Maximum Likelihood Estimation of Closed Queueing Network Demands from Queue Length Data

Weikun Wang Giuliano Casale
Department of Computing
Imperial College London
London, UK
{weikun.wang11, g.casale}@imperial.ac.uk

Ajay Kattepur Manoj Nambiar
Performance Engineering Research Center
TCS Innovation Labs
Mumbai, India
{ajay.kattepur,m.nambia}@tcs.com

ABSTRACT

Resource demand estimation is essential for the application of analyical models, such as queueing networks, to real-world systems. In this paper, we investigate maximum likelihood (ML) estimators for service demands in closed queueing networks with load-independent and load-dependent service times. Stemming from a characterization of necessary conditions for ML estimation, we propose new estimators that infer demands from queue-length measurements, which are inexpensive metrics to collect in real systems. One advantage of focusing on queue-length data compared to response times or utilizations is that confidence intervals can be rigorously derived from the equilibrium distribution of the queueing network model. Our estimators and their confidence intervals are validated against simulation and real system measurements for a multi-tier application.

1. INTRODUCTION

Guaranteeing Quality-of-Service (QoS) is an important concern for cloud providers and software vendors in order to minimize service-level agreement violations. Performance models such as queueing networks are commonly used for performance analysis and prediction and therefore can support engineers in coping with these problems. Closed queueing networks, in particular, are often preferred for software systems since real applications are layered and thus operate under pooling constraints that limit the maximum parallelism level at each layer and the underpinning hardware resources. However, despite solution methods for these models have been systematically investigated, their parametrization from real measurements is often difficult, but still it is essential to obtain accurate predictions [31]. Among these parameters, the *resource demand*, i.e., the cumulative time a request seizes from a server excluding contention overheads, is particularly challenging to estimate, since demands are difficult to measure directly without introducing substantial overheads. Statistical inference can therefore be used to determine accurate estimates from indirect measurements, such as throughputs, utilizations, queue-length, or response time data [4,14,17,18,20,22,37] .

Linear regression methods that use CPU utilization and request throughputs have attracted much attention in the last decade for demand estimation. However, these methods are known to suffer from the multicollinearity problem leading to biased estimates [12]. On the other hand, the idea of exploiting response time measurements has recently been exploited in a number of works [14,22,37]. Nonetheless, collecting response time may pose a large overhead to the application system, especially if one needs to instrument separate layers of a multi-tier application. In this paper, we therefore investigate a different approach to obtain the demand estimates, where we attempt to exploit queue-length samples, i.e., measurements of the number of executing requests at each resource. Recent research [34] has also explored this problem, however the resulting algorithm based on Gibbs sampling is computationally expensive for large models, thus restraining its use to offline analysis.

Stemming from the above considerations, in this paper we investigate the problem of *efficiently* estimating demands from queue-length measurements. Using the equilibrium distribution of product-form closed queueing networks, we develop a characterization of necessary conditions for maximum likelihood solutions to the demand estimation problem. We then provide tractable expressions for the Hessian matrix that can be used to verify if a stationary point is indeed a maximum. Furthermore, we show that the Hessian matrix readily provides the confidence intervals associated to the (local) maximum likelihood estimator.

In addition to the above contributions, we show that our method applies also to load-dependent systems, which is a distinctive advantage of our approach compared to other methods. While 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 current queue-length. Real-world systems typically exhibit a load dependent behavior, often as a result of parallelism of multi-core servers or caching and shared data structures in enterprise web applications [5]. Therefore the ability to estimate how a request demand varies with the load is essential for predicting performance of real applications where these aspects are critical for performance. The work in [15] is to our knowledge the first attempt to estimate the demand in a load-dependent queueing network, however it only applies to open models and requires prior knowledge of a parametric expression for the load-dependent function. Instead, the estimation technique presented in this paper generalizes to *closed* load-dependent models, making demand estimation viable also for this class of models. While the method we propose is also more efficient if prior knowledge of the load dependent function is available, our expressions are general and can work also without this assumption.

We illustrate the efficiency and accuracy of the proposed estimators using simulated data with random parameters and a real-world enterprise application. We also compare against existing demand estimation algorithms showing our approach to be effective. We show the applicability of our methodology against a real world case study of a multi-tier application by comparing the performance of the system with the predictions of a queueing network that uses the estimated demands. In particular, we show that predictions with demands estimated at low-load can be quite effective in characterizing high-load behaviour, even in presence of load-dependence.

Summarizing, the main contribution of this paper are:

- Novel estimators of resource demands for load-independent and load-dependent closed queueing networks;
- Confidence interval expressions for the proposed estimators in both classes of models;
- An experimental study based on simulated and real system data showing the effectiveness of the estimation methods.

This paper extends a preliminary work published as an extended abstract in [35]. Compared to our initial investigation, in this paper we generalize our results to load-dependent networks and include formal proofs for all the results, which were not given in [35].

The rest of the paper is organized as follows. Section 2 reviews the previous work on demand estimation while Section 3 presents the background information. In Section 4 a motivating example is illustrated. Later, Section 5 and 6 presents the proposed algorithms of demand estimation for load-independent and load-dependent models. Finally, evaluation of the algorithms are given in Section 7 followed by the conclusion remarks.

2. RELATED WORK

Existing work for characterizing resource demand are based on statistical inference of indirect measurements, among which CPU utilization, throughput and response time are the most popular ones. In particular, methods for regressing CPU utilization and throughput to obtain service demand have gained much attention [20,26,27,37]. Later [4] 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 multicollinearity [12] that can lead to biased estimates. To overcome this shortcoming, various algorithms based on machine learning has been proposed. Kalman filters [33,36,38,39] have shown to be effective in parameter tracking. Other methods including clustering [6, 7], pattern recognition [10, 13], independent component analysis [30] have also been explored to estimate service demands. However, these methods require CPU utilization measurements which are not always available or reliable, especially in cloud environments. Compared to these algorithms, our proposed methods rely on queue-length samples, which are easier to collect since they only require to monitor the number of running worker threads in a system and the identity of the running request.

Besides CPU utilization, response times have also been used for estimating service demand. The work in [17] defines a quadratic programming using end-to-end response time and CPU utilization together with request throughputs. The methods introduced in [14] and [28] employ response time values as well, but they only apply to FCFS servers. The work in [18] focuses on estimating demands for some simple queueing systems through optimization programs that use response time data. Recent work [22] 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.

In spite of the above mentioned measurements, queue-length samples have also been exploited for demand estimation. The study in [34] uses Gibbs sampling and Bayesian estimation methods to obtain service demands. The method also allows for prior information in the estimation. However, it is computationally expensive, with running times often exceeds many tens of minutes or even hours. This makes the technique difficult to apply to online systems. The authors in [32] have also 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 [28] also using queue-length samples. Compared to the present work, the methods in [32] and [28] are limited to open models, whereas we focus here on closed models.

Finally, the above mentioned techniques do not provide confidence interval for the estimates. The work in [11] proposes an approach to estimate resource demand with confidence through linear programming. Nonetheless, it does not apply to load-dependent networks. The other work for obtaining confidence interval is introduced in [16]. However this method is limited to single-class models only and does not work for load-dependent networks.

