Accelerating Performance Inference over Closed Systems by Asymptotic Methods

GIULIANO CASALE, Department of Computing, Imperial College London

Recent years have seen a rapid growth of interest in exploiting monitoring data collected from enterprise applications for automated management and performance analysis. In spite of this trend, even simple performance inference problems involving queueing theoretic formulas often incur computational bottlenecks, for example upon computing likelihoods in models of batch systems. Motivated by this issue, we revisit the solution of multiclass closed queueing networks, which are popular models used to describe batch and distributed applications with parallelism constraints. We first prove that the normalizing constant of the equilibrium state probabilities of a closed model can be reformulated exactly as a multidimensional integral over the unit simplex. This gives as a by-product novel explicit expressions for the multiclass normalizing constant. We then derive a method based on cubature rules to efficiently evaluate the proposed integral form in small and medium-sized models. For large models, we propose novel asymptotic expansions and Monte Carlo sampling methods to efficiently and accurately approximate normalizing constants and likelihoods. We illustrate the resulting accuracy gains in problems involving optimization-based inference.

ACM Reference format:
Giuliano Casale. 2017. Accelerating Performance Inference over Closed Systems by Asymptotic Methods. *Proc. ACM Meas. Anal. Comput. Syst.* 1, 1, Article 7 (June 2017), 24 pages.
DOI: http://dx.doi.org/10.1145/3084445

1 INTRODUCTION

During the last decade there has been a growing trend among enterprises toward exploiting large volumes of monitoring data for performance management [18]. While activities such as capacity planning have been traditionally carried out by human experts, software systems to automatically forecast capacity needs are increasingly widespread in the industry. A common issue faced by these systems is automated performance model selection and parameterization, which can be dealt with using inference methods [17, 39, 45, 46]. We here focus on inference of closed queueing network models, which are often used to describe batch systems and distributed applications with parallelism limits. In such models, likelihoods can be expressed analytically if the scheduling disciplines at the resources comply with standard product-form assumptions [6].

The main challenge in computing likelihoods in closed systems is to determine the normalizing constant of state probabilities, which appears explicitly in the likelihood function. Prior work has proposed methods to exactly compute normalizing constants using recursive algorithms [9, 15, 31, 41], generating functions [8, 25], and moment-based methods [10, 11]. Furthermore, methods based on Laplace transform inversion [13], asymptotic expansions [33, 37], and Monte Carlo integration [43] have led to inexpensive approximations of the normalizing constant for large models. Still, we find that maximum likelihood estimation problems remain either too expensive to solve or return largely inaccurate solutions, depending on the method used to compute the normalizing constant.

DOI: http://dx.doi.org/10.1145/3084445

Motivated by these observations, we revisit the computational theory of product-form multiclass closed queueing networks. Our main result is to reformulate the normalizing constant in terms of an integral over the unit simplex. This development leads to novel asymptotic expansions for the normalising constant based on Laplace's method [27], which are obtained through a novel scaling that adds at every node a set of jobs that continuously loop at that node. We also obtain a novel Monte Carlo integration method, which enables the efficient sampling of the normalizing constant. Moreover, we derive novel explicit solutions for the multiclass normalizing constant in terms of algebraic sums, with time complexities similar to recursive methods such as convolution, RECAL, and mean-value analysis, but constant space requirements [9, 15, 41, 42]. We validate the effectiveness of the proposed approximations using a numerical validation involving thousands of maximum likelihood estimation problems.

The rest of the paper is organized as follows. Section 2 introduces the reference model for closed systems and surveys related work. Section 3 gives novel exact theoretical results concerning the solution of closed product-form networks. Section 4 develops asymptotic expansions and Monte Carlo integration methods. Section 5 investigates the accuracy of the proposed techniques. Lastly, Section 6 summarizes results and concludes the paper. Proofs for some technical prerequisites are given in the Appendix.

2 BACKGROUND

2.1 Notation

The reference model is a product-form closed queueing network with M nodes and R job classes [6]. Let N_r be the number of jobs in class r and define the population vector $\boldsymbol{N} = (N_1, \ldots, N_R)^T$, $N = \sum_r N_r$. We assume that the first $K \leq M$ nodes have a single-server and that the remaining ones are infinite server nodes. Matrix $\boldsymbol{\theta} = [\theta_{kr}]$ collects the demands placed by class-r jobs at node k, *i.e.*, the product of the mean number of visits by the mean service time of the job. We denote by $\sigma_r = \sum_{k=K+1}^{M} \theta_{kr}$ the sum of the class-r demands at all infinite servers.

Consider for example a product-form network of processor sharing and infinite server nodes having exponential service times. In this case, the model maps to a Markov process with state space $\mathcal{S}_M = \{\boldsymbol{n} \in \mathbb{N}^{MR} \mid n_{kr} \geq 0, \sum_{k=1}^{M} n_{kr} = N_r\}$, where $\boldsymbol{n} = (\boldsymbol{n}_1, \ldots, \boldsymbol{n}_M)$, $\boldsymbol{n}_k = (n_{k1}, \ldots, n_{kR})$, and n_{kr} is the number of class-r jobs at node k. The equilibrium distribution for this process is given by [6]

$$\pi(\boldsymbol{n}) = \frac{1}{G_{\boldsymbol{\theta}}(\boldsymbol{N})} \prod_{i=1}^{K} n_i! \prod_{k=1}^{M} \prod_{r=1}^{R} \frac{\theta_{kr}^{n_{kr}}}{n_{kr}!} \qquad \boldsymbol{n} \in \mathcal{S}_M \tag{1}$$

where $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ is a normalizing constant over $\mathcal{S}_M$ and for vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ we define $v = v_1 + \ldots + v_n$. By the given definitions, the normalizing constant may be written as

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \sum_{\boldsymbol{n} \in \mathcal{S}_M} \prod_{i=1}^{K} n_i! \prod_{k=1}^{M} \prod_{r=1}^{R} \frac{\theta_{kr}^{n_{kr}}}{n_{kr}!} \tag{2}$$

The last expression is valid for arbitrary multiclass product-form queueing networks defined in the sense of the BCMP theorem [6].

2.2 Computational methods: state-of-the-art

Since the number of states of the queueing network model grows as $O(N^{MR})$ with the job population, it is usually infeasible to obtain $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ by direct summation over the state space $\mathcal{S}_M$. To tackle this issue, several computational methods have been defined over a time span of four decades. We limit here to give a high-level review, pointing to the references for details.

2.2.1 Exact methods. The classic exact computational methods for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ are the multiclass convolution algorithm (CA) [41] and RECAL [15], which feature respectively $O(N^R)$ and $O(N^K)$ time and space requirements. Such polynomial complexities limit the application of these methods to models with a few classes or queues. Other exact algorithms with similar complexities may be found in [8, 20, 25, 42]. The method of moments (MoM) [11] lowers the requirements approximately to $O(N^2 \log N)$ time and $O(N \log N)$ space using a recursive system of linear equations, applicable under certain regularity conditions on $\boldsymbol{\theta}$. This method becomes computationally demanding as M and R grow simultaneously and solution times in large models are of the order of minutes, thus too expensive for optimization-based performance inference. Summarizing, several exact methods for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ exist, but they are hardly applicable to performance inference problems due to their cost. Experiments illustrating these limitations are shown in Section 2.3.

2.2.2 Approximate methods. Approximate mean-value analysis (AMVA) algorithms [44] provide accurate estimates of mean performance measures and are $O(1)$ with respect to job populations. Yet the focus on mean performance metrics is restrictive since inference problems typically require a probability model such as (1), for example to express prior distributions on parameters or to infer an optimal parameterization using likelihood maximization [40, 45]. AMVA does not apply to these problems as it neither computes likelihoods nor probabilities.

Flow-equivalent methods alternatively aggregate a subnetwork into a node with state-dependent service rates, which may be solved for state probabilities [12]. Unfortunately, this method is normally too expensive to apply in multiclass models, where the parameterization of the flow-equivalent server requires to determine service rates under all possible combinations of jobs residing at the node.

Such limitations are addressed by specialized approximations for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$, which include Monte Carlo integration, numerical methods, and asymptotic expansions. Monte Carlo integration methods are first introduced in [43], based on the following integral form [37]

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \frac{1}{N_1! \cdots N_R!} \int_{\mathbb{R}_+^K} e^{-y} \prod_{r=1}^{R} \left(\sigma_r + \sum_{k=1}^{K} \theta_{kr} y_k \right)^{N_r} d\boldsymbol{y} \tag{3}$$

where $\mathbb{R}_+^K = \{\boldsymbol{y} \in \mathbb{R}^K | \boldsymbol{y} \geq \boldsymbol{0}\}$, with $\boldsymbol{0} = (0, \ldots, 0)^T$ and $y = \sum_k y_k$. Expression (3) is obtained by expressing the $n_i!$ terms in (1) using the integral form of the gamma function and subsequently by repeated application of the multinomial theorem

$$\left(\sum_{k=1}^{K} x_k \right)^V = V! \sum_{\substack{\boldsymbol{v} \geq 0 \\ v = V}} \prod_{k=1}^{K} \frac{x_k^{v_k}}{v_k!} \tag{4}$$

where $\boldsymbol{v} \in \mathbb{N}^K$. The integral form (3) can be efficiently evaluated using importance sampling [43], normally requiring $J = 10^5 - 10^7$ samples to approximate $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ with low variance. Computing millions of samples is acceptable for evaluating individual models, but places an excessive overhead for use in optimization-based inference. Moreover, the variance of the Monte Carlo estimators for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ can adversely affect the identification of the search direction [47].

Numerical methods for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ include Laplace transform inversion (LTI) [13] and ODE-based methods based on Taylor expansion (TE) [47]. LTI allows for arbitrary approximation accuracy, but can still incur a significant computational cost. For instance, [13, p. 967] provides an example where LTI requires $\approx 10^{12}$ operations on a model with $K = 64$ and $R = 9$, which is beyond the acceptable cost for a single iteration of an optimization program. Instead, the approximations proposed in this paper scale efficiently to models of this size. TE is theoretically $O(1)$, but it becomes difficult to apply in large models due to the rapid growth of $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ that affects numerical precision.

Table 1. Demand estimation results. T = timeout (10 min). Runtimes are rounded up to the nearest integer. On this example, memory requirements are negligible for all methods.

$M = K = 3, R = 3$	Abs. Perc. Error (%)			Time (seconds)		
$N_r = N/R =$	*2*	*20*	*40*	*2*	*20*	*40*
CA (No timeout)	0.3	0.0	0.0	2	419	3413
CA	0.3	0.0	40.3	2	419	T
MCI3	90.9	138.9	115.1	5	5	7
MCI6	135.9	91.1	118.3	185	171	137
MoM	15.4	51.2	71.0	T	T	T
NOG	38.6	92.2	96.1	1	1	8
RAY	82.1	72.8	59.4	6	3	8
RECAL	0.3	31.7	76.5	3	T	T
TE-2	38.7	92.2	96.1	143	174	138
TE-3	35.2	91.3	95.6	T	T	T

2.2.3 Asymptotic expansions. Asymptotic expansions for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ are $O(1)$ and thus capable of accelerating optimization. Expansions appear in [29, 33–35, 37] and are discussed below. Other asymptotic methods exist but they are not relevant to the present work as they either focus on single-class models only [19] or study asymptotic values of mean-value performance metrics [2, 4, 7, 28, 30], whereas we focus here on computing likelihoods in multiclass systems.

PANACEA (PAN) [37] is applicable only to models with infinite servers and in normal usage, *i.e.*, where resources are lightly utilized so that $\max_i \alpha_i < 1$, where $\alpha_i = \sum_r N_r \theta_{ir} \sigma_r^{-1}$, $i = 1, \ldots, K$. Normal usage conditions tend to be restrictive in applications, where the analysis of heavy-load regimes is of practical importance.

