A Simple Staffing Method for Multiskill Call Centers

Auke Pot, Sandjai Bhulai, Ger Koole

Department of Mathematics, Vrije Universiteit, 1081HV Amsterdam, The Netherlands
{sapot@few.vu.nl, sbhulai@few.vu.nl, koole@few.vu.nl}

We study a simple method for staffing in multiskill call centers. The method has short computation times and determines nearly optimal staffing levels. It is in both views competitive to other methods from the literature. Because of the fast and accurate performance of the method, many different scenarios can be analyzed, and our method can be used for both tactical and strategic capacity management decisions.

Key words: contact centers; multiskill call centers; skill-based routing; staffing; work force management
History: Received: December 1, 2005; accepted: March 1, 2007. Published online in *Articles in Advance* December 17, 2007.

1. Introduction

A substantial number of contact centers can be classified as inbound; i.e., jobs are initiated by customers. A property of this type of contact centers is that jobs arrive randomly over time and, as a consequence, the work load is hard to predict; additional complexity is added by external factors such as, for instance, the weather. Because of these complexities, the allocation of labor resources over time is a hard yet important problem.

It is important because an inadequately managed work force can lead to low service levels, e.g., long waiting times, if there are not enough employees available. This can be avoided by scheduling a sufficiently large number of employees. However, it is undesired to schedule too many employees because, besides service levels, contact centers also have to meet economical objectives, in particular, minimizing costs because of employee salaries. Minimizing the number of employees is an important subject because labor is expensive; about 80% of operating costs in call centers are because of personnel (see Gans et al. 2003). Therefore the cost reductions obtained with good staffing algorithms can be substantial.

Note that optimal labor allocation in single-skill call centers is already a complex issue because it should be studied by integrating shift scheduling as well. This, however, results in intractable models, and therefore we study the staffing problem seperately from the shift scheduling problem. In practice, obtaining good rosters requires several iterations between staffing and scheduling. In these cases, it is important to have a staffing method with short computation times.

1.1. Contribution

In this paper, we deal with the staffing problem of the labor allocation process. Our main contribution is that we develop a staffing method to determine schedules in multiskill call centers such that a rough match between the predicted work load and labor capacity is realized, taking the randomness of the arrival process into account.

The staffing method has several appealing properties that make it competitive in comparison to other methods in the literature: It is easy to implement, has short computation times, and yields nearly optimal staffing levels. Because of the fast and accurate performance of the method, many different scenarios can be analyzed as well as tactical or strategic decisions.

1.2. Literature

The number of closely related papers is moderate. These papers are briefly discussed in the next paragraphs.

Koole et al. (2003) present a heuristic to optimize the division of agents among different groups. The method in this paper is an extension of that heuristic because we additionally take the costs of agents and a service-level condition into account.

Chevalier and Van den Schrieck (2005) consider overflow routing in multiskill blocking systems with

randomization parameters. Examples with four different skills are provided, and an agent group is included for each possible set of skills. They describe a method to optimize the staffing levels and randomization parameters by using the branch-and-bound algorithm (see Land and Doig 1960).

Cezik and L'Ecuyer (2006) describe a staffing method in the context of multiskill call centers. The method reduces the solution space by means of generating cuts (see Gomory 1958). The computation time of this algorithm is relatively long because each cut requires the multiskill call center to be simulated multiple times and very accurately. Hence, they are not able to solve the shift scheduling problem but are only able to determine the staffing levels that are constant over the day.

1.3. Outline

This paper is organized as follows. Section 2 defines the problem and introduces the notation that will be used throughout the paper. The main contribution of the paper is in §3: presenting an efficient method for staffing in a multiskill environment while meeting a service-level constraint. The performance of the staffing method is evaluated numerically in §4 and compared to the method of Cezik and L'Ecuyer (2006). We show that the method is fast and yields nearly optimal results. Finally, a summary of the results is given in §5, which also discusses directions for future research. For additional numerical examples, see the online appendix.[1]