3. REFERENCE MODEL

We consider product-form closed queueing networks, under the assumptions of the BCMP theorem [2]. Models have R job classes, 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. Indexes range in $1 \leq i, k \leq M, 1 \leq j, h \leq R$. 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 jobs of class j at station i. Define $n_i = \sum_{j=1}^{R} n_{ij}$ to be the total number of jobs at station i. Then the probability of observing state $\boldsymbol{n} = (n_{01}, \ldots, n_{0R}, n_{11}, \ldots, n_{1R}, \ldots, n_{MR})$ at equilibrium is known from the BCMP theorem to be

$$\mathbb{P}(\boldsymbol{n}|\boldsymbol{\theta},\boldsymbol{\gamma}) = \left(\prod_{j=1}^{R} \frac{\theta_{0j}^{n_{0j}}}{n_{0j}!}\right) \prod_{i=1}^{M} n_i! \prod_{j=1}^{R} \frac{\theta_{ij}^{n_{ij}}}{n_{ij}! G(\boldsymbol{\theta})} \prod_{u=1}^{n_i} \gamma_i(u), \quad (1)$$

where $\gamma_i(u)$ is the load-dependent function that scales the demand for station i when its queue-length is u and $G(\boldsymbol{\theta})$ is the normalizing constant that assures $\sum_{\boldsymbol{n}\in\mathcal{S}} \mathbb{P}(\boldsymbol{n}|\boldsymbol{\theta}) = 1$, being $\mathcal{S} = \{\sum_{i=0}^{M} n_{ij} = N_j, n_{ij} \geq 0\}$ the state space. 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.

In order to perform a demand estimation, let us consider independent state samples $\boldsymbol{n}^l \in \boldsymbol{D}$, being $\boldsymbol{D}$ a dataset of empirical observations of L vectors $\boldsymbol{n}^l$. The problem of estimating the true demand vector $\boldsymbol{\theta}$ and scaling factors $\boldsymbol{\gamma} = (\gamma_i(u))$ may be solved by considering a maximum likelihood (ML) estimator

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

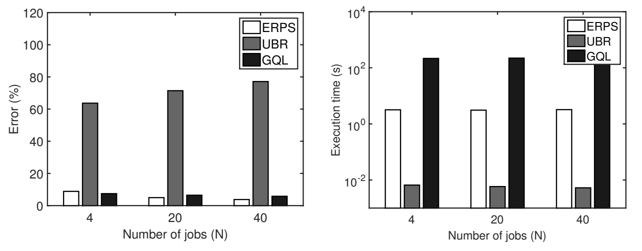
where $\mathcal{L}(\cdot)$ is the likelihood function, $\mathbb{P}(\cdot|\boldsymbol{\theta},\boldsymbol{\gamma})$ is defined as in (1), and $\boldsymbol{\Theta}$ is the parameter space composed by the candidate demand vector $\boldsymbol{\theta}$ and scaling factors $\boldsymbol{\gamma}$.

4. MOTIVATING EXAMPLE

In this section, we provide a motivating example that compares properties of three state-of-the-art demand estimation algorithms and illustrates their limitations. We consider utilization-based regression (UBR) (e.g., [37]), Gibbs Sampling for Queue Lengths (GQL) [34] and Extended Regression for Processor Sharing (ERPS) [22]. 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 as input for the analysis.

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 $\{4, 20, 40\}$. We generate 80 submodels by randomly choosing N_j and θ_{ij}. The think time is assumed to be 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. $500,000$ service completions are simulated and the results collected.

Figure 1 illustrates the mean relative absolute error and the execution time for the above algorithms. It can be noticed that UBR shows a bad estimation accuracy, likely due to problems such as multicollinearity, which lead to degraded results. Figure 1(a) instead shows that GQL and ERPS achieve good accuracy, but Figure 1(b) reveals that the execution time of GQL is fairly large and probably unacceptable for online use. ERPS performs well considering both the accuracy and execution time, nonetheless it requires response time data, which assumes the ability to track the state of individual requests, which may not be possible at server-side without introducing substantial overheads (e.g., instrumentation of worker threads at milli-second or microsecond timescales). Besides, none of the above algorithms offers confidence intervals to characterize the quality of the generated estimates. In a real system study, when the exact demands are unknown, such confidence intervals can provide guidance on the reliability of the inferred demand values and the associated predictions.



(a) Estimation Error (b) Execution time

Figure 1: Various estimation methods

5. LOAD-INDEPENDENT NETWORKS

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 focus on deriving estimators for the service demand vector $\boldsymbol{\theta}$. In addition, we derive the analytical expressions of the confidence intervals for the $\boldsymbol{\theta}$ estimates. Finally, we introduce a closed-form approximate formula that simplifies the task of obtaining an approximate, but accurate, estimate.

5.1 Necessary conditions

We begin with assuming that the parameter space $\boldsymbol{\Theta}$ is a compact set and that the think time θ_{0j} are known and strictly positive. Under the above conditions, it is simple to show that the likelihood function is continuous and that a ML estimator exists [21]. We also assume that $\boldsymbol{\Theta}$ is large enough for the true demand $\boldsymbol{\theta}^*$ to be an interior point of this set. A consequence of this assumption is that our results do not cover the estimation of demands with true 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.

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 is that*

$$Q_{ij}(\hat{\boldsymbol{\theta}}) = \widetilde{Q}_{ij}(\boldsymbol{D}), \qquad \forall i, j,$$

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

The proof of this theorem and the following ones are given in the Appendix. Note that the predicted mean queue lengths $Q_{ij}(\hat{\boldsymbol{\theta}})$ can be computed, for example, using the MVA algorithm [2].

The main contribution of Theorem 1 is to provide theoretical support to the idea that the estimation of demands in load-independent models may be simply performed by matching theoretical predictions of mean queue-lengths to the observed mean values in the real system, without need for correction terms. Furthermore, it states the less obvious fact that the demand estimation depends only on the *mean* queue-length, even though the maximum-likelihood function is probabilistic in nature. 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. Clearly, even if Theorem 1 does not prove $\hat{\boldsymbol{\theta}}$ to be the ML estimator, the condition of the theorem ensures that $\hat{\boldsymbol{\theta}}$ will achieve correct performance predictions that reproduce the training data. Hence, while in principle several vectors $\boldsymbol{\theta}$ may satisfy the same necessary condition, any of these will be a suitable choice for reproducing the observations.

The main requirement for Theorem 1 to be a sufficient condition is the availability of results that prove that a given set of queue-length values $Q_{ij}(\boldsymbol{\theta})$ can be obtained by a unique vector $\boldsymbol{\theta}$. This appears intuitive, but we are not aware of any such formal characterization in the literature of product-form models, presumably due to the complex non-linear nature of the MVA equations. Therefore, in order to support 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).

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

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

where $\boldsymbol{N} = (N_1, \ldots, N_R)$, L is the total number of samples, and $Q_{kh}^{+i}(\hat{\boldsymbol{\theta}}, \boldsymbol{N} - \boldsymbol{1}_j)$ is the mean queue length in a model obtained by adding an identical replica of queue i to the closed network under study and removing a job of class j from it.

