Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

SPEEDING UP CALL CENTER SIMULATION AND OPTIMIZATION BY MARKOV CHAIN UNIFORMIZATION

Eric Buist
Wyean Chan
Pierre L'Ecuyer

Département d'Informatique et de Recherche Opérationnelle
Université de Montréal, C.P. 6128, Succ. Centre-Ville
Montréal (Québec), H3C 3J7, CANADA

ABSTRACT

Staffing and scheduling optimization in large multiskill call centers is time-consuming, mainly because it requires lengthy simulations to evaluate performance measures and their sensitivity. Simplified models that provide tractable formulas are unrealistic in general. In this paper we explore an intermediate solution, based on an approximate continuous-time Markov chain model of the call center. This model is more accurate than the commonly used approximations, and yet can be simulated faster than a more realistic simulation (based on non-exponential distributions and additional details). To speed up the simulation, we uniformize the Markov chain and simulate only its discrete-time version. We show how performance measures such as the fraction of calls of each type answered within a given waiting time limit can be recovered from this simulation, how to synchronize common random numbers in this setting, and how to use this in the first phase of an optimization algorithm based on the cutting plane method. We also discuss various implementation issues and provide empirical results.

1 INTRODUCTION

Telephone call centers, and more generally *contact centers* where mail, fax, e-mail, and Internet contacts are handled in addition to telephone calls, are important components of large organizations (Akşin, Armony, and Mehrotra 2007, Gans, Koole, and Mandelbaum 2003). Specifying the number and types of *agents* who handle the calls, and the working schedules of these agents, under constraints on the quality of service and on admissible schedules, is one of the main optimization problems encountered in managing these centers. Large call centers are complex stochastic systems that can be analyzed realistically only by simulation; tractable queueing models oversimplify reality and are not very reliable. When simulation is combined with an optimization algorithm, its efficiency is a key issue, because optimization often requires thousands of simulation runs at different parameter settings (Ceşik and L'Ecuyer 2008, Avramidis, Gendreau, L'Ecuyer, and Pisacane 2007b, Avramidis, Gendreau, L'Ecuyer, and Pisacane 2007a). A common workaround for this problem is to use approximation formulas during the first steps of the optimization process, and then refine the solution using simulation. However, these approximations are so rough for complex models with multiple types of calls and agents that they may lead to highly-suboptimal or even infeasible solutions (Avramidis, Chan, and L'Ecuyer 2008).

One possible compromise, explored in this paper, is to model the system as a continuous-time Markov chain (CTMC) that can be simulated faster than a detailed (more realistic) discrete-event simulation model. Speedup can be achieved by uniformizing the CTMC and simulating only the embedded discrete-time Markov chain (DTMC) instead of the CTMC. Simulation must be used because this chain has an enormous complicated state space and evolves in non-stationary mode. The uniformization often increases the average transition rate significantly if done in straightforward way, and this could wipe out the performance gain, but our implementation uses an adaptive state-dependent uniformization scheme that addresses this issue. For each realization of the DTMC, performance measures are estimated by computing their conditional expectation given the DTMC and its number of steps. This quick (simplified) simulation model is used in the first stage of a staffing optimization procedure, whose second stage uses detailed simulation of a more realistic model for fine tuning of the solution. We give illustrations for a multiskill center over a single time period. The technique can be extended to cover multiple periods with different parameters.

The rest of the paper is organized as follows. The next section provides a brief description of multiskill call centers and the staffing optimization problem in these centers. Section 3 explains how a CTMC can be simulated efficiently over a finite horizon by simulating the embed-

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

ded DTMC of a uniformized chain. Section 4 presents our CTMC model of a multiskill call center, explains how our methodology can be applied in this case, and how the performance measures of interest are estimated. In particular, we will see that estimating the service level, one of the most important performance measures for call center managers, requires extra bookkeeping. Similar problems occur for other performance measures. Section 5 describes how staffing is optimized using this simplified simulation. Section 6 presents numerical results, whereas Section 7 concludes the paper and outlines ideas for future work.

2 A MULTISKILL CALL CENTER

A *call center* is a set of resources for communication between an organization and its customers over the phone. In a *multiskill* center, calls are divided into K types, and are served by agents partitioned into I groups. Each agent group $i = 1,\ldots,I$ has a skill set $S_i \subseteq \{1,\ldots,K\}$, which corresponds to the list of call types that these agents can serve. Calls arriving when no agent is available join a waiting queue, and may abandon if they do not receive service fast enough.

Managers are often interested in finding a staffing vector $\mathbf{y} = (y_1,\ldots,y_I)^{\mathrm{t}}$, where $^{\mathrm{t}}$ denotes vector transposition and y_i is the number of agents in group i, to minimize the operating costs under constraints on the quality of service. Suppose c_i is the cost of an agent of group i and let $\mathbf{c} = (c_1,\ldots,c_I)^{\mathrm{t}}$. The most widely used measure for the quality of service is the *service level*, defined as the fraction of calls answered within a given waiting time limit. For example, the service level for call type k, denoted $g_k(\mathbf{y})$, can be defined as the total expected number of calls of type k answered within a given time limit s_k, over the day, divided by the total expected number of calls of type k received over the day. In the same way, we can define the overall service level $g(\mathbf{y})$ (over all call types), for a given time limit s. A staffing optimization problem can then be formulated as the nonlinear program (Ceşik and L'Ecuyer 2008):

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{t}}\mathbf{y} = \sum_{i=1}^{I} c_i y_i \\ \text{subject to} \quad & g_k(\mathbf{y}) \geq l_k \quad \text{for all } k, \\ & g(\mathbf{y}) \geq l, \\ & \mathbf{y} \geq 0, \text{and integer}, \end{aligned} \tag{P}$$

where l and the l_k's are user-defined service level targets. This staffing problem is a relaxed version of a more complex scheduling problem (Bhulai, Koole, and Pot 2005, Avramidis, Gendreau, L'Ecuyer, and Pisacane 2007b).

