Date of draft: 29 June 2002

A Survey and Experimental Comparison of Service Level Approximation Methods For Non-Stationary *M*/*M*/*s* Queueing Systems

Armann Ingolfsson • Elvira Akhmetshina • Susan Budge • Yongyue Li • Xudong Wu

School of Business, University of Alberta, Edmonton, Alberta T6G 2R6, Canada
School of Business, University of Alberta, Edmonton, Alberta T6G 2R6, Canada
School of Business, University of Alberta, Edmonton, Alberta T6G 2R6, Canada
School of Business, University of Alberta, Edmonton, Alberta T6G 2R6, Canada
Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2R6, Canada
aingolfs@ualberta.ca • elvira@ualberta.ca • sbudge@ualberta.ca • yongyue@ualberta.ca • xudong@cs.ualberta.ca

We compare the performance of six methods in computing or approximating service levels for nonstationary *M*/*M*/*s* queueing systems: an exact method (a Runge Kutta ordinary differential equation solver), the randomization method, a closure (or surrogate distribution) approximation, a direct infinite server approximation, a modified offered load infinite server approximation, and an effective arrival rate approximation. We used all of the methods to solve the same set of 128 test problems. The randomization method was almost as accurate as the exact method, and used less than half the computational time of the exact method. The closure approximation was less accurate, and in many cases slower, than the randomization method. The two infinite server based approximations and the effective arrival rate approximation had were less accurate but had computation times that were far shorter and less problem-dependent than for the other three methods.

(*Queues, Nonstationary; Queues, Algorithms; Queues, Approximations*)

1. Introduction

Systems where time-variable and stochastic customer demand is served by a time-variable number of servers are pervasive in modern society; for example, call centers, retail stores, and emergency services. Planning for such systems involves a difficult trade-off between the cost associ-

ated with the servers (which we will call *labor cost* – although more generally some services could be provided by machines rather than humans) and the customer *service level.* Labor cost is usually relatively easy to quantify, at least compared to the level of service. A common quantification of the level of service is the fraction of customers that wait less than some threshold amount of time, τ, before being served. While one can measure service level after the fact if appropriate data have been collected, it is more difficult to predict what level of service will result from making a certain number of servers available at different times. The aim of this paper is to compare six proposed methods for calculating or approximating the service level, as defined above, for $M(t)/M/s(t)$ queueing systems (with a nonstationary Poisson arrival process, exponentially distributed service times, and a time-varying number of servers) in terms of computational effort and accuracy.

We are primarily interested in two related applications: finding staffing requirements that, if followed, will provide a specified level of service, and evaluating the time-varying service level resulting from a given staffing schedule. The latter application is generally more challenging than the former because it involves utilization (offered load divided by number of servers) that may vary widely over time and can exceed 100% temporarily. Utilization above 100% can occur when the number of servers is limited by the size of a physical facility, or if a scheduling algorithm calls a subroutine to evaluate service levels for various schedules, as in Ingolfsson and Cabral (2002).

While performance evaluation for service systems in which the number of servers can be varied is our primary motivation, the results of our comparisons may be of interest for other application areas as well, such as airport capacity evaluation (Bookbinder, 1986, Koopman, 1972), allocation of helicopters for fighting forest fires (Bookbinder and Martell, 1979), and telephone trunk line performance evaluation (Jennings and Massey 1997).

When choosing a performance evaluation method (exact or approximate) for the applications we have outlined, 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 we focus on $M(t)/M/s(t)$ systems we summarize the extent to which each method can model more general systems. We implemented all of the methods on a common platform (Matlab), and we comment on implementation issues. We compare computation speed and accuracy for the different methods for test problems designed to span a wide range of situations encountered in practice.

The methods we compare are (1) the numerical solution of the Chapman Kolmogorov forward equations using the Runge Kutta method, (2) the randomization (or uniformization) method (Jensen, 1953, Grassmann, 1977), (3) a closure (or surrogate distribution) approximation (Rothkopf and Oren, 1979, Clark, 1981, Taaffe and Ong, 1987), (4) a direct infinite server approximation, (5), an infinite server modified offered load (MOL) approximation (Jagerman, 1975, Massey and Whitt, 1997), and (6) an effective arrival rate approximation (Thompson, 1993). We will use method (1) as a baseline and refer to it as the *exact solution*, as it involves no approximations other than the ones needed to numerically solve an infinite set of differential equations.

The list of methods we compare is not exhaustive. Notable omissions are the pointwise stationary approximation (PSA – see Green and Kolesar, 1991) and the stationary independent period-by-period approximation (SIPP – see Green et al., 2001). SIPP use the average arrival rate over a period and the number of servers in that period as inputs to a stationary $M/M/s$ model, to approximate performance in that period. PSA is the limiting version of SIPP when the period length approaches zero. Implementing SIPP and PSA is straightforward and involves minimal computation. When they are sufficiently accurate, these are the methods of choice because of their simplicity. However, as Green et al. (2001) point out, the SIPP approximation results in large errors in many commonly occurring situations.

Other methods we chose not to compare include transform inversion techniques and Krylov subspace methods. See Grassmann (2000) and Stewart (1997) for a description of these methods and of some of the methods that we compared here.

All of these methods have been known for several years, and some have been used for various applications, but they do not seem to be generally known in certain obvious fields of application. For example, workforce management software packages for call centers typically use Erlang formulas based on stationary $M/M/s$ models – in effect using the PSA or SIPP approximation – even though these models often provide poor approximations to reality. Our aim is to describe the rationale for each method, mention important implementation issues, and compare

computational cost and accuracy. In implementing the methods, we tried to use straightforward and flexible approaches, as practitioners likely would. Hence, we did not use some of the sophisticated modifications that have been described in the literature, but we comment on when these modifications can potentially reduce computation.

Other researchers have compared some of these methods, but as far as we know, not in a context where both the arrival rate and the number of servers vary with time. For example, Reibman and Trivedi (1988) compared the Runge Kutta method, the randomization method, and a stable implicit ODE solver for calculating transient probabilities for stationary Markovian systems. Of the three methods, they concluded that randomization would be preferred in most situations. Leese and Boyd (1966) compared several methods that included a stable predictor-corrector ordinary differential equation (ODE) solver, a method similar to the randomization method, and simulation, for an instance of an $M(t)/M/1$ model. They concluded that a method involving a Volterra type integral equation (Wragg, 1963) was the most useful one for the application they studied. Unfortunately, this method is not applicable to multi-server systems.

We focus on $M(t)/M/s(t)$ systems for two reasons:

1. It is the simplest model that seems realistic for labor scheduling, as it incorporates both randomness and time variation in the arrival and service processes.
2. Although the methods we compare can be used for more general queueing systems, the types of possible generalizations depend on the method. All of the methods can be applied to approximate $M(t)/M/s(t)$ systems.

The two components of our model that vary with time are the arrival rate $\lambda(t)$ and the number of servers $s(t)$. The number of servers naturally changes discontinuously from one constant level to another. The arrival rate is also often modeled as piecewise constant, although it seems more natural to expect the true arrival rate to vary continuously with time. There are various reasons for modeling the arrival rate as piecewise constant:

1. The only available data on arrivals may be aggregated over time intervals of some length.
2. Estimating a continuous arrival rate function may be difficult.
3. Some methods (for example the randomization method) require the arrival rate to be piecewise constant.

As the cost of data collection and storage decreases, the first reason should become less compelling. Research continues on how to estimate arrival rate functions either from period counts

(for example, Massey et al., 1996) or transactional data (for example, Arkin and Leemis, 2000). The relevance of research on estimating continuous arrival rate functions depends, in part, on whether the use of such functions leads to different results than using a piecewise constant arrival rate function. Our comparison of the exact method, using a continuous arrival rate function, with randomization, using a piecewise constant arrival rate function over 5 minute intervals, leads to almost identical service level curves in most (but not all) cases.

The service rate μ may change with time as well, either because servers work at different rates at different times of the day or because the amount of work required to serve a typical customer varies over time. However, we expect that the service rate will typically change more slowly than the arrival rate. We therefore assume the service rate to be constant, to simplify the exposition and to limit the number of factors in our computational experiment. All of the computational methods we discuss can be extended to account for time-varying service rates.

We focus on the computation of service level, because this seems to be the most common performance measure used in determining staffing levels (for example, see Cleveland and Mayben, 1997). Some of the approximations we compare (for example, closure approximations) may perform better in approximating other performance measures, such as the mean number of customers in the system.

Two main conclusions can be drawn from our work. First, if accuracy is important, then the randomization method may be preferable to the exact method. Randomization gave almost identical results to the exact method for the problems we solved, but typically required about 35% of the computation time of the exact method. Second, when computation speed is important, the modified offered load approximation may be the preferred one, providing a median time-average absolute error of 7.5%, with median computation times of 1.4 seconds to evaluate performance for a 24-hour span.