The ray method [29] (RAY) is an approximation method for PDEs. Combined with singular perturbation theory, RAY provides an approximation for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ under an increasing number of nodes and a simultaneous scaling of their service demands. We extensively compare our results against the baseline provided by this method.

Saddle-point approximation (SPA) provides asymptotic expansions of contour integrals arising from the generating function of $G_{\boldsymbol{\theta}}(\boldsymbol{N})$. A limitation of SPA is that explicit formulas are available only for small models, although it principle the method may be generalized [33–35].

As we show in Section 4, our asymptotic expansions are based on Laplace's method [27], which may be seen as a specialization of the saddle-point method for real integrals. This substantially differs from the SPA method which applies to contour integrals in the complex domain and leads to rather different expressions for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$.

2.3 Motivating example: demand estimation from state samples

To illustrate the limitations of existing computational techniques, we compare prior art methods in a likelihood maximization application. Assume to measure a set of L state samples, $\boldsymbol{n}^{(l)} \in \mathcal{S}_M$, $l = 1, \ldots, L$. We seek for a maximum likelihood estimator (MLE) for the demand matrix $\boldsymbol{\theta}$. In practice, problems of this kind arise during model selection and calibration, where one seeks for an optimal parameterization and the $\boldsymbol{n}^{(l)}$ samples represent system state measurements. Likelihood-based estimation offers a number of advantages over other estimation techniques, for example it can cope with missing and aggregate data [40].

We assume $\sigma_r = 0$ and knowledge of $\theta_r = \sum_k \theta_{kr}, \forall r$, *i.e.*, the end-to-end response time of a single class-r request when $N = 1$. From (1) the log-likelihood of $\boldsymbol{\theta}$ is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{l=1}^{L} \log C(\boldsymbol{n}^{(l)}) + L \sum_k \sum_r \widetilde{Q}_{kr} \log \theta_{kr} - L \log G_{\boldsymbol{\theta}}(\boldsymbol{N}) \tag{5}$$

where $\widetilde{Q}_{kr} = \sum_l n_{kr}^{(l)}/L$ is the measured mean queue-length of class r at node k. The first term can be neglected upon optimizing over $\boldsymbol{\theta} \geq \mathbf{0}$. The cost of computing $\mathcal{L}(\boldsymbol{\theta})$ is thus dominated by the cost of determining $\log G_{\boldsymbol{\theta}}(\boldsymbol{N})$.

We consider (5) for a model with $M = K = 3$, $R = 3$, $\boldsymbol{N} = (N, N, N)/3$, and $\theta_{kr} = k * r$ and seek for a (local) MLE that maximizes $\mathcal{L}(\boldsymbol{\theta})/L$ subject to $\boldsymbol{\theta} \geq \mathbf{0}$. We also let $L \to \infty$ by using in place of the $\widetilde{Q}_{kr}$ the exact mean queue-lengths computed by mean-value analysis [42]. We apply MATLAB's fmincon interior point algorithm, and calculate $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ at each iteration using one among CA, RECAL, MoM, TE, or RAY. We also use Monte Carlo integration (MCI) with AMVA-based initialization [47]. The assumptions underpinning PAN and SPA are not met on this example: PAN requires infinite servers; SPA is not available for models with 3 nodes and 3 classes or larger. TE also requires infinite servers, but we can set $\sigma_r = 10^{-8}, \forall r$; a similar perturbation cannot be used with PAN since the method also requires normal usage. Lastly, we include in the experiment a variant of (5) where we neglect the normalizing constant by setting $\log G_{\boldsymbol{\theta}}(\boldsymbol{N}) = 0$. This variant is denoted by NOG and corresponds to the log-likelihood formula for a product-form open queueing network with demands $\boldsymbol{\theta}$. Each chosen method is initialized at the same point, sampled from a uniform distribution. We set a timeout of 10 minutes, after which fmincon returns after completing the running iteration.Experiments are run on a quad-core desktop computer.

Table 1 shows execution times with $N_r = N/R = 2, 20, 40$ jobs and the absolute percentage errors of the returned demands with respect to the true $\boldsymbol{\theta}$. The suffixes for MCI and TE are the number of samples J and the scale of the ODE step size τ, respectively, *e.g.*, MCI3 has $J = 10^3$ and TE-2 has $\tau = 10^{-2}$. The *CA (No timeout)* method is exact and thus provides an upper bound on achievable accuracy on this instance. For this method, we run the optimization until termination, computing the normalizing constant at each iteration using CA. This baseline is required since the problem (5) is non-convex, thus the choice of the initial point affects the achievable accuracy and it is thus undesirable to reason on absolute error alone.

We note that all methods incur a considerable degradation of accuracy and running times as the population N grows. Some methods, such as TE, have a similar (or worse) performance than NOG, which ignores the normalizing constant. CA is the best among the exact methods, but its execution times grow quickly and on larger models become infeasible. Among existing approximations, the RAY asymptotic expansion achieves the best results, although the errors remain high, around 59%-82%. However, computational times are scalable. A similar conclusion applies to MCI with a small number of samples. This motivates us to further investigate into asymptotic expansions and Monte Carlo integration methods. We also remark that on this example the expansion proposed later in Section 4 achieves less than 0.7% absolute percentage error in all the three cases, with runtimes between 2*s* and 4*s*. A validation on a broader set of instances is presented in Section 5.

3 EXACT RESULTS

In order to inexpensively approximate the normalizing constant, we first derive novel integral expressions for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$. This derivation leads to novel numerical approximations and provides a theoretical baseline for developing asymptotic results.

3.1 Integral form over the unit simplex

We first derive an exact integral form for the normalizing constant in networks without infinite servers.

Theorem 3.1. *In a multiclass closed queueing network with K single-server nodes*

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \frac{(N+K-1)!}{N_1!\cdots N_R!}\int_{\Delta_K}\prod_{r=1}^{R}\left(\sum_{k=1}^{K}\theta_{kr}u_k\right)^{N_r} d\boldsymbol{u} \tag{6}$$

where $\Delta_K = \{\boldsymbol{u} \in \mathbb{R}^K \mid u_i \geq 0, \sum_{i=1}^{K} u_i = 1\}$ *is the unit simplex.*

Proof. The multinomial theorem (4) implies that for any set of real numbers $(a_1, \ldots, a_R)$ and variables $\boldsymbol{t} = (t_1, \ldots, t_R)^T$ we can write

$$a_1^{N_1}a_2^{N_2}\cdots a_R^{N_R} = \frac{1}{N!}\frac{\partial^{N_1}\cdots\partial^{N_R}}{\partial t_1^{N_1}\cdots\partial t_R^{N_R}}\left(\sum_{i=1}^{R}a_i t_i\right)^N \tag{7}$$

Since $\sigma_r = 0$, we apply (7) to the product in the integrand of (3) with $a_r = \sum_k \theta_{kr}y_k$, finding after exchanging the order of differentiation and integration

$$\begin{aligned} G_{\boldsymbol{\theta}}(\boldsymbol{N}) &= \frac{1}{N_1!\cdots N_R!}\frac{\partial^{N_1}\cdots\partial^{N_R}}{\partial t_1^{N_1}\cdots\partial t_R^{N_R}}\int_{\mathbb{R}_+^K}\frac{e^{-y}}{N!}\left(\sum_{k=1}^{K}\theta_k y_k\right)^N d\boldsymbol{y} \\ &= \frac{1}{N_1!\cdots N_R!}\frac{\partial^{N_1}\cdots\partial^{N_R}}{\partial t_1^{N_1}\cdots\partial t_R^{N_R}} g_{\boldsymbol{\theta t}}(N) \end{aligned} \tag{8}$$

where $g_{\boldsymbol{\theta t}}(N)$ is the normalizing constant of a single-class model with demands $\theta_k = \sum_r \theta_{kr}t_r$, $k = 1, \ldots, K$, and the last passage follows by (3). We then prove in Appendix A the following equivalence

$$g_{\boldsymbol{\theta t}}(N) = [\theta_1, \ldots, \theta_K]x^{N+K-1} \tag{9}$$

where the right-hand side is the divided difference[1] of x^{N+K-1} relatively to the interpolation points $\theta_1, \ldots, \theta_K$. This expression is valid for single-class normalizing constants with arbitrary demands. We can then apply to the last expression the Hermite-Genocchi formula [3], which is a classic integral form for divided differences

$$[\theta_1, \ldots, \theta_K]f(x) = \int_{\Delta_K} f^{(K-1)}(\theta_1 u_1 + \ldots + \theta_K u_K)d\boldsymbol{u} \tag{10}$$

where $f^{(K-1)}(x)$ is the $(K-1)$th derivative of $f(x)$ and Δ_K is the unit simplex. Here we set $f(x) = x^{N+K-1}$ and show in Appendix B that, for this specific choice of $f(x)$, (10) also holds under nondistinct θ_i, a case normally not covered by the Hermite-Genocchi formula. Using (9) in (8), followed by (10), we get

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \frac{(N+K-1)!}{N_1!\cdots N_R!}\int_{\Delta_K}\frac{1}{N!}\frac{\partial^{N_1}\cdots\partial^{N_R}}{\partial t_1^{N_1}\cdots\partial t_R^{N_R}}\left(\sum_{k=1}^{K}\theta_k u_k\right)^N d\boldsymbol{u}$$

Recalling that $\theta_k = \sum_r \theta_{kr}t_r$ and applying (7) to the integrand, we find (6). □

Theorem 3.1 provides a novel integral form for the multiclass normalizing constant, with an integrand similar to (3), but defined over a bounded domain. It is also possible to verify that (3) follows from (6) using the Laplace transform, once the integration domain is reformulated in a suitable parametric form [36].

[1]For a given function $f(x)$, divided differences extend the notion of forward difference of $f(x)$ to a set of interpolation points arbitrarily located in the domain of $f(x)$. We point to [3, 38] for further details.

It is useful to note that a shorter proof of Theorem 3.1 follows by first applying the multinomial theorem (4) to each factor in the integrand of (6) and then using term-by-term the Dirichlet integral

$$\int_{\Delta_K} \prod_{k=1}^{K} u_k^{n_k} d\boldsymbol{u} = \frac{\prod_{k=1}^{K} n_k!}{(\sum_{k=1}^{K} n_k + K - 1)!} \tag{11}$$

This yields (2) after noting that in the absence of infinite server nodes it is $\sum_{k=1}^{K} n_k = N$. Compared to this simple derivation, the proof of Theorem 3.1 introduces (7), which is used in the next section to obtain explicit solutions. Similar mappings between sums and products are important also in multivariate statistical analysis [26]. Moreover, the proof of Theorem 3.1 shows that multiclass normalizing constants may be expressed as derivatives of single-class normalizing constants, and that the latter may be seen as divided differences of the power function.

3.2 Explicit solutions

While our interest is on deriving approximations, novel exact computational formulas may also be obtained from Theorem 3.1. Such expressions are not used throughout due to their cost, but they appear of theoretical interest due to the lack of similar expressions for the multiclass normalizing constant.

Corollary 3.2. *The normalizing constant of a closed multiclass queueing network is given by*

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \sum_{\mathbf{0} \le \boldsymbol{t} \le \boldsymbol{N}} \frac{(-1)^{N-t}}{N_1! \cdots N_R!} \prod_{r=1}^{R} \binom{N_r}{t_r} g_{\boldsymbol{\theta t}}(N) \tag{12}$$

where $\boldsymbol{t} = (t_1, \ldots, t_R)^T$, $t = \sum_r t_r$, *and* $g_{\boldsymbol{\theta t}}(N)$ *is the normalizing constant of a single-class model with demands* $\theta_k = \sum_{r=1}^{R} t_r \theta_{kr}$.