Many heuristics have been proposed for Problem (P) (Ceşik and L'Ecuyer 2008, Avramidis, Chan, and L'Ecuyer 2008, Wallace and Whitt 2005, Pot, Bhulai, and Koole 2007). Most methods perform a search in the space of staffing vectors, and therefore need to estimate the performance for different parameters settings. Frequently, such estimations must be computed at a large number of slightly different staffing levels, for sensitivity analysis (subgradient estimation). These estimations are generally very time-consuming, especially when simulation is used. A common practice is therefore to use quick approximations to evaluate the performance during the first steps of the optimization (Franx, Koole, and Pot 2006, Koole, Pot, and Talim 2003, Avramidis, Chan, and L'Ecuyer 2008). However, these approximations are often unreliable, especially for multiskill centers with complex routing rules. As a result, optimization heuristics driven by these approximations may produce highly-suboptimal or even unfeasible solutions. Simulation can be used in a second stage to improve the solution when it is suboptimal, and fix it if it is unfeasible, but this second stage may still produce a highly suboptimal solution if the first stage did not produce a good enough solution to start with. Our aim here is to replace the quick approximations in the first stage by more accurate simulations of a CTMC model.

3 DISCRETE-TIME SIMULATION OF CTMC MODELS

3.1 A CTMC Model Over a Finite Horizon

We consider a *continuous-time Markov chain* (CTMC) $\{X(t), t \geq 0\}$ defined over a discrete state space $S = \{1,2,\ldots\}$, where S is either finite or infinite (see Taylor and Karlin 1998 for a definition). We can think of $X(t) \in S$ as representing the state of a system of interest at time t. The sojourn time in any state $i \in S$ follows the exponential distribution, say with rate q_i. When this sojourn ends, the process jumps to state j with probability $p_{i,j}$, independently of its previous history. The jump rate to state j while in state i is thus $q_{i,j} = p_{i,j}q_i$. If we set $q_{i,i} = -q_i$ for all i, the $|S| \times |S|$ *generator matrix* of the CTMC, denoted $\mathbf{Q}$, has elements $q_{i,j}$. The corresponding embedded *discrete-time Markov chain* (DTMC) is defined as the stochastic process $\{X_n = X(\tau_n), n \geq 0\}$ where $0 = \tau_0 \leq \tau_1 \leq \tau_2 \leq \cdots$ are the jump times of the CTMC. The *transition matrix* of this embedded chain, denoted $\mathbf{P}$, has elements $p_{i,j}$.

We are interested in the evolution of the CTMC over a finite time horizon T. In the context of a call center, the interval $[0,T]$ may correspond to a period of an hour, a day, etc. Let $M(T) = \max(n : \tau_n \leq T)$ be the (random) number of transitions of the chain during $[0,T]$. This CTMC can be simulated as follows. At each step n for which $\tau_n < T$, we generate the time $\tau_{n+1} - \tau_n$ until the next transition (this is an exponential), and the next state X_{n+1} with the probabilities $p_{i,j}$ if $X_n = i$.

If a cost is incurred at the state-dependent rate $f(i)$ for $i \in S$, then the total cost can be written as

$$C = \int_0^T f(X(t))dt \quad (1)$$
$$= \sum_{n=0}^{M(T)-1} f(X_n)(\tau_{n+1} - \tau_n) + f(X_{M(T)})(T - \tau_{M(T)}).$$

3.2 Uniformization

Suppose now that the chain is *uniformizable*, in the sense that

$$q \stackrel{\text{def}}{=} \sup_{i \in S} q_i < \infty.$$

The *uniformized* CTMC has transition rate q from every state, and its embedded DTMC $\{Y_n, n \geq 0\}$ has transition matrix $\tilde{\mathbf{P}} = \mathbf{Q}/q + \mathbf{I}$, where $\mathbf{I}$ is the $|S| \times |S|$ identity matrix. In other words, we increase the transition rate out of every state to q, and each transition from state i is a "dummy" transition to the same state (or *self-jump*) with probability $1 - q_i/q$. The transitions of the uniformized chain then occur according to a stationary Poisson process $\{N(t), t \geq 0\}$ with rate q, independent of $\{Y_n, n \geq 0\}$.