The remainder of the paper is organized as follows. Section 2 presents the model formally and defines notation; Section 3 describes each method in a separate subsection, commenting on flexibility and implementation issues; Section 4 describes our experimental design; and Section 5 presents computational results. Section 6 summarizes the results and draws conclusions.

2. Model Definition and Common Notation

We are interested in calculating the service level, denoted SL(t), as a function of time over an interval (0, T]. We define the service level to be the probability that a customer arriving at time t would have to wait τ time units or less before commencing service. We denote the probability that the system has i customers at time t by $\pi_i(t)$ and use $\pi(t)$ to denote the vector $(\pi_0(t), \pi_1(t), \ldots)$. These probabilities evolve according to the Chapman Kolmogorov forward equations (see Kleinrock, 1974, for example):

$$\begin{aligned}
\pi_0'(t) &= -\lambda(t)\pi_0(t) + \mu\pi_1(t) \\
\pi_i'(t) &= \lambda(t)\pi_{i-1}(t) - (\lambda(t) + i\mu)\pi_i(t) + (i+1)\mu\pi_{i+1}(t) \text{ for } i = 1, 2, \ldots, s(t)-1 \\
\pi_i'(t) &= \lambda(t)\pi_{i-1}(t) - (\lambda(t) + s(t)\mu)\pi_i(t) + s(t)\mu\pi_{i+1}(t) \text{ for } i = s(t), s(t)+1, \ldots
\end{aligned} \tag{1}$$

We use primes to denote time derivatives. The forward equations in matrix form are $\pi'(t) = \pi(t)Q(t)$ where the nonzero entries in $Q(t)$ are:

$$\begin{aligned}
q_{i,i+1}(t) &= \lambda(t) \text{ for } i = 0, 1, \ldots \\
q_{i,i}(t) &= -(\lambda(t) + \min(i, s(t))\mu) \text{ for } i = 0, 1, \ldots \\
q_{i+1,i}(t) &= \min(i, s(t))\mu \text{ for } i = 0, 1, \ldots
\end{aligned}$$

The service level can be expressed as a function of the state probabilities, as follows (Ingolfsson et al. 2002):

$$\mathrm{SL}(t) = 1 - \sum_{i=s(t)}^{\infty} \pi_i(t) F(i - s(t), t) \tag{2}$$

where $F(x,t)$ is the CDF for a Poisson random variable with mean $\mu\int_t^{t+\tau} s(r)dr$.

Let $N(t)$ be the number of customers in the system at time t. Then $\pi(t)$ is the PMF for $N(t)$. We use $B(t) = \min(N(t), s(t))$ to denote the number of busy servers at time t.

We assume that the time interval (0, T] is divided into n_p "planning periods" of length δ_p and that the number of servers $s(t)$ can only change at the beginning of a planning period. Sometimes, we approximate the arrival rate function $\lambda(t)$ with a function $\tilde{\lambda}(t)$ that is piecewise constant over "calculation periods" of length δ_{calc}, i.e., $\tilde{\lambda}(t) = \tilde{\lambda}_i$ for $t \in ((i-1)\delta_{\text{calc}}, i\delta_{\text{calc}}]$. We assume that the length of a planning period is an integer multiple of the length of a calculation period.

We denote the average arrival rate by $\bar{\lambda} = \int_0^T \lambda(t)dt/T$ and the average number of servers by $\bar{s} = \int_0^T s(t)dt/T$. For some of the methods, we need to approximate the infinite capacity $M(t)/M/s(t)$ system with a finite capacity system. We denote the system capacity (the sum of the number of servers and the queue capacity) by K.

3. Computational Methods

In this section, we describe the computational methods and comment on the extent to which each can be used for more general systems, implementation issues, and evidence regarding accuracy and speed from other researchers.

3.1 Exact Method

We refer to the numerical solution of (1) using standard ODE solvers as the "exact method." This method does require approximation of the infinite set of equations (1) with the first $K + 1$ equations, and the approximations inherent in any numerical solution of ODEs. However, setting solver parameters appropriately can control the error from these approximations, and this is why we refer to this method as "exact." It is commonly used as a benchmark; see, for example, Green at al. (1991) and Odoni ad Roth (1983).

We used the ode45 Runge-Kutta ODE solver from the Matlab ODE suite (Shampine and Reichelt, 1997). We called the solver separately for each planning period, using the final solution from one planning period as the initial solution for the next. We approximated the infinite capacity $M(t)/M/s(t)$ system with a corresponding finite capacity system, starting with a system capacity K that was the greater of 100 or $\left\lceil \max_{t\in[0,T]} s(t) \right\rceil$. We checked whether the solution satisfied $\pi_K(t) \le \varepsilon_K = 10^{-6}$ for all t. If not, then we increased K in steps of 50% until the condition was met. 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.

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 exact 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 a $M(t)/E_{10}(t)/10/15$ system was reported to have taken about 50 hours of CPU time with the exact method (Escobar et al., 2002), and Green et al. (2001) 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.

3.2 Randomization Method

The randomization method (also known as the uniformization method) is originally due to Jensen (1953). Grassmann (1977), Gross and Miller (1983), and Reibman and Trivedi (1988) provide further details on the method and its probabilistic interpretation, applications, and implementation. The method relies on the fact that if the total transition rate out of every state in a homogeneous CTMC is the same, then the total number of state transitions in any time interval follows a Poisson distribution. The total transition rate is not, in general, the same for all states, but it can be made the same ("uniformized") by introducing fictional self-transitions.

Randomization, as usually presented, only applies to homogeneous CTMC. We now describe the details of the method for a homogeneous $M/M/s$ system. The total transition rate out of state i in such a system is $\lambda + \min(i,s)\mu$. Therefore, the maximum total transition rate is $L = \lambda + s\mu$, and states s, $s + 1$, … all have this total transition rate. States 0, 1, …, $s - 1$ can be uniformized by adding self-transitions to state i at rate $(s - i)\mu$. Then, state transitions (including self-transitions) will occur according to a Poisson process with rate L. When a transition occurs, the next state is chosen according to the transition probability matrix $P = Q/L - I$, which has the following nonzero entries:

$$p_{i,i+1} = \lambda / L \text{ for } i = 0,1,\ldots$$
$$p_{i,i} = 1 - (\lambda + \min(i,s)\mu)/L \text{ for } i = 0,1,\ldots$$
$$p_{i+1,i} = \min(i,s)\mu / L \text{ for } i = 0,1,\ldots$$

One can view P as a transition probability matrix for a discrete time Markov chain, with the times of transitions being "randomized" according to a Poisson process with rate L – hence the name "randomization."

Letting $p_N(n)$ be a Poisson PMF with mean Lt, the transient state probabilities at time t can be calculated by conditioning on the number of transitions in (0, t] using the law of total probability:

$$\pi(t) = \sum_{n=0}^{\infty} p_N(n)\pi(0)P^n \tag{3}$$

To implement the randomization method, one must truncate the infinite series in (3) after the first m terms. We used $m = \left\lceil Lt + 5\sqrt{Lt} + 4.9 \right\rceil$ as suggested by Grassmann (1989). More generally, m can be chosen to control the truncation error by using a normal approximation to the Poisson distribution and adding a constant term (the 4.9 in the expression we used) for situations where the normal approximation is poor. Efficient calculation computes the terms in (3) recursively. The first term in the series involves $p_N(0) = \exp(-Lt)$. If Lt is large, this calculation will result in zero with finite precision arithmetic and will also cause $p_N(1), p_N(2), \ldots$ to evaluate to zero. There are various ways to avoid this. We chose to limit the size of the time step t to prevent Lt from getting too large, limiting t to be at most $\ln(1/\varepsilon_t)/L$ with $\varepsilon_t = 10^{-30}$. Other ways to avoid underflow are to use a normal approximation to calculate $p_N(n)$ or to truncate the series in (3) from below and above, choosing the lower truncation point so as to ignore terms for which $p_N(n)$ is almost zero (see Grassmann (1977) and Reibman and Trivedi (1988) for details). 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 section 5.

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 standard ODE solvers.

For homogeneous CTMCs, the randomization method involves only two approximations: truncation of the infinite series in (3) and truncation of the state space if it is infinite. For inhomogeneous CTMCs, such as the system we focus on, one must approximate the time-varying elements in the generator matrix Q with piecewise constant functions, and apply the method separately over each interval where the generator matrix is constant. For our purposes, this involved approximating the arrival rate function $\lambda(t)$ with a piecewise continuous approximation $\tilde{\lambda}(t)$.

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 the randomization retains most of this computational advantage for inhomogeneous systems while providing acceptable accuracy in most cases, despite the need to approximate such systems with a sequence of homogenous systems.

