Online Supplement to: A Survey and Experimental Comparison of Service-Level-Approximation Methods for Non-Stationary $M(t)/M/s(t)$ Queueing Systems with Exhaustive Discipline

INFORMS Journal on Computing

Armann Ingolfsson, Elvira Akhmetshina, Susan Budge, Yongyue Li
School of Business, University of Alberta, Edmonton, Alberta T6G 2R6, Canada,
{armann.ingolfsson@ualberta.ca, elvira@ualberta.ca, sbudge@ualberta.ca, yongyue@ualberta.ca}

Xudong Wu
Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2R6, Canada,
xudong@cs.ualberta.ca

Abstract: This online supplement contains additional information regarding the different computational methods and additional computational results.

A. Computational Methods: Additional Information

When choosing a performance-evaluation method (exact or approximate) for the applications we mention in the paper, a practitioner will likely attempt to answer some of the following questions:

1. Are the assumptions of the method appropriate for the system?
2. Is software that implements the method available? If not, how difficult is it to implement the method?
3. If the method is approximate, how accurate is it?
4. How much computation time does the method require?

Our goal is to help practitioners answer these questions. While the paper focuses on $M(t)/M/s(t)$ systems, we summarize here the extent to which each method can model more general systems. We implemented all of the methods on a common platform (Matlab). We also comment on implementation issues and summarize evidence regarding accuracy and speed from other researchers.

A.1. Exact Method

The implementation of the exact method is relatively straightforward, as libraries of ODE solvers are widely available. Some libraries, such as the Matlab ODE suite, make it easy to experiment with different solvers with minimal changes to code. The method we used to choose a system capacity K is fairly simple to implement with such off-the-shelf solvers. Some computational savings may be possible by varying the system capacity adaptively over time, for example using the scheme described by Odoni and Roth (1983), but we did not do this.

What we refer to as the exact method consists of the Runge Kutta method, together with matrix multiplications at certain epochs to account for "instantaneous transitions" (see (1) in the paper). By setting the matrix $P(t)$ equal to the identity matrix, our exact method reduces to the Runge Kutta method. The Runge Kutta method is general in the sense that it applies to any continuous time Markov chain (CTMC), but its computation time is highly dependent on the number of system states that are included and other problem parameters. For example, the solution of an instance of an $M(t)/E_{10}(t)/10/15$ system was reported to have taken about 50 hours of CPU time with the exact method (Escobar et al. 2002, Green et al. 2001, 2003) reported that instances of $M(t)/M/s(t)$ systems with high average offered load $\bar{\lambda}/\mu$ required excessive computer time. It may be possible to approximate some systems by aggregating states that are similar and then apply the exact method to the aggregated process. Escobar et al. (2002) did this for $M(t)/E_k(t)/s/K$ systems and found that computation times were drastically reduced–for example, the instance mentioned earlier required only 70 seconds of CPU time with this approximation.

A.2. Randomization Method

The transition probability matrix $P = Q/L - I$ that is referred to in the paper has the following nonzero entries:

$$\begin{array}{rcll} p_{i,i+1}(t) & = & \lambda/L & \text{for } i = 0, 1, \ldots \\ p_{i,i}(t) & = & 1 - (\lambda + \min(i, s)\mu)/L & \text{for } i = 0, 1, \ldots \\ p_{i+1,i}(t) & = & \min(i, s)\mu/L & \text{for } i = 0, 1, \ldots \end{array}$$

We limited the size of the time step t to prevent Lt from getting too large, limiting t to be at most $\ln(1/\epsilon_t)/L$ with $\epsilon_t = 10^{-30}$. Other ways to avoid underflow are to use a normal approximation to calculate $p_J(j)$ or to truncate the series in (6) in the paper from below and

above, choosing the lower truncation point so as to ignore terms for which $p_J(j)$ is almost zero (see Grassmann 1977 and Reibman and Trivedi 1988 for details). Fox and Glynn (1988) describe a more sophisticated approach to computing the non-negligible $p_J(j)$ terms. Lower truncation can lead to computational savings (Reibman and Trivedi 1988) for homogeneous systems, but since we applied the randomization method over short time intervals, where the arrival rate was approximated by a constant, it was natural to limit the step size as described. However, lower truncation might have led to computational savings for some of our test problems, as we mention in the next section.

The randomization method is probably easier to implement from scratch than an ODE solver, such as the Runge Kutta method. However, off-the-shelf code is harder to find for the randomization method than for general purpose ODE solvers (although see, for example, the MARCA software package, Stewart 1994)).

The randomization method shares many characteristics with the exact method: it is applicable to any system that can be modeled as a homogeneous CTMC, computation times are highly dependent on model structure and parameter values, and the number of states K used to approximate an infinite state process can be changed adaptively (Van Moorsel 1994).

Reibman and Trivedi (1988) reported computation times for the randomization method that were on the order of 25% of those for the Runge Kutta method, for homogeneous CTMC instances. Our results indicate that randomization retains most of this computational advantage for inhomogeneous systems while providing excellent accuracy, despite the need to approximate such systems with a sequence of homogenous systems.

