Combining Integer Programming and the Randomization Method to Schedule Employees

Armann Ingolfsson[1], Edgar Cabral, Fernanda Campello, and Xudong Wu
School of Business, University of Alberta, Edmonton, Alberta T6G 2R6, Canada

Abstract: We describe a method to find low cost shift schedules with a time-varying service level that is always above a specified minimum. Most previous approaches used a two-step procedure: (1) determine staffing requirements and (2) find a minimum cost schedule that provides the required staffing in every period. Approximations in the first step sometimes cause the two-step approach to find infeasible or suboptimal solutions. Our method iterates between a schedule evaluator and a schedule generator. The schedule evaluator calculates transient service levels using the randomization method and identifies infeasible intervals, where the service level is lower than desired. The schedule generator solves a series of integer programs to produce improved schedules, by adding constraints for every infeasible interval, in an attempt to eliminate infeasibility without eliminating the optimal solution. We present computational results for several test problems and discuss factors that make our approach more likely to outperform previous approaches.

Keywords: service operations management; employee scheduling; staffing requirements; nonstationary queues; randomization method; integer programming.

1 Introduction and Literature Review

The need to schedule employees is ubiquitous in the service sector. Bank branches, restaurants, retail stores, and airline check-in areas are but a few examples of organizations that need to schedule employees to match the demand for services—which is typically random and varies over time—to the supply of employees providing services. Call centers are perhaps the largest sector in need of employee scheduling. Modern call centers are complex organizations, both technologically and operationally. As this sector grows and matures, and as technology advances, there are increasing opportunities to employ models to improve operations (Gans et al., 2003). Labor is typically the largest cost for a call center (60-70% of total cost according to Gans et al., 2003) and therefore efficient employee scheduling provides substantial opportunities for productivity improvements.

Models for employee scheduling have a long history in the operations research literature. Edie's (1954) classic study of traffic delays at tollbooths used a combination of empirical

[1]Corresponding author, armann.ingolfsson@ualberta.ca, tel.: 1-780-492-7982, fax: 1-780-492-3325

analysis and formulas for stationary queueing systems to generate the staffing requirements needed to ensure a specified level of service. Soon after, Dantzig (1954), referring to Edie's work, showed how a linear integer program could find shift schedules that provide enough staffing to meet specified requirements in each planning period—such as the ones developed by Edie—at minimum cost.

A typical sequence of steps in scheduling employees is (Buffa et al. 1976):

Step 1: Forecast demand,

Step 2: Convert demand forecasts into staffing requirements,

Step 3: Schedule shifts optimally, and

Step 4: Rostering: Assign employees to shifts.

Current practice (e.g., Fukunaga et al., 2002) and most research on employee scheduling has followed this approach. Edie's paper demonstrated one way of performing Step 2. Dantzig's tour scheduling model addressed Step 3. Steps 1 and 4 are important, but outside the scope of this paper.

The papers by Edie and Dantzig were among the first in two distinct streams of research: one on how to set staffing requirements and the other on how to optimally schedule employees subject to staffing requirements.

Step 2 often uses formulas for stationary $M/M/s$ queueing systems to determine the smallest number of servers (employees) needed to provide a specified level of service (often expressed as the percent of customers who experience queue delay of less than some threshold time). Green et al. (2001) termed this the SIPP (Stationary Independent Period by Period) approach. A stream of research in queueing theory has developed better methods to determine employee requirements (for example, see Jennings et al., 1996, Green, et al., 2007, and Feldman et al., 2008).

Research on shift scheduling has developed efficient algorithms for special cases (e.g., Bartholdi et al., 1980), heuristics (e.g., Brusco and Jacobs, 1993), and reformulations to allow larger problems to be solved to optimality (e.g., Aykin, 1996).

We will refer to performing Steps 2 and 3 once in sequence as the "approximate approach." The main simplifying assumption in the approximate approach is that the staffing requirement for a period can be determined independent of staffing in prior periods. The extent to which this assumption is valid determines whether it is reasonable to decouple Steps 2 and 3. Kolesar et al. (1975) demonstrated that this approximation is not always warranted.

More recently, Green et al. (2001, 2003) conducted extensive experiments to investigate the *reliability* of the SIPP approach, which they defined as the number of half-hours during which the desired service level falls below a desired minimum. They demonstrated that the SIPP approach is, unfortunately, unreliable in many situations. They explored various ways of modifying the SIPP approach while retaining the simplicity of calculations with $M/M/s$ queueing formulas. The most promising of these heuristics was the *lag max* approach, which replaces the average arrival rate over a planning period with the maximum of the arrival rate function over that planning period, shifted forward by one average service time. The lag max approach extends the range of situations where SIPP generates reliable staffing requirements considerably, as Green et al. (2001, 2003) showed. When the approximate approach is justified (see Green et al., 2001, for guidelines), then it should be used, because it is simpler and faster than the approach we will describe. Our focus is on situations where the approximate approach (using either SIPP or lag max to generate staffing requirements) has been demonstrated to be unreliable. However, our approach can also result in cost savings in situations where the approximate approach is reliable, as we demonstrate.

Ingolfsson et al. (2002) described an approach to integrating Steps 2 and 3. This article presents an improved implementation of that method, involving two algorithmic components: a schedule generator and a schedule evaluator. The schedule generator searches for good schedules using exact or heuristic optimization. The schedule evaluator estimates the cost and service level of a schedule. In Ingolfsson et al. (2002), the schedule generator used a genetic algorithm, and the schedule evaluator used numeric integration of the forward differential equations for an $M(t)/M/s(t)$ system to evaluate the service level. In this paper, we use an integer programming heuristic to generate schedules and we use the randomization method (Grassmann, 1977) to compute service levels. These algorithmic improvements result in a substantial reduction in computation time, which has allowed us to perform computational experiments to generate insight into when decoupling Steps 2 and 3 is justified and when it is not. While the method does not guarantee optimality, it provides a good feasible solution and a lower bound on the minimum cost.

We define the service level at time t as the probability that the virtual waiting time is less than a maximum acceptable waiting time τ. Because we solve the forward differential equations, we can compute instantaneous service levels for as many time points as desired, and we define our optimization problem in terms of instantaneous service levels. In related research that uses simulation, as well as in practice, service levels are typically defined as averages over some time period, such as an hour. It is straightforward to modify our approach to conform with such definitions.

Thompson (1997) and Atlason et al. (2004, 2008) described other approaches to integrating Steps 2 and 3. Thompson generated staffing requirements using stationary $M/M/s$ formulas, with a heuristic adjustment (described in Thompson, 1993) for transient effects. As discussed in Ingolfsson et al. (2007), this heuristic is similar to Green et al.'s lag max approach and, therefore, shares its limitations. Thompson used slack and surplus decision variables for deviations above or below the requirement for each planning period, with coefficients derived from $M/M/s$ formulas to quantify the impact of these deviations on the service level. He solved the resulting model using Brusco and Jacobs' (1993) simulated annealing heuristic. Atlason et al. (2004) iterate between simulation (to evaluate service levels) and integer programming, with constraints being added to the integer program at each iteration based on approximate subgradients of the service level (estimated using simulation) as a function of staffing in each planning period. Atlason et al. (2008) improved on the approach in Atlason et al. (2004), using "pseudogradients" rather than subgradients. Our approach adds constraints at each iteration as well, but our constraints do not require evaluation of subgradients or pseudogradients of the service level. We compare the performance of our approach to that of Atlason et al. (2008) as part of our computational experiments and we discuss these related approaches further in the last section.

The paper is organized as follows. Section 2 presents an example to illustrate the main issues, Section 3 states the problem formally, Section 4 reviews the randomization method, Section 5 describes an initial parameter estimation procedure, Section 6 presents our integer programming heuristic, Section 7 outlines our computational experiments and results, and Section 8 concludes with observations on how our approach performs and how it could be generalized. An online supplement contains appendices with supplementary material and additional computational results.

2 Example

We will use the following example to illustrate potential shortcomings of performing Steps 2 and 3 sequentially and how these shortcomings can be addressed. A service system is open 12 hours each day and has a sinusoidally varying arrival rate with two daily peaks (Figure 1). The planning period (the shortest time interval over which staffing is constant) is 15 minutes. For Step 2, we approximate the time-varying arrival rate by its average over each 5-minute interval and we use $M/M/s$ queueing formulas (with a service rate of 2 customers per hour) to determine, for each planning period, the smallest number of servers needed to ensure that

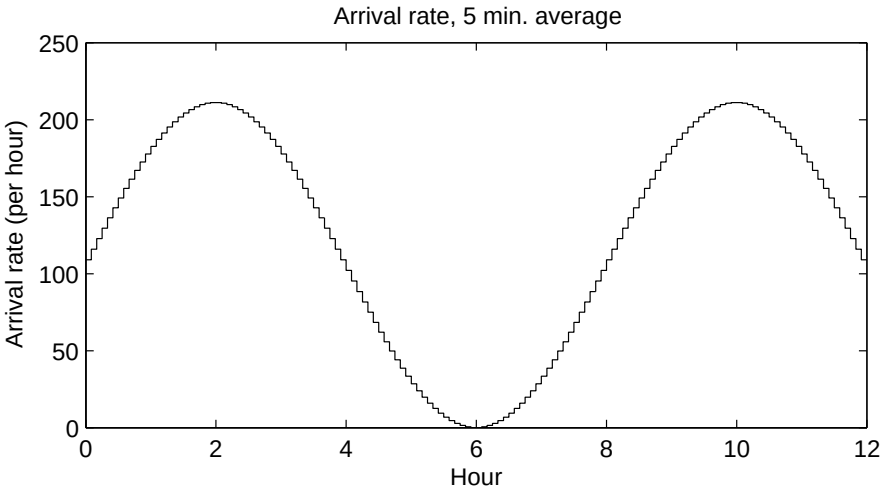

Figure 1: Arrival rate for example.

at least 80% of customers do not have to wait before commencing service (this is the SIPP approach).

