Submitted to *Management Science*
manuscript MS-00998-2004.R1

Optimizing Call Center Staffing using Simulation and Analytic Center Cutting Plane Methods

Júlíus Atlason, Marina A. Epelman
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109-2117, USA,
jatlason@umich.edu, mepelman@umich.edu

Shane G. Henderson
School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, USA, sgh9@cornell.edu

We consider the problem of minimizing staffing costs in an inbound call center, while maintaining an acceptable level of service in multiple time periods. The problem is complicated by the fact that staffing level in one time period can affect the service levels in subsequent periods. Moreover, staff schedules typically take the form of shifts covering several periods. Interactions between staffing levels in different time periods, as well as the impact of shift requirements on the staffing levels and cost should be considered in the planning. Traditional staffing methods based on stationary queueing formulas do not take this into account. We present a simulation-based analytic center cutting plane method to solve a sample average approximation of the problem. We establish convergence of the method when the service level functions are discrete pseudoconcave. An extensive numerical study of a moderately large call center shows that the method is robust and, in most of the test cases, outperforms traditional staffing heuristics that are based on analytical queueing methods.

1. Introduction

A call center (also referred to as a contact center) is an important component of the operations of many organizations. A significant portion of the cost of operating a call center is staffing cost (Gans et al. 2003). In this paper we describe a method for finding staff (agent) level requirements, while simultaneously selecting staff shifts that cover these requirements, that minimize cost while ensuring satisfactory customer service. An analytic center cutting plane method guides the selection of staffing levels, and simulation is used to report service level performance for each set of staffing levels.

This problem has received a great deal of attention in the literature, and so one can reasonably ask why there is a need for a computational tool of this sort. To answer that question we first need to describe the overall staffing process. There are variations on the following theme (e.g., Castillo et al. 2003), but the essential structure is sequential in nature and is as follows (Mason et al. 1998).

1. (Forecasting) Obtain forecasts of customer load over the planning horizon, which is typically one or two weeks long. The horizon is usually broken into short periods that are typically between 15 minutes and 1 hour long.

2. (Work requirements) Determine the minimum number of agents needed during each period to ensure satisfactory customer service. Service is typically measured in terms of customer waiting times and/or abandonment rates in the queue.

3. (Shift construction) Select staff shifts that cover the requirements. This problem is usually solved through a set-covering integer program; see Mason et al. (1998) for more details.

4. (Rostering) Allocate employees to the shifts.

The focus in this paper is on Steps 2 and 3. Steps 1 and 4 are not considered further.

Step 2 is usually accomplished through the use of analytical results for simple queueing models. Green et al. (2001) coined the term SIPP (stationary, independent, period by period) to describe the general approach. In the SIPP approach, each period of the day is considered independently of other periods, the arrival process is considered to be stationary in that period, and one approximates performance in the period by a steady-state performance measure that is usually easily obtained from analytical results for particular queueing models. Heuristics are used to select input parameters for each period that yield a close match between the predictions and reality. For a wide variety of queueing models, this procedure results in some form of the "square-root rule," which is a rule of thumb that provides surprisingly successful staffing level suggestions. See, e.g., Borst et al. (2004), Kolesar and Green (1998), Jennings et al. (1996) for more on the square-root rule.

The SIPP approach is appealing from the standpoint of computational tractability and due to the insights it provides. However, there are cases where the SIPP approach does not do as well as one might hope (Green et al. 2001, 2003). For example, this can occur when the service times are relatively long; see Whitt (1991) for more on this point. Moreover, call centers can be somewhat complex in structure, and this complexity can make it difficult to identify a queueing model of the center that is both mathematically tractable and a reasonable match to the true system. In such cases, simulation is a viable alternative. Indeed, simulation is now increasingly used in Step 2; see Section VIII of Mandelbaum (2003) for many examples.

Further motivation for the use of simulation involves the linkage between staffing decisions in adjacent periods. Boosting staffing levels in one period can often help in reducing workload in subsequent periods, so there can be linkage in performance between different periods. Such linkage can imply that there are multiple solutions to the work requirements problem that can offer valuable flexibility in Step 3. Traditional queueing approaches are not satisfactory in such cases, and then one turns to simulation or other numerical methods. Indeed, Green et al. (2001, 2003) solve a system of ordinary differential equations through numerical integration to get the "exact" results for their models in order to compare the performance of various heuristics.

Assuming that one uses simulation or some other numerical method to predict performance in the call center, one then needs to devise a method to guide the selection of potential staffing levels to be evaluated. There have been several suggestions in the literature, all of which explicitly capture the linkage between periods in an attempt to realize cost savings. Like Green et al. (2001, 2003), Ingolfsson et al. (2002) use numerical integration to compute service level performance for a proposed set of staffing levels, and a genetic algorithm to guide the search. Ingolfsson et al. (2003) again use a numerical method to compute service level performance, and integer programming to guide the search. Castillo et al. (2003) devise a method for randomly generating sets of staff shifts, then use simulation to evaluate the service level performance of each set of generated staff shifts, and finally, plot the cost versus service level of the potential solutions to identify an efficient frontier. Atlason et al. (2004) use simulation to evaluate service level performance of a proposed set of shifts, and use integer programming in conjunction with Kelley's cutting plane method (Kelley, Jr. 1960) to guide the search.

In Atlason et al. (2004), performance in each time period is measured as the fraction of calls responded to on time, and analysis relies on the assumption that service in a period is concave and componentwise-increasing as a function of the staffing level vector. To understand this assumption, consider a single period problem. Increasing the staffing level should lead to improved performance, i.e., an increasing fraction of calls responded to on time. Furthermore, one might expect "diminishing returns" as the staffing level increases, so that performance would be concave in staffing level. Empirical results suggest that this intuition is correct, at least for sufficiently high staffing levels. But for low staffing levels, the empirical results suggest that performance is increasing and *convex* in the staffing level. So performance appears to follow an "S-shaped" curve (Ingolfsson et al. 2003, Atlason et al. 2004) in one dimension.

This non-concavity can cause the cutting plane method of Atlason et al. (2004) to cut off feasible solutions, and the problem can be so severe as to lead to the algorithm suggesting impractical staffing plans. Nevertheless, the ability of the Atlason et al. (2004) approach to efficiently sift through the combinatorially huge number of potential staffing plans is appealing. One might ask whether there is a similar optimization approach that can satisfactorily deal with S-shaped curves and their multidimensional extensions. That is the subject of this paper. We replace the assumption of concavity with a weaker one (namely, pseudoconcavity), and use a different cutting plane method that, together with additional techniques, successfully handles multi-dimensional extensions of the S-shaped curves alluded to above and seen in examples like that depicted in Figure 2; see Section 4.1 for more details.

Assuming that the service level functions are pseudoconcave, an analytic center cutting plane algorithm can be used to efficiently search the combinatorially large space of potential staffing plans. In essence the algorithm works as follows. One begins with a polyhedron that contains an optimal solution to the staffing problem. At each stage, the algorithm selects a staffing plan that lies close to the analytic center of the polyhedron and runs a simulation at that point to determine service level performance. Depending on the results of the simulation an "optimality cut" or one or more feasibility cuts are added, thereby shrinking the polyhedron. The algorithm terminates when the polyhedron contains no integer points, or when the difference between upper and lower bounds on the optimal objective is sufficiently small.

We view the contributions of this paper as follows.

1. Under realistic assumptions on the service level functions, we give an algorithm for solving to optimality the combined Step 2 - Step 3 problem. The combined procedure explicitly addresses linkage between periods and the impact of shift-based scheduling.

2. We give conditions under which the algorithm provably converges.

3. We compare the proposed algorithm to what can be reasonably viewed as the current best practice to better understand the properties of the algorithm. This comparison has to take place with a model of limited complexity so that the existing methods can be applied, but is nevertheless quite illuminating.

The numerical results in Section 5 show that the analytic center cutting plane method outlined here outperforms, or at least equals, the SIPP heuristics in every case in which shift structure of staff scheduling is explicitly considered, which is the setting we are interested in. In that sense, it is a robust procedure that can be applied in a near black-box fashion. Of course, these extremely appealing properties have to be traded off against the computational cost of the procedure, which is not insignificant. We are actively considering methods for reducing the computational effort of the procedure; see Section 6.

The remainder of this paper is organized as follows. We formulate the call center staffing problem in Section 2. In Section 3 we review cutting plane methods for continuous problems with pseudoconcave constraints. Section 4 describes modifications for discrete problems, such as the call center staffing problem, and proves that the algorithm converges. We compare the proposed algorithm with the SIPP methods described in Green et al. (2001, 2003) in Section 5. Section 6 concludes the paper and discusses future research questions.

2. Problem formulation

In this section we formulate the call center staffing problem of minimizing staffing cost over a planning horizon while maintaining an acceptable level of service in each of a number of time periods, with particular emphasis on the use of simulation to estimate the service levels. We say that service is acceptable in a period if the long-run percentage of calls that arrive in this period and are answered within a certain time limit τ meets or exceeds a threshold π. This is the usual "π percent of calls are answered within τ seconds" requirement that is something of an industry standard, cf. the "80/20 rule" for which π is 80% and τ is 20 seconds.