With 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 $\boldsymbol{H}(\hat{\boldsymbol{\theta}})$ is positive definite, i.e. all eigenvalues are positive, then $\hat{\boldsymbol{\theta}}$ is a local minimum. If $\boldsymbol{H}(\hat{\boldsymbol{\theta}})$ is negative definite, i.e. all 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 $\hat{\boldsymbol{\theta}}$ 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. $c = 1.96$ means 95% confidence).

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. $\boldsymbol{I}(\hat{\boldsymbol{\theta}}) = -\boldsymbol{H}(\hat{\boldsymbol{\theta}})$.

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, the situation is more complex in load-dependent models.

5.3 Approximate Closed-Form Expression

We now turn our attention to obtaining numerically an estimator $\hat{\boldsymbol{\theta}}$ that satisfies the necessary conditions of Theorem 1. One simple possibility is to apply search method such as numerical optimization and fixed point iteration to find a vector $\hat{\boldsymbol{\theta}}$ that matches the empirical mean queue-lengths. However, this turns out to be expensive in the case of numerical optimization. Also, we were unable to find fixed point iteration schemes that were converging on all instances.

To cope with the above problems, we develop in this section an accurate approximation of $\hat{\boldsymbol{\theta}}$ using the Bard-Schweitzer (BS) approximation [1, 29]. Our idea is to relax the necessary condition of the theorem by requiring that the queue-length $Q_{ij}(\hat{\boldsymbol{\theta}})$ is computed not by exact methods, but by the BS approximate mean-value analysis, which leads to a simple analytical form for $Q_{ij}(\hat{\boldsymbol{\theta}})$. Such approximation is fairly accurate for multiclass models, except in some contrived examples, and therefore the degree of approximation of the necessary condition is quite limited.

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

$$\theta_{ij}^{bs} = \frac{\widetilde{Q}_{ij}(\boldsymbol{D})}{(N_j - \sum_{k=1}^{M} \widetilde{Q}_{kj}(\boldsymbol{D}))} \frac{\theta_{0,j}}{(1 + \sum_{h=1}^{R} \widetilde{Q}_{ih}(\boldsymbol{D}) - \widetilde{Q}_{ij}(\boldsymbol{D})/N_j)} \qquad (3)$$

It can be noted that Theorem 3 is a closed-formula that can be readily computed using the empirical mean queue lengths. This makes it suitable for online use. For ease of reference, we refer to the demand vector $\boldsymbol{\theta}$ obtained with (3) as the *QMLE* demand estimator.

6. LOAD-DEPENDENT NETWORK

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 that includes 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 $\boldsymbol{\gamma}$. 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 [19].

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*

$$Q_{ij}(\hat{\boldsymbol{\chi}}) = \widetilde{Q}_{ij}(\boldsymbol{D}), \qquad \forall i, j,$$

and

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

where $\mathbb{P}(\widetilde{n}_k = v|\boldsymbol{D})$ are the empirical marginal queue length probabilities obtained from the dataset $\boldsymbol{D}$.

The Hessian matrix for the load-dependent case is generalized as follows.

THEOREM 5. *The Hessian matrix of $\mathcal{L}(\hat{\boldsymbol{\chi}})$ at the ML estimates $\hat{\boldsymbol{\chi}}$ is given as*

$$\boldsymbol{H}(\hat{\boldsymbol{\chi}})_{ij,i'j'} = \begin{cases} \dfrac{L}{\hat{\theta}_{ij}^2}(Q_{ij}(\hat{\boldsymbol{\chi}})^2 - E[n_{ij}^2|\hat{\boldsymbol{\chi}}] + Q_{ij}(\hat{\boldsymbol{\chi}}) - \widetilde{Q}_{ij}(\boldsymbol{D})) & \text{if } i = i', j = j' \\ \dfrac{L(Q_{ij}(\hat{\boldsymbol{\chi}})Q_{i'j'}(\hat{\boldsymbol{\chi}}) - E[n_{ij}n_{i'j'}|\hat{\boldsymbol{\chi}}])}{\hat{\theta}_{ij}\hat{\theta}_{i'j'}} & \text{otherwise} \end{cases}$$

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

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

The result is qualitatively similar to the one in Theorem 1, and analogous considerations apply. An optimization program can be formulated by minimizing the difference between theoretical and the observed mean queue lengths and marginal probabilities. In particular, the load-dependent MVA algorithm [2] 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 space requirements, being N the total population in the model and R the number of classes. That is, complexity is roughly quadratic in the total number of jobs in the model. 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 $\mathbb{P}(\widetilde{n}_k \geq v|\boldsymbol{D})$ 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.

Similar as the discussion in Section 5.2, we are able to use the above expression to determine if the generated estimates are local maximum or not. The confidence interval of load-dependent network can be characterized in the same way as in Corollary 1. Differently from the load-independent case, computing confidence intervals here requires to obtain the second-order moments of marginal queue-length, i.e., the terms $E[n_{ij}n_{i'j'}|\hat{\boldsymbol{\chi}}]$, for determining the Hessian matrix. This assumes the availability of efficient computational algorithms for such moments, which yet do not exist in the load-dependent setting. Moreover, marginal probability and mean queue-length is also required to compute the confidence intervals. Therefore, without specialized algorithms, the applicability of confidence intervals will be limited to models with a small or a medium-sized population, where these moments can be obtained by direct computation over the state space. It is therefore an interesting line of future research in closed queueing networks to develop efficient algorithms that can determine such joint moments.

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.

Noticing that the structure of (1) allows us to apply logarithms, we write

$$\begin{aligned}\log(\mathbb{P}(\boldsymbol{n}|\boldsymbol{\theta}, \boldsymbol{\gamma})) = &\sum_{i=1}^{M}\sum_{j=1}^{R} n_{ij}\log(\theta_{ij}) + \sum_{u=1}^{n_i}\log(\gamma_i(u)) \\ &- \log(G(\boldsymbol{\theta})) + \sum_{j=1}^{R} n_{0j}\log(\theta_{0j}) \\ &+ \sum_{i=1}^{M}\log(n_i!) - \sum_{i=0}^{M}\sum_{j=1}^{R}\log(n_{ij}!)\end{aligned} \tag{4}$$

For each observed state $\widetilde{\boldsymbol{n}} \in S'$, where S' is the observed state space, we can rewrite (4) as

$$\begin{aligned}\log(\mathbb{P}(\widetilde{\boldsymbol{n}}|\boldsymbol{D})) = &\sum_{i=1}^{M}\sum_{j=1}^{R} \widetilde{n}_{ij}\log(\theta_{ij}) + \sum_{u=1}^{\widetilde{n}_i}\log(\gamma_i(u)) \\ &- \log(G(\boldsymbol{\theta})) + I\end{aligned} \tag{5}$$

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

It is now possible to observe that by treating $\log(\mathbb{P}(\widetilde{\boldsymbol{n}}|\boldsymbol{D}))$ as response variable and $\log(\boldsymbol{\theta})$, $\log(\boldsymbol{\gamma})$ and the normalizing constant $\log(G(\boldsymbol{\theta}))$ as unknown variables, we can easily solve (5) as a multivariate linear regression. 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 $G(\boldsymbol{\theta})$. The above approach can therefore assist in the optimization program in identifying a suitable initial point in negligible computational time. As we show later in the validation, this initial point substantially improves the optimization compared to the use of a random initial point.