In the case where a cost is incurred at state-dependent rate $f(i)$, the total cost has expectation (Gross and Miller 1984, Fox and Glynn 1990)

$$c = \mathbb{E}[C] = \mathbb{E}\left[\sum_{n=0}^{N(T)} \frac{T}{N(T)+1} f(Y_n)\right]$$
$$= T \sum_{k=0}^{\infty} \mathbb{P}[N(t) = k] \sum_{n=0}^{k} \frac{\mathbb{E}[f(Y_n)]}{k+1}$$
$$= T \sum_{k=0}^{\infty} \frac{e^{-qT}(qT)^k}{k!} \sum_{n=0}^{k} \frac{\mathbb{E}[f(Y_n)]}{k+1}. \quad (2)$$

Gross and Miller (1984), Grassmann (1989), Grassmann (1991) provide numerical methods based on uniformization and (2) to compute the expected cost over a fixed time interval $[0,T]$, when $|S|$ is finite. The general idea is to compute the probabilities $p^{(n)}(i) = \mathbb{P}[Y_n = i]$ iteratively for $n = 1, 2, \ldots$, use them to compute $\mathbb{E}[f(Y_n)]$ for each n, and compute c via (2). This technique can actually provide the state distribution of the CTMC at any given time t, if we replace the inside sum (over n) by $\mathbb{I}[Y_n = i]$, where $\mathbb{I}$ is the indicator function, for each $i \in S$. However, this idea is viable only when the state space S is not too large.

3.3 Discrete-time Conversion for Simulation

When S is very large (or infinite), one can exploit (2) to estimate c by simulation as follows: Generate a realization of the DTMC $\{Y_n, n \geq 0\}$, and estimate c by the expression (2) in which $\mathbb{E}[f(Y_n)]$ is replaced by $f(Y_n)$:

$$\mathbb{E}[C \mid Y_0, Y_1, Y_2, \ldots] = T \sum_{k=0}^{\infty} \frac{e^{-qT}(qT)^k}{k!} \sum_{n=0}^{k} \frac{f(Y_n)}{k+1}.$$

Since this estimator is the expectation of C conditional on $\{Y_n, n \geq 0\}$, its variance is guaranteed to be smaller than that of C. This is a special case of *discrete-time conversion*, studied by Fox (1990), who also examine alternative implementations and other variants of the estimation problem. In practice, the sum over k can be truncated at a point where the remaining Poisson probabilities are deemed negligible. When qT is large, a good truncation point is easily estimated via a normal approximation to the Poisson distribution. Another possibility is to sample the Poisson tail instead of truncating it.

A simpler (unbiased) estimator of c is

$$\mathbb{E}[C \mid N(T), Y_0, \ldots, Y_{N(T)}] = \frac{T}{N(T)+1} \sum_{n=0}^{N(T)} f(Y_n), \quad (3)$$

where $N(T)$ is generated from the Poisson distribution with mean qT, and the DTMC $\{Y_0, \ldots, Y_{N(T)}\}$ is generated as usual. The estimator (3) has more variance than (3.3), because (3.3) is a conditional expectation of (3), but (3) can be less expensive to compute, so it could be preferable in some situations. Another possibility, which generalizes (3), is to generate $m \geq 1$ independent replicates of $N(T)$, say $N_1, \ldots, N_m$, use them in (3) with the same realization of $\{Y_n, n \geq 0\}$, and take the average. This gives:

$$\frac{T}{m} \sum_{i=1}^{m} \frac{1}{N_i + 1} \sum_{n=0}^{N_i} f(Y_n). \quad (4)$$

Increasing m reduces the variance, but since the variance of (4) always remains larger than that of (3.3), there is an optimal value of m beyond which the gain does not justify the extra work.

An improved version of (4) stratifies on $N(T)$: generate m independent uniform random variates $U_1, \ldots, U_m$ where U_i is uniform over $[(i-1)/m, i/m)$, then generate N_i from U_i by inversion, for each i. That is, $N_i = F_{N(T)}^{-1}(U_i)$, where $F_{N(T)}$ is the distribution function of a Poisson random variable with mean qT. The estimator remains the same.

In the context of call center staffing problems, the quantities that we typically want to estimate, such as the service level, turn out *not* to be of the form (1). However, they are of the general form

$$\sum_{n=0}^{N(T)} \tilde{C}_n, \quad (5)$$

where $\tilde{C}_n = \mathbb{E}[C_n \mid N(T), Y_0, \ldots, Y_{N(T)}]$ can be computed analytically or numerically for each $n \le N(T)$, as we will see in the next section. The estimators (3.3) to (4) can be adapted rather easily to this setting. The adaptations of (3.3) and (4) give

$$\sum_{k=0}^{\infty} \frac{e^{-qT}(qT)^k}{k!} \sum_{n=0}^{k} \tilde{C}_n \tag{6}$$

and

$$\frac{1}{m} \sum_{i=1}^{m} \sum_{n=0}^{N_i} \tilde{C}_n, \tag{7}$$

respectively. In situations where $\tilde{C}_n$ depends on $N(T)$ in a complicated way, computing either (6), or (7) with a large value of m, can be much more expensive than computing (7) with $m = 1$, so the latter may turn out to be the most efficient alternative. The next section will illustrate this.

4 CTMC MODEL OF A CALL CENTER

4.1 Call Center Model

Our CTMC model of a multiskill call center is defined as follows. For each $k = 1, \ldots, K$, we suppose that calls of type k arrive according to a Poisson process with rate λ_k. Service times for calls of type k served by agents in group i are i.i.d. exponential with rate $\mu_{k,i}$. A call of type k not served immediately balks (abandons immediately) with probability ρ_k. The patience times for calls of type k joining the queue without balking are i.i.d. exponential with rate ν_k. A call becomes an abandonment whenever its waiting time exceeds its patience time. Abandoning calls are removed from the queue and lost.

