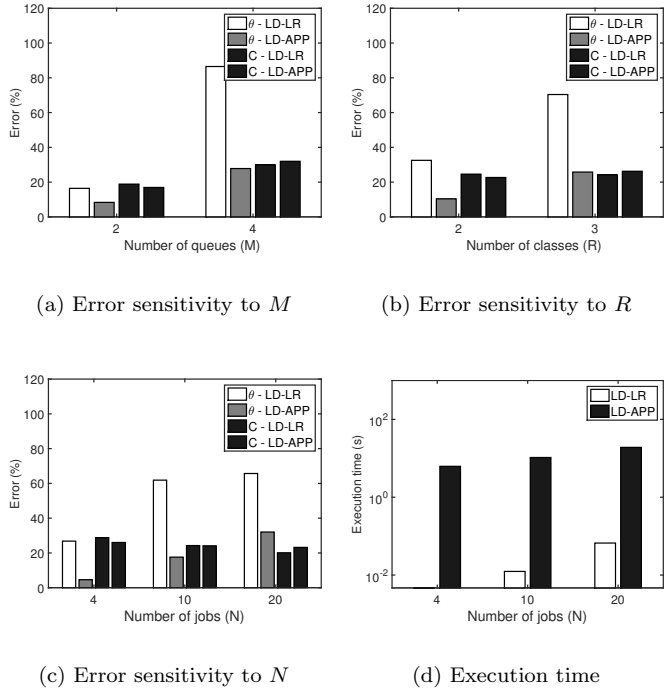
In this section, we present a case study based on two multi-tier applications, Apache OFBiz and JPetStore. The goal of this and the next section is to show that QMLE and QMLE-LD are applicable to real-world performance models. Note that since the goal of the validation is to reproduce the measurement, we do not focus on prediction scenarios, which are meant instead to validate primarily the accuracy of the queueing model, as opposed to the demand estimation method. Ample evidence exist that closed queueing networks can be used to accurately model distributed multi-tier applications, see e.g. [43] and references therein.

8.1 Apache OFBiz

We evaluate the proposed algorithms based on data collected from a real application, Apache Open for Business (OFBiz)[2], which is an open source enterprise resource planning system. We generate a set of user requests for this application using the OFBench tool [24] with an installation where the web server is co-located with the default Apache Derby database. The experiment is run on Amazon EC2, with OFBiz running simultaneously on two load-balanced virtual machines, a *c1.medium* instance and a *c1.xlarge* instance with 2 and 8 virtual CPUs, respectively. Our goal is to estimate the demands at both virtual machines.

We use OFBench to send 8 classes of requests to OFBiz and we parse the OFBiz logs to determine the $\boldsymbol{D}$ dataset. In order to avoid assuming knowledge on the population of the model, which may be unrealistic in some applications, the class populations are estimated from the dataset as the maximum number of concurrently executing jobs in the system, in each class. In order to estimate the think time, the mean throughput is also obtained

[2]https://ofbiz.apache.org/



(a) Error sensitivity to M

(b) Error sensitivity to R

(c) Error sensitivity to N

(d) Execution time

Fig. 8. Errors and execution times for load-dependent estimation algorithms

from the log files, so that by Little's Law $\theta_{0j} = n_{0j}/T_j$ where $n_{0j}, 1 \leq j \leq R$ represents the average number of users in thinking state and T_j the mean throughput of class j. The number of jobs (i.e., clients) is varied from 5 to 25 for the deployment on the *c1.medium* instance and 10 to 90 for the *c1.xlarge* instance.

8.2 Results

Since in the real system the exact demand is unknown, we have compared the estimation algorithms in terms of the observed performance metrics and the predictions obtained from the models parameterized with the estimated demands. We focus on QMLE-LD, QMLE, and the state-of-the-art estimators introduced in Section 7.1.