A.3. Closure Approximations

We begin by describing the details of Taaffe and Ong's (1987) $Ph(t)/M(t)/s/K$ closure approximation, specialized to $M(t)/M/s(t)$ systems and modified to allow for infinite system capacity. Define "partial moments" of the number of customers in the system, as follows:

$$E_1^{(p)} = \sum_{i=0}^{s(t)-1} i^p \pi_i(t), \; E_2^{(p)} = \sum_{i=s(t)}^{\infty} i^p \pi_i(t) \tag{A-1}$$

Note that $E_1^{(0)}$ is the probability that at least one server is idle and $E_2^{(0)}$ is the probability that all servers are busy. The first three partial moments satisfy the following differential equations, as shown by Taaffe and Ong (we have suppressed the time arguments for λ, s,

and π_i):

$$\begin{aligned}
E_1^{(0)\prime} &= -\lambda\pi_{s-1} + \mu s\pi_s \\
E_1^{(1)\prime} &= \lambda\left\{E_1^{(0)} - s\pi_{s-1}\right\} + \mu\left\{-E_1^{(1)} + (s-1)s\pi_s\right\} \\
E_1^{(2)\prime} &= \lambda\left\{E_1^{(0)} + 2E_1^{(1)} - s^2\pi_{s-1}\right\} + \mu\left\{E_1^{(1)} - 2E_1^{(2)} + (s-1)^2 s\pi_s\right\} \\
E_2^{(0)\prime} &= \lambda\pi_{s-1} - \mu s\pi_s \\
E_2^{(1)\prime} &= \lambda\left\{E_2^{(0)} + s\pi_{s-1}\right\} + s\mu\left\{-E_2^{(2)} - (s-1)\pi_s\right\} \\
E_2^{(2)\prime} &= \lambda\left\{E_2^{(0)} + 2E_2^{(1)} + s^2\pi_{s-1}\right\} + s\mu\left\{E_2^{(0)} - 2E_2^{(1)} - (s-1)^2\pi_s\right\}
\end{aligned} \tag{A-2}$$

We used the ode45 ODE solver to solve (A-2).

The number of customers, conditional on at least one free server $(N(t)|N(t) < s(t))$ is assumed to follow a Polya Eggenberger (PE) distribution (Johnson et al. 1993) and the number of customers conditional on all servers being busy $(N(t)|N(t) \geq s(t))$ is assumed to follow a shifted PE distribution. Consequently, the (approximate) state probabilities can be expressed as

$$\Pr\{N(t) = i\} = \pi_i(t) = \begin{cases} \gamma(i; n_1, p_1, \alpha_1)E_1^{(0)} & \text{for } i < s(t) \\ \gamma(i - s(t); n_2, p_2, \alpha_2)E_2^{(0)} & \text{for } i \geq s(t) \end{cases} \tag{A-3}$$

where $\gamma(i; n, p, \alpha)$ is a PE probability mass function with parameters n, p, and α and support $0, 1, \ldots, n$. Clark's original rationale for using a two-part specification for the closure distribution is that the stationary distribution for an $M/M/s$ system has two functional forms, one for $i < s$ and another for $i \geq s$.

Expression (A-3) has eight parameters: $E_i^{(0)}$, n_i, p_i, and α_i, for $i = 1, 2$. The probabilities $E_i^{(0)}, i = 1, 2$ are calculated directly when solving (A-2). The parameters p_i and α_i are chosen by matching moments of the two PE distributions with the moments of $N(t)|N(t) < s(t)$ and $N(t)|N(t) \geq s(t)$. See Clark (1981) and Taaffe (1982) for details of the moment matching. The parameter n_1 is set equal to $s(t) - 1$. For systems with finite capacity K, n_2 would be set equal to $K - s(t)$, but we followed Clark's (1981) recommendation to set $n_2 = \lceil 6.6E_2^{(1)}/E_2^{(0)} + 0.5\rceil$ to approximate infinite waiting space.

We found two aspects of the implementation of this method to be crucial: ensuring valid parameter values for the PE distribution and determining how to "restart" the method at epochs when the number of servers $s(t)$ changes. We discuss these two issues in turn.

Valid ranges for the first two parameters of a PE distribution are $p \in [0, 1]$ and $\alpha \in (-\min(p, 1-p)/(n-1), \infty)$. When the estimated p was outside the valid range we rounded up to zero or down to one as appropriate. Following Clark (1981), when the estimated α was

at or below its valid minimum, we set $\alpha = -\min(p, 1-p)/(n-1) + 0.001$. The equations for estimating p and α involve division, and it is possible for the denominators to be zero. We tested whether the absolute value of the denominator was less than 10^{-6}. When this happened for p, we set p to zero. When this happened for α, we set α to 400. Limited experimentation suggested that the performance of the method was insensitive to the values of p and α chosen in these situations.

The definitions of the partial moments change when the number of servers changes. Suppose the number of servers changes at epoch t. It seems natural to convert the final values of the partial moments with the previous number of servers (just before epoch t) into an approximate state probability distribution using the assumed closure distribution (see (A-3)), apply an instantaneous transition as in (1) in the paper, if appropriate calculate new partial moments using their definition and the new number of servers, and use these partial moments as initial values at epoch t. However, as mentioned in the paper, experimentation indicated that doing this increased computation time and reduced accuracy. Simply using the final partial moments with the previous number of servers as initial conditions worked much better, when the numerical integration is restarted with a new number of servers. Figure 1 illustrates this, showing the service level for one of our test problems as calculated by the exact method and by the closure approximation, with and without adjusting the partial moments at times when the number of servers changes. The number of servers changed only at epochs 0, 8, and 16 for the test problem illustrated in Figure 1 (see Figure 2, Section B).