There are I types of agents (or agent groups). For each $i = 1, \ldots, I$, group i contains y_i agents. A router assigns agents to newly arriving calls, and assigns queued calls to agents when they become free, in the following way. Each call type has a list of agent groups that can handle it, say $i_{k,1}, i_{k,2}, \ldots$, by order of preference. An arriving call of type-k is assigned to the first group in this list having an available agent. If no agent is available from the groups in the list, the call is queued. Each call type has its own FIFO queue. Similarly, each agent group has an ordered list of call types these agents can handle, say $k_{i,1}, k_{i,2}, \ldots$. When an agent of group i becomes free, he picks a call from the first non-empty queue in its list. Other similar (state-dependent) routing schemes can be implemented with this model without affecting the CTMC methodology that we are presenting here.

The state of the CTMC must keep track of the number of calls of each type being served by each agent group, as well as the number of calls in each queue. Thus, each element of the state space S is a vector whose elements are $S_{k,i}$, the number of calls of type k currently served by agents of group i, for $1 \le k \le K$ and $1 \le i \le I$, and Q_k, the number of type-k calls waiting in queue, for $1 \le k \le K$. Admissible states must satisfy the constraints $0 \le \sum_{k=1}^{K} S_{k,i} \le y_i$ for each i, and $S_{k,i} = 0$ whenever group i cannot handle call type k. We assume that the system starts empty, so initially we have $S_{k,i} = 0$ and $Q_k = 0$ for all k and i. Call centers with more than a dozen call types and a few dozen agent groups, a total of several hundred agents, and queues that may sometimes grow up to over a hundred calls, are not uncommon. The size of the state space is then much too large to consider exact numerical methods as in Gross and Miller (1984), and one must rely on simulation.

The $(I+3)K$ types of transitions (or events) for our CTMC are summarized in Table 1. The table gives the type of event that causes the transition, the state-dependent transition rate, and its usual effect on the state. We briefly discuss these different types of events.

Upon arrival of a type-k call, the router selects an agent group i' that can handle the call, if there is one, in which case $S_{k,i'}$ is increased by 1. Otherwise, the call either abandons immediately (balks), or joins a queue and Q_k increases by 1. When an agent of group i completes the service of a type-k call, $S_{k,i}$ is decreased by 1, and a queued call of certain type k' that can be handled by group i is chosen by the router if there is one such call in the queues, in which case $Q_{k'}$ decreases by 1 and $S_{k',i}$ increases by 1. Otherwise, the agent becomes free. When a call of type k abandons, Q_k decreases by 1.

4.2 Uniformization

Because of the unlimited queue size, the total abandonment rate (for all calls) is unbounded; therefore the CTMC has an unbounded total jump rate and is not uniformizable in its original form. For this reason, we impose a finite limit H on the total queue size $\sum_{k=1}^{K} Q_k$. Any call arrival that would increase the total queue size beyond H is just dismissed. With this modification, the CTMC has maximal transition rate q given by the sum of maximal arrival, service, and abandonment rates:

$$q = \sum_{k=1}^{K} \lambda_k + \sum_{i=1}^{I} y_i \mu_i + H\nu,$$

where $\mu_i = \max_{k \in S_i} \mu_{k,i}$, and $\nu = \max_{k \in \{1,\ldots,K\}} \nu_k$.

Another important issue is that at a given iteration of the optimization algorithms we use, one simulates a large number of very similar staffing configurations of the center, that differ by only one or two agents, for sensitivity analysis. These simulations are done with common random numbers, and it is important to make sure that synchronization is

Table 1: Possible Types of Events with their Usual Effects on the State

Event Type	Rate	Usual Effect
Type-k arrival when an agent is available	λ_k	Increases the number of busy agents.
Type-k arrival with balking, no agent available	$\rho_k\lambda_k$	State is unchanged.
Type-k arrival, no balking, no agent available	$(1-\rho_k)\lambda_k$	Increases the queue size Q_k by 1.
Type-k abandonment	$Q_k\nu_k$	Decreases the queue size Q_k by 1.
Group-i agent ends type-k service	$S_{k,i}\mu_{k,i}$	Decreases the number of busy agents or the queue size.

maintained, i.e., that the same random numbers are used for the same purposes, so they produce mostly the same types of transitions, when simulating the DTMC. In particular, a random number generating an arrival with a given staffing vector should not trigger, say, a service termination with another staffing vector. For this reason, we use a common overall transition rate for the uniformized chain, taken as

$$q = \sum_{k=1}^{K} \lambda_k + \sum_{i=1}^{I} \bar{y}_i \mu_i + H\nu,$$

where $\bar{y}_i$ is the maximal number of agents in group i in all the staffing configurations considered.

4.3 Simulation by Indexed Search

Because of the relatively large number of event types, generating transitions can be time-consuming. Indexed search is used to generate transitions efficiently. For this, we split the interval $[0,1]$ in 2^a sub-intervals of equal size, for some integer a, and we associate an *event type* with each sub-interval. Some of these event types correspond to a single type of transition in Table 1. Others correspond to a subinterval that must be split again, say in 2^b subintervals, and an event is associated with each subinterval. Some of them determine a DTMC transition, others must be split again, and so on.