7. NUMERICAL VALIDATION

We now present the validation methodology for evaluating the proposed algorithms. We have evaluated the algorithms using randomly generated queueing models. Service completions data are simulated from the underlying Markov Chain of a closed network, which is described in [2]. From these data, we have generated typical monitoring measurements such as response time, CPU utilization, throughput and queue-length samples. In particular, to obtain 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.

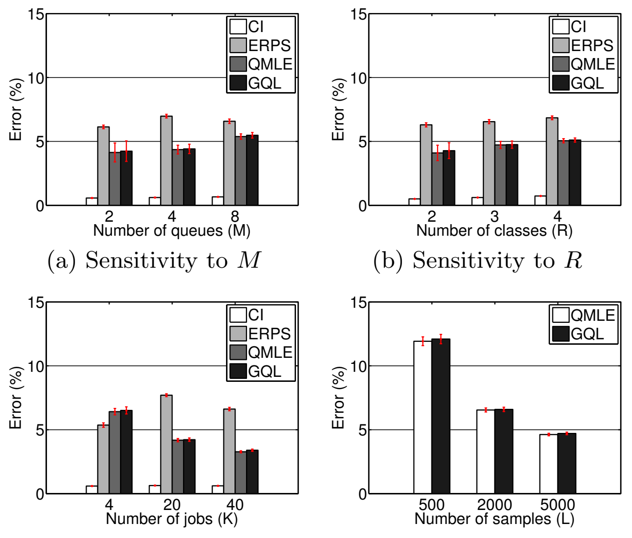
7.1 Load-independent network

7.1.1 QMLE evaluation

We begin with evaluating the proposed algorithm for load-independent network. For comparison, we have also implemented several other demand estimation algorithms. They are CI [22], UBR [37], GQL [34] and ERPS [22]. 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 events as of the queue-length samples. $500,000$ service completions are generated.

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

Figure 2 presents a sensitivity analysis of the considered algorithms. The result of UBR is not included since the error is around 100% due to multicollinearity. 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 cannot be applied in production systems, where only sample measurements are available. The error



(a) Sensitivity to M (b) Sensitivity to R (c) Sensitivity to K (d) Sensitivity to L

Figure 2: Sensitivity analysis of the estimators

of QMLE and GQL is almost the same, around 5%, which shows the effectiveness of QMLE since GQL is defined using a much more complex algorithm featuring Gibbs sampling and iterative approximation of the normalising constant $G(\boldsymbol{\theta})$. ERPS is worse 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 only 500 queue-length samples QMLE already achieves a small 10% error.

Figure 3(a) shows the execution time of each method. Clearly the proposed QMLE is orders magnitude better than the other algorithms with average 0.0002 second against 1400 (CI), 3 (ERPS) and 148 (GQL) seconds.

7.1.2 Confidence Interval validation

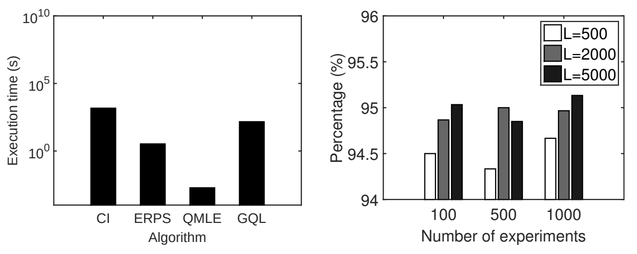
Validation on the confidence interval requires computing the maximum likelihood estimates of the service demand $\hat{\boldsymbol{\theta}}$. For this purpose, we have implemented a fixed point iteration method based on Theorem 1 to estimate $\hat{\boldsymbol{\theta}}$. The test is based on a queueing model with $M = 2, R = 3, K = 4$. 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. We use 95% confidence intervals.

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

7.2 Load-dependent network

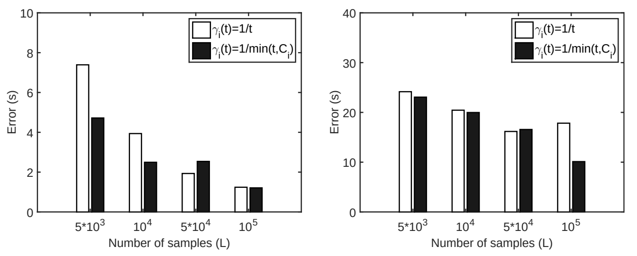
7.2.1 Exact analysis

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



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

Figure 3: Execution time and confidence interval validation



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

Figure 4: Load-dependent experiment result

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 CPUs at queueing node i. These two represent the most typical load-dependent scenarios, e.g. think time and multi-core feature of servers.

The random models generated here consider $M = 2$ queues, $R = 2$ classes, $K = 8$ 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 considering the high computational cost 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 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, $K = 4$ 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 95% confidence intervals. The result is presented in Figure 5. Clearly, it shows the computed confidence interval is correct given the estimates for different L and H.

7.2.3 Initial value determination

Here, we present the evaluation for the proposed method in Section 6.2 to determine the initial point for the optimization

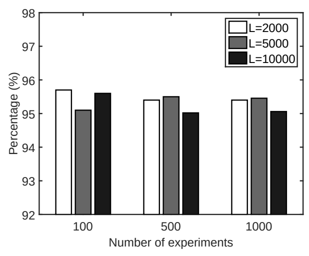
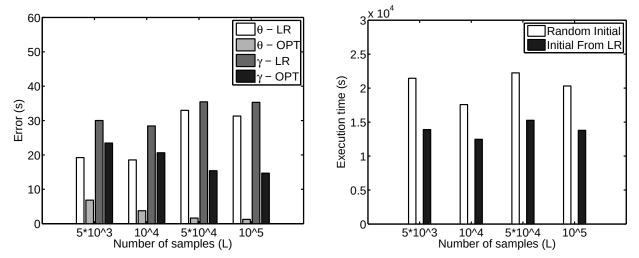


Figure 5: Confidence interval validation (95%)



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

Figure 6: Error and execution time with initial points

program and the impact on the evaluation process.

The random queueing models are generated with $M = 2$, $R = 2$, $K = 4$, $C \in \{2, 3, 4\}$, $\theta_{0j} \in \{1, 5, 10\}$. 8 sub-models are generated. Figure 6 presents the evaluation result. 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 output very accurate estimates, which are useful to guide the optimization program. Figure 6(b) shows the execution time of the optimization program for using random initial points and the initial points returned from LR. It can be noticed the execution time drops significantly with the heuristic initials. We do not include the execution time of LR since it is based on linear regression and takes less than 0.1 second.

8. CASE STUDY

In this section, we present a case study based on a real application.