Shifts are four, six, or eight hours long and can start at the beginning of any planning period that allows the shift to end before the facility closes (Appendix A describes the 243 possible shifts). Figure 2 (upper panel) shows the SIPP staffing requirements and the number of scheduled servers that minimizes the number of server-hours (by solving an integer program), while satisfying the staffing requirements. The lower panel shows the transient service level for this schedule (calculated with the randomization method). The service level goal is not met during a large fraction of the day.

Figure 2 illustrates how the SIPP approach can be unreliable, as discussed by Green et al. (2001, 2003). As they showed, the lag max approach often improves reliability. Figure 3 shows the results of using the lag max approach. The curves for the required and scheduled number of servers have shifted forward by 30 minutes (one average service time). The resulting service level is much improved and stays above the 80% target during all but 4 (of 48) planning periods. Not surprisingly (because a maximum rather than an average arrival rate is used), the improved service level comes at the expense of higher labor cost: the number of server-hours increased by 3.5%, from 954 (with SIPP staffing requirements) to 987 (with lag max staffing requirements).

However, one can do better. With the solution illustrated in Figure 4, the service level remains above 80% at all times, and the labor cost (947.5 server-hours) is 0.7% lower than with SIPP staffing requirements and 4% lower than with lag max staffing requirements.

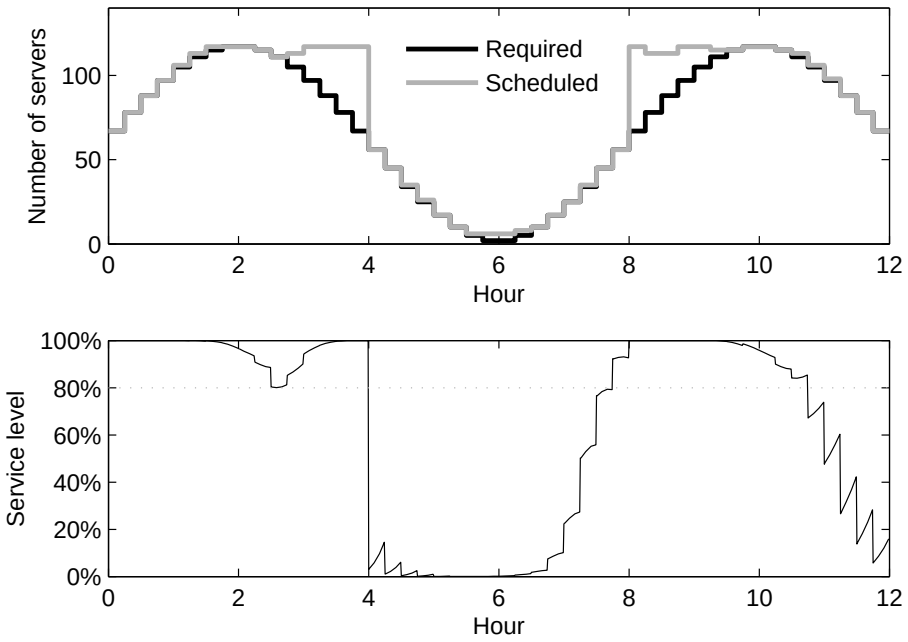

Figure 2: SIPP staffing requirements, scheduled number of servers, and resulting service level.

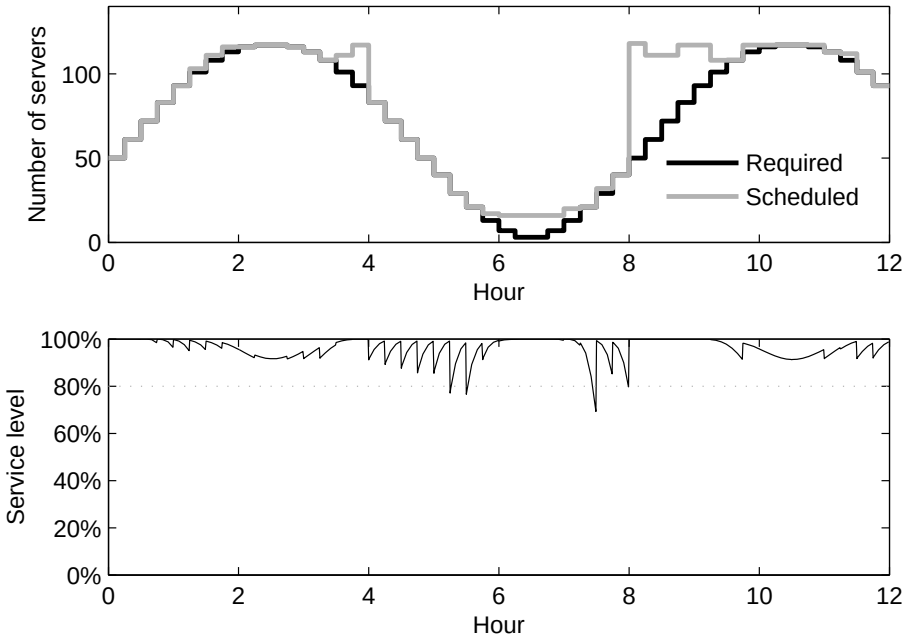

Figure 3: Lag max staffing requirements, scheduled number of servers, and resulting service level.

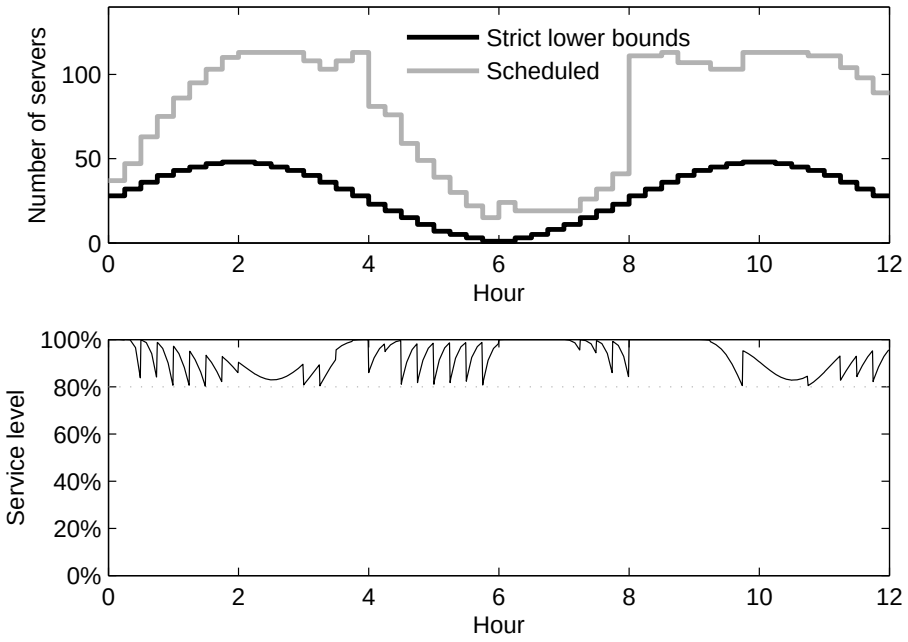

Figure 4: Strict lower bound staffing requirements, scheduled number of servers, and resulting service level for a solution generated with our approach.

Sections 5 and 6 describe the algorithm used to generate the solution shown in Figure 4. Like the SIPP and lag max approaches, the algorithm sets bounds on the staffing in each planning period that we refer to as *strict lower bounds*. In contrast to SIPP and lag max staffing requirements, which are intended to provide *sufficient* conditions for meeting the service level requirements, the strict lower bounds provide *necessary* conditions and are therefore usually smaller. We illustrate the calculation of the strict lower bounds later in this section and elaborate in Section 5.