The closure-approximation method occasionally produced sharp fluctuations in the service level that were not caused by changes in the arrival rate or the number of servers. Figure 1 shows one such fluctuation, beginning around epoch 4. Both variants of the closure-approximation method produced such a fluctuation in this test problem.

Once the partial moments had been obtained for the interval $[0, T]$, we used (A-3) to convert them into state probabilities and (4) (or (5) when appropriate) from the paper to convert the state probabilities into service levels.

Closure approximations have been developed for a variety of systems, including systems with a phase arrival process, as mentioned earlier, and certain priority queueing systems (Taaffe 1982). We found the closure approximation to be the most challenging one to implement of all the methods. Implementation challenges include the number of algorithm parameters that need to be specified, the nonlinear moment matching equations, and the need for efficient code to calculate probabilities from the PE distribution.

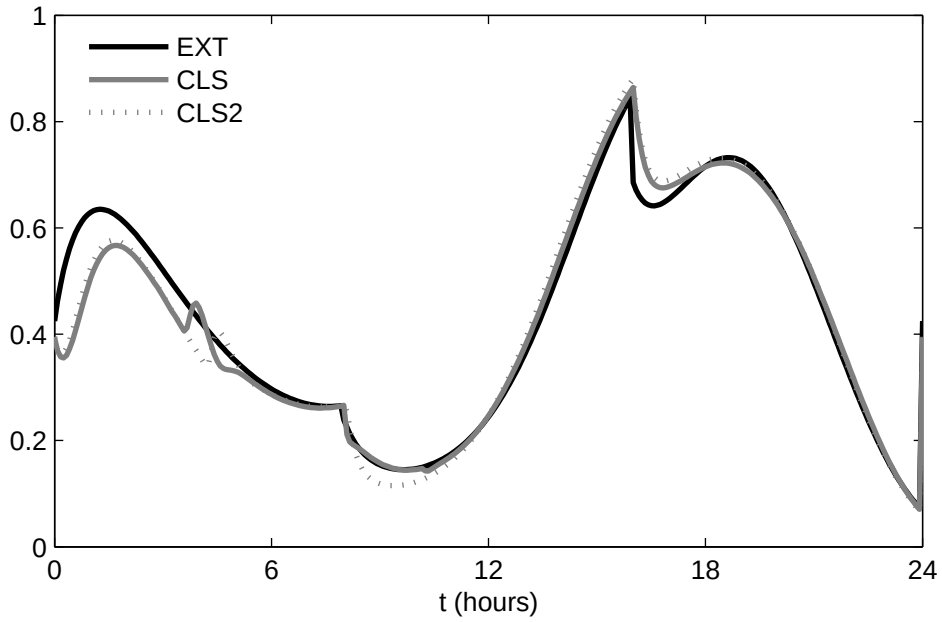

Figure 1: Service Levels Calculated by the Exact Method (EXT) and Two Variants of the Closure Approximation (CLS and CLS2). CLS2 Adjusts the Partial Moments at the Epochs when the Number of Servers Changes (0, 8, and 16); CLS Does Not. The Parameters for this Test Problem are Indicated in Figure 2, Section B.

A.4. Infinite Server Approximations

The MOL approximation retains the benefit of Clark's as well as Taaffe and Ong's closure approximations of having a two-part specification for $\pi(t)$, although the PE distribution is more flexible and may therefore lead to greater accuracy. In contrast to these closure approximations, the MOL approximation requires the solution of only one linear differential equation, as opposed to five or six nonlinear differential equations.

In approximating service levels with the MOL method, it is not necessary to evaluate all of the state probabilities. It is sufficient to evaluate the probability that all servers are busy and then closed-form expressions for the stationary queue delay distribution in an $M/M/s$ system can be used to evaluate the service level. For epochs during the last τ time units of a planning period, where the number of servers increases at the beginning of the next planning period, we use a closed-form expression that is slightly different from the standard one. This expression is obtained by substituting the steady state probabilities for the approximating stationary system into (5) from the paper instead of (4). The same applies for the effective arrival rate and lagged stationary approximation methods.

Many of the results for $M(t)/M/\infty$ systems hold for $M(t)/G/\infty$ systems (Eick et al. 1993), and some results are available for $G(t)/G/\infty$ systems as well (Jennings et al. 1996). These results could in principle be used in combination with approximations for stationary $M/G/s$ or $G/G/s$ systems to develop MOL approximations for $M(t)/G/s(t)$ or $G(t)/G/s(t)$ systems.

A.5. Effective-Arrival-Rate Approximation

It is straightforward to implement this method. We approximated the integral in (10) from the paper with a discrete sum, sampling the arrival rate $\lambda(t)$ at 0.01-hour (36-second) intervals. Service levels are calculated using the closed-form expression for the stationary queue delay distribution in an $M/M/s$ system, as for the MOL approximation. One benefit of this method is that it requires no additional computation for systems that operate continuously–all that is needed is evaluation of the arrival rate function during the interval $(-Wq - 1/\mu, T - 1/\mu]$ instead of the interval $(0, T - 1/\mu]$.

A.6. Lagged Stationary Approximation

We implemented the lagged stationary approximation similarly to the effective-arrival-rate approximation. Like that approximation, the lagged stationary approximation requires no additional computation for systems that operate continuously.

B. Experimental Design: Additional Information

We kept the cycle length of the sine wave for the arrival-rate function fixed at 24 hours. However, the impact of different cycle lengths can be seen by comparing cases that are identical except that the arrival rate and the service rate are both multiplied by the same constant, because the effect of doing this is the same as dividing the cycle length with that constant (see, for example, Green et al. 1991).