A transition is simulated by generating a independent random bits to obtain an index, and executing the event corresponding to the selected sub-interval. An event uses its corresponding sub-interval $[u_1,u_2]$, internal information, and may use additional random bits to make further decisions such as selecting a smaller sub-interval, or determining the type of a call. This indexed search allows to select the call type of an arrival, and the agent group concerned by a service termination, in constant time, because the required information only depends on the sub-interval $[u_1,u_2]$, which can be precomputed. But a linear search is required to select the type of a served or abandoned call, because its distribution depends on $S_{k,i}$, or Q_k, so it changes with time. For example, if the selected event corresponds to the end of a service by an agent in group i, a call of type k is selected with probability $S_{k,i}\mu_{k,i}/(y_i\mu_i)$, and a self-jump occurs with probability $1-\sum_{k=1}^{K} S_{k,i}\mu_{k,i}/(y_i\mu_i)$. Other search methods such as binary search impose additional overhead, and would be efficient only for large values of K. Alternative rejection methods have been proposed (Fox and Glynn 1990, Fox and Young 1991, Rajasekaran and Ross 1993), but we prefer inversion because it provides better synchronization when using common random numbers.

The selection of a call type k can be avoided at the end of a service by an agent in group i if $\mu_{k,i} = \mu_i$ for $k = 1,\ldots,K$, and if we do not need to collect statistics on the proportion of calls of each type in service by agents in group i. The selection of a call type at the end of a service or when abandonment occurs is also unnecessary if $K = 1$. In both cases, the linear search is replaced by a simple test to determine if the DTMC transition is a self-jump or not.

4.4 Reducing the Self-jumps

When the agents' occupancy is low, or if the queue capacity is high, the DTMC has a large proportion of self-jumps, whose simulation eventually adds significant overhead. Fox and Glynn (1990) propose to telescope the self-jumps by noting that the number of self-jumps of the uniformized chain between any two jumps of the original chain is a geometric random variable with parameter q_i/q when we are in state i, and generating this geometric variable directly. However, this requires an efficient method to generate the transitions of the original DTMC (without the self-jumps), for which the transitions depend strongly on the current state. This is incompatible with the fast indexed search method described earlier, because we would need a different index for each current state i.

To reduce the number of self-jumps, we adopt a solution of compromise based on the idea of constructing a limited number of search indexes. The idea is to partition the state space in, say, R pieces, and compute an upper bound $q(r)$ on the jump rate for each piece r. Whenever the current state is in piece r, we generate the next transition based on a DTMC with jump rate $q(r)$, and we add a random number Z of self-jumps before this next transition (which may actually be a self-jump itself), where Z has the geometric

distribution with parameter $p(r) = q(r)/q$. We thus obtain a valid realization for the DTMC $\{Y_n, n \geq 0\}$, from which we can compute estimators for the performance measures we have in mind.

Implementing this scheme efficiently requires setting up a different search index for each piece r. In our implementation, the index incorporates a precomputation of the inverse distribution function of the geometric distribution with parameter $p(r)$, so both Z and the next transition are generated together from the index, using a single uniform random number. It is usually better to keep the number R of pieces rather small (say, no more than a dozen, or even less), because computing and storing them can be costly.

In the context of our call center, we partition the state space according to the total queue size. The idea is that when there are many calls in queue, the overall abandonment rate becomes quite high, and this forces us to use a large value of q, which in turn implies a large fraction of self-jumps when there are few calls in the queues. More specifically, let $0 = \eta_0 < \eta_1 < \cdots < \eta_R = H$ be an increasing sequence of thresholds on the queue size. Each threshold η_r corresponds to an operating mode with a specific transition rate $q(r) = q - (H - \eta_r)\nu \leq q$ for the corresponding uniformized chain and search index. The current operating mode r is given by the smallest integer for which $\sum_{k=1}^{K} Q_k \leq \eta_r$.

4.5 Estimating the service level from the uniformized DTMC

The service level, one of the primary performance measure in which call center managers are interested, is not of the form (1). To estimate the service level (for each call type and globally), we proceed as follows. For each queued call, the transition number of its arrival is saved. These numbers are stored in arrays representing the K waiting queues of the model. The current transition number is added to the kth array when a call of type k enters the queue. When a call of type k abandons, an element of array k is chosen randomly and uniformly, and removed. Whenever a service completion triggers a start of service for a call of type k', the first call in queue k' is removed, and we count the number D of DTMC transitions since its arrival. Keeping these arrays up-to-date involves some overhead, especially when removing elements, which generally requires shifting all subsequent elements of the array. Since the first array element is often removed, we store the array in a circular buffer, to avoid the linear-time shifting for this (frequent) special case.