3.3 Closure Approximations

Rothkopf and Oren (1979) presented a closure approximation that can be applied to $M(t)/M/s(t)$ queues, whereby they attempted to approximate the infinite set of differential equations (1) with two differential equations – one for the mean $\mathrm{E}[N(t)]$ and another for the variance $\mathrm{var}[N(t)]$ of the number of customers in the system. The equations are

$$\begin{aligned} \mathrm{E}[N(t)]' &= \lambda(t) - \mu s(t) + \mu \sum_{i=0}^{s(t)-1} (s(t)-i)\pi_i(t) \\ \mathrm{var}[N(t)]' &= \lambda(t) + \mu s(t) - \mu \sum_{i=0}^{s(t)-1} (2E[Q(t)]+1-2i)(s(t)-i)\pi_i(t) \end{aligned} \tag{4}$$

Unfortunately, these equations cannot be solved without knowing the transient state probabilities $\pi(t)$. To get around this difficulty, Rothkopf and Oren assumed that $\pi(t)$ could be approximated with a negative binomial distribution. Under this assumption, the transient state probabilities can be expressed in terms of $\mathrm{E}[N(t)]$ and $\mathrm{var}[N(t)]$, thereby "closing" the set of equations (4). One could refer to the negative binomial distribution as the *closure distribution*

chosen by Rothkopf and Oren. They found that this choice worked well for single server systems, but was less accurate when the number of servers was greater than one.

Earlier, Rider (1976) had used a conceptually similar approach for $M(t)/M/1$ queues. For such systems, the differential equation for the mean number of customers in the system becomes $\mathrm{E}[N(t)]' = \lambda(t) - \mu(1-\pi_0(t))$, so Rider only needed to approximate $\pi_0(t)$ as a function of $\mathrm{E}[N(t)]$.

Clark (1981) extended Rothkopf and Oren's work, approximating the forward equations (1) with five differential equations for the first two moments of the number of customers in the system and the number of busy servers, and for the probability that no customers are waiting in queue. Clark's closure approximation was more accurate than the earlier one by Rothkopf and Oren, especially when the number of servers exceeds one. Taaffe and Ong (1987) generalized Clark's approach to $Ph(t)/M(t)/s/K$ systems with a phase arrival process and referred to the general approach as a surrogate distribution approach. Taaffe and Ong's approach, when specialized to $M(t)/M/s(t)$ systems, is slightly different from Clark's original approach. We implemented both approaches, and found Taaffe and Ong's approach to be more accurate and faster. Therefore, we only describe their approach, modified to allow for infinite system capacity.

Define "partial moments" of the number of customers in the system, as follows:

$$E_1^{(p)} = \sum_{i=0}^{s(t)-1} i^p \pi_i(t), \quad E_2^{(p)} = \sum_{i=s(t)}^{\infty} i^p \pi_i(t), \tag{5}$$

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

$$\begin{aligned}
E_1^{(0)\prime} &= -\lambda(t)\pi_{s(t)-1}(t) + \mu\pi_{s(t)}(t) \\
E_1^{(1)\prime} &= \lambda(t)\left\{E_1^{(0)} - s(t)\pi_{s(t)-1}\right\} + \mu\left\{-E_1^{(1)} + (s(t)-1)s(t)\pi_{s(t)}(t)\right\} \\
E_1^{(2)\prime} &= \lambda(t)\left\{E_1^{(0)} + 2E_1^{(1)} - s(t)^2\pi_{s(t)-1}\right\} + \mu\left\{E_1^{(1)} - 2E_1^{(2)} + (s(t)-1)^2 s(t)\pi_{s(t)}(t)\right\} \\
E_2^{(0)\prime} &= \lambda(t)\pi_{s(t)-1}(t) - \mu\pi_{s(t)}(t) \\
E_2^{(1)\prime} &= \lambda(t)\left\{E_2^{(0)} + s(t)\pi_{s(t)-1}\right\} + s(t)\mu\left\{-E_2^{(2)} - (s(t)-1)\pi_{s(t)}(t)\right\} \\
E_1^{(2)\prime} &= \lambda(t)\left\{E_2^{(0)} + 2E_2^{(1)} + s(t)^2\pi_{s(t)-1}\right\} + s(t)\mu\left\{E_2^{(1)} - 2E_2^{(2)} - (s(t)-1)^2\pi_{s(t)}(t)\right\}
\end{aligned} \tag{6}$$

We used the ode45 ODE solver to solve (6).

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

$$\Pr\{Q(t) = i\} = \pi_i(t) = \begin{cases} \gamma(i; n_1, p_1, \alpha_1) E_1^{(0)} & \text{for } i < s(t) \\ \gamma(i - s(t); n_2, p_2, \alpha_2) E_2^{(0)} & \text{for } i \geq s(t) \end{cases} \quad (7)$$

where $\gamma(i;n,p,\alpha)$ is a Polya Eggenberger PMF with parameters n, p, and α and support 0, 1, …, n. (The PE distribution can be thought of as a generalization of the binomial distribution with parameters n and p, where the probability of success on a particular trial is influenced by the outcomes of previous trials through the parameter α – see Johnson et al. (1993) for details.) 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 (7) 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 (6). The parameters p_i and α_i are chosen by matching moments of the two PE distributions with the moments of $N(t) \mid N(t) < s(t)$ and $N(t) - s(t) \mid N(t) \geq s(t)$. 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 $n_2 = \lceil 6.6 E_2^{(1)} / E_2^{(0)} + 0.5 \rceil$ 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 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 it to zero. When this happened for α, we set it to 400. Limited experimentation suggested that the performance of the method was insensitive to the values of p and α chosen in these situations.

Suppose the number of servers changes at epoch t. Then the definitions (5) of the partial moments change as well. It seems natural in this case to convert the final values of the partial moments with the previous number of servers into an approximate state probability distribution using (7), then calculate new partial moments using the definition (5) and the new number of servers, and use these partial moments as initial values at epoch t. However, 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, when the numerical integration of (6) is restarted with a new number of servers, worked much better. 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. Figure 1 also illustrates that the closure approximation method is subject to occasional oscillations. The oscillation around time 6 was not caused by a sudden change in the arrival rate or the number of servers (these are shown in Figure 2, Section 4).

Once the partial moments had been obtained for the interval [0, T], we used (7) to convert them into state probabilities and (2) to convert the state probabilities into service levels.

The benefit of closure approximations is that the number of differential equations that need to be solved remains constant regardless of the system parameters, whereas the exact approach requires the solution of K differential equations, and the value of K needed to adequately approximate an infinite capacity system varies depending on the system parameters. The disadvantage is that the set of linear differential equations (1) is replaced, in general, by a set of nonlinear differential equations. The equations are nonlinear because the probabilities for states $s(t) - 1$ and $s(t)$ must be expressed as functions of the variables being integrated (the partial moments in (6)), in order to "close" the set of differential equations. The resulting expressions for $\pi_{s(t)-1}(t)$ and $\pi_{s(t)}(t)$ are nonlinear. Although the number of equations to be integrated remains constant, the computational effort to solve them does depend on system parameters, as our results show.