Define the staffing level vector as $y = (y_1, \ldots, y_p)^T$, where p is the number of time periods in the planning horizon, y_i is the number of agents working in period i, $i = 1, \ldots, p$, and $(\cdot)^T$ denotes the transpose operation. Let N_i be the random number of calls that are received in period i, and let $S_i(y)$ be the random number of those calls that are satisfactorily handled, $i = 1, \ldots, p$. We can express the service level constraint $E[S_i(y)]/E[N_i] \geq \pi$ as

$$g_i(y) = ES_i(y) - \pi EN_i \geq 0, \quad i = 1, \ldots, p.$$

Here $g_i(y)$ gives the expected number of successful services over and above the required expected number in period i. (Note that requiring that $E[S_i(y)/N_i] \geq \pi$ is not a valid way to enforce the constraint, since it gives more weight to service on days where the call volume is low. Also note that we include a separate service level constraint for each period to ensure uniform quality of service throughout the planning horizon, which is consistent with industry practices.)

For a staffing level vector y, there is an associated cost of covering the staffing levels with tours. The term "tour" refers to a collection of periods that a single employee works over the planning horizon. A tour must agree with all provisions of employee contracts, such as rules on meal breaks and a limit on the number of hours an agent can work over the planning horizon. All feasible tours are enumerated prior to problem formulation and stored in the tour matrix A such that $A_{ij} = 1$ if tour j includes period i, and 0 otherwise. Also, the jth component of the cost vector c is the cost of the jth tour. We let x be a vector with jth component x_j giving the number of employees working the jth tour. We assume that there are m feasible tours, so $x \in \mathbb{Z}_+^m$. Using this notation, the minimal cost of covering work requirement vector y is

$$\begin{aligned} f(y) = \min\ & c^T x \\ \text{s.t.}\ & Ax \geq y \\ & x \geq 0 \text{ and integer.} \end{aligned} \tag{1}$$

We assume that every period is covered by at least one tour, i.e., for every i there is at least one j such that $A_{ij} = 1$. It then follows that (1) is feasible for any y.

The call center staffing problem can now be formulated as

$$\begin{array}{lll} \min & f(y) & \\ \text{s.t.} & g_i(y) & \geq 0 \text{ for } i=1,\ldots,p \\ & y & \geq 0 \text{ and integer.} \end{array} \tag{2}$$

Recall that we cannot compute the (vector-valued) service level function $g(y)$ exactly, and instead use simulation. The call center staffing problem is then a simulation optimization problem. As in Atlason et al. (2004) we adopt "sample-average approximation" (see Shapiro 2003, Kleywegt et al. 2001) for solving such problems. This approach, specialized to our setting, is as follows. One first generates the simulation input data sets for n independent replications of the operations of the call center over the planning horizon. Each data set includes call arrival times and service times, and other input data needed to simulate operations of the call center at hand, if appropriate, e.g., call abandonment times and so forth. The simulation data sets are fixed, leading to a deterministic optimization problem that chooses staffing levels so as to minimize staffing cost, while ensuring that average service *computed only over the generated realizations* is satisfactory. We can then attempt to identify or develop deterministic optimization methods to solve the resulting problem.

Suppose that service level functions $g_i(y),\ i=1,\ldots,p$, are estimated by corresponding sample averages $\bar{g}_i(y;n)$, where n is the sample size used. The sample average approximation of the call center staffing problem is then

$$\begin{array}{lll} \min & f(y) & \\ \text{s.t.} & \bar{g}_i(y;n) & \geq 0 \text{ for } i=1,\ldots,p \\ & y & \geq 0 \text{ and integer.} \end{array} \tag{3}$$

Proposition 1 below gives conditions under which solutions to (3) converge to those of the true problem (2) as $n \to \infty$. We first make the following natural assumption about the cost vector.

ASSUMPTION 1. *The cost vector c is positive and integer valued.* [1]

Assumption 1 implies that $f(y)$ is integer valued. Moreover, since $c > 0$, the z-level set of f,

$$\{y \geq 0 \text{ and integer} : \exists x \geq 0 \text{ and integer}, Ax \geq y, c^T x \leq z\},$$

is finite for any $z \in \mathbb{R}$.

Let Y^* denote the set of optimal solutions to the true problem (2).

[1] The integrality part of Assumption 1 can be satisfied in practice by multiplying by the appropriate power of 10 if necessary, e.g., by 100 if the cost is in dollars and cents. This does not change the computational complexity of the problem, since the number of significant digits to represent the data remains the same. In this case attention should be paid to the choice of the weight parameter w defined in Section 3 to get a significant reduction in the objective value between the associated optimality cuts.

PROPOSITION 1. *Suppose that Assumption 1 holds. Suppose further that for each* y, $\bar{g}_i(y;n) \to g_i(y)$ *as* $n \to \infty$ *with probability 1. Finally, suppose that there is an optimal solution* $y^* \in Y^*$ *that is "strictly feasible," i.e.,* $g_i(y^*) > 0$ *for all* $i = 1, \ldots, p$. *If* y_n^* *is an optimal solution to* (3) *for each* n, *then* $y_n^* \in Y^*$ *for* n *sufficiently large, with probability 1.*

This result is easily proved using techniques very similar to those used in Atlason et al. (2004) and so we omit the proof. The requirement of a "strictly feasible" optimal solution is easily justified in practice: it is almost surely impossible for the service level function g_i to be exactly zero when y takes on only discrete values.

Proposition 1 establishes the validity of the sample average approximation approach in our setting. Much more can be said about the convergence of solutions of the sample average approximation problem and their objective values, but that is not the emphasis of this paper. We refer the interested reader to Atlason et al. (2004), Kleywegt et al. (2001).

3. The analytic center cutting plane method

In this section we state a version of the traditional analytic center cutting plane method (ACCPM) to fix ideas and provide a departure point for a reader unfamiliar with cutting plane methods in continuous optimization.

There are many cutting plane methods for solving convex optimization problems, including what may be termed boundary methods, such as Kelley's algorithm (Kelley, Jr. 1960) and its extensions (e.g., Westerlund and Pörn 2002), and interior point methods, recently reviewed by Mitchell (2003). We will focus our attention on one of the latter, namely the ACCPM, which was first implemented in du Merle (1995), as pointed out in Elhedhli and Goffin (2003). The ACCPM has proven effective in terms of both theoretical complexity (Atkinson and Vaidya 1995, Nesterov 1995, Goffin et al. 1996, Mitchell 2003) and practical performance on a variety of problems (Bahn et al. 1995, Mitchell 2000, and other references in Mitchell 2003). Software packages implementing the method are available (e.g., Peton and Vial 2001).

Many versions of the ACCPM for convex feasibility and optimization problems have been explored in the literature. The description we chose below borrows from Nesterov (1995), du Merle et al. (1998), and Mitchell (2003), among others.

Consider an optimization problem in the following general form:

$$\begin{array}{l} \min\ b^T y \\ \text{s.t. } y \in Y, \end{array} \tag{4}$$

where $Y \subset \mathbb{R}^n$ is a convex set, and $b \in \mathbb{R}^n$. (A problem of minimizing a general convex function over a convex set can be easily represented in this form.) To simplify the presentation, assume that the set Y is bounded and has a non-empty interior.

To apply the ACCPM (or any other cutting plane algorithm), the feasible region Y needs to be described by a separation oracle. Such an oracle will, given an input $\hat{y} \in \mathbb{R}^n$, either correctly assert that $\hat{y} \in Y$, or otherwise return a non-zero vector $q \in \mathbb{R}^n$ such that

$$q^T(y - \hat{y}) \geq 0 \ \forall y \in Y,$$

i.e., produce a *feasibility cut*. (Depending on how the set Y is described, the oracle might produce a deep/shallow cut, which has the same form as the constraint above, but a positive/negative right hand side.)

We now describe a typical iteration of the ACCPM. At the beginning of an iteration, we have available a finite upper bound z on the optimal value of (4), and a polyhedron $P = \{y \in \mathbb{R}^n : Dy \geq d\}$ that is known to contain the set Y. Here $D \in \mathbb{R}^{r \times n}$ and $d \in \mathbb{R}^r$ for some finite r. We first compute the (weighted) analytic center of the set $P \cap \{y \in \mathbb{R}^n : b^T y < z\}$ (for ease of presentation we assume that the set $P \cap \{y \in \mathbb{R}^n : b^T y < z\}$ is bounded), defined as the solution of the convex optimization problem

$$\min_{y \in \text{int}(P \cap \{y \in \mathbb{R}^n : b^T y \leq z\})} \{-w \log(z - b^T y) - \sum_{l=1}^{r} \log(D_{l\cdot} y - d_l)\}, \tag{5}$$

where $D_{l\cdot}$ is the lth row of D and $w > 0$ is a weight constant that affects the convergence rate of the algorithm (see, for example, du Merle et al. 1998). The set $P \cap \{y \in \mathbb{R}^n : b^T y < z\}$ is often referred to as the *localization set*, since it contains all feasible solutions with objective function values lower than z.

Finding a solution to (5) with high degree of precision is a relatively simple task from a practical standpoint and can be done, e.g., via Newton's method. Let $\hat{y}$ be the analytic center found. Next, the oracle is called with $\hat{y}$ as the input. If $\hat{y} \in Y$, then, by construction, $b^T \hat{y} < z$, and the upper bound is lowered by taking $z := b^T \hat{y}$. Otherwise, if $\hat{y} \notin Y$, the oracle will produce a vector q providing a feasibility cut, which is then added to the description of the polyhedron P. The procedure is then repeated. A slightly more detailed description of the algorithm is presented in Figure 1.