The performance metrics considered here are average throughput (T), average queue length (Q) and average response time (W). To produce these performance metrics, we consider using both load-independent models (LI) solved by the Bard-Schweitzer (BS) approximation and load-dependent models (LD) solved using the QD-AMVA method [7]. For load-dependent models, we consider two cases:

- Estimation focuses only on service demands, while the load-dependent scaling is set to $\gamma(u_i) = 1/\min(u_i, C_i)$ with C_i set to the number of CPUs in the virtual machine. This case is called *fixed* $\boldsymbol{\gamma}$.
- Both demands and effective parallelism are estimated, generating an approximate value of C_i. This case is referred to as *estimated* $\boldsymbol{\gamma}$ and only QMLE-LD can be applied to this scenario, since the other methods cannot estimate C_i.

We remark that while the QMLE-LD method may be used more generally to fit a whole set of different $\gamma(u_i)$ rates, not bound to a specific functional shape, this may result in over-fitting the queue-length dependence. It is thus best to focus on simpler parametric functions which captures the key dependencies, such as the multi-server rate function. In both cases, the resulting load-dependent model is solved using QD-AMVA. The observed performance metrics include average throughput and response times, whereas the observed mean queue length is computed by Little's law.

Table 2 and Table 3 present the results. Note that the CI method requires knowledge of the state of each resource along the entire sample path of the system, thus it is not included in the table as this tends to constrain its applicability.

From the tables, we see that LD estimation improves over LI estimation; this is expected since LD models have more degrees of freedom than LI models. In addition, the QMLE-LD method typically produces more accurate results than the other estimators. The only exception is for the light-load case with $N = 5$ in Table 2; the error is by large due to an inaccurate estimate of the parallelism C_i. However, if we assume the C_i is given in advance, as in the fixed $\boldsymbol{\gamma}$ case, the error on Q and W can be reduced to 27%. Even though in this specific experiment the fixed $\boldsymbol{\gamma}$ algorithm performs better than the estimated $\boldsymbol{\gamma}$ one, the other experiments in the following tables suggest that typically the two methods feature similar accuracy. Therefore, we recommend to use fixed $\boldsymbol{\gamma}$ in first instance, with the option of switching to the more complex estimated $\boldsymbol{\gamma}$ method if predictive accuracy of the fixed $\boldsymbol{\gamma}$ needs to be improved. Metrics such as average throughputs or average queue-lengths predicted by the closed queueing network can be compared against the empirical values collected in the dataset $\boldsymbol{D}$ and used to discriminate the more accurate estimation method for a specific system configuration.

The results also indicate that QMLE fares similarly to GQL, as also seen on the random experiments. This is expected since GQL asymptotically converges to the maximum-likelihood estimator. However, GQL takes several minutes to run, whereas QMLE takes milliseconds. The CI method alternatively gives lower or higher error than the QMLE-LD approach. It should be noted that the two methods use different measurements, therefore their accuracy is not simple to relate; however as mentioned before CI is difficult to apply to production systems.

8.3 JPetStore

In this section, we present a second case study based on the MyBatis JPetStore[3] application.

8.4 Experimental setting

The benchmarking application chosen is JPetStore, an open source version of Sun's J2EE pet store application. It is an e-commerce application that allows customers to login, browse pet categories, select pets and checkout payments. A single end-to-end transaction is considered, with customers visiting all the application pages sequentially. 5GB of data (2×10^6 items) for viewing and selection by customers is used in the testbed.