Proof. We consider finite differences [38], where (7) is replaced by [5]

$$a_1^{N_1} a_2^{N_2} \cdots a_R^{N_R} = \sum_{\mathbf{0} \le \boldsymbol{t} \le \boldsymbol{N}} \frac{(-1)^{N-t}}{N!} \prod_{s=1}^{R} \binom{N_s}{t_s} \left(\sum_{r=1}^{R} t_r a_r \right)^N \tag{13}$$

The result follows by applying (13) to the product in the integrand of (6) and recognizing $g_{\boldsymbol{\theta t}}(N)$ in the resulting expression. □

Computing $g_{\boldsymbol{\theta t}}(N)$ explicitly using the closed-form formulas in [8, Eq. 3.12] implies for (12) a theoretical complexity of $O(N^R)$ time and $O(1)$ space. For example, in the special case where demands are distinct, the normalizing constant can be computed in $O(1)$ as [8, 32]

$$g_{\boldsymbol{\theta t}}(N) = \sum_{k=1}^{K} \frac{\theta_k^{N+K-1}}{\prod_{i \ne k} (\theta_k - \theta_i)} \tag{14}$$

yielding by (12) the following explicit expression for the normalizing constant

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \sum_{\mathbf{0} \le \boldsymbol{t} \le \boldsymbol{N}} \frac{(-1)^{N-t}}{N_1! \cdots N_R!} \prod_{r=1}^{R} \binom{N_r}{t_r} \sum_{k=1}^{K} \frac{(\sum_{s=1}^{R} t_s \theta_{ks})^{N+K-1}}{\prod_{i \ne k} (\sum_{s=1}^{R} t_s (\theta_{ks} - \theta_{is}))} \tag{15}$$

A similar formula holds for the general case if one uses [8, Eq. 3.12] in place of (14). Consider the single-class demands $\theta_k = \sum_{r=1}^{R} t_r \theta_{kr}$. Assume that θ_k has multiplicity $m_k \ge 1$ and let K' be the number of distinct demands.

Plugging [8, Eq. 3.12] into (12) yields the general expression

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \sum_{\mathbf{0}\leq \boldsymbol{t}\leq \boldsymbol{N}} \frac{(-1)^{N-t}}{N_1!\cdots N_R!} \prod_{s=1}^{R} \binom{N_s}{t_s} \sum_{j=1}^{K'} (-1)^{m_j-1} (\textstyle\sum_{s=1}^{R} t_s \theta_{js})^{N+M-m_j} \times \sum_{\substack{\boldsymbol{r}\geq 0 \\ r=m_j-1}} (-1)^{r_j} \binom{N+r_j}{r_j} \prod_{\substack{k=1 \\ k\neq j}}^{K'} \binom{m_k+r_k-1}{r_k} \frac{(\sum_{s=1}^{R} t_s \theta_{ks})^{r_k}}{(\sum_{s=1}^{R} t_s(\theta_{js}-\theta_{ks}))^{m_k+r_k}} \quad (16)$$

where $\boldsymbol{r} \in \mathbb{N}^{K'}$, $r = \sum_{j=1}^{K'} r_j$. We are also in condition to derive another explicit formula for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$.

Corollary 3.3. *The normalizing constant of a closed multiclass queueing network model can be expressed as*

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \sum_{\substack{\boldsymbol{h}\geq \mathbf{0}: \\ h\leq N}} \frac{(-1)^{N-h}}{N_1!\cdots N_R!} \binom{N+K-1}{N-h} \prod_{r=1}^{R} \left(\sum_{i=1}^{K} h_i \theta_{ir} \right)^{N_r} \quad (17)$$

where $\boldsymbol{h} \in \mathbb{N}^K$ *and* $h = \sum_i h_i$.

Proof. Observe that the specialization of (1) to single-class models is

$$g_{\boldsymbol{\theta}\boldsymbol{t}}(N) = \sum_{\boldsymbol{m}\in \mathcal{S}_K} \prod_{k=1}^{K} \theta_k^{m_k} \quad (18)$$

Applying (13) to (18) with $a_k = \theta_k$, and using the definition of $\mathcal{S}_K$ yields

$$g_{\boldsymbol{\theta}\boldsymbol{t}}(N) = \sum_{\substack{\boldsymbol{m}\geq \mathbf{0}: \\ m=N}} \sum_{\mathbf{0}\leq \boldsymbol{h}\leq \boldsymbol{m}} \frac{(-1)^{N-h}}{N!} \prod_{j=1}^{K} \binom{m_j}{h_j} \left(\sum_{i=1}^{K} \sum_{r=1}^{R} h_i t_r \theta_{ir} \right)^{N}$$

with $h = \sum_i h_i$. Plugging the last formula into (12) and using (13) to eliminate the dependence on $\boldsymbol{t}$ gives after rearranging terms

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \sum_{\substack{\boldsymbol{m}\geq \mathbf{0}: \\ m=N}} \sum_{\mathbf{0}\leq \boldsymbol{h}\leq \boldsymbol{m}} \frac{(-1)^{N-h}}{N_1!\cdots N_R!} \prod_{j=1}^{K} \binom{m_j}{h_j} \prod_{r=1}^{R} \left(\sum_{i=1}^{K} h_i \theta_{ir} \right)^{N_r}$$

We can here use a single summation on $\boldsymbol{h} \geq \mathbf{0}$, $h \leq N$, after noting that

$$\sum_{\substack{\boldsymbol{m}\geq \boldsymbol{h}: \\ m=N}} \prod_{i=1}^{K} \binom{m_i}{h_i} = \binom{N+K-1}{N-h} \quad (19)$$

This identity can be proved by first rewriting the expression in terms of $\boldsymbol{v} = \boldsymbol{m} - \boldsymbol{h} \geq \mathbf{0}$ and then iteratively applying a corollary of Vandermonde's convolution [21, Eq. 3]. □

This explicit form requires $O(N^K)$ time and $O(1)$ space, which makes it preferable to (12) on models with many classes, but a small number of nodes. To the best of our knowledge, (12) and (17) are the only exact and tractable algebraic expressions for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ with a $O(1)$ space requirement. This improves over the space requirements of recursive algorithms such as CA and RECAL, while retaining the same time complexities, which may be useful in cases where one wants to solve several models in parallel without incurring into memory bottlenecks. However, in practice the above expressions are applicable only to models where $H = \min(K, R)$ is not too large (*e.g.*, $H \leq 4$), typically with up to a few tens of jobs. Moreover, due to the large magnitude of the terms, multi-precision

arithmetic should be used to avoid numerical issues upon computing (12) and (17). The techniques developed later do not suffer these problems and can help to approximate larger models.

3.3 Infinite server nodes

Consider now a model where the first K nodes are single-server queues and the remaining $M-K$ nodes are infinite servers. The following corollary generalizes the integral form.

Corollary 3.4. *In a model including infinite server nodes*

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \int_{\Delta_K} \int_{v=0}^{+\infty} \frac{e^{-v} v^{K-1}}{N_1! \cdots N_R!} \prod_{r=1}^{R} \left(\sigma_r + v \sum_{k=1}^{K} \theta_{kr} u_k \right)^{N_r} dv \, d\boldsymbol{u} \tag{20}$$

Proof. We plug the definition of $\sigma_r = \sum_{i=K+1}^{M} \theta_{kr}$ in (20) and use the multinomial theorem (4) to write

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \sum_{\boldsymbol{n} \in \mathcal{S}_M} \int_{\Delta_K} \int_{v=0}^{+\infty} e^{-v} v^{n+K-1} \prod_{i=1}^{M} \prod_{r=1}^{R} \frac{\theta_{ir}^{n_{ir}}}{n_{ir}!} \prod_{k=1}^{K} u_k^{n_k} dv \, d\boldsymbol{u}$$

where $n = \sum_{k=1}^{K} n_k$ and $n_k = \sum_r n_{kr}$. Noting that $\int_{v=0}^{+\infty} e^{-v} v^{n+K-1} dv = (n+K-1)!$, we obtain (2) by (11). □

3.4 Numerical evaluation

Cubature rules are interpolation formulas that approximate a multidimensional integral by computing the integrand at a finite set of points [14]. For polynomial integrands, cubature rules may also allow the exact evaluation of the integral, if interpolation occurs at a large enough set of points. An advantage of (6) over (3) is that it expresses $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ as an integral of a polynomial over the simplex, making it suitable for application of cubature rules.

We focus here on Grundmann-Möller (GM) cubature rules, which are tailored to the exact and approximate integration of polynomials over the simplex [23]. Applying directly the definition of GM cubature rule of degree $2S+1$ to (6) leads to the following expression [23]

$$G_{\boldsymbol{\theta}}(\boldsymbol{N}) = \frac{(N+K-1)!}{N_1! \cdots N_R!} \sum_{i=0}^{S} w_i \sum_{\substack{\boldsymbol{b} \geq \boldsymbol{0}: \\ b = S-i}} \prod_{r=1}^{R} \left(\sum_{j=1}^{K} \frac{(2b_j+1)\theta_{jr}}{(2S+K-2i)} \right)^{N_r} \tag{21}$$

where $\boldsymbol{b} \in \mathbb{N}^K$, $b = \sum_{i=1}^{K} b_i$, and the weights are

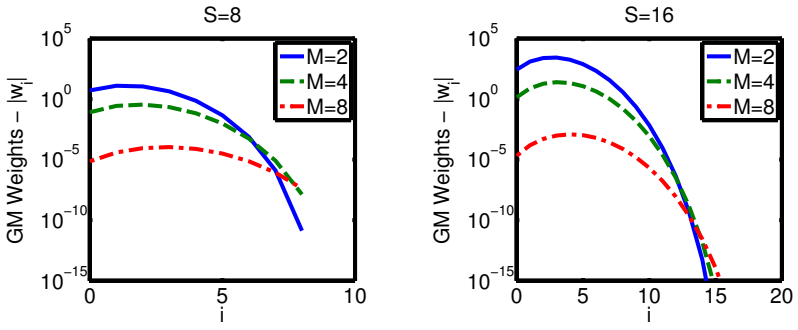
$$w_i = (-1)^i 2^{-2S} \frac{(2S+K-2i)^{2S+1}}{i!(2S+K-i)!}$$

The number of points in the rule (21) is $L = \binom{K+S}{S}$, thus worst-case complexity is $O(S^K)$ time and $O(1)$ space.

If the integrand is a multivariate polynomial of degree N, then a GM cubature rule of degree $S = \lceil (N-1)/2 \rceil$ returns the exact value of the integral in $O((N/2)^K)$ time and $O(1)$ space [23]. This is indeed the case for both (6) and (20). In the case without infinite servers, this is evident since the integrand is a product of linear forms. We now show that the same conclusion holds in models with infinite servers. Let $G^{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{N})$ be the normalizing constant for a model composed of an infinite server node with demand σ_r and K identical single-server nodes having class-r demand $\tilde{\theta}_r = \sum_k \theta_{kr} u_k$. We have the following result.