Figure 2 shows an arrival-rate function and a servers function, as we have parameterized them, for the same test problem as in Figure 1.

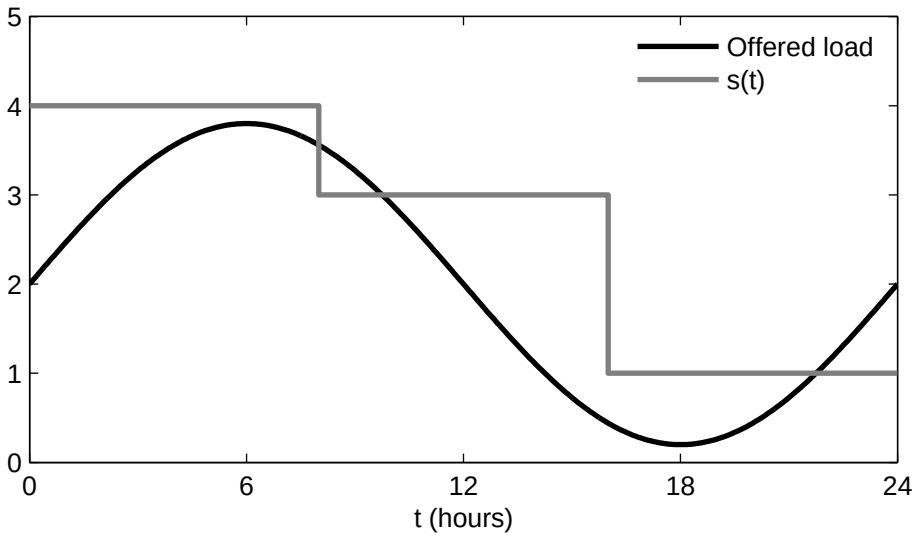

Figure 2: The Offered Load and the Number of Servers as a Function of Time for the Test Problem in Figure 1. The Parameter Values are $r = 2$, $\mu = 2$, $\alpha = \beta = 0.9$, $\gamma = 0$, $\bar{\rho} = 0.95$, $\delta_p = 8$, and $\tau = 0$.

B.1. Error Measurement

Letting $\bar{\mathrm{SL}}_{\mathrm{EXT}}(t)$ be a value for the complementary service level computed by the exact method and $\bar{\mathrm{SL}}_{\mathrm{APPROX}}(t)$ a value computed by some other method, the relative error is

$$\frac{\left|\bar{\mathrm{SL}}_{\mathrm{EXT}}(t) - \bar{\mathrm{SL}}_{\mathrm{APPROX}}(t)\right|}{\bar{\mathrm{SL}}_{\mathrm{EXT}}(t)}. \tag{B-4}$$

As indicated in the paper, the relative error can, unfortunately, sometimes be misleading. To illustrate the reasons for this, we show in Figure 3 the results of computations with the exact method and the randomization method for one test case in our experimental design, with parameters $\mu = 32, r = 32, \alpha = \beta = 0.9, \bar{\rho} = 0.5, \gamma = 0, \delta_p = 0.5$, and $\tau = 0$. The instantaneous offered load $\lambda(t)/\mu$ and the number of servers $s(t)$ are shown in the top panel of Figure 3. The bottom two panels show the complementary service level calculated by the two methods, first on a linear and then a logarithmic scale. In this case, the complementary service level is simply the probability of delay (because $\tau = 0$).

Visually, it is difficult to see any differences between the two delay-probability curves when viewed on a linear scale. The logarithmic scale reveals substantial *relative* differences between the curves during the first 12 hours of the day, when the delay probability is close to zero. Although unlikely to be important in practice, the relative-error formula (B-4)