Two tiers of servers are used, with both the Web/Application and Database servers consisting of 4-core Intel Xeon E5620 CPUs with 8 GB of memory. Thus, the resulting queueing network model has two queues and an infinite server node. The Grinder (http://grinder.sourceforge.net) open source testing framework is used for load injection, with each test lasting 10 minutes to eliminate transient values. The experiment setup is

[3] `https://github.com/mybatisjpetstore-6`

Table 2. Percentage error on performance metrics computed from estimated demands (%) - OFBiz - deployment on 2 core virtual machine

		LI			LD				
N_j					fixed γ				est. γ
		ERPS	GQL	QMLE	ERPS	GQL	QMLE	QMLE-LD	QMLE-LD
5	T	1.5	1.5	1.6	0.3	0.4	0.5	0.5	0.6
	Q	99.9	42.7	43.5	9.5	12.9	14.0	28.7	51.9
	W	99.9	44.2	45.0	10.8	14.3	15.4	27.7	52.4
10	T	1.1	1.7	1.6	0.6	1.1	1.0	0.4	0.4
	Q	99.9	37.1	37.0	14.3	22.8	23.3	10.9	16.8
	W	99.9	38.9	38.7	15.9	24.5	24.9	12.1	17.3
15	T	1.2	1.7	1.8	1.2	1.7	1.8	0.9	0.2
	Q	99.9	24.6	26.6	17.3	25.0	27.1	17.5	2.4
	W	99.9	27.8	29.7	20.3	28.1	30.2	15.8	0.8
20	T	0.7	0.8	0.9	0.7	0.8	0.8	1.0	0.3
	Q	100.0	11.7	12.5	9.3	11.1	11.9	24.0	3.7
	W	100.0	13.4	14.2	11.0	12.8	13.6	24.5	2.8
25	T	1.5	0.6	0.6	1.4	0.5	0.5	1.0	0.4
	Q	100.0	8.2	7.4	17.0	7.4	6.6	21.7	4.3
	W	100.0	11.2	10.7	20.6	10.4	9.9	19.7	1.6
All	T	1.2	1.3	1.3	0.8	0.9	0.9	0.8	0.4
	Q	99.9	24.9	25.4	13.5	15.8	16.6	20.6	15.8
	W	99.9	27.1	27.7	15.7	18.0	18.8	19.9	15.0

listed in Table 4, with Database server CPU utilization values listed corresponding to tested concurrency values. As bottlenecks are observed at the Database server CPU (maximum contribution to overall service demands), these metrics are used in the performance analysis (i.e., network, memory and disk are negligible).

8.5 Results

Here we still consider to compare the observed performance metrics and the theoretical ones computed with the estimated demands to evaluate the proposed algorithm.

Table 5 presents the analysis result. The findings here is consistent with the findings in the OFBiz case study. Methods with QD-AMVA evaluation is more accurate and the LD approach in general is more accurate than all the other algorithms. We do not include the performance prediction experiment reported in [41] here since the result remains the same.

9 CONCLUSION

In this paper, we have proposed a class of maximum likelihood estimators for resource demands in closed queueing networks. Our method is applicable to both load-independent and load-dependent nodes. After identifying necessary conditions for an estimator to be a maximizer of the likelihood function, we have derived explicit and tractable approximations for demand estimators. An advantage of our methodology is that it generalizes to load-dependent nodes, allowing the efficient estimation of load-dependent demands. Confidence interval expressions have also been derived for the proposed estimators. Finally, evaluation

Table 3. Percentage error on performance metrics computed from estimated demands (%) - OFBiz - deployment on 8 core virtual machine