Intuitively, the algorithm's efficiency stems from the fact that at each iteration the cut being added passes through the analytic center of the localization set, which is often located near a geometric center. Thus, the volume of the localization set reduces rapidly with each iteration.

Suppose the feasible set Y is specified by $Y = \{y \in \mathbb{R}^n : g_i(y) \geq 0,\ i = 1, \ldots, p\}$, where the functions $g_i(y),\ i = 1, \ldots, p$ are pseudoconcave, as defined below:

Figure 1 The analytic center cutting plane method (ACCPM) for Problem (4)**.**

Initialization Start with a polyhedron $P^0 := \{y \in \mathbb{R}^n : D^0 y \geq d^0\}$ such that $Y \subset P^0$, a solution $y^* \in Y$, and an upper bound $z^0 := b^T y^*$. Let $w^0 > 0$, and set the iteration counter $k := 0$.
Step 1 If termination criterion is satisfied, then stop, and return y^* as a solution. Otherwise, solve Problem (5) with $w = w^k$, $z = z^k$, and $P = P^k := \{y \in \mathbb{R}^n : D^k \geq d^k\}$; let y^k be the solution.
Step 2a If $y^k \in Y$ then let $z^{k+1} := b^T y^k$, $y^* := y^k$, $D^{k+1} := D^k$ and $d^{k+1} := d^k$.
Step 2b If $y^k \notin Y$ then generate one or more feasibility cuts at y^k. Update D^k and d^k to include the new constraints, and let D^{k+1} and d^{k+1} represent the new constraint set. Let $z^{k+1} := z^k$.
Step 3 Compute w^{k+1} and let $k := k + 1$. Go to Step 1.

DEFINITION 1. (Cf. Definition 3.5.10 in Bazaraa et al. 1993) Let $h : S \to \mathbb{R}$ be differentiable on S, where S is a nonempty open set in $\mathbb{R}^n$. The function h is said to be *pseudoconcave* if for any $\hat{y}, y \in S$ with $\nabla h(\hat{y})^T (y - \hat{y}) \leq 0$ we have $h(y) \leq h(\hat{y})$. Equivalently, if $h(y) > h(\hat{y})$, then $\nabla h(\hat{y})^T (y - \hat{y}) > 0$. Pseudoconcavity of a function is a weaker property than concavity; see also Figure 3.12 in Bazaraa et al. (1993).

With Y in the above form, the feasibility cut at point $\hat{y} \notin Y$ which violates the ith constraint can be specified as

$$\nabla g_i(\hat{y})^T (y - \hat{y}) \geq 0, \tag{6}$$

since any solution y that satisfies $g_i(y) \geq 0$ also satisfies $g_i(y) > g_i(\hat{y})$.

4. A cutting plane method for discrete problems

In this section we describe how the ACCPM algorithm of Section 3 can be modified to solve the sample average approximation (3) of the call center staffing problem (2).

The most significant modification to the ACCPM for continuous problems is needed to take into account the fact that the feasible region of (3) is no longer a convex, or even connected, set, due to the integrality restriction on the variables. Cutting plane algorithms for nonlinear mixed integer programs have been explored in the past; see, for example, Westerlund and Pörn (2002). However, in this and other similar papers it is assumed that the constraint functions are in fact differentiable functions of continuous variables; the integrality restrictions on the variables are, in a sense, exogenous. In such a setting the concept of a convex (continuous) nonlinear relaxation of the integer program is straightforward, and feasibility cuts are generated simply using the gradients of these continuous functions. In our setting, however, the service level functions and their sample average approximations are not defined at non-integer values of y, and devising their continuous extension, especially one that is easy to work with from the algorithmic point of view, is non-trivial at best. Therefore, we take a different approach in adapting the ACCPM to the discrete case.

As far as we know, the use of ACCPM as a solution method for nonlinear integer programs has not been reported, although the method has been successfully used to solve the linear relaxation subproblems in branching algorithms for integer programs (see, for example, Mitchell (2000) and Elhedhli and Goffin (2003), among many others).

In Section 4.1 we extend the notion of pseudoconcavity to functions of integer variables, and show how feasibility cuts can be generated assuming that the functions $\bar{g}_i(y;n),\ i=1,\ldots,p$ are, in fact, discrete pseudoconcave. This leads to an ACCPM method for (mixed) integer programming problems satisfying the pseudoconcavity assumption; the algorithm is applicable to the types of problems considered in Westerlund and Pörn (2002), for example. We also discuss whether the S-shaped form of the service level functions in the call center staffing problem is consistent with this assumption, and propose alternative feasibility cuts at points where it is violated.

The following Section 4.2 discusses other modifications of the original algorithm for the problem (3) and details of our implementation. Section 4.3 gives a proof of convergence.

4.1. Discrete pseudoconcave functions and feasibility cuts

We begin by defining the notions of a discrete convex set and a discrete pseudoconcave function. We denote the convex hull of the set C by $\text{conv}(C)$.

DEFINITION 2. We say that the set $C \subseteq \mathbb{Z}^n$ is a *discrete convex set* if $C = \text{conv}(C) \cap \mathbb{Z}^n$, i.e., the set C equals the set of integer points in $\text{conv}(C)$.

DEFINITION 3. Let $C \subseteq \mathbb{Z}^n$ be a discrete convex set and $h : C \to \mathbb{R}$. Then h is *discrete pseudoconcave* on C if for any $\hat{y} \in C$ there exists a vector $q(\hat{y}) \in \mathbb{R}^n$ such that for any $y \in C$,

$$q(\hat{y})^T(y-\hat{y}) \leq 0 \Rightarrow h(y) \leq h(\hat{y}). \tag{7}$$

Equivalently, if $h(y) > h(\hat{y})$, then $q(\hat{y})^T(y-\hat{y}) > 0$. We call such a vector $q(\hat{y})$ a *pseudogradient* of h at $\hat{y}$.

Discrete pseudoconcavity can be viewed as a relaxation of the concave extensible function property defined in Murota (2003, p. 93). Note also the similarity between the above definition and Definition 1. For a differentiable pseudoconcave function, the gradient at point $\hat{y}$ plays the role of a pseudogradient $g(\hat{y})$ in condition (7).

If the functions $\bar{g}_i(y;n)$ in (3) are discrete pseudoconcave, then a feasibility cut at an integer point $\hat{y}$ that violates the ith constraint can be obtained in the form $\bar{q}^i(\hat{y};n)^T(y-\hat{y}) \geq \epsilon$, where $\bar{q}^i(\hat{y};n)$ is the pseudogradient of $\bar{g}_i(\cdot;n)$ at $\hat{y}$, and $\epsilon > 0$ is sufficiently small.

Are the service level functions we consider indeed discrete pseudoconcave? As an illustrative example, consider a simple call center with two 30-minute periods. The service times are exponential

with mean of 15 minutes. The arrival process is a nonhomogeneous Poisson process with rate varying between 42 and 120 calls/hour with an average load of 18 in period 1 and 25 in period 2 (*load* is the arrival rate divided by the service rate). Figure 2 shows the number of calls answered on time (here $\tau = 90$ seconds) in period 2 as a function of the staffing levels in periods 1 and 2. (Notice that this is equivalent to $\bar{g}$ with $\pi = 0$). The number of servers ranges from 1 to 30 in period 1 and from 1 to 40 in period 2. The function appears to follow a multi-dimensional extension of an S-shaped curve discussed in Section 1 (see also Ingolfsson et al. 2003, Atlason et al. 2004).

To verify that the function is discrete pseudoconcave one can solve, for each point, the linear (feasibility) program

$$\begin{array}{lll} \min & 0 & \\ \text{s.t.} & q^T(y-\hat{y}) & \geq 1 \; \forall \, y \in \{y : \bar{g}_i(y;n) > \bar{g}_i(\hat{y};n)\} \\ & q & \geq 0. \end{array} \tag{8}$$

Without the nonnegativity constraint, Problem (8) has a solution q if and only if a pseudogradient exists, and then q is such a pseudogradient. We added this constraint, since the service level function can reasonably be assumed to be nondecreasing in each component of y. The number of constraints in Problem (8) can be very large and it cannot be constructed without simulating the service level function for all practical values of y. In practice, however, one can solve the LP using only the points visited by the algorithm to check whether they are consistent with the assumption of pseudoconcavity.