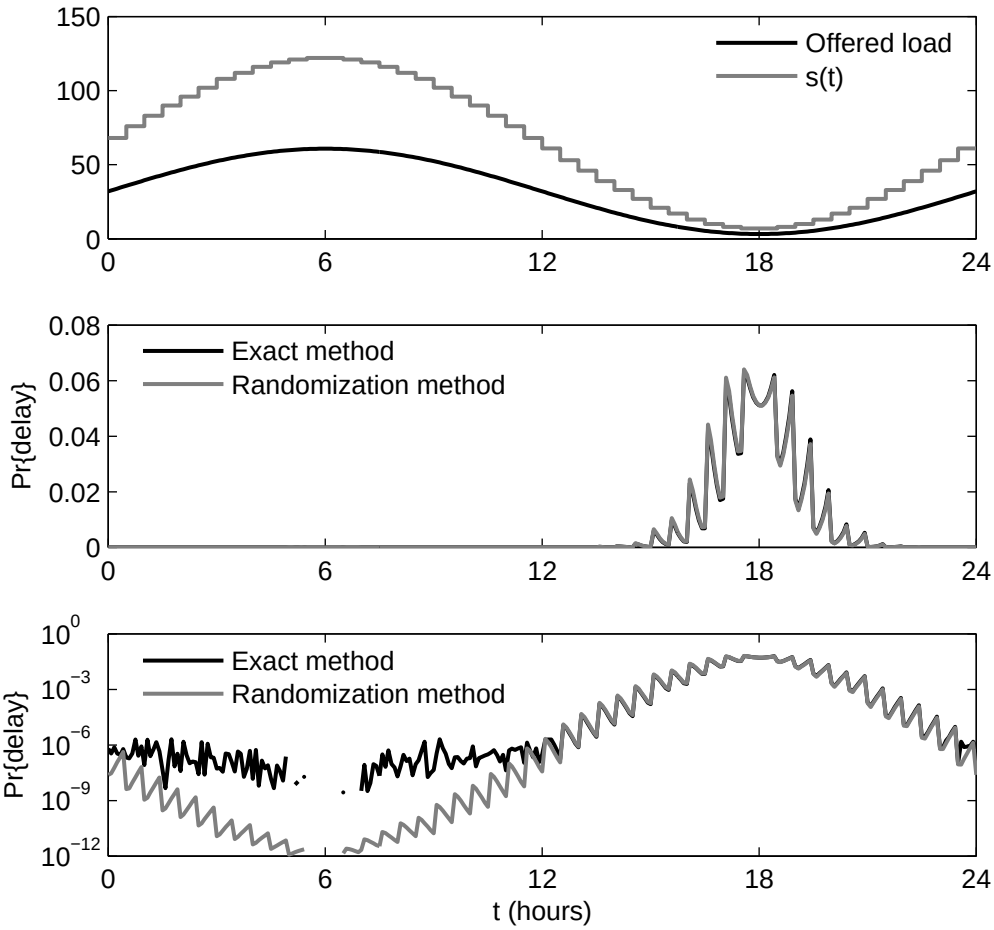

Figure 3: Probability of Delay, as Calculated by the Exact Method and the Randomization Method, for a System with Parameters $\mu = 32, r = 32, \alpha = \beta = 0.9, \bar{\rho} = 0.5, \gamma = 0, \delta_p = 0.5$, and $\tau = 0$

magnifies these differences in a similar way as the logarithmic scale. We contend that when the absolute error is small, the relative error is not important, even when it is large.

The limits on the accuracy of the exact method are another reason not to assign much importance to situations where the absolute error is small but the relative error is large. The case illustrated in Figure 3 is a low-utilization system where the number of servers changes once every half hour. The offered load is nine times higher at its peak than at its lowest point, and the ratio between the offered load and the number of servers is approximately constant over time. Because of economies of scale, we expect the delay probability to be lowest soon after the offered load peaks (at time $t = 6$) and highest soon after the offered load reaches its lowest point (at $t = 18$), and we expect the shape of the delay-probability curve to be related to the offered-load curve. The randomization method generates a delay-probability curve that is in good agreement with these expectations (especially when viewed on a logarithmic scale), but the "exact method" seems to "break down" for very small delay probabilities. There is nothing inconsistent about this: we used a default "absolute error tolerance" of 10^{-6} for Matlab's `ode45` solver, which implies that the estimated error in each integration step, for each component $\pi_i(t)$, can be as high as 10^{-6}. Figure 3 shows that the exact method does indeed compute delay probabilities that look plausible as long as they are above 10^{-6}, but below that point, the computed values from the exact method do not look plausible.

The bottom panel in Figure 3 has some missing values, corresponding to epochs when the calculated probability of delay equals zero. This illustrates a further problem with using (B-4): the possibility that the denominator $\bar{\mathrm{SL}}_{\mathrm{EXT}}(t)$ is zero. The reciprocal of the denominator, $1/\bar{\mathrm{SL}}_{\mathrm{EXT}}(t)$ acts to magnify the absolute error, $\left|\bar{\mathrm{SL}}_{\mathrm{EXT}}(t) - \bar{\mathrm{SL}}_{\mathrm{APPROX}}(t)\right|$. When the denominator is smaller than the accuracy that one can expect from the exact method, this magnification can lead to misleading results.

We modified the relative-error calculation as follows, to reduce the impact of the difficulties discussed above:

$$\frac{\max\left(0, \left|\bar{\mathrm{SL}}_{\mathrm{EXT}}(t) - \bar{\mathrm{SL}}_{\mathrm{APPROX}}(t)\right| - 10^{-3}\right)}{\bar{\mathrm{SL}}_{\mathrm{EXT}}(t) + 10^{-3}}. \tag{B-5}$$

As this expression shows, we ignore the relative error (that is, set it to zero) when the absolute error is less than one in a thousand, and we limit the magnification of the absolute error to be at most thousand-fold (by adding 10^{-3} to the denominator). Absolute errors of less than one in a thousand seem unlikely to be important in practice and magnifying absolute

errors by more than one thousand does not seem likely to produce useful information. The average relative error for the randomization method, for the case illustrated in Figure 3, computed using (B-5), is 0.06%, while the maximum relative error is 2.2%. These numbers seem consistent with a visual assessment of the Figure. All errors reported in the paper and in the next section are relative errors, computed using (B-5).

C. Additional Computational Results

In Table 1, we examine whether the approximation methods perform better when server staffing is matched to demand (we will refer to such systems as "balanced"), i.e., when $\alpha = \beta$ and $\gamma = 0$. The summary statistics in Table 1 are broken down by whether the system is balanced and by whether the wait threshold τ is zero or positive. We observe that computation times for balanced systems are considerably lower for the EXT, RND, and CLS methods but not much affected for the ISA, MOL, EAR, and LST methods. The accuracy of the RND method is insensitive to whether the system is balanced. The other approximation methods are generally more accurate for balanced systems but this effect is not consistent. For example, the median time-average relative error for the MOL, EAR, and LST methods is lower for balanced systems when the wait threshold is zero, but higher for balanced systems when the wait threshold is positive.