Suppose now that a call waits for D transitions before starting its service. The waiting time W of this call is the time required for these D transitions in the CTMC. Conditional on $N(T)$ and $D < N(T)$, this W has the same distribution as the time required for the first D transitions, which in turns has the same distribution as the Dth order statistic in a sample of $N(T)$ independent uniform random variables over $[0,T]$. (See David 1981 for standard results on order statistics.) We have

$$\begin{aligned} & \mathbb{P}[W > s \mid N(T), D] \\ = \; & \mathbb{P}[B < D] \\ = \; & \sum_{j=0}^{D-1} \binom{N(T)}{j} (s/T)^j (1 - s/T)^{N(T)-j}, \end{aligned}$$

where B has the binomial distribution with parameters $n = N(T)$ and $p = s/T$. Then we can easily compute

$$\mathbb{P}[W > s \mid N(T), D] = \mathbb{E}[\mathbb{I}[W > s] \mid N(T), X_0, X_1, \ldots, X_{N(T)}]$$

for each waiting call. The sum of these conditional probabilities over all calls of type k gives an unbiased estimator of the expected number of calls who wait more than s. To speed up these computations, once we know $N(T)$, we precompute the distribution function of B, up to some cut-off value where this function is close to 1, and store it in a table. It is important to notice that this binomial distribution depends on $N(T)$. For this reason, computing an estimator of the form (6) becomes too expensive here, because a new binomial table must be computed for each value of n in that equation. Similar ideas can be used to estimate other performance measures depending on the waiting time of calls.

5 APPLICATION TO OPTIMIZATION

Ceşik and L'Ecuyer (2008) solve a sample average approximation (P1) of Problem (P), in which the service level functions g_k are estimated by their sample averages $\hat{g}_k$ based on a certain number of simulation runs over the time period of interest. The sample problem (P1) is solved using a cutting plane method adapted from Atlason, Epelman, and Henderson (2004), combined with a number of heuristics, which are detailed in Ceşik and L'Ecuyer (2008). In a nutshell, the method relaxes the service level constraints in (P), and adds linear constraints based on estimates on the subgradients of the g_k's associated with service level constraints that are violated in (P1). The aim of these constraints is to remove unfeasible pieces from the solution space. Each subgradient estimation actually requires simulating the system at $I+1$ staffing configurations. The ith coordinate of the subgradient estimate has the form $q_i(\bar{\mathbf{y}}) = [\hat{g}(\bar{\mathbf{y}} + d\mathbf{e}_i) - \hat{g}(\bar{\mathbf{y}})]/d$, where $\bar{\mathbf{y}}$ is the current solution, $\mathbf{e}_i$ is the i-th unit vector, and $d \geq 1$ is an integer. The algorithm adds linear constraints iteratively to the sample problem until its solution becomes feasible (i.e., satisfies the service level constraints).

Because of the noise and the potential non-concavity of the functions $\hat{g}$, these linear cuts sometimes remove feasible solutions, including the optimal solution, in which case the

algorithm will return a suboptimal solution to (P). It also happens that although the final solution is feasible for (P1), it is unfeasible for (P). Increasing the number of simulation runs by a large factor generally alleviates this problem, but at a cost. Solving large staffing problems may then require hours of simulation.

After a feasible solution $\mathbf{y}^*$ is found for (P1), we refine this solution using an expanded sample problem, say (P2), obtained from a larger number of simulation runs. The solution of (P1) is adjusted to a feasible solution for (P2) by adding more agents if it is infeasible for (P2), removing agents while keeping the solution feasible for (P2), and moving agents to cheaper groups, if possible.

Optimizing a given sample problem implies that the simulations are performed with well-synchronized common random numbers across all staffing configurations. For the detailed simulations, we use common random numbers and synchronize them using separate streams of random numbers for arrivals, service times, abandonments, etc. For the simplified simulations with the uniformized DTMC, we maintain synchronization in the estimation of any given subgradient by using a common transition rate q for all the simulations involved for this subgradient, as explained earlier. Ideally, it would be best to use the same rate q and the same indexes for *all* the simulations performed during the optimization. However, to do that, we would need to impose a maximum staffing vector and use it to define q and the index. Choosing such a maximum staffing vector is difficult; if we take it too large, we obtain a large value of q and a high rate of self jumps, while if we take it too small, we may miss some relevant staffing vectors. For the experiments reported in this paper, we took a new q and a new index for each subgradient estimation.

6 NUMERICAL EXAMPLES

We now compare the simulator based on the uniformized DTMC against the detailed discrete-event simulator implemented in ContactCenters (Buist and L'Ecuyer 2005), first in terms of simulation speed, then when used in an optimization algorithm.

6.1 Simulation

Our first example has a single call type and agent group. The parameters are $\rho_1 = 1/10$, $\nu_1 = 1/1000$, and $\mu_1 = 1/100$.

In the second example, we have three call types and six agent groups. For call type k, $k = 1,2,3$, we have $\rho_k = 0.1$ and $\nu_k = 1/1000$. The total arrival rate is λ, and those per call type are $\lambda_1 = \lambda_2 = 4\lambda/11$, and $\lambda_3 = 3\lambda/11$. The service rates are $\mu_{1,1} = \mu_{2,2} = \mu_{3,3} = 11/3600$, and $\mu_{1,4} = \mu_{1,5} = \mu_{2,5} = \mu_{2,6} = \mu_{3,6} = \mu_{3,4} = 1/360$. Table 2 gives the lists of groups and types, $i_{k,1},\ldots$, and $k_{i,1},\ldots$ used for the routing.