To understand why the SIPP and lag max approaches do not generate reliable staffing requirements, it is instructive to recalculate the service level under certain changes in the scenario, to illustrate how the service level in one period depends on staffing in previous periods. For simplicity, we focus on the first two planning periods. First, schedule 8 servers for the first period ($s_1 = 8$). This is far too little capacity to keep up with the load (the SIPP approach recommends almost 70 servers for the first period) and Figure 5 (top panel) demonstrates how the transient service level drops quickly during the first period. At the end of the first period, a sizable queue will have built up. Given the first-period staffing, consider how many servers need to be scheduled for the second period to ensure that the service level remains above 80%. By varying the second-period staffing (s_2) and computing

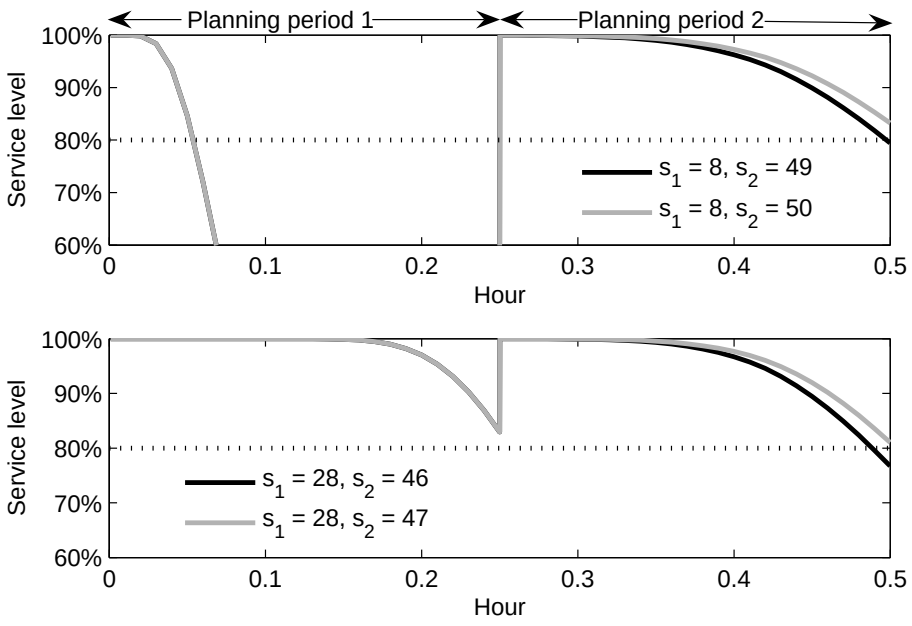

Figure 5: Service level during first two planning periods.

the service level, we discover that the minimum required staffing is 50 servers. Figure 5 (top panel) shows curves for $s_2 = 49$, which causes the service level to drop below 80% at the end of the second period, and $s_2 = 50$, which ensures that the service level constraint is satisfied.

Second, suppose we schedule 28 servers for the first period. Although still far below the SIPP staffing requirement, 28 servers is enough to keep the service level above 80% until the end of the first period. With this first-period staffing, there will be far less congestion at the beginning of the second period than with $s_1 = 8$. As a consequence, less second-period staffing is required to achieve the desired service level. As Figure 5 (bottom panel) illustrates, the minimum second-period staffing drops from 50 to 47 when the first-period staffing increases from 8 to 28. The reason is clear: if we understaff in the first period, then we need more servers in the second period. Conversely, first-period overstaffing may reduce the minimum second-period staffing. In general, it is not possible to determine staffing requirements for one period independent of staffing in other periods as with the SIPP and lag max approaches.

Observe that we can determine the minimum first-period staffing ($s_1 = 28$) to ensure that the service level constraint is met without considering staffing in any other periods, because the system is empty at the beginning of the first period. We can determine an *optimistic* estimate of the needed second-period staffing by pretending that the system is empty at the beginning of that period. We refer to the resulting staffing estimate as a

strict lower bound because staffing below this bound guarantees violation of the service level constraint. Figure 4 shows the strict lower bounds for all periods. Section 5 elaborates on the computation of the strict lower bounds.

3 Problem Description

We model the system as an $M(t)/M/s(t)$ queue, specified as follows. Customers arrive according to an inhomogeneous Poisson process with rate $\lambda(t)$ at time t, and are served by the first available server. Servers have identical capabilities and service durations are independent and exponentially distributed with mean $1/\mu$. The performance measure of primary interest is the service level $\mathrm{SL}(t)$, which we define to be $\Pr\{W(t) \leq \tau\}$, where $W(t)$ is the virtual waiting time at time t and τ is an "acceptable waiting time threshold." The final part of the $M(t)/M/s(t)$ specification is to describe what happens when servers are scheduled to leave. Like Green et al. (2001, 2003), we adopt a *pre-emptive service discipline*, to facilitate comparison of our computational results to theirs. Under this discipline, if a server is busy with a customer when scheduled to go off duty, then the customer will either be transferred to another server (if available) or return to the queue.

We consider facilities that operate for a finite time interval $[0, T]$ (e.g., retail stores that are open for part of the day) and also facilities that operate continuously (e.g., emergency services). For continuously operating systems, we assume a periodic arrival rate $\lambda(t)$ with period T (e.g., 24 hours or one week). We assume planning periods of length δ (typically 15 minutes, 30 minutes, or 1 hour). The scheduled number of servers $s(t)$ is constant during each planning period $((j-1)\delta, j\delta], j = 1, 2, \ldots, n$ and we assume $T = n\delta$.

We define the state variable $N(t)$ to be the number of customers in the $M(t)/M/s(t)$ system at epoch t and we let $\pi_k(t) = \Pr\{N(t) = k\}$ and $\pi(t) = (\pi_0(t), \pi_1(t), \ldots)$. Under the pre-emptive discipline, $N(t)$ evolves as a continuous time Markov chain, according to the well-known Chapman-Kolmogorov equations for an $M/M/s$ queue (with time arguments added for λ and s). To compute transient service levels, first consider epochs t that are more than τ time units before the end of the current planning period. At such epochs, if the queue is not empty, then the event $W(t) > \tau$, conditional on $N(t) = k \geq s(t)$, is the same as the event that there will be $k - s(t)$ or fewer service completions during $(t, t + \tau]$. This leads to the following formula for the service level: $\mathrm{SL}(t) = 1 - \sum_{k=s(t)}^{\infty} \sum_{j=0}^{k-s(t)} a^j e^{-a} \pi_k(t)/j!$ where $a = \mu \int_t^{t+\tau} s(u)du$—see Ingolfsson et al. (2002). Second, for epochs that are less than τ time units before the end of the current planning period, we need to account for the possibility that some of the customers that are currently in service will return to the queue during

$(t, t+\tau]$ because their server leaves. At these epochs, we use the algorithms described in Green and Soares (2007).

The scheduled number of servers $s(t)$ is determined by how many servers are assigned to permissible shifts that cover time t, excluding servers that are on break. The set of permissible shifts is I and each permissible shift $i \in I$ is represented by a binary row vector $a_i = (a_{ij}, j = 1, \ldots, n)$ of length n, with $a_{ij} = 1$ if shift i includes planning period j and zero otherwise, for $j = 1, 2, \ldots, n$. A schedule x is a column vector $(x_i, i \in I)$ with x_i indicating the number of servers scheduled to work shift i. Therefore, given a schedule x, $s(t)$ will equal $s_j(x) = \sum_{i \in I} a_{ij} x_i$ for $t \in ((j-1)\delta, j\delta]$. The cost of a schedule is $C(x) = \sum_{i \in I} c_i x_i$, where c_i is the variable labor cost per server assigned to shift i.

The functions $\lambda(t)$ and $s(t)$ and the service rate μ determine the transient state probabilities $(\pi_0(t), \ldots, \pi_K(t))$ where the system capacity K is chosen to approximate an infinite capacity system. For continuously operating systems, we compute the periodic stationary state probabilities, which satisfy $\pi_k(t) = \pi_k(t + pT)$ for all states k, epochs t, and integers p.

Our shift scheduling problem is:

$$\begin{aligned} \text{minimize} \quad & C(x) \\ \text{subject to} \quad & \mathrm{SL}(t) \geq \mathrm{SL}_{\min} \text{ for } t \in [0, T] \\ & \sum_{j=1}^{n} s_j(x) \geq \left\lceil \int_0^T \lambda(t) dt / \mu \right\rceil \\ & x_i \geq 0, \text{integer, for } i \in I \end{aligned} \tag{1}$$

The second constraint is included to ensure that the total scheduled server hours over the planning horizon are sufficient to handle the total amount of work. The service level $\mathrm{SL}(t)$ depends on the threshold τ and the number of servers but we suppress this dependence to simplify notation. The parameter $\mathrm{SL}_{\min}$ is the minimum desired service level.

Formulation (1) is in contrast to the following approximate formulation:

$$\begin{aligned} \text{minimize} \quad & C(x) \\ \text{subject to} \quad & s_j \geq b_j, j = 1, 2, \ldots, n \\ & x_i \geq 0, \text{integer, for } i \in I \end{aligned} \tag{2}$$

Here, the server requirement b_j is determined using a stationary $M/M/s_j$ queueing model with arrival rate $\lambda_j = \int_{(j-1)\delta}^{j\delta} \lambda(t) dt/\delta$, (the average arrival rate in planning period j). Specifically, if $W(\lambda, \mu, s)$ is the steady state waiting time before service commences in an $M/M/s$

queueing system, then $b_j = \min\{s_j : \Pr\{W(\lambda_j, \mu, s_j) \leq \tau\} \geq \mathrm{SL}_{\min}\}$. The generation of staffing requirements in this fashion is what Green et al. (2001, 2003) termed the SIPP approach. The staffing constraints $s_j \geq b_j$ in Problem (2) are intended to guarantee, to an adequate approximation, that the service level will remain at or above $\mathrm{SL}_{\min}$ for $t \in [0, T]$. Green et al.'s (2001, 2003) lag max modification to the SIPP approach changes the arrival rate used to determine the server requirement b_j to $\lambda_j = \max\{\lambda(t) : t \in [(j-1)\delta - 1/\mu, j\delta - 1/\mu]\}$, i.e., shift the arrival rate forward by one average service time and take the maximum rather than the average over the planning period.