Proposition 3.5.

$$\int_{v=0}^{+\infty} \frac{e^{-v} v^{K-1}}{N_1! \cdots N_R!} \prod_{r=1}^{R} \left(\sigma_r + v \sum_{k=1}^{K} \theta_{kr} u_k \right)^{N_r} dv = \Gamma(K) G^{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{N})$$



(a) $S = 8$ (b) $S = 16$

Fig. 1. Grundmann-Möller weights w_i for models with $M = K$ single-server nodes

Proof. Let $\boldsymbol{n} \in \mathbb{N}^R$, $n = \sum_r n_r$. Using [11, Thm. 2] we get

$$G^{\boldsymbol{u}}_{\theta}(\boldsymbol{N}) = \sum_{\boldsymbol{0} \leq \boldsymbol{n} \leq \boldsymbol{N}} \binom{n+K-1}{n} n! \prod_{r=1}^{R} \frac{\tilde{\theta}_r^{n_r}}{n_r!} \cdot \frac{\sigma_r^{N_r - n_r}}{(N_r - n_r)!} = \sum_{\boldsymbol{0} \leq \boldsymbol{n} \leq \boldsymbol{N}} \frac{\Gamma(n+K)}{\Gamma(K)} \prod_{r=1}^{R} \frac{\tilde{\theta}_r^{n_r}}{n_r!} \cdot \frac{\sigma_r^{N_r - n_r}}{(N_r - n_r)!}$$

We now plug the integral expression of $\Gamma(n+K)$ and the statement follows by the multinomial theorem (4) and by definition of $\tilde{\theta}_r$. □

Since $G^{\boldsymbol{u}}_{\theta}(\boldsymbol{N})$ is a normalizing constant with demands $\tilde{\theta}_r$ that are linear functions of $\boldsymbol{u}$, by definition it is a multivariate polynomial of degree N in $\boldsymbol{u}$. The theorem thus confirms that the inner integral in (20) is a polynomial of degree N. Thus a GM rule with $S = \lceil (N-1)/2 \rceil$ returns the exact value of $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ also in the presence of infinite servers. Therefore, similarly to (12) and (17), (21) provides an exact expression for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ that is both explicit and tractable.

Using smaller values of S trades accuracy for speed, as it is possible to approximate $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ by truncation of the outer summation of (21). This is an effective procedure thanks to the rapid decay of the weights w_i, as illustrated in Figure 1. For large enough i, the weights quickly and monotonically decrease, thus a few outer iterations of (21) are sufficient to return a good approximation. As we show later, GM rules perform very well on small and medium-sized inference problems. However, as the model size grows, the asymptotic expansions introduced in the next sections are normally more efficient.

4 ASYMPTOTIC EXPANSION

4.1 Preliminaries

We now exploit the geometry of the unit simplex to derive an asymptotic expansion for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$. Our approach first applies a logistic transformation to the integration variables in its integral form [1]. This is a classic method to map integrands defined over the n-dimensional simplex to $\mathbb{R}^n$. We find after this transformation that, as N grows, the integrand becomes increasingly peaked at a unique point in the interior of the integration domain, satisfying the conditions for Laplace's method [27].

A technical requirement for our argument to hold is that all queue-lengths grow asymptotically large as $N \to +\infty$, a property which is violated by non-bottleneck nodes. To address this issue, we introduce a novel scaling where we also slowly increase at every node a population of jobs that permanently reside at the node itself, perpetually self-looping. That is, we introduce K additional classes, each with population ϵN, $\epsilon > 0$, where

the ith class is composed by jobs that self-loop at node i, placing a unit service demand at each visit. In this way we are considering the perturbed normalizing constant

$$G^{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{N}) = \frac{(\eta_\epsilon(N)-1)!}{N_1!\cdots N_R!(\epsilon N!)^K}\int_{v=0}^{+\infty} e^{-v} v^{K(1+\epsilon N)-1}\int_{\Delta_K}\prod_{i=1}^{K} u_i^{\epsilon N}\prod_{r=1}^{R}\left(\sigma_r + v\sum_{k=1}^{K}\theta_{kr}u_k\right)^{N_r} d\boldsymbol{u}\, dv \tag{22}$$

where $\eta_\epsilon(N) = N + K(1+\epsilon N)$ and $\lim_{\epsilon\to 0^+} G^{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{N}) = G_{\boldsymbol{\theta}}(\boldsymbol{N})$.

Our asymptotic expansion is proved for an auxiliary function $I_\epsilon(N)$, which uniquely defines $G^{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{N})$. This function is defined as follows. First, we allow for real values of ϵ by expressing where needed factorials in $G^{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{N})$ using the gamma function $\Gamma(\cdot)$. Without loss of generality, we then normalize the service demands in the auxiliary function to range in $[0,1]$. That is, we set

$$G^{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{N}) = \frac{\Gamma(\eta_\epsilon(N))}{N_1!\cdots N_R!(\Gamma(1+\epsilon N))^K} I_\epsilon(N)\prod_{r=1}^{R}\alpha_r^{N_r} \tag{23}$$

where $\alpha_r = \sigma_r + \max_i \theta_{ir}$ and we define the auxiliary function

$$I_\epsilon(N) = \frac{1}{\Gamma(\eta_\epsilon(N))}\int_{v=0}^{+\infty} e^{-v} v^{K(1+\epsilon N)-1}\int_{\Delta_K}\prod_{i=1}^{K} u_i^{\epsilon N}\prod_{r=1}^{R}\left(\tilde{\sigma}_r + v\sum_{k=1}^{K}\tilde{\theta}_{kr}u_k\right)^{N_r} d\boldsymbol{u}\, dv \tag{24}$$

with $\tilde{\theta}_{kr} = \alpha_r^{-1}\theta_{kr}$, and $\tilde{\sigma}_r = \sum_{k=K+1}^{M}\tilde{\theta}_{kr}$. From now on, and without loss of generality, we focus on $I_\epsilon(N)$ and to simplify notation use θ_{kr} and σ_r in place of $\tilde{\theta}_{kr}$ and $\tilde{\sigma}_r$, subject to $\theta_{kr} \le 1$ and $\sigma_r \le 1$. Moreover, we assume that the ratios $\beta_r = N_r/N$ remain constant while increasing N.

4.2 Laplace's method

We first obtain the asymptotic approximation for $I_\epsilon(N)$ at $\hat{\boldsymbol{u}}$ in a model with single-server nodes only. This method requires to verify a set of well-known analytical conditions [27]. We here verify a slightly stronger set of assumptions. After showing that

$$I_\epsilon(N) = \int_{\mathbb{R}^{K-1}} e^{-N h_N(\boldsymbol{x})} d\boldsymbol{x} \tag{25}$$

for smooth and infinitely differentiable $h_N(\boldsymbol{x})$, having constant order with respect to N and bounded derivatives, we prove the validity of Laplace's method $\forall N > 0$ by showing that there exist a $\epsilon_N > 0$ such that $\forall \epsilon \ge \epsilon_N$:

- Condition 1: $I_\epsilon(N)$ exists and it is finite;
- Condition 2: $h_N(\boldsymbol{x})$ attains a unique stationary point in the interior of the integration domain of (25);
- Condition 3: the Hessian of $h_N(\boldsymbol{x})$ has a positive determinant at its stationary point.

Under these conditions it is possible to apply Laplace's method [27], which provides a $O(N^{-1})$ asymptotic approximation. Higher-order expansions may also be considered, but their computational cost grows quickly with the number of nodes in the model [27].

Theorem 4.1. *In a closed network without infinite servers, for all $N > 0$ there exists an $\epsilon_N > 0$ such that $\forall \epsilon \ge \epsilon_N$*

$$I_\epsilon(N) = \sqrt{\frac{(2\pi)^{K-1}}{\det(\boldsymbol{A})}}\prod_{r=1}^{R}\left(\sum_{k=1}^{K}\theta_{kr}\hat{u}_k\right)^{N_r}\prod_{i=1}^{K}\hat{u}_i^{1+\epsilon N} + O(N^{-1}) \tag{26}$$

where $\hat{\boldsymbol{u}}$ is the unique solution in Δ_K of the system of nonlinear equations

$$u_i = \eta_\epsilon^{-1}(N)\left(1 + \epsilon N + \sum_{r=1}^{R}\xi_r(\boldsymbol{u})\theta_{ir}u_i\right) \quad i = 1,\ldots,K-1 \tag{27}$$

with $\xi_r(\boldsymbol{u}) = N_r(\sum_{k=1}^{K}\theta_{kr}u_k)^{-1}$, *and where* $\det(\boldsymbol{A}) > 0$ *with* $\boldsymbol{A} = [A_{ij}]$ *having entries*

$$A_{ij} = \begin{cases} -\sum_{\substack{h=1 \\ h\neq i}}^{K} A_{ih} & i = j \\ \left(\sum_{r=1}^{R} \frac{\xi_r^2(\hat{\boldsymbol{u}})}{N_r}\theta_{ir}\theta_{jr} - \eta_\epsilon(N)\right)\hat{u}_i\hat{u}_j & i \neq j \end{cases} \tag{28}$$

for $i, j = 1, \ldots, K-1$.

The last result provides by (23) an approximation for $G^{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{N})$. The role of the ϵ parameter is to ensure that the stationary point of the integrand of (6) belongs to the interior of the integration domain and that $\det(\boldsymbol{A}) > 0$. For models with a finite N, the first condition holds irrespective of the value of ϵ, which is needed only asymptotically, and choosing smaller values of ϵ generally returns more accurate results. We also show that, for a sufficiently large ϵ, matrix $\boldsymbol{A}$ is positive definite, which is later used to develop a Monte Carlo integration method.

Lastly, we note that (26) is a product. This is highly beneficial in applications, since we can avoid numerical difficulties associated to the rapid growth of the normalizing constant by directly computing $\log G_{\boldsymbol{\theta}}(\boldsymbol{N})$. As a result, across thousands of models that we have solved in the numerical validation, we have never experienced numerical issues with (26). On the contrary, normalizing constant methods based on summations such as CA or (21) eventually fail on large models due to floating-point range exceptions and round-off errors.

4.3 Proof of Theorem 4.1

The result follows by proving the assumptions of Laplace's method. Since $\sigma_r = 0$, (24) here simplifies to

$$I_\epsilon(N) = \int_{\Delta_K} \prod_{i=1}^{K} u_i^{\epsilon N} \prod_{r=1}^{R}\left(\sum_{k=1}^{K}\theta_{kr}u_k\right)^{N_r} d\boldsymbol{u} \tag{29}$$

We first apply an additive logistic transformation [1]

$$u_i = \begin{cases} e^{x_i}\left(1 + \sum_{k=1}^{K-1} e^{x_k}\right)^{-1} & i = 1, \ldots, K-1 \\ \left(1 + \sum_{k=1}^{K-1} e^{x_k}\right)^{-1} & i = K \end{cases} \tag{30}$$

with Jacobian

$$\left|\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}\right| = \prod_{i=1}^{K-1} e^{x_i}\left(1 + \sum_{k=1}^{K-1} e^{x_k}\right)^{-K}$$

to obtain (25) with

$$-Nh_N(\boldsymbol{x}) = \sum_{i=1}^{K-1}(1+\epsilon N)x_i + \sum_{r=1}^{R} N_r \log\left(\theta_{Kr} + \sum_{k=1}^{K-1}\theta_{kr}e^{x_k}\right) - \eta_\epsilon(N)\log\left(1 + \sum_{k=1}^{K-1} e^{x_k}\right)$$

Note that $h_N(\boldsymbol{x})$ is smooth and infinitely differentiable. Moreover, $h_N(\boldsymbol{x})$ has a constant order with respect to N and its partial derivatives are also smooth and bounded at all orders. We are now ready to verify the conditions for Laplace's method given in Section 4.2.

4.3.1 Condition 1: existence and finiteness. Since the logistic transformation does not affect existence and finiteness, it is sufficient to verify these properties on (29). Existence follows since Δ_K is a finite domain and the integrand of (29) exists at all points of Δ_K. $I_\epsilon(N)$ is also finite for all $\epsilon > 0$ and $N > 0$, since we assumed throughout that $\theta_{kr} \leq 1$ and the domain Δ_K has a constant volume irrespective of the value of N.