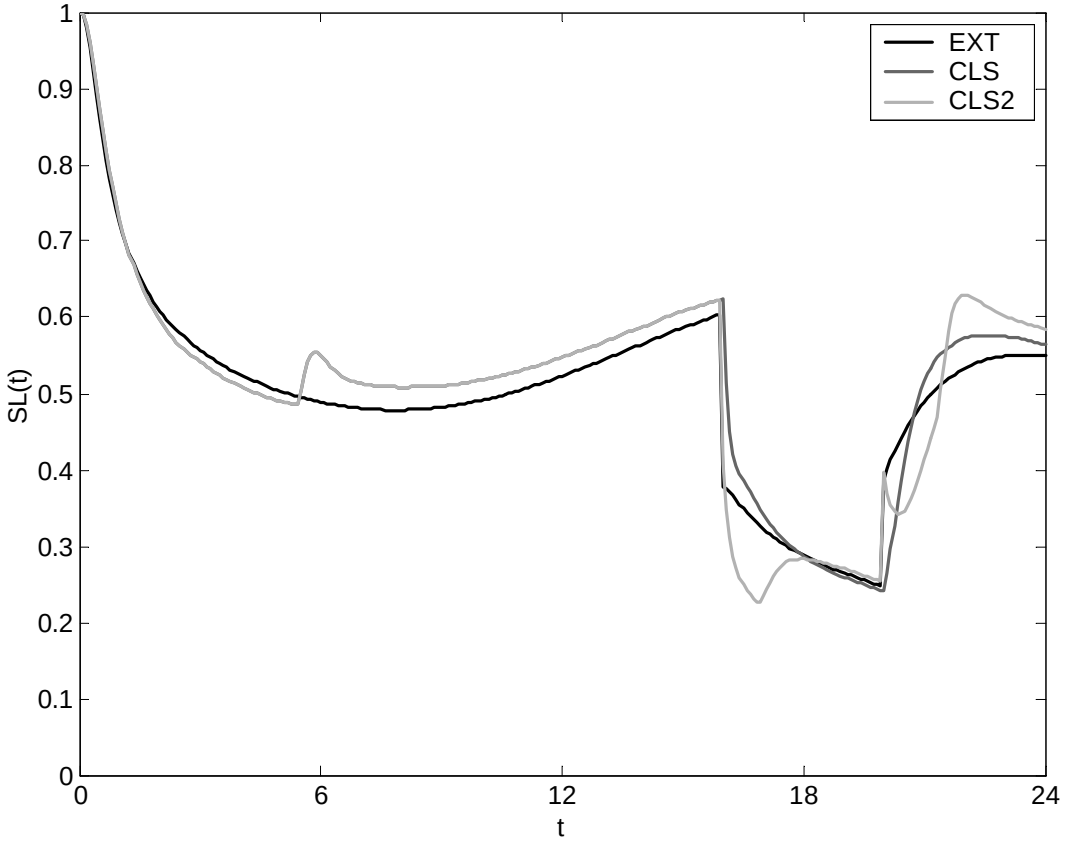

Figure 1: Service levels calculated by the exact method (EXT) and two variants of the closure approximation (CLS and CLS2). CLS2 adjusts the partial moments at the epochs when the number of servers changes (16 and 20); CLS doesn't. The parameters for this test problem are indicated in Figure 2, Section 4.

Closure approximations have been developed for a variety of systems, including $Ph(t)/M(t)/s/K$ systems, as mentioned earlier, and certain priority queueing systems (Taaffe, 1981).

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.

Taaffe and Ong (1981) mention that the closure approximation can be expected to be more accurate in light traffic (when $\sum_{i=0}^{s(t)-1}\pi_i(t) \approx 1$) or heavy traffic ($\sum_{i=s(t)}^{\infty}\pi_i(t) \approx 1$) than in medium traffic since in those situations the partial moment differential equations do not depend on the approximate state probabilities $\pi_{s(t)-1}(t)$ and $\pi_{s(t)}(t)$, and are therefore exact.

3.4 Infinite Server Approximations

The closure approximations described in the last section start with the set of forward equations (1) and derive from them a smaller set of differential equations. The approaches described in this section can also be viewed as closure approximations, but they involve an additional level of approximation, whereby an $M(t)/M/s(t)$ system is approximated with an $M(t)/M/\infty$ system. This leads to a simple differential equation for the expected number of busy servers in the system (Eick et al. 1993a).

$$\mathrm{E}[B(t)]' = \lambda(t) - \mu\,\mathrm{E}[B(t)] \tag{8}$$

We used the ode45 ODE solver to solve this equation, and its solution served as the basis for the two approximations described in this subsection. Note that this differential equation (8) is already closed – no knowledge of the state probabilities is required to solve it.

In an infinite server system, the number of customers in the system equals the number of busy servers, so equation (8) would hold with $B(t)$ replaced by $N(t)$. These two viewpoints lead to different approximations. The number of customers in an $M(t)/M/\infty$ system follows a Poisson distribution with mean determined by (8), if the system started empty in the distant past (Eick et al. 1993a). Therefore, if one views (8) as describing the evolution of the mean number in system, then one can use a Poisson PMF to approximate the state probabilities and then calculate service levels using (2). We will refer to this as a direct infinite server approximation.

In contrast, viewing equation (8) as describing the number of busy servers motivates a different choice of distribution to approximate $\pi(t)$. In a stationary $M/M/s$ system, Little's law can be applied to the servers to obtain the expected number of busy servers as λ/μ. The modified offered load (MOL) approximation uses the stationary distribution for an $M/M/s$ system to approximate $\pi(t)$, with the number of busy servers chosen to match $\mathrm{E}[B(t)]$ as obtained from solving (8). The matching is done by solving $\lambda/\mu = \mathrm{E}[B(t)]$ for λ, which gives $\lambda = \mathrm{E}[B(t)]\mu$. Thus, we approximate $\pi(t)$ with the stationary distribution for an $M/M/s$ system with arrival rate

$\mathrm{E}[B(t)]\mu$, service rate μ, and number of servers $s(t)$ (or equivalently arrival rate $\mathrm{E}[B(t)]$, service rate 1, and number of servers $s(t)$, since this stationary distribution is insensitive to multiplication of both the arrival and service rates with the same constant). The MOL approximation was originally used for $M(t)/M/s/s$ loss systems (Jagerman, 1975) and has been used for $M(t)/M/s$ systems in Massey and Whitt (1997).

The MOL approximation retains the benefit of Clark's and 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.

The MOL approximation can be expected to work best when utilization is low enough that the solution to (8) provides a good approximation to the number of busy servers over time. Note that (8) allows solutions where $E[B(t)] > s(t)$ and when this happens, the MOL approximation can be expected to be poor. When the utilization $\lambda(t)/(s(t)\mu)$ exceeds 100%, the MOL approximation may break down in the sense that the approximating stationary system is unstable. When this happened, we set the service level to zero.

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.

Many of the results for $M(t)/M/\infty$ systems hold for $M(t)/G/\infty$ systems (Eick et al., 1993a), 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.

As shown in Eick et al. (1993a), the expected number of busy servers in an $M(t)/G/\infty$ system can be expressed as:

$$\mathrm{E}[B(t)] = \int_{-\infty}^{t} G^c(t-u)\lambda(u)du \tag{9}$$

where $G^c(u)$ is the complementary service time distribution. The MOL approximation uses $\mathrm{E}[B(t)]\mu$ as an arrival rate for a stationary *M*/*M*/*s* system. One can view this quantity as an "ef-

fective arrival rate" $\lambda_{\text{eff}}(t)$. By specializing (9) to $M(t)/M/\infty$ systems, the effective arrival rate can be expressed as an exponentially weighted moving average of the arrival rate:

$$\lambda_{\text{eff}}(t) = \mathrm{E}[B(t)]\mu = \int_{-\infty}^{t} \mu e^{-\mu(t-u)} \lambda(u) du \tag{10}$$

This expression will help put the last approximation method we discuss in context.

3.5 Effective Arrival Rate Approximation

Thompson (1993) developed an effective arrival rate approximation to generate staffing requirements that take linkages between planning periods into account. Like the MOL approximation, this method develops a time-varying effective arrival rate and uses this effective arrival rate, along with the service rate and the number of servers, as input to a stationary *M*/*M*/*s* model.

We present a slightly modified version of Thompson's approximation, which is simpler to implement and should be at least as accurate. Thompson assumes the arrival rate $\lambda(t)$ to be constant over each planning period. His method is designed to generate an effective arrival rate $\lambda_{\text{eff}}(t)$ that is constant over each planning period as well. This requirement seems unnecessary, and it complicates the calculations, so we allow the effective arrival rate to vary continuously. (If desired, the resulting effective arrival rates could be averaged over each planning period.)

The method is based on two simplifying assumptions: (1) service times are deterministic and (2) waiting times in queue are deterministic and equal to *Wq*. The waiting time in queue *Wq* is estimated as the expected wait in queue in a stationary *M*/*M*/*s* model with arrival rate $\bar{\lambda}$ and service rate μ. Thompson suggested choosing the number of servers to estimate *Wq* as the minimum required to provide a pre-specified level of service in this stationary system. We would like to use the method to *predict* service levels, *given* the number of servers, so we used $\lceil \bar{s} \rceil$ to estimate *Wq*.

Under the two simplifying assumptions, the customers in service at time *t* will be the ones that arrived during the interval $[t - Wq - 1/\mu, t - Wq]$. The effective arrival rate is then calculated as the average of the true arrival rate during this interval, that is:

$$\lambda_{\text{eff}}(t) = \int_{t-Wq-1/\mu}^{t-Wq} \mu\lambda(r) dr \tag{11}$$

It is instructive to compare equations (10) and (11). The effective arrival rate under the MOL approximation is an exponentially weighted moving average of the arrival rate over the interval $(-\infty, t]$, whereas the effective arrival rate under the present approximation is a moving average over the window $[t - Wq - 1/\mu, t - Wq]$. Green and Kolesar (1998) use a similar method, which they refer to as "lagged PSA," to estimate peak probabilities of delay. In their method, $\lambda_{\text{eff}}(t) = \lambda(t - t_{\text{lag}})$, where t_{lag} is an estimate of the time lag between peaks in the arrival rate and in the expected number in system.

Like the MOL approximation, the effective arrival rate method may result in an effective arrival rate that is larger than μ times the number of servers. When this happened, we set the service level to zero.