8.1 Experimental setting

The benchmarking application chosen is MyBatis JPetStore[1], 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 (2,000,000 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

[1]`https://github.com/mybatis/jpetstore-6`

Table 1: Performance prediction with same dataset where demands are estimated

Z(s)	Metric	LD-LR	BS				AMVA-QD			
			CI	ERPS	GQL	QMLE	CI	ERPS	GQL	QMLE
0.1	X	13.6	9.6	10.6	13.5	11.1	2.3	3.5	4.2	5.7
	Q	9.7	9.6	9.5	13.9	8.3	3.4	3.3	7.0	2.2
	CN	19.8	13.6	15.0	22.2	16.3	1.6	3.3	10.1	7.2
0.5	X	6.1	7.5	8.8	9.3	7.6	3.9	7.0	4.4	4.2
	Q	14.0	64.0	41.0	39.4	39.8	12.3	10.6	17.7	10.6
	XN	15.8	40.8	40.2	45.9	41.9	8.3	8.8	15.0	10.1
1	X	3.6	3.7	9.4	4.1	4.4	3.6	12.3	4.2	4.9
	Q	17.7	43.4	44.4	45.3	40.1	14.6	21.1	18.3	14.4
	CN	18.8	43.9	58.3	46.2	41.1	12.0	39.2	18.4	15.5
5	X	4.8	2.3	17.9	3.9	3.0	3.8	20.0	5.7	5.4
	Q	37.5	63.0	161.3	44.4	60.2	37.1	177.4	12.0	14.6
	CN	38.9	62.6	443.0	46.2	59.9	36.0	464.6	14.2	14.9
All	X	7.0	5.8	11.7	7.7	6.5	3.4	10.7	4.6	5.1
	Q	19.7	45.0	64.1	35.8	37.1	16.8	53.1	13.7	10.4
	CN	23.3	40.2	139.1	40.1	39.8	14.5	129.0	14.4	11.9

Table 3: Experiment setup with Think Time, Concurrency and Database server CPU Utilization.

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	$\{1,10,20,40\}$	$\{0.01,0.06,0.12,0.22\}$

E5620 CPUs with 8 GB of memory. The Grinder[2] open source testing framework is used for load injection, with each test lasting 10 minutes to eliminate transient values. The experiment setup is listed in Table 3, 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 effects are negligible).

8.2 Explaining observed performance

Considering that in real system the exact demand is unknown, we evaluate the proposed algorithm by comparing the observed performance metrics and the theoretical ones computed with the estimated demands. Here, the algorithms considered are the proposed linear regression method for determining the initial values, which is referred as *LD-LR*, as well as the QMLE approach. The load-independent algorithms introduced in Section 7.1 are also included.

The performance metrics considered here are average throughput (X), average queue length (Q) and average response time (CN). To produce these metrics, we consider using both the Bard-Schweitzer (BS) approximation, which we already used in Section 5.3 and the recently proposed AMVA-QD method, which is an approximate algorithm for mean performance metrics in load-dependent models [5]. For the previous case, we scale the demands by the minimum of the number of jobs and number of CPUs to approximate the multi-core behavior of the server. For the latter case, we explicitly use the *softmin* function, which is defined as

$$\gamma_i(u) = \frac{u e^{\alpha u} + C_i e^{\alpha C_i}}{e^{\alpha u} + e^{\alpha C_i}} u$$

This is an approximation of the multi-core function and converges to the exact value as $\alpha \to -\infty$. We set α to be $\alpha = -10$ throughout the experiment. For load-independent demand estimation algorithms, we assume the number of CPUs is given. For *LD-LR*, we use the Piecewise Cubic Hermite interpolation method [9] to fit the estimated $\boldsymbol{\gamma}$ and then use AMVA-QD to estimate the performance.

Table 1 presents the analysis result. From the table, it can first be noticed that evaluation with AMVA-QD returns much accurate result than using AMVA-BS for all the algorithms. In addition, compared to CI and GQL, QMLE outputs stable and accurate performance for all different kind of models. Finally, the linear regression method produces accurate result as well except the case with $Z = 5$, which is caused by the large K in that experiment. Considering that this method does not have a-priori knowledge of how many cores the server has, the finding is more significant than the others.

8.3 Performance prediction

Here we study another common problem which is to predict the performance of the application using benchmarked dataset. In particular, the demand is first estimated for the benchmarked dataset, then it is used to predict the performance for a number of requests that is different from the one used for demand fitting. For ease of comparison, we assume the think time is the same as the one of the experiment on which the demand was fitted.

Evaluation results are given in Table 2 focusing on the case $Z = 0.1s$. For each dataset, we estimate the demands first and predict the performance for $K \in \{1,3,5,10,15\}$ and obtain the average error. We limit our study to use load-independent algorithms only for demand estimation since it is unfair to use methods such as the *LD-LR* to predict the performance considering for small K there is not enough

[2] `http://grinder.sourceforge.net/`

Table 2: Performance prediction with different datasets

K	Metric	BS				AMVA-QD			
		CI	ERPS	GQL	QMLE	CI	ERPS	GQL	QMLE
1	X	18.5	18.5	37.6	13.2	8.2	8.2	25.2	3.7
	Q	15.5	15.5	27.1	11.9	7.9	7.9	18.1	5.0
	CN	25.0	25.0	45.0	19.7	13.1	13.1	33.9	7.7
3	X	14.1	14.2	13.1	15.2	6.7	6.9	3.7	5.4
	Q	10.0	10.0	11.6	13.0	3.7	3.7	4.8	5.8
	CN	21.2	21.4	19.1	21.9	9.1	9.6	7.1	9.9
5	X	14.4	14.5	13.6	14.2	7.8	8.0	4.1	6.9
	Q	9.9	9.9	12.2	10.0	3.8	3.8	5.2	3.7
	CN	22.3	22.5	20.3	21.4	11.2	11.7	8.3	9.6
10	X	13.3	14.2	13.2	13.7	4.1	7.0	3.7	5.3
	Q	11.2	10.0	11.9	10.4	4.4	3.7	4.9	3.8
	CN	18.9	21.5	19.6	19.9	6.9	9.7	7.6	7.8
15	X	13.1	13.1	13.6	13.8	3.8	3.8	4.8	5.6
	Q	11.6	11.6	10.7	10.3	4.7	4.7	4.1	3.8
	CN	18.9	21.5	19.6	19.9	7.1	7.1	7.4	8.1
All	X	14.7	14.9	18.2	14.0	6.1	6.8	8.3	5.4
	Q	11.7	11.4	14.7	11.1	4.9	4.8	7.4	4.4
	CN	21.3	22.4	24.7	20.6	9.5	10.2	12.9	8.6

information on γ. From the table, in general algorithms with AMVA-QD performs much better than the ones with AMVA-BS method. The proposed QMLE method with AMVA-QD is able to predict the performance with less than 9% error.

9. CONCLUSION

In this paper, we have proposed a class of maximum likelihood estimator for resource demand in closed queueing networks with both load-independent and load-dependent service. After identifying necessary conditions for an estimator to be a maximizer of the likelihood function, we derived explicit and tractable expressions for the confidence interval of the estimates. For load-independent models, a closed form formula has been presented for demand estimation, which allows to obtain good estimates very quickly. Moreover, a heuristic method is proposed to accelerate searching for the estimates in load-dependent networks. Finally, evaluation 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

This work has been funded by the European Commission as part of the DICE action (H2020-644869).

10. REFERENCES

[1] Y. Bard. Some extensions to multiclass queueing network analysis. In *Proc. of SPECTS*, pages 51–62. 1979.

[2] 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.

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

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

[5] G. Casale, J. F Pérez, and W. Wang. Qd-amva: Evaluating systems with queue-dependent service requirements. *Perf. Eval.*, 91:80–98, 2015.