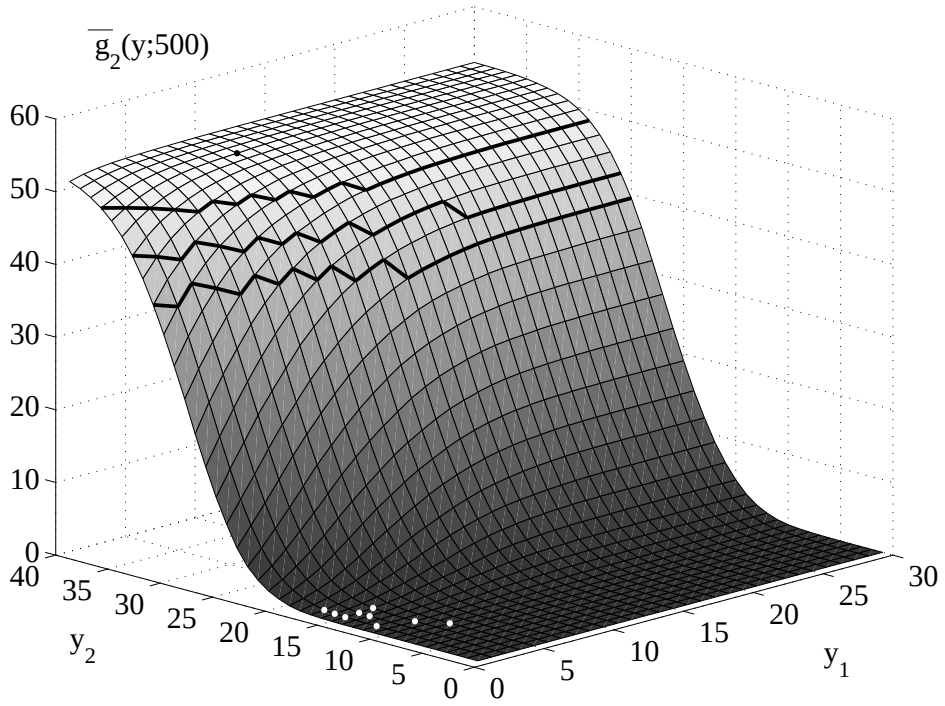
The results of Problem (8) for the function in Figure 2 indicate that the function is very close to being discrete pseudoconcave. There are only a few points in the entire domain where a pseudogradient does not exist. These points are depicted in Figure 2 by the white dots on the bottom and a single black dot on top of the graph. Note that these points are in relatively flat areas of the graph. In the next section we describe a simple way to advance the algorithm at such points, which we successfully used in the implementation of our cutting plane method.

Moreover, in a flat area, for a finite sample size, each additional call answered on time can have a significant effect on the order of the values of the service level function. Therefore, it seems reasonable to assume that this situation would be alleviated for larger sample sizes, and in particular, to assume that the *expected* service level function is discrete pseudoconcave. Thus, to prove the convergence results in Section 4.3 we assume that the sample averages of the service level functions are discrete pseudoconcave.

4.2. A simulation-based cutting plane method for the call center staffing problem with pseudoconcave service level functions

In this subsection we describe our modification of ACCPM for Problem (3). At the beginning of a typical iteration, we have available a feasible solution y^* of (3), and an upper bound $z = f(y^*)$ on

Figure 2 **The sample average (sample size $n = 500$) of the number of calls answered on time in period 2 as a function of the staffing levels in periods 1 and 2.**



Note. The white dots on the bottom and the black dot on the top represent points where a pseudogradient does not exist. The black lines show the boundaries of the satisfactory service levels for π equal to 70%, 80% and 90%.

the optimal value of the problem. The point y^* is the best feasible solution found by the algorithm so far. We also have available a polyhedron $P = \{y \in \mathbb{R}^p : y \geq 0,\ Dy \geq d\}$ that is known to contain the feasible region.

Suppose y^*, z and P are as above. If y^* is not an optimal solution, then, since $f(y)$ takes on integer values, the localization set

$$\{y \in P : f(y) \leq z - 1,\ y \text{ integer}\}$$

is nonempty and contains all feasible solutions with objective values better than z. The localization set is empty precisely if y^* is an optimal solution. This observation provides grounds for the termination criteria we specify in our algorithm.

Computing the next iterate. First we find the analytic center of a polyhedral relaxation of the localization set. In particular, we solve the following optimization problem:

$$\begin{array}{ll} \min & -w\log(z - a - c^T x) - \sum_{i=1}^{p} \log y_i - \sum_{j=1}^{r} \log(D_{j\cdot} y - d_j) \\ \text{s.t.} & Ax \geq y \\ & x \geq 0, \end{array} \tag{9}$$

where the constant $a \in (0, 1)$ ensures that only points with an objective value better than z are considered. Due to Assumption 1 the effective feasible region of (9) is bounded; hence the problem

has an optimal solution so long as it has a feasible point in the effective domain of the objective. If (9) has no solution, the algorithm terminates with y^* as an optimal solution; otherwise, let $(y^{\text{ac}}, x^{\text{ac}})$ be a solution of (9). Here, $w > 0$ is the weight constant; we discuss later how this constant is determined in the algorithm.

Note that the analytic center found in the previous step is almost certainly not integer, and rounding y^{ac} to the nearest integer can result in a point outside of the localization set. To capitalize on the centrality of the analytic center in the localization set, we instead find the closest integer point in the effective feasible region of (9), i.e., solve

$$\begin{array}{lll} \min & \|y - y^{\text{ac}}\| & \\ \text{s.t.} & Ax & \geq y \\ & c^T x & \leq z - 1 \\ & Dy & \geq d \\ & x, y & \geq 0 \text{ and integer.} \end{array} \tag{10}$$

If (10) is infeasible, the algorithm terminates with y^* as an optimal solution; otherwise, let $(\hat{y}, \hat{x})$ be the solution of (10) and choose $\hat{y}$ as the next iterate. Here, $\|y - y^{\text{ac}}\|$ is a measure of the distance between y and y^{ac}. If we choose the L_1-norm as the measure, i.e., $\|y - y^{\text{ac}}\| = \sum_{i=1}^{p} |y_i - y_i^{\text{ac}}|$, then (10) is a linear integer program. We discuss the computational requirements of solving this problem at each iteration when we talk about the overall computational effort in relation to the computational experiments.

Estimating the service levels. Next we compute $\bar{g}_i(\hat{y}; n)$ for all i via simulation.

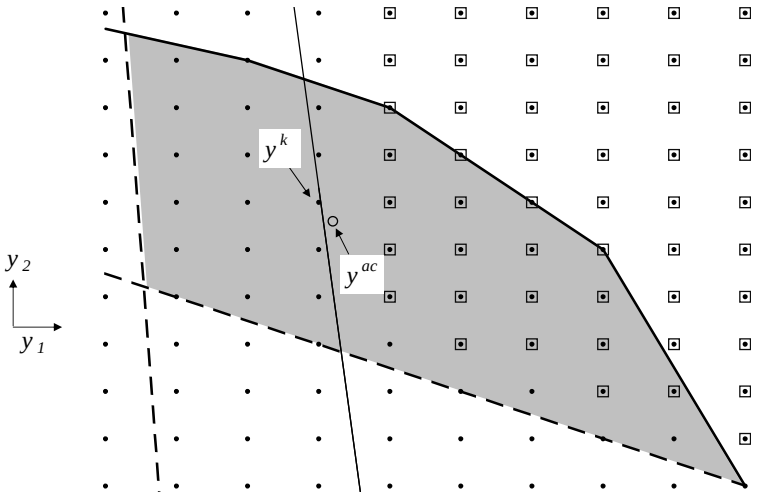
Adding an optimality cut. If $\bar{g}_i(\hat{y}; n) \geq 0$ for all i, then $\hat{y}$ satisfies the service level requirements. Since $c^T \hat{x} \leq z - 1$, $\hat{y}$ is contained in the localization set, i.e., it is the best staffing level so far. Note that $c^T \hat{x}$ is not necessarily the cost associated with staffing level $\hat{y}$, since $c^T x$ is not being minimized in (10). To compute the cost associated with $\hat{y}$ we instead solve (1) to get $f(\hat{y})$ and update $z := f(\hat{y})$.

Adding a feasibility cut. If $\bar{g}_i(\hat{y}; n) < 0$ for some i, then we add a feasibility cut for each i such that $\bar{g}_i(\hat{y}; n) < 0$. In particular, we estimate a pseudogradient, $\bar{q}^i(\hat{y}; n)$, of $\bar{g}_i(\cdot; n)$ at $\hat{y}$ (we will discuss techniques for estimating the pseudogradients in Section 5.2). If $\bar{q}^i(\hat{y}; n) \neq 0$, we add a feasibility cut of the form

$$\bar{q}^i(\hat{y}; n)^T y \geq \bar{q}^i(\hat{y}; n)^T \hat{y} + \epsilon \tag{11}$$

for some small constant $\epsilon > 0$ (we discuss the role of ϵ in the discussion of the convergence of the method in Section 4.3). If $\bar{q}^i(\hat{y}; n) = 0$, the feasibility cut takes the form of a lower bound on the number of servers (see discussion in Section 5.2). We update D and d to reflect the cuts added.

Figure 3 Illustration of the feasible region and an iterate in the SACCPM.



The above procedure is then repeated. An illustration of the localization set and the feasible regions and solutions of problems (9) and (10) in each iteration is shown in Figure 3. The picture shows points in the space of the y variables in the case $p = 2$. The shaded area represents the localization set $\{y \in P : f(y) \leq z - 1\}$ (ignoring integer constraints). The squares represent the points that are feasible for the sample average approximation problem (3). The thick solid line segments result from an optimality cut $f(y) \leq z - 1$ and the dotted lines represent the feasibility constraints $Dy \geq d$. Point y^{ac} is the solution of the analytic center problem, and y^k is the closest integer in the localization set. The line just to the right of y^k represents a feasibility cut that would be added in this iteration.

We summarize the simulation-based analytic center cutting plane method (SACCPM) for the call center staffing problem in Figure 4. (To shorten the presentation, the description of the algorithm in Figure 4 is only specified for the case when the constraint functions $\bar{g}_i(\cdot; n)$ are in fact discrete pseudoconcave.)