It is straightforward to implement this method. We approximated the integral in (11) 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 almost no additional computation for systems that operate continuously – all that is needed is evaluation of the arrival rate function during the interval $(-Wq - 1/\mu, T]$ instead of the interval $(0, T]$.

4. Experimental Design

We applied the six methods to a set of 128 test problems. Our intent was to include a sufficiently wide range of conditions to permit inferences regarding the relative merit of the six computational methods. In this section, we discuss how we specified the arrival rate function $\lambda(t)$, the number of servers $s(t)$, and other problem parameters. Then we describe the combinations of parameter values that specified our test problems. Finally, we describe how we measured the accuracy and speed of each method.

Service systems often experience cyclical demand, with cycles recurring on various time scales, for example, daily, weekly, monthly, and yearly. We focus on daily cycles because the amplitude of demand variation is often greatest on this time scale. All of our experiments assume the facility begins empty of customers at time zero and operates for $T = 24$ hours.

Similarly to Green et al. (1991) and Eick et al. (1993b), we used a sinusoidal arrival rate function, specified as

$$\lambda(t) = \bar{\lambda}(1 + \alpha \sin(2\pi t / 24)) \tag{12}$$

where $\bar{\lambda}$ is the average arrival rate and $\alpha \in [0,1]$ is the relative amplitude.

We realize that a sinusoidal arrival rate with one peak per day does not capture all possible demand patterns. Arrival rates may have multiple peaks per day, the peaks may be of different heights, the duration of peaks may differ from the duration of troughs, and the function may not be well approximated by a sine wave. Yet we believe that the average arrival rate and the relative amplitude may be the two parameters of the arrival rate with the largest impact on the computation speed and accuracy of the methods that we compare and therefore we limit our attention to these parameters.

In reality, facilities differ in their open hours and days. If a system operates continuously, and the average demand cycle is the same every day, then it is often desirable to evaluate system performance in *periodic steady state* (Heyman and Whitt, 1984). This can be done by evaluating performance for n days, until performance on day $n - 1$ is sufficiently similar to performance on day n. Our experience suggests that this typically takes 3 days, but that it takes longer for extremely low service rates. In our computational experiments, we only evaluated performance for one day.

Ideally, the number of servers (employees) over time will be chosen to match the variation in demand. However, the match may be less than ideal for various reasons:

1. For a fixed number of person-hours, employing an unchanging number of servers is typically less expensive than a variable number of servers.
2. There may be an upper limit on the number of servers, determined, for example, by the number of available employees or the size of the facility.
3. Servers typically only go on or off duty at a discrete set of time instants, i.e., the number of servers remains constant in each planning period.
4. Server schedules may not have been prepared to minimize cost or maximize service level.

We are interested in the performance of these methods for situations where the number of servers is not necessarily well matched to the arrival rate for two reasons: (1) because such situations occur in reality, and so it is important to evaluate performance for such situations, and (2)

because a scheduling algorithm that calls one of the methods we are comparing as a subroutine may need to do so for some "poor" schedules on its way to the "optimal" schedule.

We model the number of servers as a sinusoidal function that is discretized to be constant over each planning period and may have a phase shift relative to the arrival rate function. Thus, we start with the continuous function $s_c(t) = \bar{s}_c(1 + \beta \sin(2\pi(t - \gamma)/24))$ where $\bar{s}_c$ is the average value of the continuous function, β is the relative amplitude, and γ is the phase shift. The number of servers peaks γ time units after the arrival rate peaks. Then, for each planning period $((i-1)\delta_p, i\delta_p]$, we replace $s_c(t)$ with its rounded up average, i.e.,

$$s(t) = \left\lceil \int_{(i-1)\delta_p}^{i\delta_p} s_c(u)du / \delta_p \right\rceil$$

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.

Table 1 shows the parameters that we varied and the values that we used for these parameters. All combinations of these values resulted in 128 different test problems. The service rates we chose correspond to a high of half-hour average service times and a low of just less than 2 minutes per service. The average offered load *r* (which equals the average number of busy servers in stationary systems) had a low value of 2 and a high value of 32, which translated to values for the average arrival rate $\bar{\lambda} = r\mu$ of 4, 64, and 1024 per hour. The relative amplitudes for both the arrival rate and the number of servers took values of 0.1 and 0.9. The average utilization had values of 0.5 and 0.95, which together with the average offered load determined the parameter $\bar{s}_c = r/\bar{\rho}$, which ranged from 2.1 to 64. The phase shift γ had values of 0 and 3 hours and the planning periods were either 15 minutes or 4 hours long.

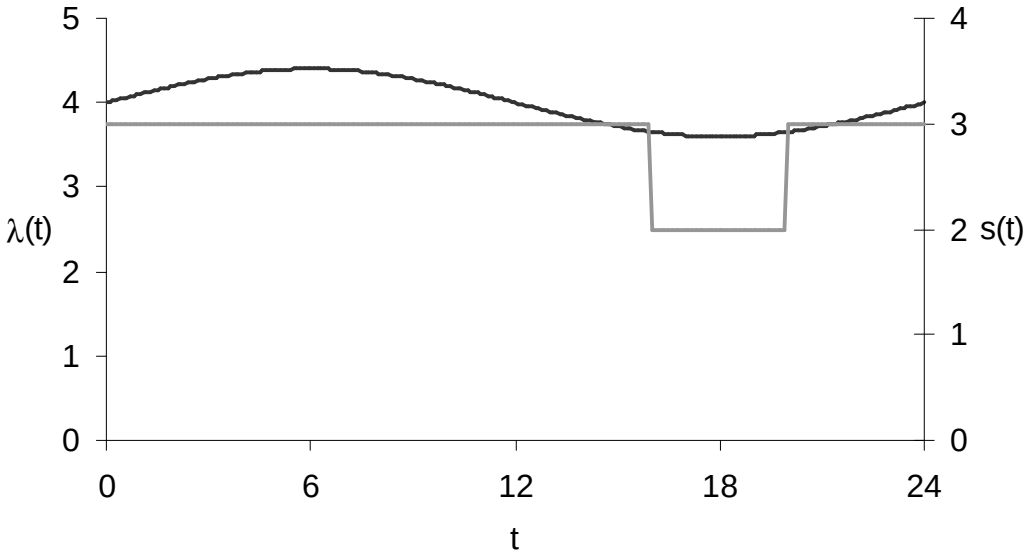

Figure 2: The arrival rate function and servers function for the test problem in Figure 1. The parameter values are $r = 2, \mu = 2, \alpha = \beta = 0.1, \gamma = 0, \bar{\rho} = 0.95$, and $\delta_p = 4$.

Table 1: Parameter values for computational experiments

Factor	Low value	High value
Service rate (μ)	2 / hour	32 / hour
Offered load ($r = \bar{\lambda}/\mu$)	2	32
Arrival rate relative amplitude (α)	0.1	0.9
Servers relative amplitude (β)	0.1	0.9
Average utilization ($\bar{\rho}$)	0.5	0.95
Phase shift (γ)	0 hours	3 hours
Planning period (δ_p)	0.25 hours	4 hours

The best way to maintain a uniform service level may be to have the number of servers peak slightly *before* the arrival rate peaks, i.e., use a negative value for the phase shift γ. Theory and experiments (Massey and Whitt, 1997, Green and Kolesar, 1997) suggest $1/\mu$ as a first order estimate of the time lag between an arrival rate peak and the peak in the expected number in system that follows it, which implies a phase shift of $\gamma = -1/\mu$. We wanted to challenge the methods with test problems where service levels could be expected to vary widely over time so we used a positive phase shift of 3 hours as our high value.

The service level threshold value τ was set to zero in all the experiments, and therefore the service level corresponded to the fraction of customers that received service without having to wait. The calculation period δ_{calc} was set to zero for all methods except the randomization method, where the arrival rate function was discretized to constant values over five minute intervals.

We calculated service levels at 5-minute intervals, starting at time zero and ending at time 24, so the output from each method for each test problem was a service level vector with 289 elements. We calculated errors and relative (percent) errors by comparing to the exact method, and we measured the CPU time used by each method for each test problem. Since the service level is bounded between zero and one, we will only present results based on errors. Analysis of relative errors led to similar results. We calculated time averages and maxima of the absolute values of the errors for each test problem and each method.

5. Computational Results

In this section, we will sometimes use abbreviations for the six methods: EXT for the exact method, RND for the randomization method, CLS for the closure approximation, ISA for the direct infinite server approximation, MOL for the modified offered load approximation, and EAR for the effective arrival rate approximation. All computations were performed on a 1 GHz Linux workstation.

Table 2 shows summary statistics (means and medians) for computation times, time-average absolute errors, and maximum absolute errors. Figure 3 uses box-and-whisker plots on a logarithmic scale to summarize the distribution of computation times for each method across the 128 test problems. Each box-and-whisker plot is centered on the median computation time per test problem, which ranged from 1.4 seconds for the MOL approximation to 59 seconds for the exact method. The randomization method had a median computation time of 20 seconds, for about 65% savings over the exact method – comparable to the 75% savings observed by Reibman and Trivedi (1988) for homogenous test problems.