Problem (2), with or without the lag max modification, is an approximation to Problem (1) for two reasons:

1. The service level during period j depends not only on the staffing s_j during that period but also on staffing in preceding periods. If $\tau > 0$, then the service level will also depend on staffing in subsequent periods.

2. The service level varies during a period. It may not be well approximated by the limiting value obtained by assuming that the arrival rate and number of servers are constant and continue indefinitely.

For these reasons, the approximate approach, which solves Problem (2), often results in either infeasible or suboptimal solutions to Problem (1).

4 Randomization Method

Also known as the uniformization method, the randomization method (described in Grassmann 1977) provides a computationally stable and efficient method to calculate transient state probabilities for a homogeneous, discrete-state, continuous time Markov chain. The computational cost of this method is significantly lower than for the Runge-Kutta numerical integration method (Grassmann 1977) used in Ingolfsson et al. (2002).

The randomization method only applies to homogenous processes, but in our problem, the arrival rate may vary continuously. Therefore, we approximate the arrival rate function $\lambda(t)$ by a piecewise constant function $\tilde{\lambda}(t) = \tilde{\lambda}_l \equiv \int_{(l-1)\delta_{\mathrm{calc}}}^{l\delta_{\mathrm{calc}}} \lambda(s)ds/\delta_{\mathrm{calc}}$ for $t \in ((l-1)\delta_{\mathrm{calc}}, l\delta_{\mathrm{calc}}]$, were, δ_{calc} is the "calculation period." We will always use a calculation period that is shorter than the planning period ($\delta_{\mathrm{calc}} \leq \delta$) and which divides evenly into a planning period ($\delta \bmod \delta_{\mathrm{calc}} = 0$). We apply the randomization method in each calculation period, during which the parameters of the $M(t)/M/s(t)$ queueing system remain constant. The state

probabilities at the end of one calculation period are used as the initial state probabilities at the beginning of the next. Importantly, this procedure calculates transient probabilities throughout each calculation period—no steady state approximation is used.

We approximate the infinite capacity system with a finite capacity $M(t)/M/s(t)/K$ system, with the system capacity K chosen large enough to ensure that $\pi_K(t) < \epsilon_K$ for all t, with $\epsilon_K = 10^{-4}$. For continuously operating systems, we calculate the state probabilities for p periods, until $|\pi_k(t + (p-1)T) - \pi_k(t + pT)|$ is less than a specified tolerance $\epsilon_{SS} = 10^{-2}$ for all $t \in [0, T]$ and for all $k = 0, 1, \ldots, K$.

Ingolfsson et al. (2007) demonstrated that the randomization method remains highly accurate compared to the Runge-Kutta method when applied to $M(t)/M/s(t)$ systems and that most of the computational speed advantage of the randomization method over the Runge-Kutta method for time homogenous systems is retained for $M(t)/M/s(t)$ systems. See Ingolfsson et al. (2007) for details of our implementation of the randomization method.

5 Parameter Estimation Procedure

The first step in our procedure uses the randomization method to calculate *strict lower bounds* $b_j^{\min}$ on the staffing levels s_j in each period and to estimate how the service level in each period increases as a function of increased staffing.

The strict lower bounds have a different meaning and are calculated differently from the SIPP (or lag max) staffing requirements used in Problem (2). The strict lower bound $b_j^{\min}$ is the minimum number of servers needed in period j to ensure the service level constraint is satisfied during that period, assuming that the system is empty at the beginning of period j and that all waiting customers will enter service at the beginning of period $j + 1$. Both assumptions are optimistic and therefore the strict lower bound specifies a necessary (but not sufficient) condition on the number of servers. To compute the strict lower bound for period j, one pretends that the system is empty at the beginning of the period (by setting $\pi((j-1)\delta) = (1, 0, \ldots))$, chooses a staffing level s_j, and computes the resulting transient service level $\mathrm{SL}(t)$ for $t \in ((j-1)\delta, j\delta]$. If $\mathrm{SL}(t)$ drops below $\mathrm{SL}_{\min}$ during the period, then one increases s_j and repeats the calculation, until one finds the smallest s_j such that $\mathrm{SL}(t) \geq \mathrm{SL}_{\min}$ for $t \in ((j-1)\delta, j\delta]$. In performing these calculations, one needs to set $s_{j+1} = \infty$ (in practice, a very large number), to remove dependence of the service level in period j on staffing in period $j + 1$. Referring back to the example in Section 2, the strict lower bound for planning period 1 was 28. As Figure 5 illustrates, if the system starts empty, with 28 scheduled servers in period 1, then the service level stays above 80% throughout the

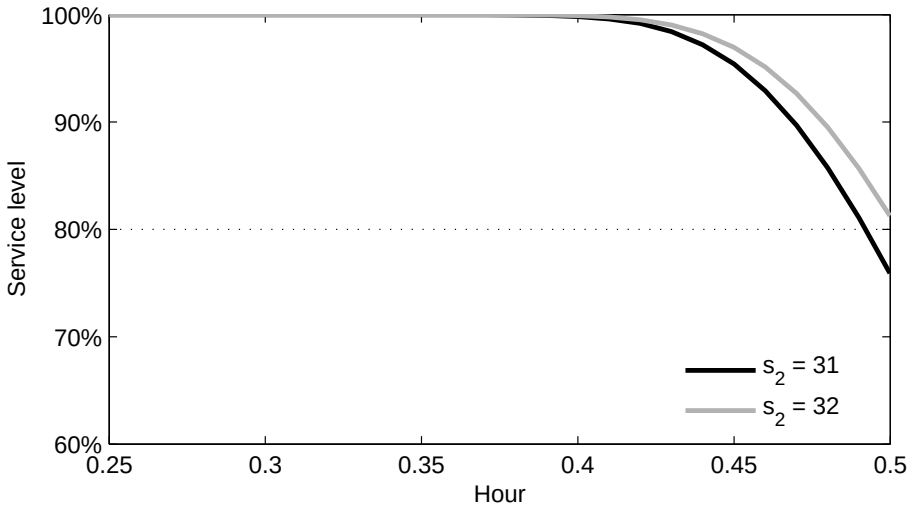

Figure 6: Illustration of strict lower bound calculation for period 2.

first period. With 27 servers in the first period (not shown), the service level drops below 80% before the end of the period. To illustrate the calculation of the strict lower bound for the second period, Figure 6 shows the transient service level during that period if the system is empty at the beginning of the period and staffing is either 31 or 32. The lowest staffing level that achieves the desired service level throughout the period is 32, so $b_2^{\min} = 32$.

If s_j is smaller than $b_j^{\min}$, then the service level constraint is certain to be violated during period j, regardless of staffing in other periods. If the system is not empty at the beginning of the period or if the number of servers is reduced at the end of the period, then s_j may need to be strictly greater than $b_j^{\min}$ for the service level constraint to be satisfied during period j; hence the term strict lower bound. We use bisection search to compute the strict lower bounds, using the randomization method at each iteration to calculate transient service levels.

If SIPP server requirements truly represented sufficient conditions that guaranteed that the service level was met, then these server requirements would always be larger than or equal to the corresponding strict lower bounds. This is usually the case. There are exceptions, however—particularly when the arrival rate changes considerably during a planning period. Such exceptions are more common with the lag max approach. For example, in Figures 3 and 4, the lag max staffing requirement for the 6:30-6:45 planning period is 3 servers but the strict lower bound is 5 servers.

Our estimate of how the minimum service level in a planning period increases with increased staffing in that planning period is based on an approximation that we describe in

a moment. To motivate the approximation, consider how the service level in a stationary $M/M/s$ system changes when s increases. For $s > \lambda/\mu$, the service level is concave increasing in s and approaches 100% in the limit (from results in Jagers and Van Doorn 1991). Therefore, it seems plausible to expect $\min\{\mathrm{SL}(t) : t \in ((j-1)\delta, j\delta]\}$ to be concave increasing in s_j and to approach 100% in the limit, for s_j above some inflection point. We assume, for our approximation, that this inflection point is at or below the strict lower bound $b_j^{\min}$.