4.3.2 Condition 2: unique stationary point. Condition 2 is verified as follows. We seek to solve $\nabla h_N(\boldsymbol{x}) = \mathbf{0}$ and use the inverse transformation of (30) to express the result over Δ_K. The inverse transformation is given by [1]

$$\hat{\boldsymbol{x}} = (\log \hat{u}_1/\hat{u}_K, \ldots, \log \hat{u}_{K-1}/\hat{u}_K) \tag{31}$$

and yields the system of nonlinear equations (27). We now show that this system admits a unique solution $\hat{\boldsymbol{u}}$. Moreover we also show that if $\epsilon > 0$ then $\hat{\boldsymbol{u}} \in \Delta^{\epsilon}_{K,N} \subset \Delta_K$ for all $N > 0$, with

$$\Delta^{\epsilon}_{K,N} = \left\{\boldsymbol{u} | \boldsymbol{u} \in \Delta_K, u_i \geq (1+\epsilon N)\eta_{\epsilon}^{-1}(N), \forall i\right\}$$

Mapping back $\hat{\boldsymbol{u}}$ to $\mathbb{R}^{K-1}$ using the logistic transformation (30), this implies that the stationary point of $h_N(\boldsymbol{x})$ is in the interior of the integration domain, *i.e.*, $\hat{\boldsymbol{x}}$ is finite in $\mathbb{R}^{K-1}$.

We begin by proving the existence of a solution in $\Delta^{\epsilon}_{K,N}$. Let us consider a point $\boldsymbol{u}^{(n)} \in \Delta_K$, $n \in \mathbb{N}$, and the continuous mapping

$$u_i^{(n+1)} = \eta_{\epsilon}^{-1}(N)\left(1 + \epsilon N + \sum_{r=1}^{R} \xi_r(\boldsymbol{u}^{(n)})\theta_{ir}u_i^{(n)}\right) \tag{32}$$

for $i = 1, \ldots, K-1$. Since Δ_K is convex, non-empty and compact, (32) has a fixed point $\hat{\boldsymbol{u}} \in \Delta_K$ and this must also be a solution of (27) by definition. We now show by contradiction that (27) has no solution in $\Delta_K \setminus \Delta^{\epsilon}_{K,N}$. Assume that a solution $\boldsymbol{u} \in \Delta_K \setminus \Delta^{\epsilon}_{K,N}$ exists. Then there exists a node i such that $0 \leq u_i < (1+\epsilon N)\eta_{\epsilon}^{-1}(N)$. Plugging $\boldsymbol{u}$ into (27) now makes the ith equation infeasible, since $\sum_{r=1}^{R} \xi_r(\hat{\boldsymbol{u}})\theta_{ir}u_i \geq 0$ implies $u_i \geq (1+\epsilon N)\eta_{\epsilon}^{-1}(N)$, against the assumptions.

To prove uniqueness, we focus on $\Delta^{\epsilon}_{K,N}$, since no solutions exist outside this sub-domain. Consider the nonlinear program

$$\min_{\boldsymbol{u} \in \Delta^{\epsilon}_{K,N}} \sum_{i=1}^{K-1} \frac{u_i^2}{2} - \eta_{\epsilon}^{-1}(N)\left((1+\epsilon N)\sum_{i=1}^{K-1} u_i + \sum_{r=1}^{R} N_r \log\left(\sum_{k=1}^{K} \theta_{kr}u_k\right)\right) \tag{33}$$

with first-order Karush-Kuhn-Tucker (KKT) conditions

$$\begin{aligned} -u_i + \eta_{\epsilon}^{-1}(N)\left((1+\epsilon N) + \sum_{r=1}^{R} \xi_r\theta_{ir}u_i\right) &= \lambda + \mu_i \\ \sum_{k=1}^{K} u_k &= 1 \\ u_i\mu_i &= 0 \\ \eta_{\epsilon}(N)u_i &\geq (1+\epsilon N) \\ \mu_i &\geq 0 \end{aligned}$$

for all $i = 1, \ldots, K-1$. A feasible solution in $\Delta^{\epsilon}_{K,M}$ requires $\mu_i = 0, \forall i$. It is possible to verify that the objective is strictly convex over $\Delta^{\epsilon}_{K,N}$, being the sum of functions that are convex and strictly convex over this domain. Thus the KKT conditions admit a unique solution, which must be $\hat{\boldsymbol{u}}$ since this is feasible for $\lambda = 0$. Since the solutions to the above KKT conditions include all the solutions of (27) in $\Delta^{\epsilon}_{K,N}$, we conclude that (27) has a unique solution $\hat{\boldsymbol{u}}$ in $\Delta^{\epsilon}_{K,N}$.

4.3.3 Condition 3: positive Hessian determinant. Let $\boldsymbol{A} = [A_{ij}]$, $i, j = 1, \ldots, K-1$, be the Hessian of $Nh_N(\boldsymbol{x})$ evaluated at the stationary point $\hat{\boldsymbol{x}}$. Computing $\boldsymbol{A}$ by the definition we obtain (28). We now prove that for all $N > 0$ there exists an $\epsilon_N > 0$ such that $\boldsymbol{A}$ is positive definite for all $\epsilon \geq \epsilon_N$.

From (27) we have $\lim_{\epsilon\to+\infty}\hat{u}_i = K^{-1}$,$\forall i$ which implies that $\lim_{\epsilon\to+\infty} A_{ij} < 0$, $i \neq j$. Thus, there exist an ϵ_N such that $-\boldsymbol{A}$ has positive off-diagonal entries for all $\epsilon > \epsilon_N$. By (28), this implies that

$$\boldsymbol{Q} = -\begin{bmatrix} \boldsymbol{A} & \boldsymbol{a}_K^T \\ \boldsymbol{a}_K & A_{KK} \end{bmatrix}$$

with $\boldsymbol{a}_K = (A_{1K}, \ldots, A_{K-1,K})$ is an irreducible infinitesimal generator. Being $-\boldsymbol{A}$ the principal sub-matrix of an irreducible generator, it is negative definite and thus $\boldsymbol{A}$ is positive definite, implying $\det(\boldsymbol{A}) > 0$, $\forall\epsilon > \epsilon_N$. Since $N > 0$, this verifies Condition 3.

4.3.4 Final expression. To conclude the proof of Theorem 4.1 we can apply Laplace's method in $\mathbb{R}^{K-1}$ to obtain the expansion

$$I_\epsilon(N) = \sqrt{\frac{(2\pi)^{K-1}}{\det(\boldsymbol{A})}} e^{-Nh_N(\hat{\boldsymbol{x}})} + O(N^{-1})$$

The final expression (26) follows by the expression of $h_N(\hat{\boldsymbol{x}})$ and the inverse transformation (31). By the initial definitions and using the α_r terms to remove the condition $\theta_{kr} \leq 1$, the asymptotic expansion is finally given by

$$G^\epsilon_{\boldsymbol{\theta}}(\boldsymbol{N}) = \frac{\Gamma(\eta_\epsilon(N))}{N_1!\cdots N_R!(\Gamma(1+\epsilon N))^K} \sqrt{\frac{(2\pi)^{K-1}}{\det(\boldsymbol{A})}} \prod_{r=1}^{R}\left(\sum_{k=1}^{K}\theta_{kr}\hat{u}_k\right)^{N_r} \prod_{i=1}^{K}\hat{u}_i^{1+\epsilon N} \tag{34}$$

4.4 Extensions

4.4.1 Models with infinite servers. When the population at one or more nodes does not scale asymptotically, the asymptotic stationary point of $h_N(\boldsymbol{x})$ does not lie anymore in the interior of the integration domain and this complicates the asymptotic validity of Laplace's method. In the presence of infinite servers, it does not seem easy to scale parameters in (25) to address the problem. Thus, in models with infinite servers we propose Laplace's method only as a sub-asymptotic heuristic. The sub-asymptotic method amounts to fitting the integrand of (25) to a multivariate normal density and using the resulting closed-form expressions to approximate $I_\epsilon(N)$, for $N < \infty$. This needs to be coupled with an approximation of the indefinite integral in (20).

The heuristic follows a very similar argument as in the case without infinite servers, thus we just give a sketch. We first apply the change of variable $v = e^{x_0}$ and the logistic transformation (30) so that

$$I_\epsilon(N) = \int_{\mathbb{R}^K} b_N e^{-Nh_N(\boldsymbol{x})} d\boldsymbol{x}$$

with $\boldsymbol{x} = (x_1, \ldots, x_{K-1}, x_0)$, $b_N = (\Gamma(\eta_\epsilon(N)))^{-1}$, and

$$-Nh_N(\boldsymbol{x}) = -e^{x_0} + K(1+\epsilon N)x_0 + \sum_{r=1}^{R} N_r \log\left(\sigma_r + e^{x_0}\theta_{Kr} + \sum_{k=1}^{K-1} e^{x_k}(\sigma_r + \theta_{kr}e^{x_0})\right) + (1+\epsilon N)\sum_{k=1}^{K-1} x_k - \eta_\epsilon(N)\log\left(1 + \sum_{k=1}^{K-1} e^{x_k}\right)$$

Note that this is a K-dimensional integral, whereas in the case without infinite servers we have used $K-1$ dimensions. Setting $\nabla h_N(\boldsymbol{x}) = 0$ and simplifying terms, we find that the stationary point is written in terms of

the original integration variables as the solution $(\hat{\boldsymbol{u}}, \hat{v})$ of the system

$$\begin{aligned} u_i &= \eta_\epsilon^{-1}(N)\left(1 + \epsilon N + \sum_{r=1}^{R} \xi_r(\boldsymbol{u}, v)(\sigma_r + v\theta_{ir})u_i\right) \qquad \forall i \neq K \\ v &= \eta_\epsilon(N) + 1 - \sum_{r=1}^{R} \xi_r(\boldsymbol{u}, v)\sigma_r \end{aligned} \tag{35}$$

where $\boldsymbol{u} \in \Delta_K$, $\eta_\epsilon(N) = N + K(1+\epsilon N)$, $\xi_r(\boldsymbol{u}, v) = N_r(\sigma_r + v\sum_{k=1}^{K}\theta_{kr}u_k)^{-1}$ and in which the equation for v uses that $\sum_{r=1}^{R}\xi_r(\boldsymbol{u}, v)(\sigma_r + v\sum_{k=1}^{K}\theta_{kr}u_k) = N$.

Explicit formulas for the entries of $\boldsymbol{A}$ are obtained by computing the Hessian matrix of $Nh_N(\boldsymbol{x})$ expressed in terms of the variables $(\hat{\boldsymbol{u}}, \hat{v})$. Define $\hat{\theta}_{kr} = \sigma_r + \hat{v}\theta_{ir}$, $\forall k, r$, the entries of $\boldsymbol{A} = [A_{ij}]$ are given by

$$\begin{aligned} A_{ii} &= -\sum_{j\neq i} A_{ij} \\ A_{ij} &= \sum_{r=1}^{R}\left(\frac{\xi_r^2(\hat{\boldsymbol{u}}, \hat{v})}{N_r}\hat{\theta}_{kr}\hat{\theta}_{jr} - \eta_\epsilon(N)\right)\hat{u}_i\hat{u}_j \qquad i \neq j \\ A_{00} &= \hat{v}\left(1 - \sum_{r=1}^{R}\frac{\xi_r^2(\hat{\boldsymbol{u}}, \hat{v})}{N_r}\sigma_r\left(\sum_{k=1}^{K}\theta_{kr}\hat{u}_k\right)\right) \\ A_{i0} &= A_{0i} = \hat{v}\hat{u}_i\sum_{r=1}^{R}\left(\frac{\xi_r^2(\hat{\boldsymbol{u}}, \hat{v})}{N_r}\hat{\theta}_{ir}\left(\sum_{k=1}^{K}\theta_{kr}\hat{u}_k\right) - \xi_r(\hat{\boldsymbol{u}}, \hat{v})\theta_{ir}\right) \end{aligned}$$

for all $i, j = 1, \ldots, K-1$. The above expressions use the inverse transformations (31) and $x_0 = \log v$. The knowledge of $(\hat{\boldsymbol{u}}, \hat{v})$ and $\boldsymbol{A}$ provides a Laplace-type approximation for (20)

$$I_\epsilon(N) \approx \frac{e^{-\hat{v}}\hat{v}^{K(1+\epsilon N)}}{\Gamma(\eta_\epsilon(N))}\sqrt{\frac{(2\pi)^K}{\det(\boldsymbol{A})}}\prod_{r=1}^{R}\left(\sigma_r + \hat{v}\sum_{k=1}^{K}\theta_{kr}\hat{u}_k\right)^{N_r}\prod_{k=1}^{K}\hat{u}_k^{1+\epsilon N} \tag{36}$$

where the exponent of $\hat{v}$ includes the contribution of the Jacobian. Equation (36) can be readily applied to approximating models with infinite server nodes, leading to

$$G_{\boldsymbol{\theta}}^{\epsilon}(\boldsymbol{N}) = \frac{e^{-\hat{v}}\hat{v}^{K(1+\epsilon N)}}{N_1!\cdots N_R!(\Gamma(1+\epsilon N))^K}\sqrt{\frac{(2\pi)^K}{\det(\boldsymbol{A})}}\prod_{r=1}^{R}\left(\sigma_r + \hat{v}\sum_{k=1}^{K}\theta_{kr}\hat{u}_k\right)^{N_r}\prod_{k=1}^{K}\hat{u}_k^{1+\epsilon N} \tag{37}$$

As before, the $\epsilon > 0$ parameter should be chosen as small as possible, but such that $\det(\boldsymbol{A}) > 0$.