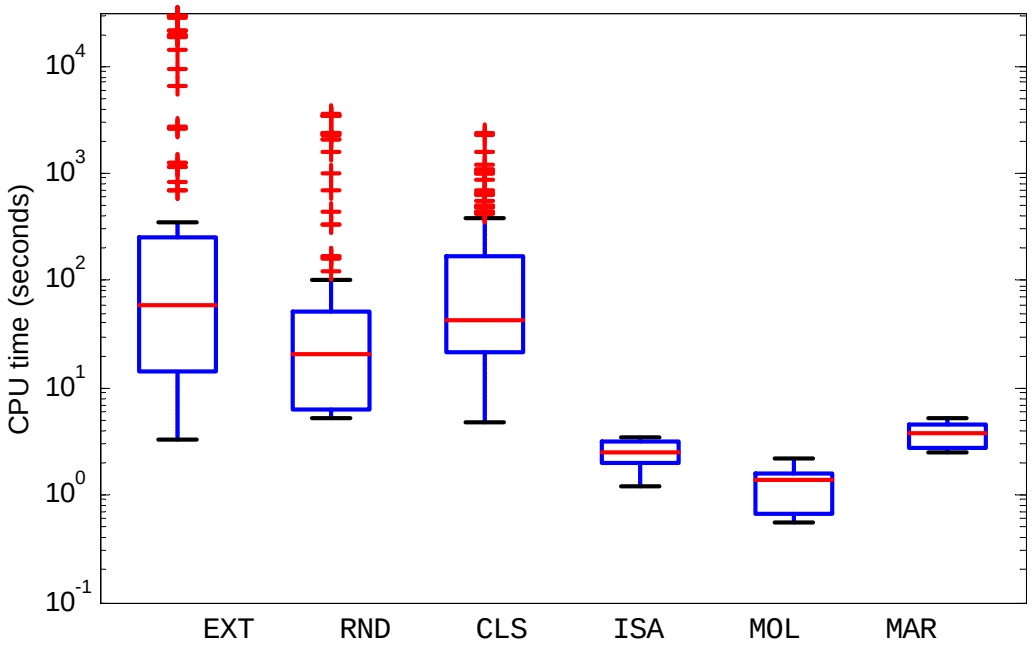

Figure 3: Box-and-whisker plots of the distribution of computation times for the six methods across 128 test problems. EXT = exact method, RND = randomization, CLS = closure approximation, ISA = direct infinite server approximation, MOL = modified offered load approximation, and EAR = modified arrival rate approximation.

Table 2: Summary of results.

	Method					
Summary statistics	**EXT**	**RND**	**CLS**	**ISA**	**MOL**	**EAR**
Computation time: median (sec.)	58.695	20.170	42.000	2.520	1.370	3.705
Computation time: average (sec.)	2,890.268	342.518	211.173	2.556	1.241	3.690
Time-average absolute error: median	N/A	0.0011	0.0456	0.1496	0.0752	0.0724
Time-average absolute error: average	N/A	0.0026	0.0847	0.1666	0.1452	0.1434
Maximum absolute error: median	N/A	0.0068	0.5095	0.5841	0.5831	0.5773
Maximum absolute error: average	N/A	0.0248	0.5172	0.5442	0.5558	0.5440

Figure 4 summarizes the distribution of time-average absolute errors, compared to the exact method, for the five approximation methods. Using the median time-average absolute error as a yardstick, the methods are ranked as follows, in order of decreasing accuracy (see Table 2): RND, CLS, EAR, MOL, ISA. RND is off by about 0.5 percentage points, CLS by about 5 percentage points, EAR and MOL by 7-8 percentage points, and ISA by 15 percentage points. Using means instead of medians or maxima instead of time-averages as yardsticks leads to similar rankings. RNS is always most accurate, CLS comes next, MOL and EAR have similar accuracy, and ISA is least accurate when one considers time-average absolute error but similar to MOL and EAR when one considers maximum absolute error.

We investigated the accuracy of RND further by taking a closer look at test problems where it was least accurate. Figure 5 shows the results of applying the exact method and the randomization method to a test problem where RND resulted in a time-average absolute error of 2.6%. RND resulted in time-average absolute error of less than 2% for all other test problems, so Figure 5 illustrates the worst-case accuracy for RND, among the problems we solved.

We investigated the potential of left truncation (described in section 3.2) to reduce computation for RND by further analyzing a test problem where RND required a comparatively long computation time (2,370 seconds of CPU time). 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 0.8 to 3.6 minutes) to prevent Lt from getting too large. The average computation time per step was 1.97 seconds. Left truncation would have reduced the number of steps from 1202 to 288 but it would also have increased the computation time per step. As long as the increase per step is less than 4-fold, left truncation would result in reduced total computation time for this test problem.

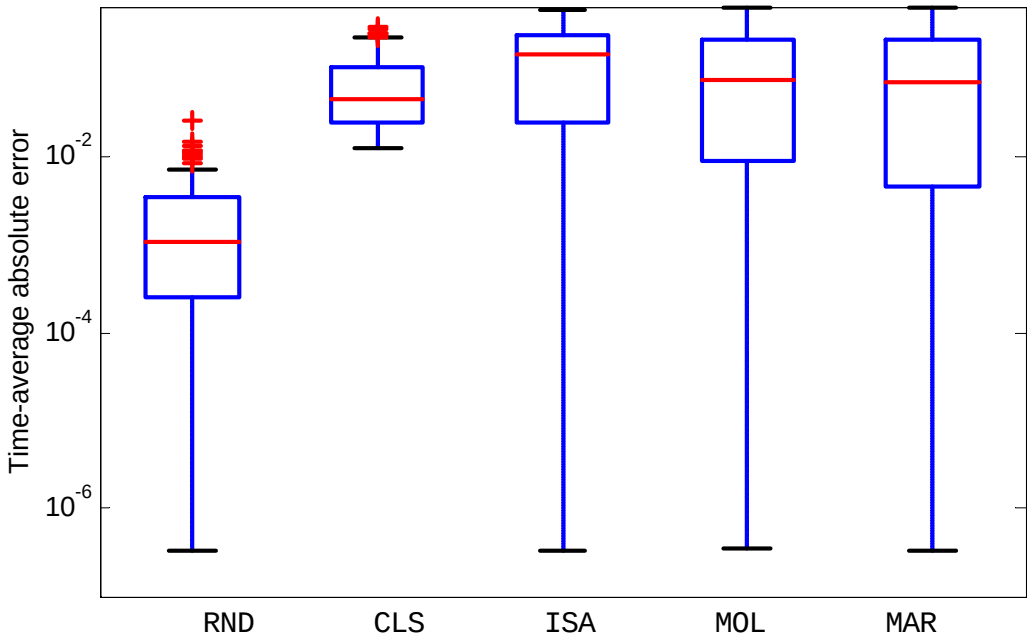

Figure 4: Box-and-whisker plots of the distribution of time-average absolute errors for the five methods (using the exact method as a standard) across 128 test problems.

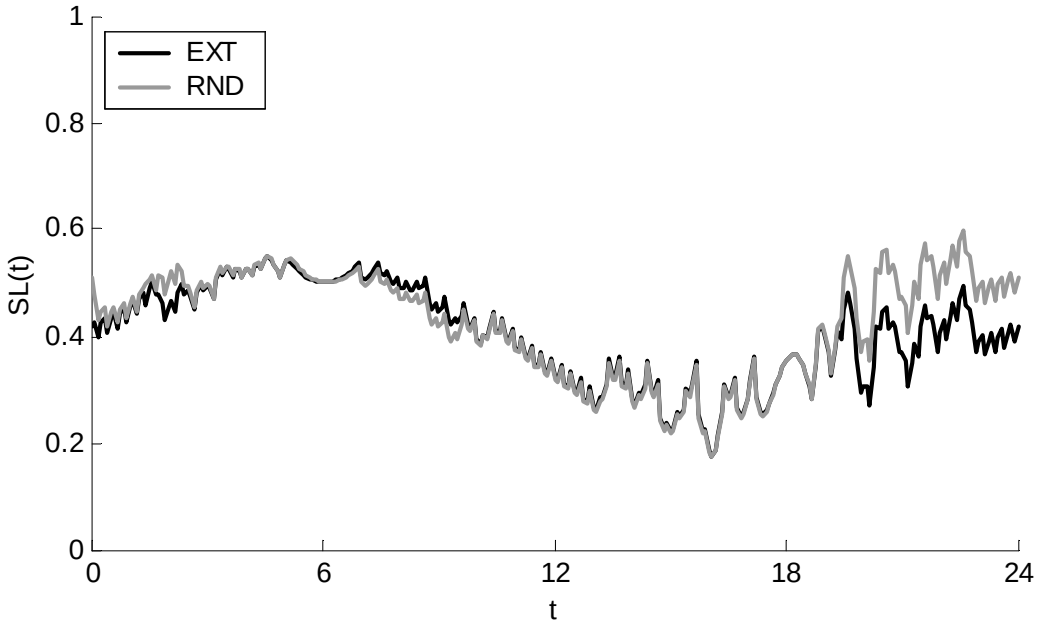

Figure 5: Service levels calculated by the exact and randomization methods for the case where they differed most, on average. The parameter values are $r = 32$, $\mu = 32$, $\alpha = 0.9$, $\beta = 0.9$, $\gamma = 0$, $\bar{\rho} = 0.95$, and $\delta_p = 0.25$.