2. Problem Formulation

Consider a call center that handles calls that require a skill from the set $\mathcal{M} := \{1, 2, \ldots, M\}$. Calls of type $m \in \mathcal{M}$ arrive according to a Poisson process with rate λ_m. Every agent in the call center belongs to an agent group from the set $\mathcal{G} = \{1, 2, \ldots, G\}$. The service rates are skill- and group-dependent, denoted by rate $\mu_{m,g}$ for skill $m \in \mathcal{M}$ and group $g \in \mathcal{G}$. We assume that a control policy π is given that defines the call selection and agent selection rule. Call assignment occurs according to the agent selection rule. If a call is not

---

[1] The online supplement to this paper is available on the *Manufacturing & Service Operations Management* website (http://msom.pubs.informs.org/ecompanion.html).

assigned to an agent group, it is queued, after which it is served according to the call selection rule.

We define the service level as the percentage of arrivals that waits less than the acceptable waiting time (AWT) in the queue. This service level is denoted by $SL^\pi(s)$ when control policy π is used, where s is a vector of length G denoting the staffing levels of the different groups. The minimal requirement is denoted by α. In practice, call centers often take $\alpha = 80\%$ and an AWT of 20 seconds. The objective is to calculate the number of agents s that is required to meet the service-level constraint.

Note that the service-level constraints are only included to a limited extent. Service levels cannot be specified per job type. In our opinion, this is not a severe restriction. Schedules are most often generated at least a few weeks ahead of time based on predictions. Call centers often have to reschedule during the day when the real work load deviates from the predictions. Thus, service levels often can be and need to be controlled during operations. In addition, service levels can also be controlled by means of optimizing the routing policies.

The staffing costs are denoted by $K(s)$ and are determined by calculating the expected costs of scheduling s_g agents in group g. Mathematically,

$$K(s) := \sum_{g \in \mathcal{G}} K^g(s_g),$$

where $K^g(s_g)$ are the costs of scheduling s_g agents. The objective function is given by

$$f(s) := -K(s).$$

The objective of finding optimal staffing levels can be translated into the following optimization problem:

$$(\text{P1}) \quad \begin{cases} \max_{s \in \mathbb{Z}_+^G} \; f(s) \\ \\ \text{s.t.} \;\; SL^\pi(s) \geq \alpha, \end{cases}$$

with $\mathbb{Z}_+$ the set of nonnegative integers.

3. Staffing Algorithm

In this section, we present an efficient method to determine the staffing levels for each agent group, based on Problem (P1). However, Problem (P1) is intractable because the number of different agent configurations s is huge and, hence, suffers from the curse

of dimensionality. Experiments show that there exists no simple ordering of states such that simplification by local search would be possible. For this reason, we reformulate Problem (P1) as

$$\max_{n \in \mathbb{Z}_+} \left(\max_{s \in \mathbb{Z}_+^G : \, se = n} f(s) \right)$$
$$\text{s.t.} \quad \text{SL}^\pi(s) \geq \alpha.$$

Our solution approach is to solve the Lagrangean dual of this integer problem. Note that we are solving a discrete optimization problem so that there can be a duality gap. Moreover, the service level is only concave in the number of agents if the system is stable, assuming no abandonments occur (see Jagers and Van Doorn 1991). Therefore we should ignore staffing vectors that yield unstable systems. However, in case of abandonments, a problem could arise because the service level is not a concave function anymore. To this end, we formulate the algorithm in such a way that we only require weaker monotonicity properties.

For the formulation of the Lagrangean dual, we define

$$f(\beta, s) := \beta(\text{SL}^\pi(s) - \alpha) - K(s) \tag{1}$$

and

$$f^{(n)}(\beta) := \max_{s \in \mathbb{Z}_+^G : \, se = n} f(\beta, s).$$

Then, Problem (P1) becomes

$$\text{(P2)} \quad \begin{cases} \max_{n > L^n} f^{(n)}, & \text{where} \\ f^{(n)} := \min_{\beta \in \mathbb{R}} f^{(n)}(\beta), \end{cases}$$

and L^n denotes the smallest number of agents such that the service-level constraint can be satisfied. The variable β is the Lagrange multiplier of the service-level constraint, denoting the sensitivity of $K(s)$ for changes in α. Finally, in the remainder of this section, we need to define $\text{SL}^{(n),\pi}(\beta)$:

$$\text{SL}^{(n),\pi}(\beta) := \text{SL}^\pi \left(\arg\max_{s \in \mathbb{Z}_+^G : \, se = n} f(\beta, s) \right).$$

We now present a simple heuristic for solving Problem (P2), and explain the heuristic in detail afterwards.