Table 2: Routing Tables for Call Center Example

Type	Groups		
1	1	4	5
2	2	5	6
3	3	6	4

Group	Types	
1	1	
2	2	
3	3	
4	1	3
5	2	1
6	3	2

We simulate 1000 independent replications with $T = 46800$ seconds, for different choices of λ, i.e., of the expected total numbers of arrivals during $[0,T]$. For each scenario (each λ), the number of agents is adjusted to have approximately 80% of the calls served after less than 20 seconds of wait. Table 3 presents the simulation results for the two examples. Each row corresponds to a scenario. The columns give the expected total number of arrivals during $[0,T]$, the total number of agents in the center (staffing), the total queue capacity H, the expected total number of transitions $\mathbb{E}[N(T)]$, the CPU time taken by the DTMC simulator, the CPU time taken by the discrete-event simulator, and the ratio of the two times. The results confirm that the DTMC simulator is faster than the general discrete-event simulator, and the gain is higher when there are fewer call types and when the arrival rate is higher. It is not surprising that the gain is higher with a single call type, because no linear search is required for the call selections in that case. Moreover, the number of objects created by the discrete-event simulator increases with the number of calls simulated while the DTMC simulator does not create objects to represent calls.

6.2 Optimization

We consider Example 2 with $\lambda = 14300$. The cost of the six agent types are given by the vector $\mathbf{c} = (1,1,1,1.05,1.05,1.05)^t$, and the service level targets are $l = l_k = 80\%$ of calls within 20 seconds of wait. We optimize the staffing with the cutting-plane algorithm of Ceşik and L'Ecuyer (2008), using the DTMC-based simulation (CP1) and the discrete-event simulator (CP2). We impose a CPU time limit of 2 minutes for the optimization process, and use $d = 2$ for all the subgradient estimators. Each time we add linear constraints, the updated problem is solved as an integer program with CPLEX 9.0. To obtain comparable execution times for both CP1 and CP2, the simulations use 40 replications for the DTMC and 8 for the discrete-event simulator.

We evaluate the quality of the results produced by these optimization algorithms as follows. First, we found

Table 3: Performance Comparison of the CTMC and the Discrete-Event Simulators

	$\mathbb{E}$[Num. arrivals]	Staffing	H	$\mathbb{E}[N(T)]$	DTMC Time	DES Time	Ratio
Example 1	1,660	5	30	5404	1 s	6 s	4.6
	25,000	52	80	57,292	11 s	1 min 29 s	8.2
	50,000	100	130	104,756	18 s	3 min 08 s	10.0
	75,000	148	170	152,220	25 s	4 min 59 s	12.0
Example 2	14,300	110	100	34,554	9 s	59 s	6.6
	30,000	221	100	65,958	19 s	2 min15 s	7.1
	45,000	331	150	99,041	27 s	3 min29 s	7.7
	60,000	431	150	128,302	37 s	4 min 38 s	7.5

Table 4: Example 2 with $\lambda = 14300$, Performance Comparison of the Optimization with DTMC and the Discrete-Event Simulators, based on 100 replications

Algo	ε	Min	Med	F	$O_{0.2}$	$O_{0.5}$	O_1
CP1	0.01	110.70	111.40	73	16	27	58
CP2	0.01	110.85	111.50	36	0	9	23
CP1	0.1	110.65	111.35	80	20	32	65
CP2	0.1	110.70	111.48	42	1	13	28

an empirical optimum by running the optimization algorithm with a very large budget. This gave the vector (36, 35, 27, 3, 5, 4) as our best estimate of the optimal staffing, with a cost of 110.60. We call a solution ε-feasible if its service level is greater or equal to $(l-\varepsilon)\%$ for every target. Then, we executed CP1 and CP2 100 times each, independently. For each execution, we verify the feasibility of the final solutions for problem (P) by simulating with 2000 runs of a discrete-event simulation; this gives a 95% confidence interval of width smaller than 0.4% for the service level. Most replications of CP1 and CP2 found a feasible solution after 60 seconds; only a few needed about 90 seconds. The remaining time was devoted to the local search. Table 4 compares the costs of the minimum and median solutions conditional to ε-feasibility (for $\varepsilon = 0.01$ and 0.1), the number of ε-feasible solutions (F), and the number of these solutions that are within $p\%$ of the optimal cost (O_p). Although the minimum and median costs are comparable, the results show that there is a much higher probability of finding low-cost feasible solutions when optimizing with DTMC.

7 CONCLUSION

We have constructed a simulator for multiskill call centers based on the discrete-time conversion of a uniformized CTMC. This simulator executes faster than a fully-detailed discrete-event simulator of the CTMC. We have shown that there is a potential gain in using this simulator in an optimization algorithm, instead of a discrete-event simulator, when the call center can be modeled reasonably well as a continuous-time Markov process. If the original model is not a CTMC, then we can still use the DTMC simulator as an approximation, to save time in the (expensive) first stages of an optimization algorithm.

Future work includes the generalization of the DTMC-based simulator to time-varying parameters (arrival rates, staffing, etc.), and its use in a scheduling optimization algorithm over multiple time periods.

ACKNOWLEDGMENTS

This research has been supported by Grants OGP-0110050 and CRDPJ-320308 from NSERC-Canada, and a grant from Bell Canada via the Bell University Laboratories, and a Canada Research Chair to the third author. The first author benefited from an Industrial Scholarship from NSERC-Canada and the Bell University Laboratories. The second author benefited from an Industrial Innovation Scholarship from NSERC-Canada, FQRNT-Québec, and the Bell University Laboratories. Part of this paper was inspired by Grassmann's section in (Grassmann, Puterman, L'Ecuyer, and Ingolfsson 2008).