The ISA, MOL, and EAR approximations have significantly lower and more consistent computation times than the EXT, RND, and CLS methods. Of the former three, the MOL approximation was the fastest for every test problem, and it provided similar or better accuracy than ISA and EAR for all test problems. MOL has a speed advantage over ISA because MOL uses a closed form expression (from the steady state *M*/*M*/*s* model) to evaluate the service level whereas ISA requires approximation of the service level through truncation of the series in (2). EAR uses the same closed form expression as MOL, but the approximation of the integral in (11) takes longer than numerical solution of the differential equation (8), at least with our implementation.

ISA results in upper bounds on the exact service level, at all times and for all test problems. The ability to quickly generate upper bounds on the service level may be useful in some applications. For example, the employee scheduling algorithm described in Ingolfsson and Cabral (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 quickly generate such lower bounds that could then be refined using either the exact method or the randomization method.

The MOL approximation often results in systematic errors when the number of servers changes, as illustrated in Figure 6 for a test problem where the number of servers changed at epochs 16 and 20. 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 equation (2). The calculation of the mean number of busy servers using (8) 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.

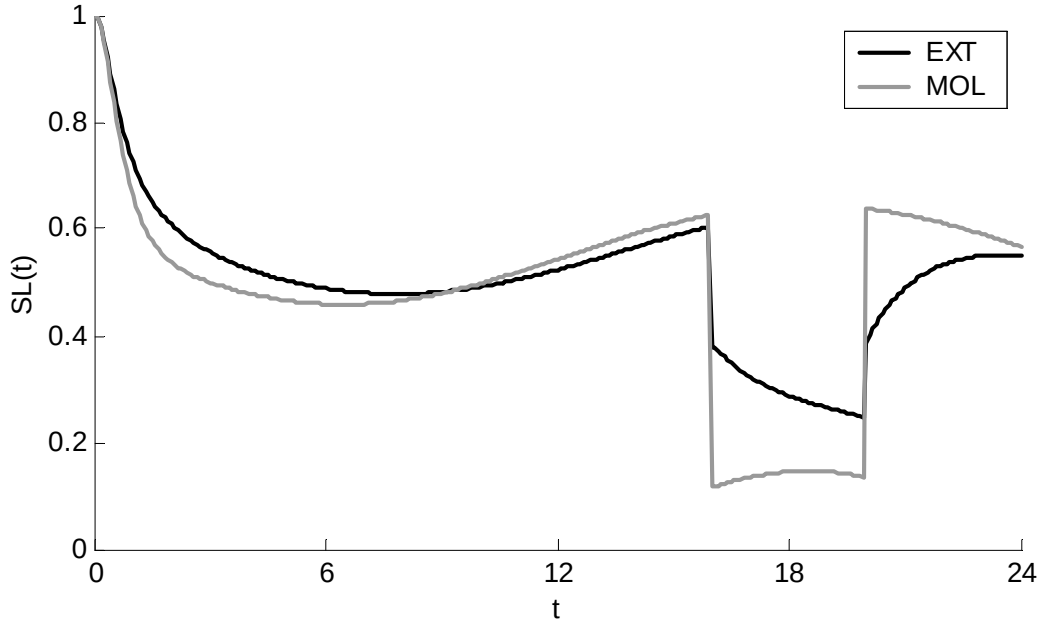

Figure 6: Service levels calculated by EXT and MOL for a test problem with parameters $r = 2, \mu = 2, \alpha = 0.1, \beta = 0.1, \gamma = 0, \bar{\rho} = 0.95$, and $\delta_p = 4$. The number of servers changed at epochs 16 and 20.

We examined the impact of the individual factors and their interactions on the computation time and accuracy of each method. Two factors, the service rate μ and the average offered load $r = \bar{\lambda}/\mu$, have by far the largest impact on computation time, for all methods. Figure 7 illustrates this for the exact method. The computation times for this method are on the order of 10 seconds when both μ and r equal 2, on the order of 100 seconds when one of μ and r equals 32 and the other equals 2, and on the order of 10,000 seconds when both μ and r equal 32. The other methods exhibit similar behavior, except that computation times grow more slowly with increasing service rate or increasing average offered load.

The average offered load is a useful measure of system size (it equals the average number of busy servers for stationary $M/M/s$ systems) and the higher the average offered load, the larger the system capacity K needed by EXT and RND to adequately approximate an $M(t)/M/s(t)$ infinite capacity system. The service rate is an indication of the *event frequency* (Green et al., 1991), i.e., the total frequency of arrivals and service completions. With our parameterization, the event frequency can be expressed as $\lambda(t) + \mu s(t) = \mu(r(t) + s(t))$ where $r(t) = \lambda(t)/\mu$, i.e., the event

frequency is proportional to the service rate for fixed $r(t)$ and $s(t)$. The higher the event frequency, the faster the number in system will grow during time intervals when the utilization is close to or above 100%. Hence, higher service rates cause the system capacity K required by EXT and RND to be higher. This explains why computation time for the EXT and RND methods increases with average offered load and service rate.

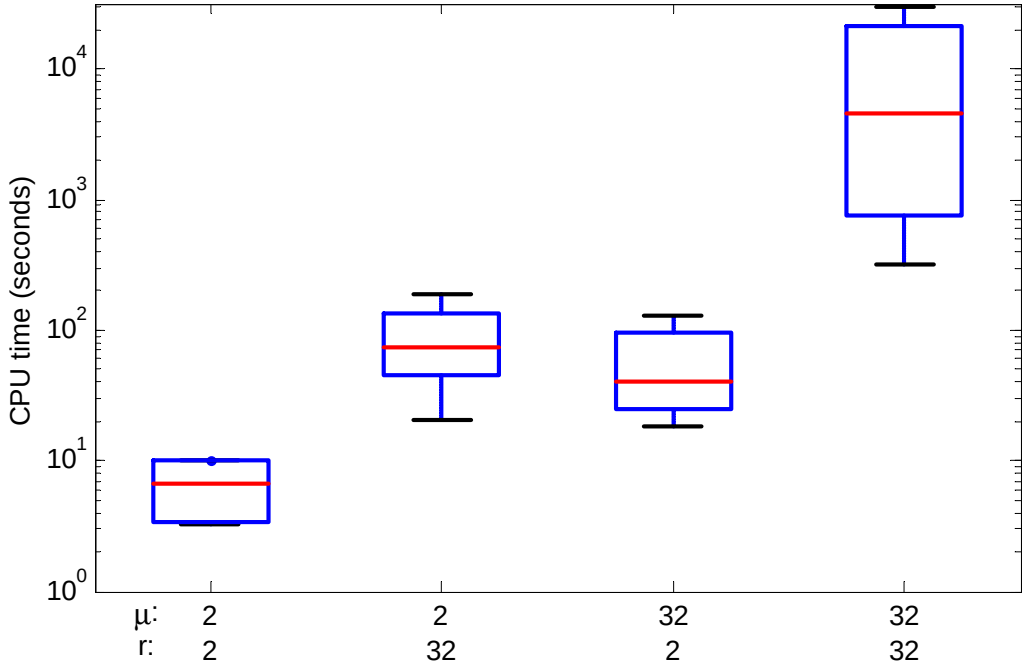

Figure 7: Distribution of calculation times for the exact method, as a function of the service rate μ and the average offered load r.

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 was slower than EXT were in a well-defined category – they all had a low service rate of 2 per hours, a low average offered load of 2, and long planning periods of 4 hours. Figure 8 demonstrates that for test problems in this category, RND required 46-66% longer computation times than EXT, whereas outside this category RND required only 11-57% 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.