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

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

[8] 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.

[9] F. N Fritsch and R. E Carlson. Monotone piecewise cubic interpolation. *SIAM SINUM*, 17(2):238–246, 1980.

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

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

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

[13] 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*, pages 1287–1294. 2012.

[14] S. Kraft, S. Pacheco-Sanchez, G. Casale, and S. Dawson. Estimating service resource consumption from response time measurements. In *Proc. of ICST*,

page 48. 2009.

[15] D. Kumar, L. Zhang, and A. Tantawi. Enhanced inferencing: Estimation of a workload dependent performance model. In *Proc. of ICST*, page 47. 2009.

[16] 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.

[17] Z. Liu, L. Wynter, C. H Xia, and F. Zhang. Parameter inference of queueing models for it systems using end-to-end measurements. *Perf. Eval.*, 63(1):36–60, 2006.

[18] D. A Menascé. Computing missing service demand parameters for performance models. In *Int. CMG Conference*, pages 241–248, 2008.

[19] I.J. Myung. Tutorial on maximum likelihood estimation. *JMP*, 47 (1):90–100, 2003.

[20] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi. Cpu demand for web serving: Measurement analysis and dynamic estimation. *Perf. Eval.*, 65(6):531–553, 2008.

[21] Y. Pawitan. *In all likelihood: statistical modelling and inference using likelihood*. Oxford University Press, 2001.

[22] J. F Perez, G. Casale, and S. Pacheco-Sanchez. Estimating computational requirements in multi-threaded applications. *IEEE TSE*, 41(3):264–278, 2015.

[23] 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.

[24] M. Reiser and H. Kobayashi. On the convolution algorithm for separable queuing networks. In *Proc. of ACM SIGMETRICS*, pages 109–117. 1976.

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

[26] J. Rolia and V. Vetland. Parameter estimation for performance models of distributed application systems. In *Proc. of CASCON*, page 54. IBM Press, 1995.

[27] J. Rolia and V. Vetland. Correlating resource demand information with arm data for application services. In *Proc. of ACM WOSP*, pages 219–230. 1998.

[28] J. V Ross, T. Taimre, and P. K Pollett. Estimation for queues from queue length data. *Queueing Systems*, 55(2):131–138, 2007.

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

[30] 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.

[31] S. Spinner, G. Casale, F. Brosig, and S. Kounev. Evaluating Approaches to Resource Demand Estimation. *Perf. Eval.*, 92:51–71, 2015.

[32] C. Sutton and M. I Jordan. Bayesian inference for queueing networks and modeling of internet services. *The Annals of Applied Statistics*, pages 254–282, 2011.

[33] 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*, pages 439–446. IEEE, 2012.

[34] W. Wang and G. Casale. Bayesian service demand estimation using gibbs sampling. In *Proc. of IEEE MASCOTS*, pages 567–576. 2013.

[35] W. Wang and G. Casale. Maximum likelihood estimation of closed queueing network demands from queue length data. *ACM SIGMETRICS PER*, 43(2):45–47, 2015.

[36] X. Wu and M. Woodside. A calibration framework for capturing and calibrating software performance models. In *Computer Performance Engineering*, pages 32–47. Springer, 2008.

[37] 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*, pages 27–27. 2007.

[38] T. Zheng, M. Woodside, and M. Litoiu. Performance model estimation and tracking using optimal filters. *IEEE TSE*, 34(3):391–406, 2008.

[39] T. Zheng, J. Yang, M. Woodside, M. Litoiu, and G. Iszlai. Tracking time-varying parameters in software systems with extended kalman filters. In *Proc. of CASCON*, pages 334–345. IBM Press, 2005.

Appendix

Proof of Theorem 1

A sufficient condition for existence of a ML estimator is that the parameter space is compact and the likelihood function continuous. Observe that the parameter space is compact since $\boldsymbol{\theta}$ is assumed bounded. Since the normalising constant is continuous in $\boldsymbol{\theta}$ and $G(\mathbf{0}) \neq 0$ since $\theta_{0,j} > 0$, then $\mathcal{L}(\boldsymbol{\theta})$ is also continuous in $\boldsymbol{\theta}$.

Given existence, we now determine the ML estimator. Recall the following relationship proved in [8]

$$\frac{\partial G(\boldsymbol{\theta})}{\partial \theta_{ij}} = \frac{Q_{ij}(\boldsymbol{\theta})}{\theta_{ij}} G(\boldsymbol{\theta}) \tag{6}$$

This allows us to take the first derivative of the likelihood function as follows

$$\begin{aligned} \frac{d\mathcal{L}(\boldsymbol{\theta})}{d\theta_{ij}} =& \frac{\partial}{d\theta_{ij}} \Big[\prod_{l=1}^{L} \frac{1}{G(\boldsymbol{\theta})} \prod_{i=1}^{M} n_i^l! \prod_{j=1}^{R} \frac{\theta_{ij}^{n_{ij}^l}}{n_{ij}^l!} \Big] \\ =& \sum_{l=1}^{L} \left(-\frac{Q_{ij}(\boldsymbol{\theta})}{\theta_{ij}} + \frac{n_{ij}^l}{\theta_{ij}} \right) \mathcal{L}(\boldsymbol{\theta}) \end{aligned} \tag{7}$$

A stationarity point is then found at

$$-Q_{ij}(\boldsymbol{\theta}) L + \sum_{l=1}^{L} n_{ij}^l = 0$$

which implies the condition $Q_{ij}(\hat{\boldsymbol{\theta}}) = \widetilde{Q}_{ij}$. This completes the proof. $\square$

Proof of Theorem 2

Given the maximum likelihood estimator $\hat{\boldsymbol{\theta}}$, the Hessian matrix is defined as

$$\boldsymbol{H}(\boldsymbol{\theta})_{ij,kh} = \frac{\partial^2 \log \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_{ij} \partial \theta_{kh}}\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \tag{8}$$

Using (7), the partial derivative of $\log \mathcal{L}(\boldsymbol{\theta})$ with respect to θ_{ij} is

$$\frac{\partial \log \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_{ij}} = \frac{1}{\mathcal{L}(\boldsymbol{\theta})} \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_{ij}} = L \frac{\widetilde{Q}_{ij} - Q_{ij}(\boldsymbol{\theta})}{\theta_{ij}} \tag{9}$$

It can be seen from (9) that in order to obtain the Hessian the partial derivative of $Q_{ij}(\boldsymbol{\theta})$ is required. According to [24]

$$Q_{ij}(\boldsymbol{\theta}) = \frac{\theta_{ij} G^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j)}{G(\boldsymbol{\theta}, \boldsymbol{N})}$$

where $\boldsymbol{1}_j$ is a vector has all components zeros except for j and $G^{+i}(\cdot)$ refers to a model with an additional queue identical to queue i in the original model (i.e. same demands).