We investigated the accuracy of RND further by taking a closer look at test problems where it was least accurate. Figure 4 shows the results of applying the exact method and the randomization method to a test problem where RND resulted in a time-average relative error of 0.45%. RND resulted in time-average relative error of less than 0.4% for all other test problems, so Figure 4 illustrates the worst-case accuracy for RND, among the problems we solved.

We investigated the potential of left truncation (described in Section 3.2 of the paper) to reduce computation for RND by further analyzing a test problem where RND required a comparatively long computation time (866 seconds for a 24-hour interval). A 24-hour interval has 288 five-minute calculation periods, which RND uses as time steps by default. For this test problem, the duration of the time step was always reduced below 5 minutes (actual step sizes ranged from 0.2 to 4 minutes) to prevent Lt from getting too large. The average computation time per step was 0.92 seconds. Left truncation would have reduced the number of steps from 864 to 288 but it would also have increased the computation time

Table 1: Comparison of Balanced (BAL, where $\alpha = \beta$ and $\gamma = 0$) and Unbalanced Systems, for Wait Thresholds (WT, or τ) that are Zero or Positive.

		Method						
BAL	**WT**	**EXT**	**RND**	**CLS**	**ISA**	**MOL**	**EAR**	**LST**
Computation time: median (sec.)								
yes	0	26.61	12.15	26.58	1.07	1.05	0.63	0.71
no	0	44.14	19.19	30.58	0.98	0.87	0.56	0.66
yes	> 0	26.37	12.14	24.13	1.07	0.95	0.63	0.70
no	> 0	44.20	19.76	30.58	0.98	0.86	0.56	0.66
Computation time: mean (sec.)								
yes	0	281.6	52.2	145.0	1.15	0.98	0.57	0.66
no	0	3886.1	327.3	6401.2	1.18	0.99	0.56	0.65
yes	> 0	281.3	52.0	117.8	1.03	0.90	0.57	0.66
no	> 0	3886.2	327.3	6204.7	1.07	0.91	0.56	0.65
Time-average relative error: median								
yes	0	N/A	0%	40%	30%	8%	8%	21%
no	0	N/A	0%	39%	32%	16%	18%	40%
yes	> 0	N/A	0%	13%	40%	74%	73%	111%
no	> 0	N/A	0%	18%	45%	60%	61%	87%
Time-average relative error: mean								
yes	0	N/A	0.036%	1731%	35%	28%	24%	1988%
no	0	N/A	0.016%	938%	46%	41%	41%	1076%
yes	> 0	N/A	0.012%	34%	43%	141%	144%	2279%
no	> 0	N/A	0.003%	56%	58%	144%	146%	948%
Maximum relative error: median								
yes	0	N/A	0%	116%	72%	98%	97%	114%
no	0	N/A	0%	347%	100%	125%	121%	465%
yes	> 0	N/A	0%	99%	89%	234%	235%	509%
no	> 0	N/A	0%	158%	99%	318%	343%	852%
Maximum relative error: mean								
yes	0	N/A	0.349%	3725%	1404%	1924%	1688%	10975%
no	0	N/A	0.305%	5555%	4897%	5505%	5487%	13739%
yes	> 0	N/A	0.125%	605%	143%	4172%	4141%	13228%
no	> 0	N/A	0.102%	1732%	3025%	9816%	9874%	15532%

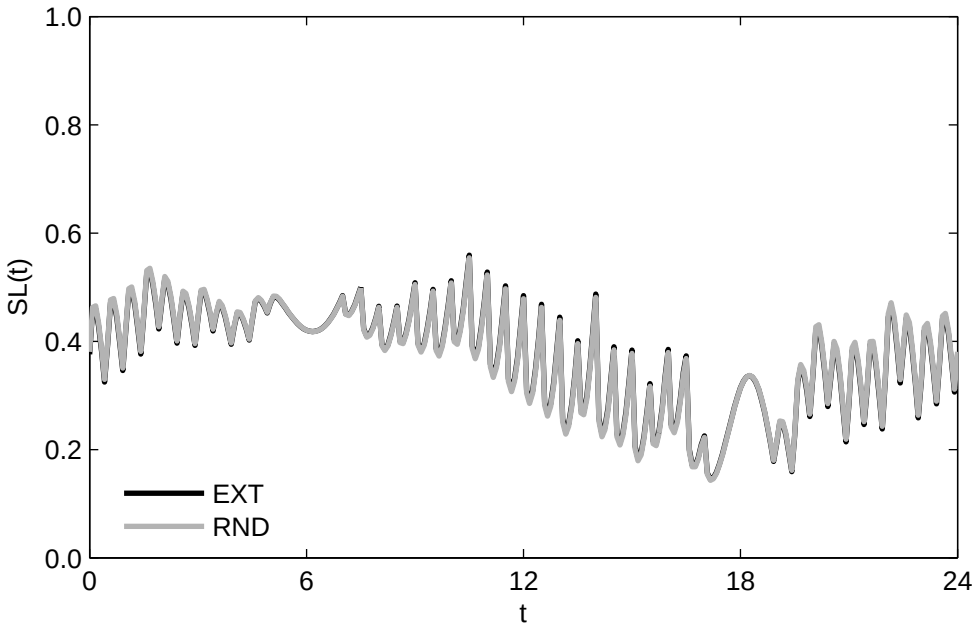

Figure 4: Service Levels Calculated by the Exact and Randomization Methods for the Case where they Differed Most, on Average. The Parameter Values are $r = 32$, $\mu = 32$, $\alpha = 0.9$, $\beta = 0.9$, $\gamma = 0$, $\bar{\rho} = 0.95$, $\delta_p = 0.5$, and $\tau = 0$. The Time-Average Relative Error was 0.4% and the Maximum Relative Error was 1.1%.

per step. As long as the increase per step is less than 3-fold, left truncation would result in reduced total computation time for this test problem.