4.4.2 Combining Laplace's method with AMVA. In some inference problems, measurements for both system state and mean performance metrics are available. In this case, in addition to likelihood-based inference, one may require that the estimated model also matches the empirical mean value of some performance metrics. This typically requires to run an AMVA algorithm alongside the likelihood maximization algorithm.

In this section we argue that a more efficient way to optimize these models is to heuristically compute $\hat{\boldsymbol{u}}$ using the results of AMVA. This effectively doubles the speed of the optimization, since the same AMVA fixed-point iteration can be used both to calculate likelihood and mean measures. A limitation of this method is that no formal guarantee is in place to ensure that $\det(\boldsymbol{A}) > 0$ on all instances. However, no problematic instance in this sense is observed throughout the numerical validation. In cases where $\det(\boldsymbol{A}) \leq 0$, one may try to heuristically increase ϵ in order to resolve this issue.

The proposed method works as follows. Let us observe that, as $N \to \infty$, (27) tends to

$$u_i = \frac{\epsilon}{1 + K\epsilon} + \sum_{r=1}^{R} \hat{\xi}_r(\hat{\boldsymbol{u}}) \theta_{ir} u_i \quad i = 1, \ldots, K \tag{38}$$

where $\hat{\xi}_r(\boldsymbol{u}) = \beta_r (1 + K\epsilon)^{-1} (\sum_{k=1}^{K} \theta_{kr} u_k)^{-1}$. As $\epsilon \to 0$ the last expression coincides with the expression of the mean-value analysis algorithm's queue-length equations when u_i is the total queue-length at node i divided by N, and $N \to \infty$. This suggests the following way to determine the point $\hat{\boldsymbol{u}}$ at which we instantiate the Laplace's method. Instead of using (27), we choose $\hat{\boldsymbol{u}} \approx \hat{\boldsymbol{q}} = (\hat{q}_1, \ldots, \hat{q}_M)$, in which $\hat{q}_i = \sum_r q_{ir}(\boldsymbol{N})/N$, $\forall i$, where $q_{ir}(\boldsymbol{N})$ is the mean queue-length of class r at node i in a model with population $\boldsymbol{N}$, a value which can be accurately approximated in $O(1)$ time and space using AMVA [44].

4.4.3 Monte Carlo integration. The applicability of Laplace's method indicates that the integral $I_\epsilon(N)$ may be approximated using a multivariate normal distribution centered at the stationary point of $h_N(\boldsymbol{x})$ and with covariance matrix $\boldsymbol{A}^{-1}$. The resulting normal distribution is non-degenerate if $\boldsymbol{A}$ is positive definite. In situations where asymptotic expansions are expensive or inaccurate, one may thus apply an importance sampling method to $I_\epsilon(N)$, using samples from a multivariate normal distribution. Denote by $\boldsymbol{x}_j$ the jth sample drawn, out of a total of J. We have the importance sampling estimator

$$I_\epsilon(N) \approx J^{-1} \sum_{j=1}^{J} \frac{b_N e^{-N h_N(\boldsymbol{x}_j)}}{\phi(\boldsymbol{x}_j)} \qquad \boldsymbol{x}_j \sim \mathcal{N}(\hat{\boldsymbol{x}}, \boldsymbol{A}^{-1}) \tag{39}$$

where $\hat{\boldsymbol{x}}$ is the stationary point of $h_N(\boldsymbol{x})$ and $\phi(\cdot)$ stands for the normal density function. For the case without infinite servers, $b_N = 1$ and $h_N(\boldsymbol{x})$ is defined as in Section 4.3. For models with infinite servers, one needs to use the expressions of b_N and $h_N(\boldsymbol{x})$ given in Section 4.4.1. From Monte Carlo integration theory, (39) converges to $I_\epsilon(N)$ as $O(J^{-1/2})$ under a growing sample size J.

5 NUMERICAL RESULTS

5.1 Algorithms

In this section we assess accuracy and speed of the proposed methods. We distinguish the proposed algorithms in two groups:

- *Deterministic methods*, such as the asymptotic expansions, which return the same answer in successive invocations on the same model and therefore are suitable for use within deterministic optimization programs. We include in this group RAY and the asymptotic expansion (25), referred to as the *logistic expansion* (LE), and the heuristic variant of LE calibrated with AMVA, denoted by LE-A. We also consider in this group the cubature rules given in (21) with $S = \{1, 3, 5, 7\}$ and denote, *e.g.*, by CUB5 a cubature rule with $S = 5$. We have also experimented with TE, CA, and MoM, but computational times are far larger than those of the other methods and incompatible with the scale of the experimental validation, which encompasses thousands of optimization programs.
- *Randomized methods*, which use sampling to achieve the desired accuracy in return for an increased effort. We include in this group MCI and the Monte Carlo integration method in (39), which we call *logistic sampling* (LS). Monte Carlo methods are instantiated with $J \in \{10^1, 10^2, 10^3\}$ samples, *e.g.*, MCI2 stands for MCI with $J = 10^2$, and similarly LS3 has $J = 10^3$.

For deterministic methods, we are interested in assessing both accuracy and ability to guide optimization-based search. For randomized methods, we verify accuracy as the number of samples grows. Remarks on the implementations are as follows:

- In LE we use fixed-point iteration to solve (32), setting the convergence tolerance on the 1-norm of $\hat{\boldsymbol{u}}$ to $\tau = 10^{-10}$. The initial point has $u_i = 1/K, \forall i$. We also set $\epsilon = 10^{-10}$. Out of the thousands of models solved, none failed to converge and none required to increase ϵ beyond its initial value.
- LE-A is implemented using the Bard-Schweitzer AMVA [44] for determining the point $\hat{\boldsymbol{q}}$ with a $\delta = 10^{-6}$ convergence tolerance on the 1-norm of the mean queue-lengths. Also in this case none of the models failed to converge.
- For cases where $\boldsymbol{A}$ is not positive definite, LS is instantiated by increasing ϵ in steps of $10^{-3}N$, until obtaining a positive definite matrix. In small and medium-sized models, this calibration is not normally required. However, on large models where the entries of $\boldsymbol{A}$ are small, very few increments of ϵ are normally sufficient to address the issue. In the random validation on large models, this calibration is required on 44% of the instances and occurs prior to computing (39). The computational cost of the calibration is negligible.

5.2 Methodology

An important issue for the validation methodology is that for large models $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ cannot be computed exactly due to the large cost of the exact algorithms. Thus we first carry out a validation on small and medium-sized models where the normalizing constant can be obtained exactly. Afterwards, we report a similar validation on larger models where we estimate $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ by Monte Carlo integration with a large number of samples (10^7). In the large-scale setting, we attempt to compensate the variance of the estimator of the normalizing constant by assessing percentage error with respect to the scale, *i.e.*, $\log G_{\boldsymbol{\theta}}(\boldsymbol{N})$. Note that on most large-scale models the order of the normalizing constant is the dominant factor in the likelihood expressions.

The validation does not include mean performance metrics. This is because their computation can be performed very efficiently using AMVA methods [44]. Our methods are instead proposed for the accurate computation of likelihoods and probabilities, which require $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ and are still difficult to compute in practice.

The computational times to run LE and LE-A inside optimization programs are very small, typically a fraction of a second. This is due to the rapid convergence of the fixed-point algorithms used to determine the location of the stationary point. For example, on the largest model with $K = R = 64$ and $N = 4096$ LE takes $26s$ to find at the first iteration the stationary point, but just $2s$ for a new prediction after a 10^{-6} increment of $\boldsymbol{\theta}$, provided that the fixed point equations (32) are re-initialized at the previously-found stationary point. When the model size is decreased to $K = R = 8$ the first solution requires just $0.21s$, while successive updates about $0.015s$. Since the optimization-based study considers an identical timeout for all methods, we do not provide details on the running times of individual algorithms. Lastly, we remark that space complexity is negligible and does not grow significantly with the model size. This is because all approximation methods considered throughout have $O(1)$ space complexity as the population sizes N grow, with N typically being the largest model parameter.

5.3 Computing a single normalizing constant

5.3.1 Small and medium-sized models. We consider randomly-generated models with $K \in \{2, 4, 6\}$ nodes, $R \in \{2, 4, 6\}$ classes, and where each class has the same number of jobs equal to $N/R = \{2, 4, 8\}$. Thus the largest model in this group has 6 nodes, 6 classes and 48 jobs. We use less jobs than in the motivating example in Table 1 since we now consider models with $R = 6$ classes that are much more expensive to solve exactly. For any given triplet (K, R, N), we solve 100 random instances, for a total of 2700 models. In each instance, demands are generated at random in $[0, 1]$. $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ is computed exactly using the CA algorithm.

Note that RAY is the only method that incurs failures during execution. This occurs on 9 models out of 2700 and it is due to a singular determinant in its expression. Indeed, RAY does not provide correctness guarantees in the sub-asymptotic setting [29]. We count as a failure a run that either stops due to excessive memory requirements, or that returns a 0, NaN, $\pm\infty$, or a complex value for $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ due to numerical issues.

Table 2. Small and medium models - deterministic methods

$K, R \in \{2, 4, 6\}$	MAPE (%)		
$N/R =$	*2*	*4*	*8*
CUB1	0.0	0.4	8.9
CUB3	0.0	0.4	7.4
CUB5	0.0	0.0	0.7
CUB7	0.0	0.0	0.1
LE	25.7	25.2	24.1
LE-A	26.7	23.3	18.3
RAY	362.0	142.7	96.5

Table 3. Small and medium models - randomized methods

$K, R \in \{2, 4, 6\}$	MAPE (%)		
$N/R =$	*2*	*4*	*8*
LS1	23.1	21.1	31.6
LS2	12.1	12.9	12.1
LS3	6.0	7.0	7.4
MCI1	28.9	41.7	55.5
MCI2	9.3	13.0	18.1
MCI3	3.0	4.1	5.3

Tables 2 and 3 give the *mean absolute percentage error* (MAPE) on $G_{\boldsymbol{\theta}}(\boldsymbol{N})$ for deterministic and randomized methods. The results indicate that the CUB dominates all other methods, including the randomized ones. Execution times of CUB on these models are in the order of a few milliseconds, making this method preferable in small and medium-sized models. As expected, asymptotic expansions incur smaller errors as the number of jobs grows. The Monte Carlo integration methods, MCI and LS, are instead more accurate with fewer jobs. However, increasing the number of samples in both methods quickly lowers errors to the desired level.