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

$$\begin{aligned}\frac{\partial Q_{ij}(\boldsymbol{\theta})}{\partial \theta_{ij}} &= \frac{G^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j)}{G(\boldsymbol{\theta}, \boldsymbol{N})} - \frac{\theta_{ij} G^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j) Q_{ij}(\boldsymbol{\theta})}{\theta_{ij} G(\boldsymbol{\theta}, \boldsymbol{N})} \\ &\quad + \frac{\theta_{ij}}{G(\boldsymbol{\theta}, \boldsymbol{N})} \frac{Q^{+j}_{ij}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j)}{\theta_{ij}} G^{+j}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j) \\ &= \frac{G^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j)}{G(\boldsymbol{\theta}, \boldsymbol{N})} (1 + Q^{+i}_{ij}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j) - Q_{ij}(\boldsymbol{\theta})) \\ &= \frac{Q_{ij}(\boldsymbol{\theta})}{\theta_{ij}} (1 + Q^{+i}_{ij}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j) - Q_{ij}(\boldsymbol{\theta}))\end{aligned} \tag{10}$$

For the other cases, we have

$$\begin{aligned}\frac{\partial Q_{ij}(\boldsymbol{\theta})}{\partial \theta_{kh}} &= \frac{\theta_{ij}}{G(\boldsymbol{\theta}, \boldsymbol{N})} \frac{Q^{+i}_{kh}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j)}{\theta_{kh}} G^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j) \\ &\quad - \frac{\theta_{ij} G^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j) Q_{kh}(\boldsymbol{\theta})}{\theta_{kh} G(\boldsymbol{\theta}, \boldsymbol{N})} \\ &= \frac{\theta_{ij} G^{+i}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j)}{\theta_{kh} G(\boldsymbol{\theta}, \boldsymbol{N})} (Q^{+i}_{kh}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j) - Q_{kh}(\boldsymbol{\theta})) \\ &= \frac{Q_{ij}(\boldsymbol{\theta})}{\theta_{kh}} (Q^{+i}_{kh}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j) - Q_{kh}(\boldsymbol{\theta}))\end{aligned} \tag{11}$$

Combine (10)(11) with (7) and (8), the diagonal elements of the Hessian matrix is

$$\begin{aligned}\boldsymbol{H}(\hat{\boldsymbol{\theta}})_{ij,ij} &= \frac{\partial^2 \mathcal{L}(\hat{\boldsymbol{\theta}})}{\partial \hat{\theta}_{ij} \partial \hat{\theta}_{ij}} = L \frac{Q_{ij}(\hat{\boldsymbol{\theta}}) - \widetilde{Q}_{ij}}{\hat{\theta}^2_{ij}} - \frac{L}{\hat{\theta}_{ij}} \frac{\partial Q_{ij}(\hat{\boldsymbol{\theta}})}{\partial \hat{\theta}_{ij}} \\ &= L \frac{Q_{ij}(\hat{\boldsymbol{\theta}})(Q_{ij}(\hat{\boldsymbol{\theta}}) - Q^{+i}_{ij}(\hat{\boldsymbol{\theta}}, \boldsymbol{N} - \boldsymbol{1}_j)) - \widetilde{Q}_{ij}}{\hat{\theta}^2_{ij}}\end{aligned}$$

The non-diagonal elements are

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

which completes the proof. □

Proof of Corollary 1

According to [21, Chapter 9], the distribution of the maximum likelihood estimator $\hat{\boldsymbol{\theta}}$ is asymptotically normal with mean $\hat{\boldsymbol{\theta}}$ and the covariance matrix being approximated by the inverse of the Fisher Information matrix $\boldsymbol{I}(\hat{\boldsymbol{\theta}})$.

The corresponding confidence interval for $\hat{\theta}_{ij}$ is

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

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

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

According to [21, Chapter 2], for the maximum likelihood estimator $\hat{\boldsymbol{\theta}}$, it can be further simplified to

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

The Fisher Information matrix for the maximum likelihood estimates is also referred to as the observed Fisher Information matrix. This completes the proof. □

Proof of Theorem 3

By the Arrival Theorem [25] we have:

$$\theta_{ij} = \frac{Q_{ij}(\boldsymbol{\theta})}{X_j(\boldsymbol{\theta})(1 + A_{ij}(\boldsymbol{\theta}))}$$

where $A_{ij} = \sum_{r=1}^{R} Q_{ir}(\boldsymbol{\theta}, \boldsymbol{N} - \boldsymbol{1}_j)$. Since Little's law on a closed network implies $X_j(\boldsymbol{\theta})\theta_{0,ij} + \sum_{i=1}^{M} Q_{ij}(\boldsymbol{\theta}) = N_j$, we get

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

Direct substitution can be used to check that his expression holds also for the Bard-Schweitzer fixed point. Simplifying we can write

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

for all demand vectors $\boldsymbol{\theta}$. For the Bard-Schweitzer estimator, the last expression becomes (3) since $Q_{ij}(\boldsymbol{\theta}^{bs}) = \widetilde{Q}_{ij}$. This completes the proof. □

Proof of Theorem 4

Similar to the proof of Theorem 1, it is not difficult to verify the existence of the ML estimator. From [8], the relationship in (6) still holds for the load-dependent queueing network. Define $\boldsymbol{\chi} = (\boldsymbol{\theta}, \boldsymbol{\gamma})$, therefore considering θ_{ij}, we have

$$\frac{d\mathcal{L}(\boldsymbol{\chi})}{d\theta_{ij}} = \sum_{l=1}^{L} \left(-\frac{Q_{ij}(\boldsymbol{\chi})}{\theta_{ij}} + \frac{n^l_{ij}}{\theta_{ij}} \right) \mathcal{L}(\boldsymbol{\chi})$$

which implies the condition $Q_{ij}(\hat{\boldsymbol{\chi}}) = \widetilde{Q}_{ij}$.

For $\gamma_k(v)$, we have

$$\begin{aligned}\frac{\partial \log \mathcal{L}(\boldsymbol{\chi})}{\partial \gamma_k(v)} &= \frac{1}{\mathcal{L}(\boldsymbol{\chi})}\frac{\partial \mathcal{L}(\boldsymbol{\chi})}{\partial \gamma_k(v)}\\ &= \frac{L\mathbb{P}(\widetilde{n}_k \geq v)}{\gamma_k(v)} - \frac{L}{G(\boldsymbol{\chi})}\frac{\partial G(\boldsymbol{\chi})}{\partial \gamma_k(v)}\\ &= L\frac{\mathbb{P}(\widetilde{n}_k \geq v) - \mathbb{P}(n_k \geq v|\boldsymbol{\chi})}{\gamma_k(v)}\end{aligned} \tag{12}$$

The stationarity point is $\mathbb{P}(\widetilde{n}_k \geq v) = \mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}})$ which further simplifies to $\mathbb{P}(\widetilde{n}_k = v) = \mathbb{P}(n_k = v|\hat{\boldsymbol{\chi}})$. □

Proof of Theorem 5

Similar as the proof of Theorem 2, derivation of Hessian matrix $\boldsymbol{H}(\hat{\boldsymbol{\chi}})$ requires the evaluation of second-order partial derivatives. First, we consider the partial derivative of $\log \mathcal{L}(\boldsymbol{\chi})$ regarding θ_{ij}

$$\frac{\partial \log \mathcal{L}(\boldsymbol{\chi})}{\partial \theta_{ij}} = \frac{1}{\mathcal{L}(\boldsymbol{\chi})}\frac{\partial \mathcal{L}(\boldsymbol{\chi})}{\partial \theta_{ij}} = L\frac{\widetilde{Q}_{ij}(\boldsymbol{D}) - Q_{ij}(\boldsymbol{\chi})}{\theta_{ij}} \tag{13}$$