The weight parameter w on the optimality cut can be increased to "push" the weighted analytic center away from the optimality cuts. There are no theoretical results on how to choose the weights, but some computational experiments have been done to test different values of the weights, and in fact, weight parameters on the feasibility cuts (Goffin et al. 1992, du Merle et al. 1998). The choice of the weights is problem and data dependent (see Section 5.2 for the particular choice used in our implementation).

In practice it is useful to maintain a lower bound on the optimal value of the problem (3) throughout the algorithm, in addition to the upper bound z. In particular, the algorithm can be terminated as soon as the gap between the upper and lower bounds is sufficiently small, indicating

Figure 4 The simulation-based analytic center cutting plane method (SACCPM).

Initialization Start with a solution y^0 feasible to Problem (3). Let $z^0 := f(y^0)$, and $y^* := y^0$. Let $P^0 = \{y \geq 0 : D^0 y \geq d^0\}$, where D^0 and d^0 are empty. Choose an $\epsilon > 0$, $a \in (0,1)$ and $w^0 > 0$. Let $k := 0$.
Step 1 Solve Problem (9) with $w = w^k$, $z = z^k$, and $P = P^k$. If (9) does not have a (feasible) solution, then terminate with y^* as the optimal solution and z^k as the optimal value; otherwise let y^{ac} be the solution of (9).
Step 2 Solve Problem (10) with $z = z^k$ and $P = P^k$. If (10) is infeasible, then terminate with y^* as the optimal solution and z^k as the optimal value; otherwise let y^k be the optimal solution of (10) found.
Step 3a If $\bar{g}_i(y^k; n) \geq 0 \, \forall \, i = 1, \ldots, p$, let $z^{k+1} := f(y^k)$, $y^* := y^k$, $D^{k+1} := D^k$ and $d^{k+1} := d^k$.
Step 3b If $\bar{g}_i(y^k; n) < 0$ for some $i = 1, \ldots, p$, then add the constraint $\bar{q}^i(y^k; n)^T y \geq \bar{q}^i(y^k; n)^T y^k + \epsilon$ for each i such that $\bar{g}_i(y^k; n) < 0$. Update D^k and d^k to include the added inequalities, and let $P^{k+1} = \{y \geq 0 : D^{k+1} y \geq d^{k+1}\}$ represent the new constraint set. Let $z^{k+1} := z^k$.
Step 4 Compute w^{k+1} and let $k := k + 1$. Go to Step 1.

that the current "incumbent" y^* is a sufficiently good solution of the problem. A lower bound can be found as the optimal objective value of

$$\begin{array}{lll} \min & f(y) & \\ \text{s.t.} & Dy & \geq d \\ & y & \geq 0 \text{ and integer,} \end{array} \tag{12}$$

or, for a weaker but easier to compute bound, of the linear programming relaxation of (12).

4.3. Convergence of the SACCPM

In this section we give conditions under which the SACCPM terminates with y^* as an optimal solution of (3). First, we argue that the algorithm does indeed terminate and then we show that y^* is an optimal solution of (3) at termination. To prove the results we make the following two assumptions.

ASSUMPTION 2. *The functions $\bar{g}_i(y; n)$ are discrete pseudoconcave in y for all $i = 1, \ldots, p$.*

ASSUMPTION 3. *In the implementation of the SACCPM, $\bar{q}^i(\hat{y}; n)$ is a pseudogradient of $\bar{g}_i(\cdot; n)$ at $\hat{y}$.*

We also define the sets

$$\begin{array}{ll} \Gamma & := \{y \geq 0 \text{ and integer} : f(y) \leq z^0\}, \\ \Psi & := \Gamma \cap \{y : \bar{g}_i(y; n) \geq 0 \, \forall \, i = 1, \ldots p\}, \\ \Upsilon & := \Gamma \setminus \Psi \text{ and} \\ I(y) & := \{i : \bar{g}_i(y; n) < 0\}. \end{array}$$

In words, Γ is the set of points that are potentially visited by the algorithm, and it contains the set of optimal solutions. The set Ψ is the set of points in Γ that are feasible for the sample average approximation problem (3). The set Υ is the set of points in Γ that are not feasible for the sample

average approximation problem (3). The set $I(y)$ is the set of periods in which the sample average of the service level function is not acceptable given the staffing levels y. The following lemma says that all solutions in Ψ satisfy potential feasibility cuts (11) for some appropriately chosen ϵ.

LEMMA 1. *Let $\bar{q}^i(\hat{y};n)$ be a pseudogradient of $\bar{g}_i(\cdot;n)$ at $\hat{y}$ and suppose Assumptions 1 and 2 hold. Then there exists an $\tilde{\epsilon}>0$ such that $\bar{q}^i(\hat{y};n)^T(y-\hat{y}) \geq \tilde{\epsilon}\ \forall\, y\in\Psi, \hat{y}\in\Upsilon$, $i\in I(\hat{y})$.*

Proof. Let $\hat{y}\in\Upsilon$ be fixed. Suppose that $\bar{q}^i(\hat{y};n)^T(y-\hat{y})\leq 0$ for some $y\in\Psi$ and $i\in I(\hat{y})$. Then $\bar{g}_i(y;n)\leq \bar{g}_i(\hat{y};n)<0$, where the first inequality follows by Assumption 2 and the second inequality follows since $i\in I(\hat{y})$. This is a contradiction since $y\in\Psi$, and therefore $\bar{q}^i(\hat{y};n)^T(y-\hat{y})>0\ \forall\, y\in\Psi$, $i\in I(\hat{y})$.

Let $\epsilon(\hat{y}) = \min_{i\in I(\hat{y})}\min_{y\in\Psi}\bar{q}^i(\hat{y};n)^T(y-\hat{y})$. The set Ψ is finite by Assumption 1; therefore, the inner minimum is attained for some $y\in\Psi$. Since $I(\hat{y})$ is also finite, the outer minimum is also attained for some $i\in I(\hat{y})$. Therefore $\epsilon(\hat{y})>0$.

Finally, let $\tilde{\epsilon} = \min_{\hat{y}\in\Upsilon}\epsilon(\hat{y})$. Then $\tilde{\epsilon}>0$ since Υ is finite. □

THEOREM 1. *Suppose* (3) *has an optimal solution and that Assumptions 1, 2 and 3 hold. Furthermore, let $0<\epsilon\leq\tilde{\epsilon}$, where $\tilde{\epsilon}$ is as in Lemma 1. Then the SACCPM terminates in a finite number of iterations returning y^*, which is an optimal solution of* (3).

Proof. The SACCPM only visits points that are feasible solutions of (10). The set of feasible solutions of (10) at every iteration is a subset of Γ and therefore is finite by Assumption 1. Suppose that the kth iterate y^k is the same as a previous iterate y^j for some $j<k$. If y^j is in Ψ then $f(y^k)\leq z-1\leq f(y^j)-1$, where the second inequality follows since z is the cost of the best feasible solution visited before iteration k. This is a contradiction, so y^j is not in Ψ. On the other hand, if y^j is in Υ then $\bar{g}_i(y^j;n)<0$ for some $i\in I(y^j)$. Since y^k is in P, $\bar{q}^i(y^j;n)^T(y^k-y^j)\geq\epsilon>0$ which is also a contradiction and y^k can therefore not be equal to y^j for any $j<k$. Therefore, the SACCPM does not visit any point in Γ more than once. Since Γ is a finite set, the algorithm is finite.

By Lemma 1 the feasibility cuts never cut off any of the feasible solutions of (3). When at termination problems (9) or (10) do not have a feasible solution it means that all the feasible solutions for (3) have an objective value greater than $z-1$. But y^* is feasible for (3) and has an objective value of z and is, therefore, optimal, since $f(y)$ is integer by Assumption 1. □

The theorem says that there exists an $\tilde{\epsilon}$ such that the algorithm terminates with an optimal solution of (3) if $0<\epsilon\leq\tilde{\epsilon}$. In practice, the value of $\tilde{\epsilon}$ is unknown, but in general a "small" value for ϵ should be chosen, as in Section 5.4.

5. Numerical results

In this section we give an implementation of the SACCPM for a call center with time varying arrival rates. In Section 5.1 we describe the example that we use for the numerical experiments. In Section 5.2 we discuss how to compute, or estimate, the pseudogradients, and we also describe how the algorithm was implemented. We compare the results with staffing levels obtained by analytical queueing methods (see Section 1 for references). Analytical queueing methods based on the Erlang C formula (14) are widely used to determine staffing levels in call centers (Cleveland and Mayben 1997). In Green et al. (2001, 2003) several heuristics for improving the performance of the basic Erlang C model are evaluated and we give a summary of these heuristics in Section 5.3. We believe that the methods in Green et al. (2001, 2003) are among the best analytical methods for determining required staffing levels to answer a minimum fraction of calls before a threshold time limit. Therefore we use these methods as benchmarks for our method. The results of our experiments are in Section 5.4 and we comment on the computational requirements of the SACCPM in Section 5.5.

5.1. Example

Our test model is similar to the models used in Green et al. (2001, 2003), which are call centers that can be modeled as $M(t)/M/s(t)$ queueing systems. In the Green et al. (2001) paper a call center operating 24 hours a day and 7 days a week is studied, while the subject of the Green et al. (2003) paper are call centers with limited hours of operation. We consider a call center with limited hours of operation that is open from 6am to 12am.