We approximate $\min\{\mathrm{SL}(t) : t \in ((j-1)\delta, j\delta]\}$ with an exponential function of s_j, as follows. Denote the minimum service level during planning period j when $s_j = b_j^{\min}$ and $s_{j-1} = s_{j+1} = \infty$ by $\mathrm{SL}_{j,0}$. We set $s_j = b_j^{\min} + 1$, continue to assume that $s_{j-1} = s_{j+1} = \infty$ and calculate the minimum service level during the period, which we denote $\mathrm{SL}_{j,1}$. Suppose that with a particular schedule, staffing in planning period j is s_j, and the minimum service level during planning period j is $\mathrm{SL}_{j,\mathrm{current}} < \mathrm{SL}_{\min}$. If staffing increases to $s_j + k$, then we approximate the minimum service level during planning period j as $\mathrm{SL}_{j,\mathrm{current}} + (1 - \mathrm{SL}_{j,\mathrm{current}})(1 - \exp(-d_j k))$. We estimate the unknown constant d_j by solving $\mathrm{SL}_{j,0} + (1 - \mathrm{SL}_{j,0})(1 - \exp(-d_j)) = \mathrm{SL}_{j,1}$, resulting in $d_j = -\ln\left((1 - \mathrm{SL}_{j,1})/(1 - \mathrm{SL}_{j,0})\right)$. Then we use d_j to estimate the smallest number $k_j^\star$ of additional servers needed to bring the minimum service level in planning period j above $\mathrm{SL}_{\min}$:

$$\begin{aligned} & \mathrm{SL}_{j,\mathrm{current}} + (1 - \mathrm{SL}_{j,\mathrm{current}})(1 - \exp(-d_j k_j)) \geq \mathrm{SL}_{\min} \\ \Leftrightarrow \quad & k_j \geq -\ln\left((1 - \mathrm{SL}_{\min})/(1 - \mathrm{SL}_{j,\mathrm{current}})\right)/d_j \\ \Rightarrow \quad & k_j^\star = \lceil -\ln\left((1 - \mathrm{SL}_{\min})/(1 - \mathrm{SL}_{j,\mathrm{current}})\right)/d_j \rceil \end{aligned} \tag{3}$$

These estimates are used in the integer programming heuristic to generate constraints. In recognition that the procedure for estimating the required staffing increase involves approximation, we introduce (in the next section) an algorithm parameter β to "scale down" the required increase, in order to reduce the risk of adding constraints that eliminate the optimal solution.

6 Integer Programming Heuristic

Our heuristic begins by using the approximate approach, with SIPP and lag max staffing requirements, to obtain two staffing solutions. We use the randomization method to calculate the transient service level resulting from these two solutions. If one of these solutions satisfies the service level constraint throughout the interval $[0, T]$, then we store it as x_{approx} and use its cost $C(x_{\mathrm{approx}})$ as an upper bound. (If more than one approximate solution satisfies the

service level constraint then we use the least costly solution.)

Next, we solve the following integer program:

$$\begin{aligned} \text{minimize} \quad & C(x) \\ \text{subject to} \quad & s_j(x) \geq b_j^{\min} \text{ for } j = 1, 2, \ldots, n \\ & \sum_{j=1}^{n} s_j(x) \geq \left\lceil \int_0^T \lambda(t)dt/\mu \right\rceil \\ & x_i \geq 0, \text{integer, for } i \in I \end{aligned} \tag{4}$$

This integer program is linear, because $s_j(x)$ is linear in x (see Section 3). We note that, instead of x_i being the number of servers scheduled to work shift i, the vector x could be any set of decision variables used to (fully or partially) specify a schedule, as long as $s_j(x)$ is a linear function of x, to ensure that the above integer program is linear. For example, x could include "break variables" as in Aykin (1996).

The definition of the strict lower bounds $b_j^{\min}$ implies that (4) is a relaxation of Problem (1) and therefore, its optimal cost is a lower bound on the optimal cost for Problem (1). Once we have this initial solution, we begin iterating between calculating service levels for a solution and obtaining new solutions by solving the integer program again, with some constraints added and some deleted.

An iteration begins by calculating the service level $\text{SL}(t)$ for $t \in [0, T]$ for a solution x_{current} (we calculate the service level at 5-minute intervals). Then we enumerate all *infeasible planning periods*, defined as ones where the service level drops below $\text{SL}_{\min}$. Let $(l, l+1, \ldots, u)$ be an *infeasible interval* of consecutive infeasible periods. We estimate the additional staffing (measured in server-planning periods) needed to bring the service level above $\text{SL}_{\min}$ as $\sum_{j=l}^{u} k_j^\star$, using equation (3) to compute $k_j^\star$, and add the following constraint to the integer program, for all infeasible intervals:

$$\sum_{j=l}^{u} s_j(x) \geq \sum_{j=l}^{u} s_j(x_{\text{current}}) + \max\left[1, \left\lfloor \beta \sum_{j=l}^{u} k_j^\star \right\rfloor\right] \tag{5}$$

(β is a parameter between 0 and 1 whose value and role is explained later.)

When a new constraint $a'x \geq b'$ is added, we eliminate any existing constraint $a''x \geq b''$ that is dominated by the new constraint (that is, if $a'' \geq a'$ (elementwise) and $b'' \leq b'$). Every integer program that we solve can be represented as follows:

$$\text{minimize} \quad C(x)$$

$$\begin{aligned} \text{subject to} \quad & \sum_{i \in I} a_{ij} x_i \geq b_j^{\min} \text{ for } j = 1, 2, \ldots, n \\ & \sum_{j=l_v}^{u_v} \sum_{i \in I} a_{ij} x_i \geq b_v, \text{ for } v \in V \\ & x_i \geq 0, \text{integer, for } i \in I \end{aligned} \tag{6}$$

where V is the set of constraints of the form (5) that have been added so far (note that the second constraint in (4) is of the same form as (5)). Formulation (6) includes, first, period-by-period staffing constraints, just as Problem (2) does. Importantly, the second set of constraints has a different structure. Instead of constraining minimum staffing in a period, these constraints specify that staffing in an *interval* must be increased by a certain amount. The integer programming solver distributes the required staffing increase over the interval so as to minimize cost. The structure of the constraints in (6) suggests that our approach should be better able to exploit scheduling flexibility than the approximate approach.

The parameter $\beta \in [0, 1]$ in expression (5) influences both solution quality and solution speed. A low β makes it unlikely that we add a constraint that eliminate the optimal solution to Problem (1), and makes our procedure more likely return the optimal solution. However, low values for β will also lead to more iterations. We experimented with values of β from 0.3 to 1.0 (see Appendix B) and these hypotheses were generally confirmed, although solution quality appeared not to be particularly sensitive to β.

At each iteration, before solving the integer program, we solve its LP relaxation to obtain the solution x_{LP}. If the costs c_i are integer (as in our computational experiments), then we tighten the formulation by adding the valid inequality $C(x) \geq \lceil C(x_{LP}) \rceil$. We also use the LP solution to construct a solution x_{feas} by rounding all components of x_{LP} up to the nearest integer. This solution is feasible for (6), because all constraint coefficients in (6) are non-negative and all constraints are of the "greater than or equal to" type. The resulting upper bound $C(x_{\text{feas}})$ on the cost of the integer program helps when using a branch-and-bound solver. If some of the constraint coefficients are negative, as in Aykin's (1996) formulation, then a more sophisticated rounding heuristic is needed to generate a feasible solution; for example, see Aykin (1998).

After adding and deleting constraints in this fashion, we solve the integer program again. We iterate between calculating service levels and solving integer programs until either all planning periods are feasible, or the cost of the integer program exceeds $C(x_{\text{approx}})$. The final solution is guaranteed to be feasible to Problem (1), but it may not be optimal.

Other heuristics are possible for the schedule generator. For example, Atlason et al.

(2004) add constraints based on estimated subgradients of the service level function with respect to staffing in each planning period. Evaluating subgradients is considerably more computationally expensive than just the service level resulting from the current schedule, which is all that our procedure requires. Atlason et al.'s procedure has the advantage that it converges to the optimal solution, but only if the service level function is concave in the staffing vector $(s_1, \ldots, s_n)$. The service level for Problem (1) is not a concave function of staffing, if one considers all possible staffing levels—rather, the service level increases from zero to one according to an S-shaped curve as staffing increases. Atlason et al. (2008) use a different heuristic, which is optimal if the service level is pseudoconcave in the staffing levels; an assumption that appears realistic. Instead of subgradients, this heuristic requires "pseudogradients," which are equally expensive to compute. We compare the performance of our method to this method in the next section.

7 Computational Experiments

We have compared the performance of our method to the approximate approach and to the Analytic Center Cutting Plane Method (ACCPM) from Atlason et al. (2008) on a variety of test problems. Appendix C outlines how we applied ACCPM to our model. All experiments that we report on in this section were run on a 1.66 GHz Windows PC with 0.99 Gb of RAM. The randomization method was coded in Matlab, and the integer programs were solved using the Tomlab optimization environment (Holmström, 1999), using CPLEX 11 to solve all linear and integer programs and using SNOPT 7.2 to solve the nonlinear programs required by ACCPM. When solving integer programs, we set the maximum duality gap to 0.5% and the maximum number of simplex iterations to 5,000, except when solving integer programs to compute ACCPM lower bounds, where we left the maximum duality gap at its default value. The parameter β was set to 0.7, based on experimentation that is described in Appendix B. We set the calculation period δ_{calc} to 5 minutes in all of the experiments, as was done in Ingolfsson et al. (2007).