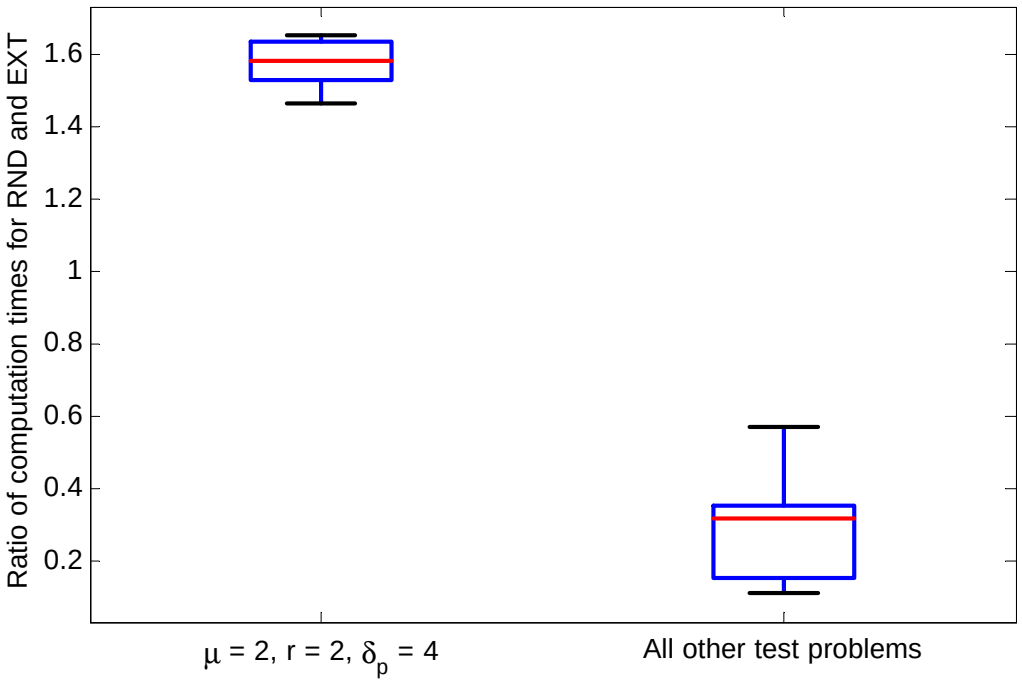

Figure 8: The ratio of computation times for RND and EXT, for test problems where $\mu = 2$/hour, $r = 2$, and $\delta_p = 4$ hours, compared to all other test problems.

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 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.

Overall, the performance of CLS was disappointing. For 21 of 128 test problems, CLS was dominated, i.e., it was both the slowest and the least accurate, as measured by time-average absolute error. No other method was dominated in this sense, for any test problem.

Given the speed advantage of the MOL approximation, we were interested in identifying domains where this method provided acceptable accuracy. The average utilization and the relative amplitude of the servers function seem to be the primary determinants of the accuracy of MOL. When both of these factors took their low values (0.5 and 0.1, respectively), the time-average absolute error for MOL was at most 3.5% and usually less than 1%. The influence of the relative amplitude of the servers function on the accuracy of MOL is consistent with the observations made earlier about systematic errors from the MOL approximation just after the number of servers changes.

6. Conclusions

Before conducting the experimental comparison of the six methods, we expected that they would fall into three categories: (1) methods that are slow but highly accurate, (2) methods that are fast but often inaccurate, and (3) an intermediate category with modest computation times and reasonably good accuracy. We expected the exact and randomization methods (EXT and RND) to fall in the first category; the infinite server based approximations (ISA and MOL) and the effective arrival rate approximation (EAR) to fall in the second category; and the closure approximation (CLS) to fall in the third category. Our expectations were generally confirmed, with some exceptions. In particular, the closure approximation performed worse than expected, being both the slowest and least accurate method for 21 of the 128 test problems.

Two caveats regarding the poor performance of the closure approximation are in order. First, the method is designed to approximate the mean number in system, not the service level. The literature on closure approximations shows them to be less accurate in approximating the variance and the second moment of the number in system (Rothkopf and Oren, 1979, Clark, 1981, Taaffe and Ong, 1987) and our experiments indicate that they are even less accurate in approximating the service level, which depends on tail probabilities for the number in system. Second, the closure approximation does have a computational advantage over the exact method for systems that require a large system capacity to approximate the system sufficiently well. We did not see such an advantage over the randomization method for the test problems we solved, but closure approximations may be faster than randomization for problems that are larger than the ones we solved.

In the “slow but accurate” category, we demonstrated that the randomization method provides close to the accuracy of the exact method at a fraction of the computational cost, even though application of the randomization method requires approximation of the arrival rate with a piecewise constant function. The speed advantage of randomization over the exact method is consistent, except for systems with low service rates, low arrival rates, and long planning periods, where the exact method is consistently faster than randomization.

In the “fast but sometimes inaccurate” category, the modified offered load approximation is, on average, the fastest and most accurate of the methods. The direct infinite server approximation may be useful in some applications to quickly generate upper bounds on the service level. The effective arrival rate approximation performed similarly to the modified offered load approximation.

Acknowledgments

This research was supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

References

Arkin, B. L., L. M. Leemis. 2000. Nonparametric estimation of the cumulative intensity function for a nonhomogeneous Poisson process from overlapping realizations. *Management Science* **46** 989–998.

Bookbinder, J. H. 1986. Multiple queues of aircraft under time-dependent conditions. *INFOR* **24** 280–288

Bookbinder, J. H., D. L. Martell. 1979. Time-dependent queuing approach to helicopter allocation for forest fire initial-attack. *INFOR* **17** 58–72

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

Cleveland B., J. Mayben. 1997. *Call Center Management on Fast Forward*. Call Center Press, Annapolis, MD.

de Souza e Silva, E. R. H. Gail. 2000. Transient solutions for Markov chains. In *Computational Probability*. W. K. Grassmann ed. Kluwer Academic Publishers, Boston, MA. 43–79.

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

Eick, S. G., W. A. Massey, W. Whitt. 1993b. The physics of the $M_t/G/\infty$ queue. *Operations Research* **41** 731–742.

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

Grassmann, W. 1989. Numerical solutions for Markovian event systems. In *Quantitative Methoden in den Wirtschaftswissenschaften*, Springer-Verlag, Berlin Heidelberg, 73–87.

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

Grassmann, W. K., ed. 2000. *Computational Probability*. Kluwer Academic Publishers, Boston, MA.

Green, L. V., P. J. Kolesar. 1991. The pointwise stationary approximation with nonstationary arrivals. *Management Science* **37** 84—97.

Green, L. V., P. J. Kolesar. 1997. The lagged PSA for estimating peak congestion in multiserver Markovian queues with periodic arrival rates. *Management Science* **41** 80–87.

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

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.

Gross, D., D. R. Miller. 1983. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research* **32** 343–361.

Heyman D. P., W. Whitt. 1984. The asymptotic behavior of queues with time-varying arrival rates. *Journal of Applied Probability* **21** 143–156.

Ingolfsson, A., E. Cabral. 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.

Jagerman, D. L. 1975. Nonstationary blocking in telephone traffic. *The Bell Systems Technical Journal* **54** 625–661.

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

Jennings, O. B., W. A. Massey 1997. A modified offered load approximation for nonstationary circuit switched networks. *Telecommunication Systems* **7** 229-251.

Jensen, A. 1953. Markoff chains as an aid in the study of Markoff processes. *Skandinavisk Aktuarietidskrift* **36** 87–91.

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

Kleinrock, L. 1974. *Queueing Systems, Volume 1: Theory*. John Wiley & Sons, New York, NY.

Koopman, B. O. 1972. Air terminal queues under time-dependent conditions. *Operations Research* **20** 1089–1114.

Leese, E. L., D. W. Boyd. 1966. Numerical methods of determining the transient behaviour of queues with variable arrival rates. *INFOR* **4** 1–13.

Massey, W. A., G. A. Parker, W. Whitt. 1996. Estimating the parameters of a nonhomogeneous Poisson process with linear rate. *Telecommunication Systems* **5** 361–388.

Massey, W. A., W. Whitt. 1997. Peak congestion in multi-server systems with slowly varying arrival rates. *Queueing Systems* **25** 157–172.

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 and Operations Research* **15** 19–36.

Rider, K.L. 1976. Simple approximation to average queue size in time-dependent M/M/1 queue. *Journal of the ACM* **23** 361–367.

Rothkopf, M.H., S.S. Oren 1979. A closure approximation for the nonstationary M/M/s queue. *Management Science* **25** 522–534.

Shampine, L. F., M. W. Reichelt. 1997. The Matlab ODE suite. *SIAM Journal on Scientific Computing* **18** 1–22.

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.

Thompson, G. M. 1993. Accounting for the multi-period impact of service when determining employee requirements for labor scheduling. *Journal of Operations Management* **11** 269–287.

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

Wragg, A. 1963. The solution of an infinite set of differential-difference equations occurring in polymerization and queueing problems. *Proceedings of the Cambridge Philosophical Society* **59** 117–124.