The call center has the following additional characteristics. The planning horizon consists of a single day's operation. The 18 hour planning horizon is broken into 72 time periods, each of length 15 minutes. In each period 80% of calls should be answered immediately. This is equivalent to setting $\tau = 0$ and $\pi = 0.8$. The service times are independent exponential random variables with rate μ; see Table 1. The size of the system is measured by average load R, which is the average arrival rate divided by the service rate. The arrival process on any given day is a nonhomogeneous Poisson process with continuous, piecewise-linear arrival rate defined at the end of period i by $\lambda_i = \lambda(1 + \theta \sin(2\pi(t_i - 6)/18))$, where t_i is the end time of period i and is measured in the hour of the day (e.g., $t_0 = 6$ and $t_{72} = 24$). The average daily arrival rate is $\lambda = R\mu$ and θ is the relative amplitude. Calls are answered in the order they are received. When there is a reduction in the number of servers at the end of a period, outgoing servers complete calls that they are already servicing before ending their shifts. A new call cannot enter service until there are fewer calls in service than there are servers for the new period.

Table 1 The parameter settings in the experiments.

Parameter	Low	High
μ	4 calls/hour	16 calls/hour
R	8	32
θ	.25	.75
Shifts	Yes	No

We study the performance both in the presence of shift constraints and when there are no shift constraints. When there are no shift constraints, the tour matrix A is the identity matrix in $\mathbb{R}^p$, and the cost for each tour is equal to 1 agent-period, i.e., the total staffing cost equals the total number of servers over all 72 periods. We assume that the shift constraints are such that each tour covers 6 hours, or 24 periods. The tours can only start on the hour and no later than 6 hours before the end of the day. This results in 13 tours: 6am-12pm, 7am-1pm, ..., 5pm-11pm and 6pm-12am. There are no meal-type breaks incorporated in the shifts, but they can easily be included by changing the appropriate values in the tour matrix A. The cost for each tour is equal to 24 agent-periods per tour.

Note that we are yet to specify a value for μ, R and θ. In the experiments, we study how the SACCPM performs under two different settings (high and low) of each of these parameters. Along with the two settings for the shift constraints this results in a total of $2^4 = 16$ experiments. The high and low values for each parameter are given in Table 1. Green et al. (2001, 2003) additionally studied the effect of the target probability, π. The target level did not appear to have as significant an effect on the reliability of each method as the other parameters did, so we do not include different settings of it in our study. Instead of shifts, they included a factor that they call the planning period which is similar to the shift factor in that the number of servers is constant in each planning period (which can be longer than what we, and they, call a period and measure the performance in).

5.2. Estimating the pseudogradients and implementing the SACCPM

Before we describe the implementation of the continuous and discrete optimization problems solved in each iteration of the SACCPM, we discuss what is perhaps the most challenging part of the implementation of the algorithm: estimating the pseudogradients. Up until now we have assumed that in the implementation of the SACCPM, $\bar{q}^i(\hat{y}; n)$ is a pseudogradient of $\bar{g}_i(\cdot; n)$ at $\hat{y}$. What makes computing (or estimating, since it may prove difficult to ensure that the resulting vector satisfies condition (7) exactly) such a pseudogradient particularly challenging is that the service

level function is a discrete function of the number of agents. In addition, we do not have a closed form expression of the function.

The simplest and perhaps the most intuitive method for estimating a gradient (or pseudogradient) when an expression for the function is unknown is the method of finite differences (see, e.g., Andradóttir 1998). The method of finite differences can easily be extended to discrete functions, i.e., we let

$$\bar{q}^i_j(\hat{y};n) = \bar{g}_i(\hat{y}+e_j;n) - \bar{g}_i(\hat{y};n) \qquad \forall\, j=1,\ldots,p, \tag{13}$$

where e_j is the jth unit vector in $\mathbb{R}^p$, be the estimate of the pseudogradient of $\bar{g}_i(y;n)$ at the staffing level $\bar{y}$. It is clear that $p+1$ function evaluations of $\bar{g}_i$ are required to compute $\bar{q}^i$. We note, however, that estimates of the pseudogradients of the service level functions $\bar{g}_i$ in *all* p periods, $i=1,\ldots,p$, can be obtained from those same $p+1$ simulations.

The number of periods p can be quite large in staffing applications. Therefore, if we could estimate a pseudogradient from a single simulation, the computation time of the SACCPM would be greatly reduced. Atlason (2004) includes, however, an implementation of the SACCPM using infinitesimal perturbation analysis to estimate pseudogradients, and the solutions from that algorithm are inferior to solutions that the algorithm gives when finite differences are used. Pseudogradient estimates based on the likelihood ratio gradient estimation method have also not been reliable (Atlason 2004, Atlason et al. 2003).

At low staffing levels there are so few servers, and calls are so backed up, that no calls are answered on time, and adding a server does little to alleviate the situation; cf. low values of y_1 and y_2 in Figure 2. It is certainly possible that we would encounter such a staffing level in the cutting plane algorithm for the sample average approximation of the service level function; an attempt to compute or estimate a pseudogradient at this point would produce a zero vector. Therefore, as a feasibility cut at such a point $\hat{y}$, we impose a lower bound on the number of servers in the period i, i.e., add the constraint $y_i \geq \hat{y}_i + 1$. This is not necessarily a valid feasibility cut, since the reason for calls backing up might be under-staffing in previous periods. In our implementation, if such a cut is added during the algorithm, we verify at termination that the corresponding constraint is not tight at the optimal solution found. If it is tight, then we lower the bound and do more iterations of the cutting plane method.

Other parts of the SACCPM were implemented as follows. We used a sample size of $n=100$. We built a simulation model using the ProModel simulation software to compute the sample average of the service level function. We used Visual Basic for Applications and Microsoft Excel to store the data and to compute the cuts. We used the AMPL modeling language to model, and call a

solver for, the analytic center problem (9) and the IP (10) of finding an integer point close to the analytic center. We used the MINOS solver to solve the analytic center problem (9). We used the CPLEX solver to solve the IP (10). We used the Excel solver to compute the cost $f(\hat{y})$ of a particular staffing level $\hat{y}$.

Finally, we describe the settings of the parameters that are specific to the SACCPM. We let $w^k = r$, where r is the number of feasibility cuts that have been added in iterations 1 through $k-1$ (we let $w^k = 1$ if no feasibility cuts have been added). We used $\epsilon = 10^{-5}$ and chose $a = 1-\epsilon$. Instead of specifying a particular feasible starting point y^0, we started with an upper bound on the staffing levels in each period to ensure boundedness of the localization set. For the parameters chosen in the experiment it is unlikely that more than 130 agents will be allocated in any period in an optimal solution. Hence we added the term $-\sum_{i=1}^{p} \log(130 - y_i)$ to the objective of the analytic center problem (9) and the constraints $y_i \leq 130 \,\forall\, i = 1, \ldots, p$ to the IP (10).

5.3. Analytical queueing methods

In this section we describe the queueing methods in Green et al. (2001, 2003) that we use as benchmarks for our computational study. Traditional queueing methods for staffing call centers assume that the process is in steady state, i.e., the arrival rate is fixed and the call center has been open long enough, so that the initial state of the call center does not matter. Then assuming that the call center can be modeled as an $M/M/s$ queueing system, the probability that a customer has to wait more than τ time units, as a function of the number of servers s, is given by (Cooper 1981, p. 91,97)

$$P(s) = C(s,R)e^{-(s\mu-\lambda)\tau} \text{ for } s > R, \tag{14}$$

where $R = \lambda/\mu$ is the offered load. If $s \leq R$ then $P(s) = 1$. The term $C(s,R)$ is the probability that an incoming call is delayed, also called the Erlang-C probability of delay. It can be computed in a numerically stable way via the recursion (Cooper 1981, p. 82,92)

$$\begin{aligned} B(0,R) &= 1, \\ B(s,R) &= \tfrac{R \cdot B(s-1,R)}{s + R \cdot B(s-1,R)} \text{ for } s \geq 1 \text{ and} \\ C(s,R) &= \tfrac{sB(s,R)}{R \cdot B(s,R) + s - R} \text{ for } s > R. \end{aligned}$$

When the arrival rate changes between periods or within periods, Green et al. (2001) proposed adjusting the value of the arrival rate λ used in (14). This is a straightforward method that can give good staffing levels, at least for a call center operation that is similar to the $M(t)/M/s(t)$ model. They consider 6 different adjustments of the arrival rate. The 6 different schemes are given in Table 2. In Green et al. (2003) only SIPPavg, LAGavg and LAGmax are considered. When we

Table 2 Different methods for adjusting the arrival rate to use in Equation (14).

Method	Λ_i
SIPPavg	$\int_{t_{i-1}}^{t_i} \lambda(t)dt$
SIPPmax	$\max_{t_{i-1}<t\le t_i} \lambda(t)$
SIPPmix	If $\lambda(t)$ is nondecreasing in period i use SIPPavg rate, otherwise use SIPPmax rate.
LAGavg	$\int_{t_{i-1}-1/\mu}^{t_i-1/\mu} \lambda(t)dt$
LAGmax	$\max_{t_{i-1}-1/\mu<t\le t_i-1/\mu} \lambda(t)$
LAGmix	If $\lambda(t)$ is nondecreasing in $[t_{i-1}-1/\mu, t_i-1/\mu]$ use LAGavg rate, otherwise use LAGmax rate.