Staffing Heuristic ( )
1. Initialization: $f \leftarrow 0$, $f^{(n)} \leftarrow \mathsf{M}$, $n_L \leftarrow 0$, $n_U \leftarrow \mathsf{M}$, $\beta_L \leftarrow 0$, and $\beta_U \leftarrow \mathsf{M}$
2. For all n (golden ratio search):

3.   Init: $f^{(n)}(\beta) \leftarrow \mathsf{M}$ and $\beta \leftarrow \beta_L + (\beta_U - \beta_L)/z$
4.   While $\beta_U - \beta_L > \epsilon$ (bisection)
5.   Initialization: $f^{(n)}(\beta) \leftarrow 0$ and $s^* \leftarrow 0$ and declare $\text{SL}^{(n),\pi}(\beta)$
6.   For all s: $se = n$ (local search)
7.   Calculate $\text{SL}^\pi(s)$ and $f(\beta, s)$
8.   If $f(\beta, s) > f^{(n)}(\beta)$
9. $s^* \leftarrow s$, $f^{(n)}(\beta) \leftarrow f(\beta, s)$, and $\text{SL}^{(n),\pi}(\beta) \leftarrow \text{SL}^\pi(s)$
10.   End if
11.   Next
12.   If $f^{(n)}(\beta) < f^{(n)}$
13. $s^{**} \leftarrow s^*$ and $f^{(n)} \leftarrow f^{(n)}(\beta)$
14.   End if
15.   If $\text{SL}^{(n),\pi}(\beta) < \alpha$, then $\beta_L \leftarrow \beta$, else $\beta_U \leftarrow \beta$ End if
16. $\beta \leftarrow \beta_L + (\beta_U - \beta_L)/z$
17.   Next
18.   If $f^{(n)} > f$, then $f \leftarrow f^{(n)}$ and $s^{***} \leftarrow s^{**}$ End if
19.   Update n_L, n_U, and n by using the golden ratio.
20. Next

The heuristic consists of several parts. The initialization occurs in lines 1–5 and uses several parameters: n_L/n_U is the lower/upper bound of the total number of agents, β_L/β_U is the lower/upper bound of β, and M is a large number, e.g., 10^4. In lines 6–11, the heuristic finds the optimal agent configuration with respect to $f(\beta, s)$ for a fixed Lagrange multiplier β and a fixed number of agents $n = se$. Lines 15–16 optimize the costs by varying β, whereas lines 18–19 vary n. Optimality with respect to β is tested in lines 12–14, which is also checked against optimality with respect to n in line 18. The result is stored in the variable s^{***}, which is the final output of the algorithm. A few lines require additional explanation, which are given next.

3.1. Apply Local Search

In the part where the optimal agent configuration is determined, lines 6–11, we mention a local search algorithm that has not yet been defined. To this end, we define an ordering on the set $\{s: se = n\}$ such that $s_1 < s_2$ iff $f(\beta, s_1) < f(\beta, s_2)$. The local search algorithm works well theoretically if there exists a path $s_1 = s^1 < s^2 < \cdots < s^p = s_2$ of length p such that $s^{i+1} = s^i - e_a + e_b$, with $a, b \in G$, and e_i a vector with a one at index i, and

zero otherwise. In practice, it also performs well in all our experiments. In the local search algorithm, the next state $s^{i+1} = s^i - e_a + e_b$ is the successor of s^i chosen such that $f(\beta, s^{i+1})$ is maximized over all $a, b \in \mathcal{G}$, and $f(\beta, s^{i+1}) > f(\beta, s^i)$.