		LI			LD				
N_j					fixed γ				est. γ
		ERPS	GQL	QMLE	ERPS	GQL	QMLE	QMLE-LD	QMLE-LD
10	T	1.9	1.9	1.9	0.2	0.2	0.2	0.0	0.0
	Q	100.0	87.9	88.3	10.4	8.5	11.4	0.2	0.2
	W	100.0	88.3	88.7	12.2	10.3	13.2	3.0	3.0
30	T	3.1	3.1	3.1	0.7	0.6	0.6	0.0	0.0
	Q	100.0	88.1	88.1	19.4	16.7	16.8	1.6	0.9
	W	100.0	88.8	88.8	22.2	19.4	19.6	4.4	3.7
50	T	3.3	3.2	3.2	0.8	0.7	0.7	0.0	0.0
	Q	100.0	86.8	86.9	20.3	17.6	18.2	0.3	0.3
	W	100.0	87.8	87.9	24.7	22.0	22.5	5.0	5.0
70	T	4.2	4.4	4.4	0.8	1.2	1.2	0.0	0.0
	Q	100.0	84.8	84.9	13.6	23.0	23.2	0.4	0.4
	W	100.0	86.2	86.3	18.8	28.0	28.1	5.7	5.7
90	T	3.2	5.2	5.2	0.6	2.3	2.3	0.0	0.0
	Q	100.0	82.6	82.8	9.6	37.1	37.2	0.5	0.5
	W	100.0	85.0	85.1	16.7	44.1	44.2	9.6	9.6
All	T	3.1	3.6	3.6	0.6	1.0	1.0	0.0	0.0
	Q	100.0	86.0	86.2	14.7	20.6	21.3	0.6	0.5
	W	100.0	87.2	87.4	18.9	24.8	25.5	5.5	5.4

Table 4. Experiment setup.

Think time	Number of jobs	CPU Utilization
0.1s	$\{1, 3, 5, 10, 15\}$	$\{0.12, 0.38, 0.58, 0.90, 0.95\}$
0.5s	$\{1, 3, 5, 10, 20\}$	$\{0.03, 0.14, 0.23, 0.44, 0.79, 0.93\}$
1s	$\{1, 2, 5, 10, 20, 40\}$	$\{0.04, 0.06, 0.14, 0.26, 0.51, 0.88\}$
5s	$\{1, 10, 20, 40, 80, 120\}$	$\{0.01, 0.06, 0.12, 0.22, 0.45, 0.67\}$

based on simulation data and traces of a multi-tier application demonstrate the applicability of the proposed methods to demand estimation in real software systems.

ACKNOWLEDGMENT

W. Wang and G. Casale have been supported by the European Commission under grant agreement 644869 (DICE), the EPSRC grant EP/M009211/1 (OptiMAM) and by an Amazon AWS in Education Research Grant Award. Data and scripts used in this paper are available at http://dx.doi.org/10.5281/zenodo.35321.

REFERENCES

[1] V. Apte and T. V. S. Viswanath and D. Gawali and A. Kommireddy and A. Gupta. AutoPerf: Automated Load Testing and Resource Usage Profiling of Multi-Tier Internet Applications. In *Proc. of ICPE*, pp.115–126. ACM, 2017.

Table 5. Percentage error on performance metrics computed from estimated demands - JPetStore.

		LI			LD				
N_j					given γ				est. γ
		ERPS	GQL	QMLE	ERPS	GQL	QMLE	QMLE-LD	QMLE-LD
0.1	T	11.5	18.2	18.9	2.3	7.9	8.6	3.6	3.6
	Q	8.6	13.1	12.6	2.0	4.7	4.3	6.0	6.0
	W	14.7	24.1	23.5	2.0	11.2	10.6	9.1	9.1
0.5	T	9.6	11.0	11.1	2.4	3.9	3.9	1.8	1.8
	Q	39.5	44.0	44.2	9.1	13.7	13.8	5.3	5.3
	W	43.0	48.2	48.4	10.7	16.2	16.3	6.9	6.9
1	T	5.3	5.5	5.6	2.1	2.3	2.4	1.6	1.6
	Q	43.1	46.2	46.0	12.0	13.7	13.5	5.7	5.7
	W	44.9	48.2	48.0	13.2	15.1	14.9	6.4	6.4
5	T	11.4	6.5	6.3	12.2	7.5	7.3	2.8	2.8
	Q	163.6	113.9	108.9	131.6	76.1	70.8	20.5	20.5
	W	383.8	155.1	146.2	350.8	116.6	107.7	22.6	22.6
All	T	9.4	10.3	10.5	4.7	5.4	5.5	2.5	2.5
	Q	63.7	54.3	52.9	38.7	27.1	25.6	9.4	9.4
	W	121.6	68.9	66.5	94.2	39.8	37.4	11.2	11.2

[2] M. Awad and D. A. Menascé. Deriving Parameters for Open and Closed QN Models of Operational Systems Through Black Box Optimization. In *Proc. of ICPE*, pp.127–138. ACM, 2017.