Here, t_i is the time when period i ends ($t_0 \equiv 0$), $1/\mu$ is the mean service time, and Λ_i is the rate to be used to determine the staffing in period i.

computed the arrival rate to use for the LAG methods in the first period we assumed that the arrival rate prior to time zero was equal to the arrival rate at the beginning of the first period.

The required staffing, y_i, in period i is computed by letting $\lambda = \Lambda_i$ in (14) and letting

$$y_i = \min\{s > 0 \text{ and integer} : P(s) \le 1 - \pi\}.$$

The cost of the resulting staffing level is $f(y)$.

5.4. Results

The parameter settings and the cost of the staffing levels obtained by each method in each of the 16 experiments are shown in Table 3. The table also shows the cost savings of the SACCPM versus the LAGavg method. We chose the LAGavg method for comparison because it performs best of the 6 analytical methods. The comparison only includes feasible solution determined by the criteria described below.

Determining feasibility. The solutions obtained by the 6 queueing methods and the SACCPM are not guaranteed to be feasible for the call center staffing problem with expected service level constraints (2) (although it is true that a solution obtained by the SACCPM is feasible for the respective sample average approximation problem). To determine feasibility we increased the sample size to $n = 999$ (the maximum in ProModel) and simulated the call center using each of the solutions found, at the staffing level vector Ax. We simulated at Ax rather than y, since the ith component of the vector Ax is the actual number of agents available in period i and can be greater than y_i because of slack in the shift constraint $Ax \ge y$ in (1). Because the simulation output of the 999 experiments is still random, we decided to declare the service level requirements not met

if the fraction of calls answered immediately is less than 75% in any period instead of the $\pi = 80\%$ from the example and was used in the implementation of the SACCPM algorithm. This is similar to what Green et al. (2001) used to determine feasibility. They computed the service level using numerical integration of transient queueing formulas and said that the service level is infeasible in a period if the target probability is exceeded by at least 10%. We chose 75% as our threshold because the largest 95% confidence interval half-width in our simulations using $n = 999$ was 4.4%, and $80\% - 4.4\% \approx 75\%$.

The feasibility of the solutions is reported in Table 4. We first note that all the methods do well in most cases. The SIPPavg method struggles when there are no shifts present and the service rates are low. In this case there is significant dependence between time periods, which none of the SIPP methods take into account. We also note that the solutions from the SACCPM do not always meet the service level requirements by the 75% criterion and in many periods fail to meet the 80% requirements. This is because the sample size was only 100 when the solution was computed, so there is a significant error in the estimates of the service level functions.

Ranking the methods. In Table 3 we put in boldface text the cost of the method that we declared a "winner." We selected a winner based on two criteria. The first criterion is that the service level must be greater than 75% in all periods based on the simulation using sample size 999. The second criterion is cost. We see that the SACCPM is a winner in all but three experiments. In two of these the solutions fail to meet the feasibility criterion. In the third case the cost of the best solution was only 0.3% lower than that obtained by SACCPM. In the case of ties it would make more sense to use one of the analytical queueing heuristics because they are much easier to implement. However, one would not know beforehand whether a tie would occur, and which heuristic to use.

The analytical queueing heuristics have been shown to do well on this particular type of queueing model. In the SACCPM, however, no assumptions are made on the arrival process or the service times, so it may apply in even more general settings.

In Section 5.2 we noted that the finite difference estimator of a pseudogradient for low staffing levels can be zero. In one or more iterations in some of the experiments the finite difference method produced a zero vector, which could not be used to generate feasibility cuts. Instead we imposed the lower bounds described in Section 5.2 and then checked upon termination of the algorithm whether these lower bounds were tight. The bounds were tight in experiment 16, so we relaxed the bounds and ran the algorithm until it terminated again, this time with a solution where these kinds of bounds were not tight.

5.5. Computational requirements

We can divide each iteration of the algorithm into 3 main parts in terms of computations:

1. *Solve the analytic center problem* (9). This usually took less than 1 second using the MINOS solver and never took more than 3 seconds.

2. *Solve the IP* (10) *to get an integer solution close to the analytic center.* In the initial stages of the algorithm this takes on the order of seconds to solve. However, when there were no shifts, meaning that the solution space for y is larger, it could take millions of branch and bound nodes to find an optimal solution after a number of cuts were added. Since it is not necessary to find an optimal solution of the IP (10) to advance the algorithm, we often terminated the IP with a suboptimal, but feasible, solution to determine the next iterate. If the IP seemed infeasible, but the infeasibility was difficult to prove and the optimality gap in the SACCPM was less than 1%, the SACCPM was terminated with a possibly suboptimal solution.

Solving the IPs was much easier when the shift constraints were included, which is the scenario one usually faces in practice. In that case, an optimal solution of the IP was found, or the IP was deemed infeasible, in a matter of seconds.

3. *Simulate to estimate the expected service level and gradients.* Simulating at each staffing level in ProModel took from 6 seconds to about 1 minute, depending on the number of arrivals to the system, on a Pentium 4 3GHz computer. Up to 73 such simulations are required per iteration, although we did not always compute all 72 differences to compute the FD pseudogradient. It appeared that dependence between time periods was not a factor over more than 10 periods, so as a rule of thumb we only computed the differences for the 10 periods preceding period i and for period i if the service level constraint was not satisfied in period i. Staffing levels in subsequent periods do not have any effect on the service level in period i since the requirement is to answer 80% of the calls immediately.

Hence, estimating the pseudogradients and solving the IP (10) requires the most computation; see Section 6 for ideas on how the computational requirements can be reduced.

The number of iterations required is given in Table 4. In the initial stages, the SACCPM sometimes produced several optimality cuts before any feasibility cuts were generated. Because the weight on the optimality cuts is only 1 in the beginning, the optimality cuts will be fairly close to each other in such a case, and progress will be slow. This happens when the starting point has much higher staffing levels than what is really needed. When this occurred, we added deeper optimality cuts until the first feasibility cuts were generated by the algorithm, i.e., we lowered z. One could start the algorithm close to the solutions of the analytical queueing heuristics, to hopefully speed up the convergence. We did not try to implement this approach.

Table 3 Cost of the solutions in agent-periods.

Experiment	μ	R	θ	Shifts	λ	SACCPM	SIPPavg	SIPPmax	SIPPmix	LAGavg	LAGmax	LAGmix	Savings
1	4	8	0.75	Y	32	**1008**	1056	1056	1056	1056	1056	1056	4.6%
2	16	8	0.75	Y	128	**1032**	1056	1056	1056	**1032**	1056	**1032**	0.0%
3	4	32	0.75	Y	128	**3456**	3552	3624	3576	**3456**	3552	3552	0.0%
4	16	32	0.75	Y	512	**3504**	3552	3624	3576	3576	3576	3528	2.0%
5	4	8	0.25	Y	32	**936**	**936**	**936**	**936**	**936**	**936**	**936**	0.0%
6	16	8	0.25	Y	128	**936**	**936**	**936**	**936**	**936**	**936**	**936**	0.0%
7	4	32	0.25	Y	128	**3024**	3048	3096	3072	3048	3048	3048	0.8%
8	16	32	0.25	Y	512	**2976**	3048	3096	3072	3024	3072	3048	1.6%
9	4	8	0.75	N	32	829	848	862	855	**848**	862	855	-
10	16	8	0.75	N	128	**838**	848	858	853	847	862	853	1.1%
11	4	32	0.75	N	128	2746	2786	2838	2812	2787	2838	**2813**	-
12	16	32	0.75	N	512	2786	2786	2838	2812	**2778**	2830	2804	-0.3%
13	4	8	0.25	N	32	**846**	856	862	859	856	862	859	1.2%
14	16	8	0.25	N	128	**850**	854	860	857	856	861	859	0.7%
15	4	32	0.25	N	128	**2774**	2802	2818	2810	2802	2818	2810	1.0%
16	16	32	0.25	N	512	**2790**	2802	2818	2810	2797	2815	2806	0.3%

The bold numbers in each row are the lowest cost solutions out of the solutions that satisfy the feasibility requirements; see Table 4. "Savings" shows the cost savings of using the SACCPM versus the LAGavg method.

6. Conclusions and future research

The SACCPM presented in this paper is a promising method for computing optimal staffing levels in a call center when traditional methods fail. Although the computational requirements of our method are large, we were able to solve, or at least approximately solve, a number of moderately sized hypothetical call center staffing problems. The results of the SACCPM were encouraging and they show that the method can potentially be applied in real world situations with good results. Furthermore, the algorithm can be automated, so that no user intervention is required while the algorithm is running.

The algorithm is robust in the sense that it works well on a range of different problems which gives it further credibility over the traditional methods. Compared to the traditional queueing methods, the SACCPM does best in the presence of shift constraints. To see why, recall that the SACCPM solves *both* the problem of determining minimum staffing levels *and* the problem of computing the best assignments to cover these staffing levels simultaneously, while the queueing methods compute the required staffing levels first and the shift assignments afterwards, using the staffing levels as input.

Table 4 Feasibility of the solutions and number of iterations of the SACCPM.