In order to better understand the differences between MCI and LS, we have investigated how the accuracy of the two methods varies under increasing number of nodes K or number of classes R. MCI decreases errors as K increases, whereas it performs worse under increasing R. For example, MCI1 goes from 35.4% to 48.2% as the number of classes goes from 2 to 6, whereas it decreases errors from 57.4% to 30.1% when the number of nodes grows of the same amount. Conversely, LS is rather insensitive to R, with LS1 error lying between 22.5% and 26.4%, but the method incurs larger errors as K grows, with LS1 going from 12.5% with 2 nodes to 38.9% with 6 nodes. This increased error is due to the larger number of integration dimensions in (25), which requires a larger N value to deliver a similar level of accuracy. Similar trends are seen also in large models and suggest that the two methods can complement each other, preferring MCI on models with many nodes and LS on models with several classes.

5.3.2 Large-scale models. We now consider a similar setup as in the previous experiment, but with K, R ranging in $\{16, 32, 64\}$. The number of jobs ranges in $N/R = \{2, 4, 8, 16, 32, 64\}$. Thus the largest models have 64 nodes, 64 classes and 4096 jobs. As explained before, in this setting it is difficult to obtain the exact value of the

Table 4. Small and medium models with infinite server nodes - deterministic methods

$K, R \in \{2, 4, 6\}$	MAPE (%)			
σ_r	$0.1\theta_{max}$	θ_{max}	$10\theta_{max}$	$100\theta_{max}$
PAN	N/A	N/A	26.2	0.1
LE	7.1	4.8	2.9	1.5

Table 5. Large models - deterministic methods

$K, R \in \{16, 32, 64\}$	MAPE (%)					
$N/R =$	*2*	*4*	*8*	*16*	*32*	*64*
CUB1	0.0	0.1	0.4	1.1	2.5	2.9
LE	1.8	1.0	0.7	0.4	0.4	0.5
LE-A	1.8	1.0	0.7	0.4	0.4	0.4
RAY	14.5	4.8	1.4	0.5	0.3	0.4

Table 6. Large models - randomized methods

$K, R \in \{16, 32, 64\}$	MAPE (%)					
$N/R =$	*2*	*4*	*8*	*16*	*32*	*64*
LS1	0.9	0.6	1.4	2.4	2.3	1.8
LS2	0.5	0.5	1.3	2.2	2.2	1.8
LS3	0.3	0.4	1.2	2.1	2.1	1.7
MCI1	0.1	0.1	0.2	0.3	1.5	6.1
MCI2	0.0	0.0	0.1	0.1	0.4	3.3
MCI3	0.0	0.0	0.0	0.0	0.2	1.6

normalizing constant, thus we compare against MCI with 10^7 samples and focus on matching $\log G_{\boldsymbol{\theta}}(\boldsymbol{N})$. The MAPE on $\log G_{\boldsymbol{\theta}}(\boldsymbol{N})$ may be seen as the percentage error introduced in log-likelihoods such as (5).

Tables 5 and 6 present the results of these experiments. Since CUB3, CUB5 and CUB7 fail on over 98% of the large instances due to floating-point range exceptions, the corresponding entries are omitted from the table. The results indicate that CUB1 remains the best method under small population sizes, however as N grows the asymptotic expansions become the most accurate. The RAY method is the least accurate in light load, but has a similar accuracy to the other methods in high-load. This is indeed the regime that matches the assumptions for the scaling used in RAY [29]; we have noted however that on models with a large number of classes, but a few queues, RAY is less accurate than LE and LE-A, which is consistent with the fact that the scaling used in [29] assumes a growing number of nodes. For example, going from $K = 16$ to $K = 64$ nodes RAY improves its error from an average of 6.82% to 1.28%. On the opposite, when the number of classes is increased from $R = 16$ to $R = 64$, RAY goes from an average error of 0.99% to 7.22%. In the same ranges for nodes and classes, LE and LE-A have narrow error bands between 0.46% and 1.23% average error.

5.3.3 Models with infinite server nodes. We have repeated the experiments in Section 5.3.1 on models with a infinite server node, focusing on the validation of the heuristic given in Section 4.4.1. Since with infinite servers RAY is no longer applicable, we have validated LE against the PANACEA (PAN) asymptotic expansion [37]. We have set in the experiments an identical think time on all classes equal to $\sigma_r \in \{0.1\theta_{max}, \theta_{max}, 10\theta_{max}, 100\theta_{max}\}$, where $\theta_{max} = \max_r \max_{k=1,\ldots,K} \theta_{kr}$. Results are shown in Table 4. PAN fails on all models with think time $\sigma_r = 0.1\theta_{max}$ and $\sigma_r = \theta_{max}$ due to violation of the normal usage assumption; it instead returns a 26.2% MAPE with $\sigma_r = \theta_{max}$, and an error less than 0.1% with $\sigma_r = 10\theta_{max}$. LE returns a valid solution in all cases, with decreasing errors for increasing σ_r values. Thus, LE appears generally more robust than PAN, which is preferable only in very lightly loaded models.

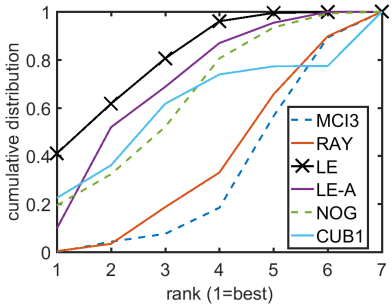
5.4 Optimization programs

We now compare the methods against the likelihood maximization problem (5) for service demand estimation, focusing on deterministic methods. For the sake of illustration of the limited performance of randomized methods in this setting, we also include MCI3 in the validation. We consider problems with $K, R \in \{2, 4, 8, 16, 32\}$ and populations with $N/R \in \{2, 20, 40\}$ jobs per class. Each experiment is carried out with the same procedure described in Section 2.3, in particular setting a timeout of $T = 10$ minutes. We repeat the same experiment 15 times randomizing demands, solving 1125 optimization programs for each method. Upon detecting an invalid normalizing constant the interior point method returns a failure. We also mark as failed all the runs returning demands that do not satisfy the constraint $\boldsymbol{\theta} \geq \mathbf{0}$.

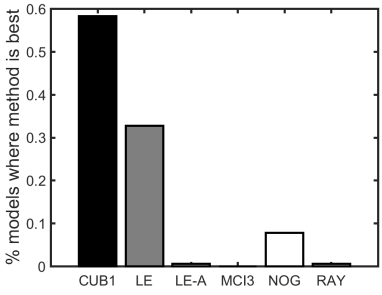
5.4.1 Metric. As mentioned in Section 2.3, a critical issue in the analysis of the results is that (5) is non-convex, thus the choice of the initial points affects the relative error on the $\boldsymbol{\theta}$ estimate, irrespective of the quality of the approximation of $G_{\boldsymbol{\theta}}(\boldsymbol{N})$. This is addressed in Section 2.3 by comparing results against an exact method, run without timeout. Here we cannot apply the same approach due to the size of the models and also MCI with a large number of samples ($J = 10^7$) fails since the variance of the estimator adversely affects the search direction of the interior-point method. Thus, on most models it does not seem possible to determine and compare the methods based on an absolute accuracy metric.

To cope with this problem, we use the same initial point for all the methods and compare them relatively to each other. For each model, we rank methods based on the absolute percentage error from the true value of $\boldsymbol{\theta}$. In this way, a method that achieves the best possible estimator given the initial point, will be ranked first, irrespective of the magnitude of the error that depends on the initial point and the local optimum found. Methods that return the same demands are assigned the same rank.

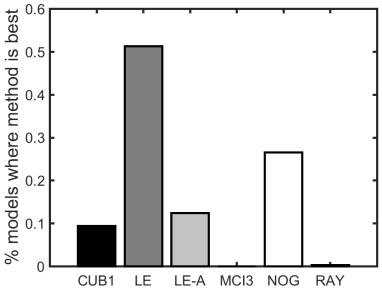
5.4.2 Results. Experimental results are given in Figure 2. We include in the study also the NOG method, which neglects the normalizing constant. In Figures 2(b)-(d) we show how frequently each model is ranked best for a given population level N. The results indicate that LE outperforms all the other methods and it is slightly better than LE-A. However, in models with a small number of jobs CUB is preferable, which is consistent with the observations in Section 5.3.1. The fair performance of NOG is explained by the fact that asymptotically the closed network approaches an open network, where the arrival rate intensity matches the cumulative departure rate from the bottleneck nodes. The methods proposed in this paper remain preferable to NOG, as they are the best ones in most models. This is evident in Figure 2(a), which indicates that LE is the best method among the considered ones. The figure shows that about 41% of the times LE is the best method, and in about 62% of the cases it ranks second, typically behind CUB1, NOG, or LE-A. Table 7 reports statistics on the number of failures, which occur only for CUB as the load grows and for RAY, similarly to what seen for small and medium-sized models. Methods not shown in the table do not incur failures.



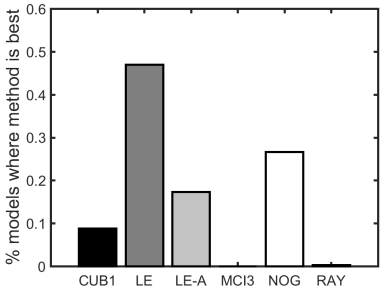
(a) $N/R \in \{2, 20, 40\}$



(b) $N/R = 2$



(c) $N/R = 20$



(d) $N/R = 40$

Fig. 2. Optimization programs: experimental results

Table 7. Percentage of failures (1125 models).

Method	Total	$N/R = 2$	$N/R = 20$	$N/R = 40$
CUB1	22.4	0.0	26.9	40.3
RAY	8.8	2.7	16.0	7.7

5.4.3 Single-class models. Lastly, for completeness we include results concerning inference in single-class models, which also arise in practice. For such models we do not study the computation of a single normalizing constant since exact $O(1)$ expressions are available, *e.g.* (14). We instead consider likelihood maximization with $K \in \{2, 4, 8, 16, 32\}$ nodes, $R = 1$ class, $N = \{2, 20, 40\}$, and single-class demands $\theta_k = k/K$. The initial guess for the demand matrix is $\theta_k = 1/K$, $\forall k$. We include in the study the exact-order asymptotic (EOA) formula recently proposed in [19].

Generally speaking, exact methods such as CA are very fast on single-class models, and return a tiny error, on average just 0.058%. However, CA complexity is $O(N)$, thus in models with very large populations N approximations may be of interest. In the above study, LE returns a MAPE of just 0.28%. The other methods are instead rather inaccurate, with a MAPE of 28.1% for CUB1, 26.0% for RAY, 24.8% for LE-A, and 66.1% for EOA.

This overall indicates that LE is fit for use also in single-class problems, although exact methods such as CA seem sufficient in practice.

5.5 Summary

Summarizing, the results indicate the following main properties for the proposed algorithms:

- On small models, CUB dominates all other algorithms.
- On large models, one should choose CUB if the population is small, or otherwise prefer LE. The same criteria applies to optimization programs involving likelihoods.
- Among randomized methods, LS is the best on models with many classes, whereas MCI is best with several nodes.

6 CONCLUSION

This paper has shown that performance inference over closed systems faces computational hurdles. If the model is a closed product-form multiclass network, we have shown that computational issues can be addressed by novel asymptotic expansions and Monte Carlo sampling methods for the normalizing constant of state probabilities. Future research may further investigate the implications of the integral form (6) for the exact theory of normalizing constants, for example on models with multi-server and load-dependent nodes.