[3] G. Bolch, S. Greiner, H. de Meer, and K. S Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[4] S. Bruell, G. Balbo and PV. Afshari. Mean value analysis of mixed, multiple class BCMP networks with load dependent service stations. *PEVA*, 4(4):241–260, 1984.

[5] G. Casale. CoMoM: Efficient class-oriented evaluation of multiclass performance models. *IEEE TSE.*, 35(2):162–177, 2009.

[6] G. Casale, P. Cremonesi, and R. Turrin. Robust workload estimation in queueing network performance models. In *Proc. of IEEE PDP*, pp.183–187. 2008.

[7] G. Casale, J. F Pérez, and W. Wang. QD-AMVA: Evaluating systems with queue-dependent service requirements. *PEVA*, 91:80–98, 2015.

[8] P. Cremonesi, K. Dhyani, and A. Sansottera. Service time estimation with a refinement enhanced hybrid clustering algorithm. In *Proc. of ASMTA*, pp.291–305. Springer, 2010.

[9] P. Cremonesi and A. Sansottera. Indirect estimation of service demands in the presence of structural changes. *PEVA*, 73:18–40, 2014.

[10] E. De Souza e Silva and R. R. Muntz. Simple relationships among moments of queue lengths in product form queueing networks. *IEEE TC*, 37(9):1125–1129, 1988.

[11] E. De Souza e Silva and S. Lavenberg. Calculating Joint Queue Length Distributions in Product Form Queueing Networks. *JACM*, 36(1):194–207, 1989.

[12] G. Franks, T. Al-Omari, M. Woodside, O. Das and S. Derisavi. Enhanced modeling and solution of layered queueing networks. *IEEE TSE*, 35(2):148–161, 2009.

[13] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Workload analysis and demand prediction of enterprise data center applications. In *Proc. of IEEE IISWC*, pp.171–180. 2007.

[14] A. Kalbasi, D. Krishnamurthy, J. Rolia, and S. Dawson. DEC: Service demand estimation with confidence. *IEEE TSE*, 38(3):561-578, 2012.

[15] A. Kalbasi, D. Krishnamurthy, J. Rolia, and M. Richter. Mode: Mix driven on-line resource demand estimation. In *Proc. of IEEE CNSM*, pp.1–9. 2011.

[16] A. Khan, X. Yan, S. Tao, and N. Anerousis. Workload characterization and prediction in the cloud: A multiple time series approach. In *Proc. of IEEE NOMS*, pp.1287–1294. 2012.