Experiment	SACCPM	SIPPavg	SIPPmax	SIPPmix	LAGavg	LAGmax	LAGmix	Iterations	Feas. cuts	Opt. cuts
1	2 79.0%	1 79.0%	1 79.0%	2 79.0%	81.3%	84.4%	85.9%	17	23	8
2	82.1%	83.1%	83.1%	83.1%	82.1%	83.1%	82.1%	20	36	9
3	1 76.7%	4 (**1**) 74.4%	81.6%	81.5%	82.1%	85.9%	82.5%	10	77	3
4	82.1%	82.0%	84.3%	83.1%	82.0%	86.2%	82.1%	15	97	4
5	82.4%	83.1%	81.3%	82.5%	81.8%	83.3%	83.1%	39	34	16
6	81.8%	81.8%	81.8%	81.8%	81.8%	81.8%	81.8%	39	115	14
7	80.1%	80.4%	82.5%	80.8%	83.5%	83.8%	83.8%	12	123	3
8	3 76.6%	82.0%	82.5%	83.0%	81.6%	82.3%	81.6%	11	119	3
9	15 (**3**) 74.2%	14 (**3**) 73.2%	3 76.2%	3 76.2%	2 76.5%	81.2%	1 79.8%	47	210	6
10	12 77.8%	3 78.4%	80.8%	80.8%	2 79.9%	81.1%	2 79.9%	24	173	7
11	21 (**3**) 74.7%	33 (**23**) 61.2%	29 (**4**) 71.5%	29 (**4**) 71.5%	4 78.7%	80.8%	80.7%	38	374	4
12	5 77.9%	11 76.0%	80.8%	80.8%	1 79.7%	81.9%	80.2%	26	271	5
13	7 76.1%	1 79.9%	80.5%	80.5%	1 80.0%	1 80.0%	1 80.0%	51	273	8
14	6 75.9%	3 79.0%	1 79.8%	2 79.6%	2 79.6%	1 79.8%	2 79.6%	27	290	7
15	17 75.1%	19 76.7%	8 78.2%	8 78.2%	3 79.0%	80.3%	2 79.0%	32	294	4
16	10 76.6%	2 79.1%	81.1%	81.1%	1 79.7%	81.1%	81.1%	53	376	7

Each cell has up to 3 values. The first value is the number of periods in which the fraction of calls answered immediately is less than 80%. The boldface values in parentheses are the number of periods in which the fraction of calls answered immediately is less than 75%. The percentages show the lowest fraction of calls answered immediately in any period. We do not display the first two values when they are equal to 0. The last three columns show the number of iterations of the SACCPM and the number of feasibility and optimality cuts added.

There are several interesting directions for related future research, some of which we are actively pursuing. From the numerical experiments we identified that both the simulations and solving the integer programs can be computationally expensive. It would be beneficial to study more efficient methods for estimating the pseudogradients that could mimic the performance of the finite difference method. In relation to the integer programs one should investigate integer programming algorithms that can utilize the special structure of the relaxed problems solved in each iteration and consider allowing approximate solutions of the IPs, especially in early stages of the algorithms. Another technique often used in the continuous case to speed up the computation of the next

iterate is to drop cuts that are redundant (see, e.g., Mitchell 2003, for more on this approach), and it is quite possible that the same technique could reduce the time required to solve the IPs in the later stages of the SACCPM. One could also generate some initial cuts by running simulations at the staffing levels suggested by heuristics.

It is conceivable that other optimization methods could perform well in this setting. The extended cutting plane method in Westerlund and Pörn (2002) seems to fit the framework particularly well, although some details of the implementation are unclear.

The problems solved in this paper were fairly simple instances of a call center staffing problem, but since no assumptions are made on the arrival and service processes and simulation is used to evaluate performance, it seems that the method would also apply in more complicated settings. Call abandonments, skill-based routing and prioritizing multiple customer classes are problems that call center managers commonly face and it would be interesting to incorporate those in the algorithm. Moreover, it is likely that the SACCPM could, with appropriate modifications, be applied to problems other than call center staffing.

Acknowledgments

This research was supported by National Science Foundation grants DMI 0230528 and DMI 0400287 and a Horace H. Rackham School of Graduate Studies Faculty Grant. The first author would like to thank the Department of Industrial and Operations Engineering at the University of Michigan for its generous financial support. The authors would like to thank the two referees for their comments, which lead to improvements of the paper.

References

Andradóttir, S. 1998. Simulation optimization. J. Banks, ed., *Handbook of Simulation*, chap. 9. John Wiley & Sons, New York, 307–333.

Atkinson, D. S., P. M. Vaidya. 1995. A cutting plane algorithm for convex programming that uses analytic centers. *Math. Programming* **69**(1, Ser. B) 1–43. Nondifferentiable and large-scale optimization (Geneva, 1992).

Atlason, J. 2004. A simulation based cutting plane method for optimization of service systems. Ph.D. thesis, University of Michigan, Ann Arbor, MI.

Atlason, J., M. A. Epelman, S. G. Henderson. 2003. Using simulation to approximate subgradients of convex performance measures in service systems. S. Chick, P. J. Sánchez, D. Ferrin, D. J. Morrice, eds., *Proceedings of the 2003 Winter Simulation Conference*. IEEE, Piscataway, NJ, 1824–1832.

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

Bahn, O., O. du Merle, J.-L. Goffin, J.-P. Vial. 1995. A cutting plane method from analytic centers for stochastic programming. *Math. Programming* **69**(1, Ser. B) 45–73. Nondifferentiable and large-scale optimization (Geneva, 1992).

Bazaraa, M. S., H. D. Sherali, C. M. Shetty. 1993. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, New York.

Borst, S., A. Mandelbaum, M. I. Reiman. 2004. Dimensioning large call centers. *Operations Research* **52** 17–34.

Castillo, I., T. Joro, Y. Li. 2003. Workforce scheduling with multiple objectives. Submitted.

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

Cooper, R. B. 1981. *Introduction to Queueing Theory. Second Edition*. Elsevier North Holland, New York, NY.

du Merle, O. 1995. Interior points and cutting planes: Development and implementation of methods for convex optimization and large scale structured linear programming. (In French). Ph.D. thesis, University of Geneva, Geneva, Switzerland.

du Merle, O., J.-L. Goffin, J.-P. Vial. 1998. On improvements to the analytic center cutting plane method. *Computational Optimization and Applications* **11** 37–52.

Elhedhli, S., J.-L. Goffin. 2003. The integration of an interior-point cutting plane method within a branch-and-price algorithm. *Math. Programming* **100, Ser. A** 267–294.

Gans, N., G. Koole, A. Mandelbaum. 2003. Telephone call centers: Tutorial, review and research prospects. *Manufacturing & Service Operations Management* **5**(2) 79–141.

Goffin, J.-L., A. Haurie, J.-P. Vial. 1992. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Science* **38**(2) 284–302.

Goffin, J.-L., Z.-Q. Luo, Y. Ye. 1996. Complexity analysis of an interior cutting plane method for convex feasibility problems. *SIAM J. Optim.* **6**(3) 638–652.

Green, L. V., P. J. Kolesar, J. Soares. 2001. Improving the SIPP approach for staffing service systems that have cyclic demands. *Operations Research* **49**(4) 549–564.

Green, L. V., P. J. Kolesar, J. Soares. 2003. An improved heuristic for staffing telephone call centers with limited operating hours. *Production and Operations Management* **12**(1) 46–61.

Ingolfsson, A., E. Cabral, X. Wu. 2003. Combining integer programming and the randomization method to schedule employees. Submitted.

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

Jennings, O. B., A. Mandelbaum, W. A. Massey, W. Whitt. 1996. Server staffing to meet time-varying demand. *Management Science* **42**(10) 1383–1394.

Kelley, Jr., J.E. 1960. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics* **8**(4) 703–712.

Kleywegt, A. J., A. Shapiro, T. Homem-de-Mello. 2001. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* **12**(2) 479–502.

Kolesar, P. J., L. V. Green. 1998. Insights on service system design from a normal approximation to Erlang's delay formula. *Production and Operations Management* **7** 282–293.

Mandelbaum, A. 2003. Call centers (centres): Research bibliography with abstracts. Version 5. Accessible online via ⟨`http://ie.technion.ac.il/serveng/References/ccbib.pdf`⟩ [accessed June 7, 2004].

Mason, A. J., D. M. Ryan, D. M. Panton. 1998. Integrated simulation, heuristic and optimisation approaches to staff scheduling. *Operations Research* **46** 161–175.

Mitchell, J. E. 2000. Computational experience with an interior point cutting plane algorithm. *SIAM J. Optim.* **10**(4) 1212–1227 (electronic).

Mitchell, J. E. 2003. Polynomial interior point cutting plane methods. *Optim. Methods Softw.* **18**(5) 507–534.

Murota, K. 2003. *Discrete Convex Analysis*. SIAM, Philadelphia, PA.

Nesterov, Y. 1995. Complexity estimates of some cutting plane methods based on the analytic barrier. *Math. Programming* **69**(1, Ser. B) 149–176. Nondifferentiable and large-scale optimization (Geneva, 1992).

Peton, O., J.-P. Vial. 2001. A tutorial on ACCPM: User's guide for version 2.01. Working Paper. University of Geneva.

Shapiro, A. 2003. Monte Carlo sampling methods. A. Ruszczynski, A. Shapiro, eds., *Stochastic Programming*. Handbooks in Operations Research and Management Science, Elsevier, 353–425.

Westerlund, T., R. Pörn. 2002. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optimization and Engineering* **3** 253–280.

Whitt, W. 1991. The pointwise stationary approximation for $M_t/M_t/s$ queues is asymptotically correct as the rates increase. *Management Science* **37** 307–314.