ISA results in approximate upper bounds on the exact service level. We calculated the fraction of time that the service level calculated by ISA was greater than or equal to the exact service level, for each test problem, and found this fraction to be 98.4% on average and 74.7% in the worst case. ISA would provide an exact upper bound were it not for the modeling of servers staying beyond their scheduled end time to finish the current service (see Section 2 in the paper). The ability to generate upper bounds quickly on the service level may be useful in some applications. For example, the employee-scheduling algorithm described in Ingolfsson et al. (2002) relies on "strict lower bounds" on the staffing required in each period to provide a specified service level: the minimum number of servers needed in a period to provide a specified level of service, given that the system is empty at the beginning of the period and all waiting customers enter service immediately at the end of the period. The strict lower-bound calculation is the most time-consuming aspect of the algorithm. ISA could be used to generate quickly such lower bounds that could then be refined using either the exact method or the randomization method.

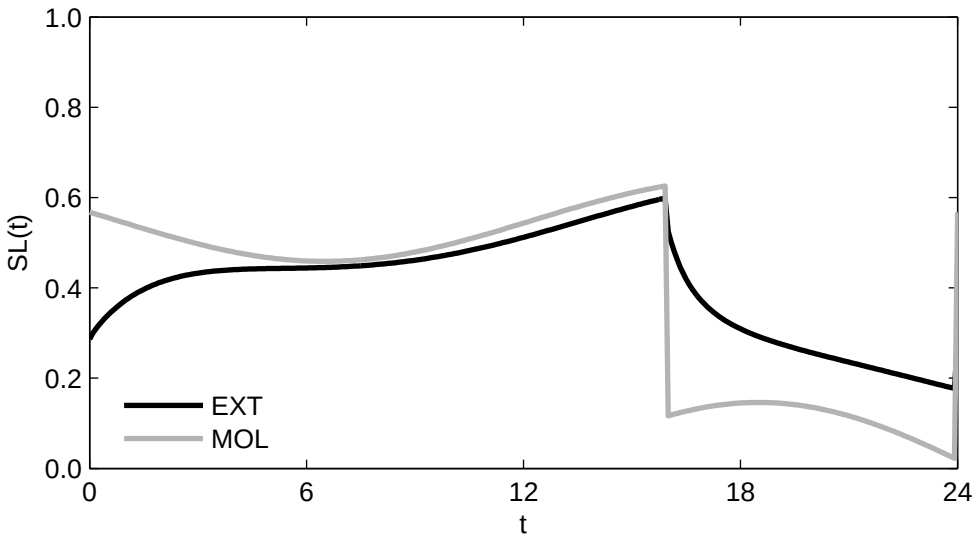

Figure 5: Service Levels Calculated by EXT and MOL for a Test Problem with Parameters $r = 2$, $\mu = 2$, $\alpha = 0.1$, $\beta = 0.1$, $\gamma = 0$, $\bar{\rho} = 0.95$, $\delta_p = 8$, and $\tau = 0$. The Number of Servers Changed at Epochs 0 and 16.

Lower bounds on the service level can also be useful. LST results in an approximate lower bound on the service level, under some conditions. We found that the average utilization had the strongest influence, with LST producing lower bounds 82.6% of the time when the average utilization was 50% but only 70.3% of the time when the average utilization was 95%.

The MOL approximation often results in systematic errors when the number of servers changes, as illustrated in Figure 5 for a test problem where the number of servers changed at epochs 0 and 16. The service level as approximated by MOL does change at these epochs, but this happens only because the set of states included in the "all-servers-busy" category changes. The probabilities for these states are used to calculate the service level using either (4) or (5) from the paper. The calculation of the mean number of busy servers using (7) from the paper is not affected by changes in the number of servers. Consequently, the MOL approximation sometimes gives the wrong sign for the slope of the service-level curve just after a change in the number of servers. The EAR approximation exhibits similar behavior.

The acceptable waiting time τ generally did not have much impact on computation time. The mean computation times for the CLS, ISA, and MOL methods are slightly lower for longer acceptable waiting times. The dominant cause of this reduction was that fewer 24-

hour cycles were needed to reach periodic steady state when τ was larger. For example, the CLS method required an average of 4.52 cycles to converge when $\tau = 0$ compared to 3.98 cycles when $\tau = 1/\mu$.

We were curious to see whether the speed advantage of RND over EXT was consistent across all test problems. This was not the case, but the test problems where RND took equal or more time than EXT were in a well-defined category—they all had a low service rate of 2 per hour and a low average offered load of 2. Figure 6 demonstrates that for test problems in this category, RND took more than twice as long as EXT when planning periods were long (8 hours) and about the same time as EXT when planning periods were short (0.5 hours). Outside this category RND required less than half of the computation time taken by EXT. Thus, EXT may be preferable to RND for situations where the arrival and service rates are low and planning periods are long.

CLS requires the solution of a fixed number of differential equations. Therefore, one would expect this method to have lower computational times than EXT and RND for large systems. We tried to determine under what conditions CLS had a computational advantage over these two methods. Comparing computation times for CLS and EXT, we found that CLS was almost always faster when the average utilization was high (95%) and the service rate or the average offered load (or both) took its high value. When both the service rate and the average offered load took their low values, EXT was almost always faster than CLS.