[17] S. Kraft, S. Pacheco-Sanchez, G. Casale, and S. Dawson. Estimating service resource consumption from response time measurements. In *Proc. of ValueTools*, page 48. 2009.
[18] D. Kumar, L. Zhang, and A. Tantawi. Enhanced inferencing: Estimation of a workload dependent performance model. In *Proc. of ValueTools*, page 47. 2009.
[19] S. S. Lam and Y. L. Lien A tree convolution algorithm for the solution of queueing networks. *CACM*, 26:203–215, 1983.
[20] S. S. Lavenberg and G. S. Shedler. Derivation of confidence intervals for work rate estimators in a closed queuing network. *SIAM Journal on Computing*, 4(2):108–124, 1975.
[21] Z. Liu, L. Wynter, C. H Xia, and F. Zhang. Parameter inference of queueing models for it systems using end-to-end measurements. *PEVA*, 63(1):36–60, 2006.
[22] D. A Menascé. Computing missing service demand parameters for performance models. In *Int. CMG Conference*, pp.241–248, 2008.
[23] D. A Menascé, V. AF Almeida, L. W Dowdy and L. Dowdy. *Performance by design: computer capacity planning by example.* Prentice Hall, 2004.
[24] J. Moschetta, and G. Casale. OFBench: An enterprise application benchmark for cloud resource management studies. In *Proc. of SYNASC*, pp.393–399, 2012.
[25] I.J. Myung. Tutorial on maximum likelihood estimation. *JMP*, 47 (1):90–100, 2003.
[26] A. Kattepur, and M. K. Nambiar. Service demand modeling and performance prediction with single-user tests *PEVA*, 110:1–21, 2017.
[27] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi. CPU demand for web serving: Measurement analysis and dynamic estimation. *PEVA*, 65(6):531–553, 2008.
[28] Y. Pawitan. *In all likelihood: statistical modelling and inference using likelihood.* Oxford University Press, 2001.
[29] J. F Perez, G. Casale, and S. Pacheco-Sanchez. Estimating computational requirements in multi-threaded applications. *IEEE TSE*, 41(3):264–278, 2015.
[30] M. Reiser, and H. Kobayashi. Queuing networks with multiple closed chains: theory and computational algorithms. *IBM journal of Research and Development*, 19(3):283–294, 1975.
[31] M. Reiser and S. S Lavenberg. Mean-value analysis of closed multichain queuing networks. *JACM*, 27(2):313–322, 1980.
[32] J. Rolia and V. Vetland. Parameter estimation for performance models of distributed application systems. In *Proc. of CASCON*, page 54. 1995.
[33] J. Rolia and V. Vetland. Correlating resource demand information with arm data for application services. In *Proc. of ACM WOSP*, pp.219–230. 1998.
[34] J. V Ross, T. Taimre, and P. K Pollett. Estimation for queues from queue length data. *Queueing Systems*, 55(2):131–138, 2007.
[35] P. J. Schweitzer. Approximate analysis of multiclass closed networks of queues. In *Proc. of Inter. Conf. on Stoc. Cont. and Opti.*, pp.25–29. 1979.
[36] A. B Sharma, R. Bhagwan, M. Choudhury, L. Golubchik, R. Govindan, and G. M Voelker. Automatic request categorization in internet services. *ACM SIGMETRICS PER*, 36(2):16–25, 2008.
[37] S. Spinner, G. Casale, F. Brosig, and S. Kounev. Evaluating Approaches to Resource Demand Estimation. *PEVA*, 92:51–71, 2015.
[38] C. Sutton and M. I Jordan. Bayesian inference for queueing networks and modeling of internet services. *The Annals of Applied Statistics*, pp.254–282, 2011.
[39] W. Wang, X. Huang, X. Qin, W. Zhang, J. Wei, and H. Zhong. Application-level cpu consumption estimation: Towards performance isolation of multi-tenancy web applications. In *Proc. of IEEE CLOUD*, pp.439–446. IEEE, 2012.
[40] W. Wang, G. Casale and C. Sutton. A Bayesian Approach to Parameter Inference in Queueing Networks. *ACM Trans. on Modeling and Computer Simulation*, 2016.
[41] W. Wang, G. Casale, A. Kattepur and M. Nambiar. Maximum Likelihood Estimation of Closed Queueing Network Demands from Queue Length Data. *Proc. of ACM/SPEC ICPE*, pp.3–14, 2016.
[42] X. Wu and M. Woodside. A calibration framework for capturing and calibrating software performance models. In *Computer Performance Engineering*, pp.32–47. Springer, 2008.
[43] Q. Zhang, L. Cherkasova, and E. Smirni. A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In *Proc. of IEEE ICAC*, pp.27–27. 2007.
[44] T. Zheng, M. Woodside, and M. Litoiu. Performance model estimation and tracking using optimal filters. *IEEE TSE*, 34(3):391–406, 2008.

Received January 2018