ACKNOWLEDGEMENT

This research has been partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644869 (DICE) and by a UK Engineering and Physical Sciences Research Council grant (EP/L00738X/1). Research data is available at (https://doi.org/10.5281/zenodo.546873) under CC-BY 4.0 licence. The author wishes to thank Tony Field for support while preparing this work and Urtzi Ayesta for his helpful comments while serving as managing editor for this paper.

REFERENCES

[1] J. Aitchison. The statistical analysis of compositional data. *J. Royal Stat. Society. Series B.*, 139–177, 1982.
[2] J. Anselmi, P. Cremonesi. A unified framework for the bottleneck analysis of multiclass queueing networks. *Perform. Eval.*, 67(4):218-234, 2010.
[3] K. E. Atkinson. *An Introduction to Numerical Analysis.* John Wiley & Sons, 2nd ed., 1989.
[4] G. Balbo, G. Serazzi. Asymptotic analysis of multiclass closed queueing networks: Multiple bottlenecks. *Perform. Eval.*, 30(3):115–152, 1997.
[5] V. Baldoni, N. Berline, J. A. De Loera, M Köppe, M. Vergne. How to integrate a polynomial over a simplex. *Math. of Computation*, 80(273):297–325, 2011.
[6] F. Baskett, K. M. Chandy, R. R. Muntz, F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *JACM*, 22:248–260, 1975.
[7] A. W. Berger, L. M. Bregman, and Y. Kogan. Bottleneck analysis in multiclass closed queueing networks and its application. *QUESTA*, 31(3-4):217–237, 1999.
[8] A. Bertozzi, J. McKenna. Multidimensional residues, generating functions, and their application to queueing networks. *SIAM Review*, 35(2):239–268, 1993.
[9] J. P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Comm. of the ACM*, 16(9):527–531, 1973.
[10] G. Casale. An efficient algorithm for the exact analysis of multiclass queueing networks with large population sizes. *Proc. of ACM SIGMETRICS*, pp. 169–180, 2006.
[11] G. Casale. Exact analysis of performance models by the method of moments. *Perf. Eval.*, 68(6):487–506, 2011.
[12] K. M. Chandy, U. Herzog, L. Woo. Parametric Analysis of Queueing Networks. *IBM J. Res. Dev.*, 19(1):36–42, 1975.
[13] G L. Choudhury, K. K. Leung, and W. Whitt. Calculating normalization constants of closed queuing networks by numerically inverting their generating functions. *JACM*, 42(5):935–970, 1995.
[14] R. Cools. An encyclopaedia of cubature formulas. *Journal of Complexity*, 19:445–453, 2003.

[15] A. E. Conway, N. D. Georganas. RECAL - A new efficient algorithm for the exact analysis of multiple-chain closed queueing networks. *JACM*, 33(4):768–791, 1986.
[16] C. de Boor. Divided differences. *Surveys in Approximation Theory*, 1:46–49, 2005.
[17] I. Perezy, D. Hodge, and T. Kypraios. Auxiliary Variables for Bayesian Inference in Multi-Class Queueing Networks. arXiv:1703.03475, 9 March 2017.
[18] C. Fletcher. Innovation Insight for Algorithmic IT Operations Platforms. Gartner report G00296380, 24 March 2016.
[19] D. K. George, C. H. Xia, and M. S. Squillante. Exact-order asymptotic analysis for closed queueing networks. *J. Applied Probability*, 49(2):503–520, 2012.
[20] A. I. Gerasimov. On normalizing constants in multiclass queueing networks. *Oper. Res.*, 43(4):704–711, 1995.
[21] H. W. Gould. Some Generalizations of Vandermonde's Convolution. *The American Mathematical Monthly*, 63(2):84–91, 1956.
[22] J. J. Gordon. The evaluation of normalizing constants in closed queueing networks. *Oper. Res.*, 38(5):863–869, 1990.
[23] A. Grundmann, H.M. Möller. Invariant integration formulas for the n-simplex by combinatorial methods. *SIAM Journal on Numerical Analysis*, 15(2):282–290, 1978.
[24] P. G. Harrison. On normalizing constants in queueing networks. *Oper. Res.*, 33(2):464–468, 1985.
[25] P. G. Harrison, T. T. Lee. A new recursive algorithm for computing generating functions in closed queueing networks. In *Proc. of IEEE MASCOTS*, 223–230. IEEE Press, 2004.
[26] R. Kan. From moments of sum to moments of product. *J. of Multivariate Analysis*, 99(3):542–554, March 2008.
[27] R. E. Kass, L. Tierney, and J. B. Kadane. The validity of posterior expansions based on Laplace's method. *Bayesian and likelihood methods in stat. and econ.*, 7:473, 1990.
[28] F. P. Kelly, L. Massoulié, and N. S. Walton. Resource pooling in congested networks: proportional fairness and product form. *QUESTA*, 63(1-4):165–194, 2009.
[29] C. Knessl, C. Tier. Asymptotic expansions for large closed queueing networks with multiple job classes. *IEEE Trans. Computers*, 41(4):480–488, 1992.
[30] C. Knessl, C. Tier. Asymptotic approximations and bottleneck analysis in product form queueing networks with large populations. *Perf. Eval.*, 33(4):219–248, 1998.
[31] H. Kobayashi. A computational algorithm for queue distributions via the Pólya theory of enumeration. *Perf. of Computer Systems*, North-Holland, 1979, pp. 79-88.
[32] E. Koenigsberg. Cyclic queues. *Operational Research Quarterly*, 9, 1:22–35, 1958.
[33] Y. Kogan. Asymptotic expansions for probability distributions in large loss and closed queueing networks. *Perform. Eval. Rev.*, 29(3):25–27, Dec. 2001.
[34] Y. Kogan, M. Shenfild. Asymptotic solution of generalized multiclass Engset model. In *Proc. of ITC*, 1239–1249, 1994.
[35] Y. Kogan, A. Yakovlev. Asymptotic analysis for closed multichain queueing networks with bottlenecks. *QUESTA*, 23:235–258, 1996.
[36] J. B. Lasserre, E. S. Zeron. A Laplace transform algorithm for the volume of a convex polytope. *JACM*, 48(6), 2001.
[37] J. McKenna, D. Mitra. Asymptotic expansions and integral representations of moments of queue lengths in closed Markovian networks. *JACM*, 31(2):346–360, 1984.
[38] L. M. Milne-Thomson. *The calculus of finite differences.* Mac Millan, London, 1933.
[39] A. Asanjarani, Y. Nazarathy, P. K. Pollett. *Parameter and State Estimation in Queues and Related Stochastic Models.* arXiv:1701.08338, 29 Jan 2017.
[40] Y. Pawitan. *In All Likelihood - Statistical Modeling and Inference Using Likelihood.* Oxford Science, 2001.
[41] M. Reiser, H. Kobayashi. Queueing networks with multiple closed chains. *IBM J. Res. Dev.*, 19(3):283–294, 1975.
[42] M. Reiser, S. S. Lavenberg. Mean-value analysis of closed multichain queueing networks. *JACM*, 27(2):312–322, 1980.
[43] K.W. Ross, D.H.K. Tsang and J. Wang. Monte carlo summation and integration applied to multiclass queueing networks. *JACM*, 41(6):1110–1135, 1994.
[44] P. Schweitzer. Approximate Analysis of Multiclass Closed Networks of Queues. In *Proc. of the Int'l Conf. on Stoch. Control and Optim.*, 25–29, 1979.
[45] S. Spinner, G. Casale, F. Brosig, S. Kounev. Evaluating approaches to resource demand estimation. *Perf. Eval.*, 92:51–71, 2015.
[46] C. Sutton, M. Jordan. Bayesian inference for queueing networks and modeling of internet services. *Annals of Applied Stat.*, 5(1), 254–282, 2011.
[47] W. Wang, G. Casale, C. Sutton. A Bayesian Approach to Parameter Inference in Queueing Networks. *ACM TOMACS*, 27(1), 2016.

A DIVIDED DIFFERENCES

We show that $g_{\boldsymbol{\theta}}(N) = [\theta_1, \dots, \theta_K] x^{N+K-1}$ holds for arbitrary demands $\theta_1 \le \theta_2 \le \cdots \le \theta_K$. In the case of distinct demands, the result follows by [38, Sec. 1.31], [23]. Otherwise note that divided differences may be recursively

defined as follows [16]

$$[\theta_1,\ldots,\theta_K]f(x) = \begin{cases} \dfrac{f^{(K-1)}(\theta_1)}{(K-1)!} & \text{if } \theta_1 = \theta_K \\ \dfrac{[\theta_2,\ldots,\theta_K]f(x) - [\theta_1,\ldots,\theta_{K-1}]f(x)}{\theta_K - \theta_1} & \text{otherwise} \end{cases} \tag{40}$$

where $f^{(n)}(x)$ denotes the nth derivative of $f(x)$. Let us then note the following relation [24]

$$\begin{aligned} g_{\boldsymbol{\theta}}(N) &= \frac{\theta_K g_{\boldsymbol{\theta}}^{-1}(N) - \theta_1 g_{\boldsymbol{\theta}}^{-K}(N)}{\theta_K - \theta_1} \\ &= \frac{g_{\boldsymbol{\theta}}^{-1}(N+1) - g_{\boldsymbol{\theta}}^{-K}(N+1)}{\theta_K - \theta_1} \end{aligned} \tag{41}$$

where $g_{\boldsymbol{\theta}}^{-i}$ refers to a network without node i and we have used that by convolution $g_{\boldsymbol{\theta}}^{-1}(N+1) = g_{\boldsymbol{\theta}}^{-1-K}(N+1) + \theta_K g_{\boldsymbol{\theta}}^{-1}(N)$ and $g_{\boldsymbol{\theta}}^{-K}(N+1) = g_{\boldsymbol{\theta}}^{-1-K}(N+1) + \theta_1 g_{\boldsymbol{\theta}}^{-K}(N)$, see *e.g.* [9].

We can now show that (40) and (41) are equivalent if $f(x) = x^{N+K-1}$. In the case $\theta_1 \neq \theta_K$, (40) readily matches (41). Since $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_K$, upon reaching $\theta_1 = \theta_K$ in (41) we have a model with balanced demands, in which case we can use the termination condition [22, Eq. 19]: $g_{\boldsymbol{\theta}}(N) = \binom{N+K-1}{K-1}\theta_1^N$. The last expression matches $\frac{f^{(K-1)}(\theta_1)}{(K-1)!}$ when $f(x) = x^{N+K-1}$, as in (40).

B HERMITE-GENOCCHI

We show that if $f(x) = x^{N+K-1}$, then the Hermite-Genocchi theorem (10) holds under nondistinct demands, *i.e.*

$$[\theta_1,\ldots,\theta_K]x^{N+K-1} = \frac{(N+K-1)!}{N!}\int_{\Delta_K}(\theta_1 u_1 + \ldots + \theta_K u_K)^N d\boldsymbol{u}$$

for arbitrary $\theta_1,\ldots,\theta_K$. Using the multinomial theorem (4)

$$[\theta_1,\ldots,\theta_K]x^{N+K-1} = (N+K-1)!\int_{\Delta_K}\sum_{\substack{\boldsymbol{n}\geq 0:\\ n=N}}\prod_{i=1}^{K}\frac{\theta_i^{n_i}u_i^{n_i}}{n_i!}d\boldsymbol{u}$$

Note that the expression can now be simplified by (11), which provides the same identity proved in Appendix A

$$[\theta_1,\ldots,\theta_K]x^{N+K-1} = \sum_{\substack{\boldsymbol{n}\geq 0:\\ n=N}}\prod_{i=1}^{K}\theta_i^{n_i} = g_{\boldsymbol{\theta}}(N)$$

where we have noted that the summation in the last expression is the specialization of (1) to single-class models.

Received October 2016; revised January 2016; accepted December 2016