In choosing test problems, our aim was to identify situations where our approach is most likely to be beneficial. Green et al. (2001, 2003) demonstrated that the lag max modification to the SIPP approach improves reliability considerably in many situations. For our first set of 27 test problems we focused on situations where the lag max approach is least reliable, namely, situations when the service facility has limited operating hours, large variation in arrival rate, long average service times, and short planning periods (see Green et al., 2003).

We fixed the operating hours to $T = 12$ hours and used a sinusoidal arrival rate function $\lambda(t) = \lambda\{1 + \gamma \sin(\pi t/4)\}$ to specify the inhomogeneous Poisson arrival process, similarly to Green et al. (2003). The parameter γ is the "relative amplitude" and the parameter λ determines (but is not equal to) the average arrival rate. This arrival rate function has two peaks, at $t = 2$ and 10, corresponding to a morning and afternoon peak. The average arrival rate over the 12 hours is $\bar{\lambda} = \lambda(1 + 2\gamma/(3\pi))$. We specified our 27 test problems by varying the service rate ($\mu = 1$, 2, or 4 customers per hour), the average offered load $r = \bar{\lambda}/\mu$ (16, 32, or 64), and the length of a planning period δ (0.25, 0.5, or 1 hours). We fixed the relative amplitude at $\gamma = 1$. The example in Section 2 is one of the problems in this set, corresponding to $\mu = 2, r = 64$, and $\delta = 0.25$.

To facilitate comparison with results from Green et al. (2003), we used a threshold $\tau = 0$—the service level is equal, in this case, to the probability of zero queue delay. The minimum service level was set to $\mathrm{SL}_{\min} = 80\%$. We also performed the same experiments with a threshold $\tau = 20$ seconds (a common target for call centers, Gans et al., 2003) and obtained very similar numerical results. We used a set of shifts that we believe are realistic, specified as follows. Shifts can be 4, 6, or 8 hours in length, including breaks. The number of breaks varies from 1 to 3 depending on the shift length and the planning period length and the timing of breaks is flexible. Shifts can start in any planning period, as long as the shift ends by time T. The number of permissible shifts varied from 45 for $\delta = 1$ to 243 for $\delta = 0.25$. Appendix A provides further details on the set of permissible shifts. We set the cost of each shift equal to the number of hours of work (excluding breaks) contained in it. Thus, the cost of a schedule equals the total number of scheduled person-hours.

We compared our method both to the approximate approach and to the ACCPM approach. We begin by describing our comparisons to the approximate approach, where we solved each test problem three times: using the approximate approach with SIPP and lag max staffing requirements, to obtain solutions x_1 and x_2, and using our method to obtain a solution x_3. If the minimum service level $\mathrm{SL}'_{\min}$ (evaluated using the randomization method) of the lag max solution was less than the desired minimum $\mathrm{SL}_{\min}$, then we used our method with the constraint $\mathrm{SL}(t) \geq \mathrm{SL}'_{\min}$ to obtain a fourth solution x_4.

Table 1 shows the cost and minimum service level of the solutions obtained with the SIPP, lag max, and our approach for each test problem. For the approximate SIPP and lag max approaches, we also show the fraction of 5-minute intervals during which the service level drops below 80%. We calculated the percent cost reduction achieved by our approach compared to the lag max approach as $1 - C(x_3)/C(x_2)$ (column (7) in Table 1). When the approximate lag max approach resulted in an infeasible solution, we used $1 - C(x_4)/C(x_2)$

to calculate an alternative measure of percent cost reduction (column (10) in Table 1).

Table 1 reveals the following trends:

- The approximate SIPP approach is highly unreliable for these test problems. The lag max modification is more reliable, but it fails to produce a solution that satisfies the service level constraint for 13 of the 27 problems. This is consistent with results in Green et al. (2003).

- Our approach resulted in lower cost than the approximate lag max approach for all problems, even ones where the lag max solution was infeasible. The cost savings ranged from 1.4% to 10.4%, averaging 5.5%.

- The cost savings are even greater when we adjust the percent savings calculation for test problems where the lag max solution was infeasible, with cost reductions ranging from 1.4% to 30.2%, averaging 11.0%.

- Solution cost always decreases with decreased planning period length when μ and r are fixed, for both the approximate lag max approach and our approach. This is what one would hope happens, because decreased planning period length implies additional scheduling flexibility, so all else being equal, the solution cost should decrease.

Table 2 shows computation times for the approximate approach and our approach. The computation time for our approach is divided into the time for the initial parameter estimation procedure and the time for iterations between solving integer programs and computing service levels. On average, 33% of the computation time was spent on the parameter estimation procedure. The total computation time per problem ranged from 0.26 to 5.31 minutes.

We found our method's computation times for these test problems to be highly predictable as a function of the experimental factors μ, r, and δ. The parameter estimation computation times in Table 2 were well approximated ($R^2 = 0.97$) by $0.0024r^{1.14}\delta^{-1.17}\mu^{0.20}$. A shorter planning period length implies more planning periods and more parameters to estimate, so the inverse relationship with δ is not surprising. A higher average offered load r implies that the randomization method will need to solve larger systems, and therefore the parameter estimation time for each planning period increases. We analyzed the number of iterations and the average time per iteration separately and found the number of iterations to be almost independent of δ, while the average time per iteration was almost independent of μ. Combining these two analyses resulted in the following approximation for the iterations computation time ($R^2 = 0.88$): $0.0067r^{1.26}\delta^{-0.63}\mu^{-0.29}$. The computation time per iteration

Table 1: Cost and service level results.

			Approximate SIPP approach			Approximate lag max approach			Our method			Our method, with SL_{min}=min(80%, (3a))		
			(1a)	(1b)	(2)	(3a)	(3b)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
			min	fraction		min	fraction		min		Cost	min		Cost
μ	r	δ	SL(t)	$< 80\%$	$C(x_1)$	SL(t)	$< 80\%$	$C(x_2)$	SL(t)	$C(x_3)$	savings	SL(t)	$C(x_4)$	savings
1	16	0.25	0.8%	39.6%	273	36.3%	15.8%	280.5	80.6%	256	8.7%	36.5%	207	26.2%
1	16	0.5	0.7%	37.5%	268	47.5%	8.3%	289.5	81.0%	263	9.2%	47.7%	222	23.3%
1	16	1	0.6%	47.6%	258	48.2%	4.2%	297	80.6%	266	10.4%	49.1%	232	21.9%
1	32	0.25	0.0%	41.7%	503	15.1%	16.7%	519	80.3%	470	9.4%	15.7%	377.5	27.3%
1	32	0.5	0.0%	41.7%	500	29.7%	8.3%	532.5	80.8%	486.5	8.6%	31.0%	410.5	22.9%
1	32	1	0.0%	44.0%	486	30.2%	5.4%	550	80.3%	494	10.2%	34.5%	427	22.4%
1	64	0.25	0.0%	41.7%	954	1.7%	17.1%	983.5	80.4%	894.5	9.0%	1.8%	686.5	30.2%
1	64	0.5	0.0%	41.7%	946	9.9%	8.3%	1012	81.2%	924.5	8.6%	10.6%	749.5	25.9%
1	64	1	0.0%	41.7%	916	10.2%	5.4%	1047	80.4%	943	9.9%	10.5%	768	26.6%
2	16	0.25	8.2%	34.2%	273	81.4%	0.0%	281.5	80.6%	266	5.5%			
2	16	0.5	6.0%	37.5%	268	83.4%	0.0%	289.5	81.2%	273	5.7%			
2	16	1	4.6%	38.7%	258	83.5%	0.0%	304	80.1%	282	7.2%			
2	32	0.25	0.6%	38.8%	503	77.9%	0.4%	519.5	81.9%	506	2.6%	78.0%	496	4.5%
2	32	0.5	0.4%	44.4%	500	78.4%	1.4%	534.5	80.2%	511	4.4%	78.5%	505.5	5.4%
2	32	1	0.1%	38.1%	486	78.5%	0.6%	559	80.6%	536	4.1%	79.2%	528	5.5%
2	64	0.25	0.0%	42.5%	954	76.6%	1.7%	987	80.0%	947.5	4.0%	77.0%	940	4.8%
2	64	0.5	0.0%	44.4%	946	80.4%	0.0%	1016	80.1%	965	5.0%			
2	64	1	0.0%	43.5%	916	80.5%	0.0%	1066	80.4%	1016	4.7%			
4	16	0.25	31.3%	28.3%	273	83.6%	0.0%	282	80.9%	272.5	3.4%			
4	16	0.5	22.1%	29.2%	268	84.0%	0.0%	289.5	80.4%	278.5	3.8%			
4	16	1	13.2%	42.3%	258	84.0%	0.0%	301	80.7%	292	3.0%			
4	32	0.25	9.5%	35.4%	503	83.1%	0.0%	518	80.7%	510.5	1.4%			
4	32	0.5	5.5%	36.1%	500	84.2%	0.0%	533	80.0%	520.5	2.3%			
4	32	1	2.1%	42.3%	486	84.2%	0.0%	559	80.1%	545	2.5%			
4	64	0.25	1.0%	38.3%	954	81.6%	0.0%	984	80.3%	970	1.4%			
4	64	0.5	0.2%	38.9%	946	82.2%	0.0%	1015.5	80.0%	993.5	2.2%			
4	64	1	0.0%	52.4%	916	85.0%	0.0%	1064	80.8%	1049	1.4%			

Table 2: Computation time (minutes).