We expected to see qualitatively similar results when comparing CLS and RND, i.e., that CLS would have a computational advantage for systems that were sufficiently large, in the sense of the average offered load or the service rate being high, but this did not happen. We were not able to find any category where CLS was consistently faster than RND.

References

Clark, G. M. 1981. Use of Polya distributions in approximate solutions to nonstationary M/M/s queues. *Communications of the ACM* **24** 206–217.

Eick, S. G., W. A. Massey, W. Whitt. 1993. $M_t/G/\infty$ queues with sinusoidal arrival rates. *Management Science* **39** 241–252.

Escobar, M., A. R. Odoni, E. Roth. 2002. Approximate solution for multi-server queueing systems with Erlangian service times. *Computers & Operations Research* **29** 1353–1374.

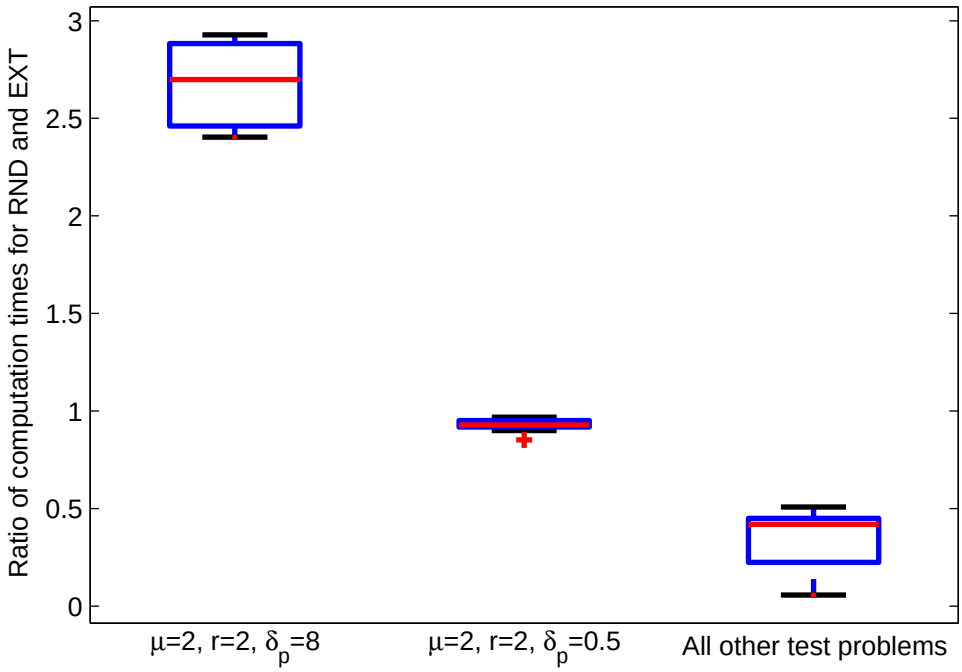

Figure 6: The Ratio of Computation Times for RND and EXT, for Test Problems where $\mu = 2$/Hour, $r = 2$, and $\delta_p = 8$ Hours and where $\mu = 2$/Hour, $r = 2$, and $\delta_p = 0.5$ Hours, Compared to all other Test Problems.

Fox, B. L., P. W. Glynn. 1988. Computing Poisson Probabilities. *Communications of the ACM* **31** 440–445.

Grassmann, W. K. 1977. Transient solutions in Markovian queueing systems. *Computers & Operations Research* **4** 47–53.

Green L. V., P. J. Kolesar, J. Soares. 2001. Improving the SIPP approach for staffing service systems that have cyclic demands. *Operations Research* **49** 549–564.

Green L. V., P. J. Kolesar, J. Soares. 2003. An improved heuristic for staffing telephone call centers with limited operating hours. *Production and Operations Management* **12** 1–16.

Green, L. V., P. J. Kolesar, A. Svoronos. 1991. Some effects of nonstationarity on multiserver Markovian queueing systems. *Operations Research* **39** 502–511.

Ingolfsson, A., E. Cabral., X. Wu. 2002. Combining integer programming and the randomization method to schedule employees. Research Report No. 02–1, Department of Finance and Management Science, Faculty of Business, University of Alberta, Edmonton, Alberta, Canada.

Jennings, O. B., A. Mandelbaum, W. A. Massey, W. Whitt 1996. Server staffing to meet time-varying demand. *Management Science* **42** 1383–1394.

Johnson, N.L., S. Kotz, A.W. Kemp. 1993. *Univariate Discrete Distributions.* Wiley, New York.

Odoni, A. R., E. Roth. 1983. An empirical investigation of the transient behavior of stationary queueing systems. *Operations Research* **31** 432–455.

Reibman, A., K. Trivedi. 1988. Numerical transient analysis of Markov models. *Computers & Operations Research* **15** 19–36.

Stewart, W. J. 1994. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, Princeton, NJ.

Taaffe, M. R. 1982. Approximating nonstationary queueing models. Ph.D. dissertation, The Ohio State University.

Taaffe, M. R., K.L. Ong. 1987. Approximating nonstationary $Ph(t)/M(t)/s/c$ queueing systems. *Annals of Operations Research* **8** 103–116.

Van Moorsel, A. P. A., W. H. Sanders. 1994. Adaptive uniformization. *Communications in Statistics Stochastic Models* **10** 619–647.