REFERENCES

Akşin, O. Z., M. Armony, and V. Mehrotra. 2007. The modern call center: A multi-disciplinary perspective on operations management research. *Production and Operations Management* 16 (6). Forthcoming.

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

Avramidis, A. N., W. Chan, and P. L'Ecuyer. 2008. Staffing multi-skill call centers via search methods and a performance approximation. *IIE Transactions*. to appear.

Avramidis, A. N., M. Gendreau, P. L'Ecuyer, and O. Pisacane. 2007a. Optimizing daily agent scheduling in a multiskill call centers. Technical Report CIRRELT-2007-44, CIRRELT, University of Montreal

Avramidis, A. N., M. Gendreau, P. L'Ecuyer, and O. Pisacane. 2007b. Simulation-based optimization of agent scheduling in multiskill call centers. In *Proceedings of the 2007 Industrial Simulation Conference*, 255–263: Eurosis.

Bhulai, S., G. Koole, and G. Pot. 2005. Simple methods for shift scheduling in multi-skill call centers. Technical report, Technical Report WS 2005-10, Free University, Amsterdam.

Buist, E., and P. L'Ecuyer. 2005. A Java library for simulating contact centers. In *Proceedings of the 2005 Winter Simulation Conference*, 556–565: IEEE Press.

Ceşik, M. T., and P. L'Ecuyer. 2008. Staffing multiskill call centers via linear programming and simulation. *Management Science* 54 (2): 310–323.

David, H. A. 1981. *Order statistics*. Second ed. Wiley.

Fox, B. L. 1990. Generating Markov-chain transitions quickly: I. *ORSA Journal on Computing* 2:126–135.

Fox, B. L., and P. W. Glynn. 1990. Discrete-time conversion for simulating finite-horizon Markov processes. *SIAM Journal on Applied Mathematics* 50:1457–1473.

Fox, B. L., and A. R. Young. 1991. Generating Markov-chain transitions quickly: II. *ORSA Journal on Computing* 3:3–11.

Franx, G. J., G. Koole, and A. Pot. 2006. Approximating multi-skill blocking systems by hyper-exponential decomposition. *Performance Evaluation* 63:799–824.

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

Grassmann, W. K. 1989. Numerical solutions for Markovian event systems. In *Quantitative Methoden in den Wirtschaftswissenschaften*, ed. P. Kall, J. Kohlas, W. Popp, and C. A. Zehnder, 73–87. Springer-Verlag.

Grassmann, W. K. 1991. Finding transient solutions in Markovian event systems through randomization. In *Numerical Solutions of Markov Chains*, ed. W. J. Stewart, 357–371.

Grassmann, W. K., M. L. Puterman, P. L'Ecuyer, and A. Ingolfsson. 2008. Four canadian contributions to stochastic modeling. *INFOR*. to appear.

Gross, D., and D. R. Miller. 1984. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research* 32 (2): 343–361.

Koole, G., A. Pot, and J. Talim. 2003. Routing heuristics for multi-skill call centers. In *Proceedings of the 2003 Winter Simulation Conference*, 1813–1816: IEEE Press.

Pot, A., S. Bhulai, and G. Koole. 2007. A simple staffing method for multi-skill call centers. Manuscript, available at ⟨http://www.cs.vu.nl/~koole/research⟩.

Rajasekaran, S., and K. W. Ross. 1993. Fast algorithms for generating discrete random variates with changing distributions. *Modeling and Computer Simulation* 3 (1): 1–19.

Taylor, H. M., and S. Karlin. 1998. *An introduction to stochastic modeling*. third ed. San Diego: Academic Press.

Wallace, R. B., and W. Whitt. 2005. A staffing algorithm for call centers with skill-based routing. *Manufacturing and Service Operations Management* 7 (4): 276–294.

AUTHOR BIOGRAPHIES

ERIC BUIST is a PhD Student at the Université de Montréal. His main interests are software engineering, object-oriented programming, and simulation. He is currently working on the development of flexible and efficient tools for the simulation of contact centers. His e-mail address is <buisteri@IRO.UMontreal.CA>.

WYEAN CHAN is a PhD Student at the Université de Montréal. His main interests are applied mathematics and optimization. He is currently working on the development of workforce management tools for call centers. His e-mail address is <chanwyea@IRO.UMontreal.CA>.

PIERRE L'ECUYER is Professor in the Département d'Informatique et de Recherche Opérationnelle, at the Université de Montréal, Canada. He holds the Canada Research Chair in Stochastic Simulation and Optimization. His main research interests are random number generation, quasi-Monte Carlo methods, efficiency improvement via variance reduction, sensitivity analysis and optimization of discrete-event stochastic systems, and discrete-event simulation in general. He is currently Associate/Area Editor for *ACM Transactions on Modeling and Computer Simulation*, *ACM Transactions on Mathematical Software*, *Statistical Computing*, *International Transactions in Operational Research*, *The Open Applied Mathematics Journal*, and *Cryptography and Communications*. He obtained the *E. W. R. Steacie* fellowship in 1995-97, a *Killam* fellowship in 2001-03, and became an INFORMS Fellow in 2006. His recent research articles are available on-line from his web page: <http://www.iro.umontreal.ca/~lecuyer>.