					Our method			
			Approximate	Lag max	Parameter			Number of
μ	r	δ	approach	approach	estimation	Iterations	Total	Iterations
1	16	0.25	0.02	0.02	0.33	0.59	0.92	22
1	16	0.5	0.01	0.01	0.11	0.29	0.41	25
1	16	1	0.01	0.01	0.06	0.23	0.29	21
1	32	0.25	0.03	0.03	0.55	1.09	1.64	30
1	32	0.5	0.02	0.02	0.22	0.55	0.77	33
1	32	1	0.01	0.02	0.12	0.65	0.76	38
1	64	0.25	0.06	0.06	1.91	3.40	5.31	44
1	64	0.5	0.03	0.03	0.61	1.62	2.23	41
1	64	1	0.03	0.03	0.26	1.93	2.19	60
2	16	0.25	0.02	0.02	0.34	0.59	0.93	22
2	16	0.5	0.01	0.01	0.13	0.21	0.34	14
2	16	1	0.01	0.01	0.08	0.18	0.26	14
2	32	0.25	0.04	0.04	0.68	1.56	2.23	37
2	32	0.5	0.02	0.02	0.28	0.49	0.76	22
2	32	1	0.02	0.02	0.14	0.55	0.70	29
2	64	0.25	0.07	0.06	1.91	3.10	5.01	35
2	64	0.5	0.03	0.03	0.67	1.13	1.81	26
2	64	1	0.03	0.03	0.30	1.39	1.70	39
4	16	0.25	0.03	0.03	0.37	0.48	0.85	16
4	16	0.5	0.01	0.01	0.17	0.18	0.35	10
4	16	1	0.01	0.01	0.09	0.21	0.30	14
4	32	0.25	0.04	0.04	0.73	0.95	1.67	20
4	32	0.5	0.02	0.02	0.33	0.34	0.67	12
4	32	1	0.02	0.02	0.16	0.37	0.53	15
4	64	0.25	0.07	0.07	2.00	2.89	4.89	30
4	64	0.5	0.04	0.05	0.75	0.76	1.52	14
4	64	1	0.04	0.04	0.39	0.93	1.32	19

should decrease with increased planning period length, because shorter planning periods imply a larger number of permissible shifts (in our experimental design, and likely in practice as well) and hence a larger integer program. The approximation equation confirms this expectation. We caution that it is not safe to extrapolate the approximation equations to situations beyond the test problems we considered, particularly ones where the factors that we kept fixed (for example, limited facility operating hours) vary.

To compare our method to ACCPM, we ran that algorithm two times for each of the 27 problems. First, we gave ACCPM the same computational budget as our method and recorded the best solution x_5 found by ACCPM. Second, we ran ACCPM until it found a solution x_6 for which the lower and upper bounds maintained by the method were within 1% of each other. Table 3 shows the minimum service level, cost, and computation time required to find solutions x_3 (by our method), x_5, and x_6. We also show optimality gaps for both methods, computed as $C(x_3)/\text{LB} - 1$ for our method and $C(x_6)/\text{LB} - 1$ for ACCPM, where LB is the best lower bound found by ACCPM.

Comparing our method to ACCPM, we observe the following:

- When given the same computational budget as our method, ACCPM found solutions that were 11% to 88% more costly than those found by our method.
- Running ACCPM until it found a solution guaranteed to be within 1% of optimality typically required about 20 times as much computation as our method.
- Comparing the best solutions found by the two methods, ACCPM found a better solution for 20 test problems, our method found a better solution for 3 problems, and the cost was the same for 4 test problems. The average optimality gap was 0.76% (maximum = 0.99%) for ACCPM and 1.19% (maximum = 2.53%) for our method.

ACCPM has the important advantage of providing good lower bounds on the optimal cost of Problem (1). Unfortunately, based on our experiments, it appears that computing the value of the lower bounds is very time consuming. Because of this, for two problem instances (indicated in Table 3), we were unable to find solutions that were guaranteed to be within 1% of optimality within 5 hours of computation time. Computing the lower bounds involves solving integer programs, and Atlason et al. (2008) mention that one could solve linear programming relaxations instead. We tried this, but it appeared that the reduction in the time needed to compute the lower bounds came at the expense of a larger number of iterations, and the total computation time was not reduced. Instead, for the two problem instances mentioned, we solved the lower bound integer programs to within 0.5% of optimality. This

Table 3: Comparison of our method and ACCPM.

			Our method			ACCPM					Optimality gaps:	
				min	Comp.		min		min	Comp.		
μ	r	δ	$C(x_3)$	SL(t)	time (min.)	$C(x_5)$	SL(t)	$C(x_6)$	SL(t)	time (min.)	ACCPM	Our Method
1	16	0.25	256	80.6%	0.92	480	99.87%	252.5	80.12%	43.92	0.60%	1.99%
1	16	0.5	263	81.0%	0.41	332.5	87.62%	259.5	80.94%	5.33	0.58%	1.94%
1	16	1	266	80.6%	0.29	329	88.84%	264	80.02%	1.04	0.00%	0.76%
1	32	0.25	470	80.3%	1.64	864	99.95%	470	80.55%	42.69	0.97%	0.97%
1	32	0.5	486.5	80.8%	0.77	590	81.26%	482	80.89%	6.61	0.94%	1.88%
1	32	1	494	80.3%	0.76	577	80.02%	493	80.61%	1.77	0.61%	0.82%
1	64	0.25	894.5	80.4%	5.31	1227.5	86.68%	891	80.25%	94.80	0.62%	1.02%
1	64	0.5	924.5	81.2%	2.23	1060	81.56%	914	80.01%	16.51	0.66%	1.82%
1	64	1	943	80.4%	2.19	1048	80.41%	943	81.45%	4.04	0.86%	0.86%
2	16	0.25	266	80.6%	0.93	480	99.50%	264.5	80.31%	25.22	0.95%	1.53%
2	16	0.5	273	81.2%	0.34	337	86.79%	273	80.01%	7.01	0.92%	0.92%
2	16	1	282	80.1%	0.26	346	84.31%	282	80.75%	1.13	0.71%	0.71%
2	32	0.25	506	81.9%	2.23	864	99.66%	496	80.46%	157.66	0.51%	2.53%
2	32	0.5	511	80.2%	0.76	624.5	86.95%	509.5	80.30%	6.27	0.79%	1.09%
2	32	1	536	80.6%	0.70	616	83.15%	533	80.07%	1.70	0.95%	1.52%
2	64	0.25	947.5	80.0%	5.01	1632	99.92%	946.5	80.14%	59.74	0.91% *	1.01% *
2	64	0.5	965	80.1%	1.81	1200	81.01%	969	78.73%	10.28	0.99%	0.57%
2	64	1	1016	80.4%	1.70	1163	80.45%	1018	80.46%	4.04	0.99%	0.79%
4	16	0.25	272.5	80.9%	0.85	480	99.24%	271	80.16%	29.87	0.93%	1.49%
4	16	0.5	278.5	80.4%	0.35	353.5	82.30%	277	80.57%	4.69	0.91%	1.46%
4	16	1	292	80.7%	0.30	367	82.67%	290	80.07%	1.20	0.35%	1.04%
4	32	0.25	510.5	80.7%	1.67	864	99.39%	508.5	80.25%	193.21	0.59%	0.99%
4	32	0.5	520.5	80.0%	0.67	621.5	83.16%	519	80.85%	79.72	0.78%	1.07%
4	32	1	545	80.1%	0.53	692	82.89%	548	81.43%	2.56	0.92%	0.37%
4	64	0.25	970	80.3%	4.89	1632	99.80%	968.5	80.60%	273.05	0.89%	1.04%
4	64	0.5	993.5	80.0%	1.52	1197	84.25%	990.5	80.52%	8.16	0.56% *	0.86% *
4	64	1	1049	80.8%	1.32	1319	81.59%	1048	80.86%	5.70	0.96%	1.06%

*: Lower bounds not guaranteed to be valid.

greatly reduced the computation time, but the lower bounds for these problem instances are not guaranteed to be valid.

Appendix C provides additional information about ACCPM computation times. Appendix D contains additional computational results for the approximate methods and our method, for 54 limited operating hours test problems for situations where Green et al. (2003) found the lag max approach to be reliable, and for 72 continuous operating hours test problems, as in Green et al. (2001). For the former set of experiments, cost savings of our approach over the lag max solution ranged from 0% (in 9 cases) to 8.6% and averaged 3.5%, and for the latter set, the cost savings ranged from 0% (in 9 cases) to 3.0%, averaging 1.0%.