The following computes the partial derivative of $Q_{ij}(\boldsymbol{\chi})$ regarding $\theta_{i'j'}$. if $i = i', j = j'$, we have

$$\begin{aligned}\frac{\partial Q_{ij}(\boldsymbol{\chi})}{\partial \theta_{i'j'}} &= \frac{\partial \sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\partial \theta_{ij}}\\ &= \frac{\sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})(n_{ij} - Q_{ij}(\boldsymbol{\chi}))}{\theta_{ij}}\\ &= \frac{\sum_{\boldsymbol{n}\in S} n_{ij}^2\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi}) - Q_{ij}(\boldsymbol{\chi})\sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\theta_{ij}}\\ &= \frac{E[n_{ij}^2|\boldsymbol{\chi}] - Q_{ij}^2(\boldsymbol{\chi})}{\theta_{ij}}\end{aligned} \tag{14}$$

Otherwise, we have

$$\begin{aligned}\frac{\partial Q_{ij}(\boldsymbol{\chi})}{\partial \theta_{i'j'}} &= \frac{\partial \sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\partial \theta_{i'j'}}\\ &= \frac{\sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})(n_{i'j'} - Q_{i'j'}(\boldsymbol{\chi}))}{\theta_{i'j'}}\\ &= \frac{\sum_{\boldsymbol{n}\in S} n_{ij}n_{i'j'}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi}) - Q_{i'j'}(\boldsymbol{\chi})\sum_{\boldsymbol{n}\in S} n_{ij}\mathbb{P}(\boldsymbol{n}|\boldsymbol{\chi})}{\theta_{i'j'}}\\ &= \frac{E[n_{ij}n_{i'j'}|\boldsymbol{\chi}] - Q_{ij}(\boldsymbol{\chi})Q_{i'j'}(\boldsymbol{\chi})}{\theta_{i'j'}}\end{aligned} \tag{15}$$

Substituting (14) and (15) into (13), for $\boldsymbol{H}(\hat{\boldsymbol{\chi}})_{ij,ij}$ we have

$$\begin{aligned}\boldsymbol{H}(\hat{\boldsymbol{\chi}})_{ij,ij} &= \frac{\partial^2 \log \mathcal{L}(\hat{\boldsymbol{\chi}})}{\partial \hat{\theta}_{ij}\partial \hat{\theta}_{ij}}\\ &= L\frac{Q_{ij}(\hat{\boldsymbol{\chi}}) - \widetilde{Q}_{ij}(\boldsymbol{D})}{\hat{\theta}_{ij}^2} - \frac{L}{\hat{\theta}_{ij}}\frac{\partial Q_{ij}(\hat{\boldsymbol{\chi}})}{\hat{\theta}_{ij}}\\ &= \frac{L}{\hat{\theta}_{ij}^2}\left(Q_{ij}^2(\hat{\boldsymbol{\chi}}) - E[n_{ij}^2|\hat{\boldsymbol{\chi}}] + Q_{ij}(\hat{\boldsymbol{\chi}}) - \widetilde{Q}_{ij}(\boldsymbol{D})\right)\end{aligned}$$

for $\boldsymbol{H}(\hat{\boldsymbol{\chi}})_{ij,i'j'}$, we have

$$\begin{aligned}\boldsymbol{H}(\hat{\boldsymbol{\chi}})_{ij,i'j'} &= \frac{\partial^2 \log \mathcal{L}(\hat{\boldsymbol{\chi}})}{\partial \hat{\theta}_{ij}\partial \hat{\theta}_{i'j'}} = -\frac{L}{\hat{\theta}_{ij}}\frac{\partial Q_{ij}(\hat{\boldsymbol{\chi}})}{\hat{\theta}_{i'j'}}\\ &= L\frac{Q_{ij}(\hat{\boldsymbol{\chi}})Q_{i'j'}(\hat{\boldsymbol{\chi}}) - E[n_{ij}n_{i'j'}|\hat{\boldsymbol{\chi}}]}{\hat{\theta}_{ij}\hat{\theta}_{i'j'}}\end{aligned}$$

For computing $\frac{\partial^2 \log \mathcal{L}(\boldsymbol{\chi})}{\partial \gamma_k(v)\partial \theta_{ij}}$, $\frac{\partial \log \mathcal{L}(\boldsymbol{\chi})}{\partial \gamma_k(v)}$ is already given in (12). Noticing that

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

Therefore we have

$$\begin{aligned}\boldsymbol{H}(\hat{\boldsymbol{\chi}})_{ij,kv} &= \frac{\partial \log \mathcal{L}(\hat{\boldsymbol{\chi}})}{\partial \hat{\gamma}_k(v)\partial \hat{\theta}_{ij}}\\ &= L\frac{Q_{ij}(\hat{\boldsymbol{\chi}})\mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}}) - E[n_{ij}|\hat{\boldsymbol{\chi}}, n_k \geq v]}{\hat{\chi}_{ij}\hat{\gamma}_k(v)}\end{aligned}$$

Finally, we consider to compute $\frac{\partial^2 \log \mathcal{L}(\boldsymbol{\chi})}{\partial \gamma_k(v)\partial \gamma_k'(v')}$. If $k = k', v = v'$

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

For cases otherwise

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

Therefore, for $\boldsymbol{H}(\hat{\boldsymbol{\chi}})_{kv,kv}$ we have

$$\begin{aligned}\boldsymbol{H}(\hat{\boldsymbol{\chi}})_{kv,kv} &= \frac{\partial^2 \log \mathcal{L}(\hat{\boldsymbol{\chi}})}{\partial \hat{\gamma}_k(v)\partial \hat{\gamma}_k(v)}\\ &= L\frac{\mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}}) - \mathbb{P}(\widetilde{n}_k \geq v|\boldsymbol{D})}{\hat{\gamma}_k(v)^2}\\ &\quad - L\mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}})\frac{1 - \mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}})}{\hat{\gamma}_k(v)^2}\\ &= L\frac{\mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}})^2 - \mathbb{P}(\widetilde{n}_k \geq v|\boldsymbol{D})}{\hat{\gamma}_k(v)^2}\end{aligned}$$

For $\boldsymbol{H}(\hat{\boldsymbol{\chi}})_{kv,k'v'}$

$$\begin{aligned}&\boldsymbol{H}(\hat{\boldsymbol{\chi}})_{kv,k'v'} = \frac{\partial \log \mathcal{L}(\hat{\boldsymbol{\chi}})}{\partial \hat{\gamma}_k(v)\partial \hat{\gamma}_{k'}(v')}\\ &= L\frac{\mathbb{P}(n_k \geq v|\hat{\boldsymbol{\chi}})\mathbb{P}(n_{k'} \geq v'|\hat{\boldsymbol{\chi}}) - \mathbb{P}(n_k \geq v, n_{k'} \geq v'|\hat{\boldsymbol{\chi}})}{\hat{\gamma}_k(v)\hat{\gamma}_{k'}(v')}\end{aligned}$$

which completes the proof. □