8 Summary and Conclusions

We developed a solution approach for the problem that appears to be tacitly assumed in most research on tour and shift scheduling (Problem (1)). A commonly used approximate approach often results in service-level-infeasible solutions to this problem. We tested our approach on three sets of problems. The first set corresponds to situations where the approximate approach has been found to be least reliable. In this setting, the lag max variant of the approximate approach generated feasible solutions to 14 of the 27 problems. Our approach, which is guaranteed to generate a feasible solution, resulted in cost savings compared to the lag max approach on all 27 test problems, ranging from 1.4% to 10.4%. For the other two sets of experiments, the approximate approach performed relatively better, but nevertheless our approach resulted in considerable cost savings in many cases. For the first set of experiments, we also compared our approach to ACCPM, an approach that provides lower bounds on the minimum cost and that can be used to find globally optimal solutions. Based on the ACCPM lower bounds, the solutions from our approach were generally within 2% of optimality, and they were obtained in considerably less time than the ACCPM solutions.

The most obvious use for our approach is in situations where the approximate approach is known to generate solutions where service level constraints are not satisfied. However, our approach can also result in cost savings in other situations, where the approximate approach is reliable, but not optimal. Furthermore, our approach can be used for verification: If the approximate approach is sufficiently accurate, then our approach will terminate after one iteration, with proof that the solution from the approximate approach is a feasible and optimal solution to Problem (1).

An important issue is whether service level constraints should apply at all times or be aggregated over a longer time period. Our formulation has one service level constraint per

planning period, which is common in the related literature. The purpose of such constraints could be to provide a consistent level of service over time. When the service level is lower than desired, customers may leave or be less likely to return in the future. When the service level is higher than desired, customers' expectations may increase to a level that the organization cannot sustain. These arguments suggest that one should minimize variations in service level. A constraint on the minimum service level together with cost minimization, as in Problem (1), will sometimes be an adequate proxy for this goal. However, a strict limit on the instantaneous service level could be counterproductive and our approach can handle such alternatives as allowing the service level to drop below the desired minimum for short time intervals, or constraining the average service level over a planning period, rather than the minimum instantaneous service level. The minimum desired service level could also vary with time, because customers could be more tolerant of waiting, for example, in the evening (Jackson 2002) or on paydays (Katz et al. 1991).

Future research could help illuminate the consequences of several algorithm design choices for this and related server scheduling problems, including:

1. How to generate schedules—using a heuristic approach (such as the genetic algorithm in Ingolfsson et al., 2002) or an integer programming constraint (or "cut") addition heuristic (as in the present paper, Atlason et al., 2004, 2008, and in Cezik and L'Ecuyer, 2008, who address a related multi-skill problem). Atlason et al. (2004) and Cezik and L'Ecuyer (2008) add cuts based on estimated subgradients of the service level. This approach would guarantee an optimal solution if the service level were a concave function of the staffing, which it is not. Atlason et al. (2008) relax the concavity requirement to one of pseudoconcavity, which appears realistic for service levels. Instead of subgradients, their cuts are based on pseudogradients. An advantage of our cuts is that they require only one service level evaluation, while estimation of sub- or pseudogradients using a finite difference approach requires one evaluation for each planning period. Our cuts could be generalized to define a "feasible period" to be one where specified goals regarding service level, probability of abandonment, and probability of blocking are achieved. As long as each planning period can be classified as feasible or infeasible, and as long as any infeasible period can be made feasible by increasing staffing, our integer programming heuristic is applicable.

2. How to evaluate schedules—using simulation or an analytical approach. Simulation has the advantage of generality, but analytical approaches should be considered seriously. Compared to simulation, they have the advantage that noise in service level estimates

is not an issue. As we have shown, time-varying inputs and transient effects can be handled analytically, using a numerical approach. Abandonments and blocking can be incorporated in an analytical model as well. Another important feature is the discipline used when servers finish their shift. The pre-emptive discipline that we used will sometimes be unrealistic. A possible alternative is an *exhaustive service discipline*, where servers complete the current service before leaving. Ingolfsson et al. (2007) and Ingolfsson (2005) demonstrate how to treat this discipline analytically. We adopted the pre-emptive discipline in this paper to facilitate comparison of our computational results to Green et al. (2001, 2003).

3. The benefit of using of strict lower bounds. Such bounds can speed convergence of any integer programming-based heuristic that iterates between staffing and scheduling (Steps 2 and 3 in Section 1).

4. Combining methods. Our computations with Atlason et al.'s (2008) ACCPM suggest that combining it with our method could be a fruitful strategy. Our method could be used to quickly generate a near-optimal feasible solution. This solution could then be used as a starting solution by ACCPM, which could improve the solution further and provide lower bounds on the solution cost.

Acknowledgments: This work was partially supported by grants to the first author from the Natural Science and Engineering Research Council of Canada. The author gratefully acknowledge useful comments on earlier versions of this paper from Ignacio Castillo, from attendees at seminar at the Free University in Amsterdam and the Tinbergen Institute, and from the anonymous referees.

References

Aykin, T. Optimal shift scheduling with multiple break windows. *Management Science.* 1996; 42; 591–602

Aykin, T. A composite branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows. *Journal of the Operational Research Society* 1998; 49; 603–615

Atlason, J, Epelman, M A, Henderson, S G. Call center staffing with simulation and cutting plane methods. *Annals of Operations Research* 2004; 127; 333–358

Atlason, J, Epelman, M A, Henderson, S G. Optimizing call center staffing using simulation and analytic center cutting plane methods. *Management Science* 2008; 54; 295–309

Bartholdi III, J J, Orlin, J B, Ratliff, H D. Cyclic scheduling via integer programs with circular ones. *Operations Research* 1980; 28; 1074–1085

Brusco, M J, Jacobs, L W. A simulated annealing approach to the cyclic staff-scheduling problem. *Naval Reseach Logistics* 1993; 40; 69–84

Buffa, E S, Cosgrove, M J, Luce, B J. An integrated work shift scheduling system. *Decision Sciences* 1976; 7; 620–630

Cezik, T, L'Ecuyer, P. Staffing multiskill call centers via linear programming and simulation, *Management Science*, 2008; 54; 310–323

Deslauriers, A, L'Ecuyer, P, Pichitlamken, J, Ingolfsson, A, Avramidis, A N. Markov chain models of a telephone call center with call blending. *Computers & Operations Research* 2005; 34; 1616–1645

Dantzig, G B. A comment on Edie's 'Traffic delays at toll booths'. *Operations Research* 1954; 2; 339–341

Edie, L C. Traffic delays at toll booths. *Operations Research* 1954; 2; 107–138

Feldman, Z, Mandelbaum, A, Massey, W A, Whitt, W. Staffing of time-varying queues to achieve time-stable performance. *Management Science*, 2008; 54; 324–338

Fukunaga, A, Hamilton, E, Fama, J, Andre, D, Matan, O, Nourbakhsh, I. Staff scheduling for inbound call centers and customer contact centers. *AI Magazine* 2002; 23(4); 30–40

Gans, N, Koole, G, Mandelbaum, A. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management* 2003; 5; 79–141

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

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

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

Green, L V, Kolesar, P J, Soares, J. An improved heuristic for staffing telephone call centers with limited operating hours. *Production and Operations Management* 2003; 12; 46–61

Green, L V, Kolesar, P J, Whitt, W. Coping with time-varying demand when setting staffing

requirements for a service system. *Production and Operations Management* 2007; 16; 13–39

Green, L V, Soares, J. Computing time-dependent waiting time probabilities in $M(t)/M/s(t)$ queueing systems. *Manufacturing & Service Operations Management* 2007; 9; 54–61

Holmström, K. The TOMLAB optimization environment in Matlab. *Advanced Modeling and Optimization* 1999; 1; 47–69.

Ingolfsson, A, Haque, M A, Umnikov, A. Accounting for time-varying queueing effects in tour scheduling. *European Journal of Operational Research* 2002; 139; 585–597.

Ingolfsson, A, Akhmetshina, A, Budge, S, Li, Y, Wu, X. A survey and experimental comparison of service level approximation methods for non-stationary $M(t)/M/s(t)$ queueing systems. *INFORMS Journal on Computing* 2007; 19; 201–214

Ingolfsson, A. Modeling the $M(t)/M/s(t)$ queue with an exhaustive discipline. Working paper. 2005. (`http://www.business.ualberta.ca/aingolfsson/publications.htm`)

Jackson, K. Thinking beyond the old 80/20 rule. *Call Center Magazine* 2002; 15 (January); 54–56.

Jagers, A A, Van Doorn, E A. Convexity of functions which are generalizations of the Erlang loss function and the Erlang delay function. *SIAM Review* 1991; 33; 281–282.

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

Katz, K L, Larson, B M, Larson, R C. Prescription for the waiting-in-line blues: entertain, enlighten, and engage. *Sloan Management Review* 1991; 32 (December); 44–53

Kolesar, P J, Rider, K L, Crabill, T B, Walker, W E. A queuing-linear programming approach to scheduling police patrol cars. *Operations Research* 1975; 23; 1045–1062

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

Thompson, G M. Labor staffing and scheduling models for controlling service levels. *Naval Research Logistics* 1997; 44; 719–740.