3.2. Avoid Simulations

We use performance evaluation methods other than simulation to determine service levels in the local search algorithm. The benefit is a considerable reduction of computation times. We refer to §3.4 for the details on the alternative methods, which are based on blocking models, and we give a summary in this paragraph. A disadvantage of the alternative methods is that the absolute values of the service-level approximations are not accurate. However, changes in service levels obtained by adjusting the sizes of the agent groups can be measured sufficiently accurately to be useful; see §3.5 for a numerical example and a discussion. To ensure that the service level converges to α, we adapted the calculation in line 15. We optimize the Lagrange multiplier β such that the service level approaches α, ensuring that the service-level condition is met. This requires that $\mathrm{SL}^{(n), \pi}(\beta)$ is a monotone increasing function in β, which is discussed in the next paragraph.

3.3. Assume Monotonicity

The second part of the heuristic deals with the optimal value of β. The implementation is based on the intuition that $f^{(n)}(\beta)$ is monotone in β for fixed n. Indeed, when β is high and increases, the service level becomes more relevant when determining s, resulting in a higher value of $f^{(n)}(\beta)$ because of more conservative scheduling. The opposite holds for small values of β. The third part of the heuristic optimizes n and relies on concavity of $f^{(n)}$. Indeed, by intuition, the service level is concave in n if the system is stable and the costs $K(s)$ will typically be linear. Note that n is not a continuous variable, so that the use of a Lagrangean relaxation can lead to further inaccuracies. The variable n is optimized by the golden ratio search (Press et al. 2002). In line 19, we always increase n_L if the service level cannot be met: for example, if the system is not stable. This is checked by taking $\beta = \mathsf{M}$, and applying the local search algorithm from lines 6–10 such that the service level is maximized.

3.4. Performance Improvements

The algorithm, as described in the previous section, has long computation times because of the large number of simulations required in line 7. Hence it is beneficial to replace simulation by another method whenever possible. We achieved this by using methods from the literature that are based on multiskill blocking models (see, for example, Franx et al. 2006, Chevalier and Tabordon 2003). These methods assume no queues and approximate the blocking probability of the system. Although these assumptions are not made in our model, the methods are useful; earlier studies show that when comparing two different staffing levels, the difference in blocking probabilities accurately reflect the difference in service levels for the original system (see Koole et al. 2003 for a first example; Chevalier et al. 2004, Chevalier and Van den Schrieck 2005 for a method that integrates one of these methods in an optimization algorithm). Thus these methods are useful for optimization purposes, and therefore used in all of our numerical examples. In particular, we used the HyperExponential Decomposition method from Franx et al. (2006) and the Hayward-Fredericks method from Fredericks (1980). The results reported in Tables 1–3 are obtained by the Hayward-Fredericks method. We prefer this method above the HyperExponential Decomposition method because it is faster.

To support these methods of blocking models, the algorithm is adjusted as follows:

• replace in lines 6–11 "SL" by "$\widetilde{\mathrm{SL}}$," which denotes one minus the blocking probability, instead of the probability of waiting less than 20 seconds,

• replace β by $\tilde{\beta}$ because the Lagrange multiplier has no longer the same meaning, and

• determine after line 11, $\mathrm{SL}^{(n), \pi}(\beta)$ by simulation, and calculate $f^{(n)}(\beta)$.

Line 15 is of crucial importance because it ensures that the service-level constraint is met.

The performance is improved further as follows. With the initialization, it is preferable to choose n_L and n_U that are as tight as possible. A formula such as the Erlang-C is helpful to calculate lower and upper bounds on the total number of agents. A lower bound is obtained by considering the system as a single-skill call center and taking the highest service rate over

all groups and skills. An upper bound is obtained by considering separate call centers for each job type.

Finally, we mention a number of technical details that help us to reduce the computation times

(1) We take $z = \text{MAX}(2.0, \text{LOG}(\beta_U - \beta_L))$.

(2) For each n, the local search starts with the agents equally distributed among the agent groups. Every time that β is adjusted according to the bisection algorithm, the group sizes are not changed.

(3) Before β is optimized for a certain n, we check whether the best objective value that has been found so far can be improved for that value of n by calculating an upper bound. Otherwise, β is not optimized and the golden ratio method continues directly by choosing another value of n.

(4) Each time the staffing vector is adjusted according to $s \leftarrow s + e_i - e_j$, we repeat the same movement several times, as long as the objective improves and the jth component of s is greater than zero, with a maximum of three times.

3.5. Optimality with Blocking Models

We validate the usage of blocking models by performing numerical experiments. We compare two performance measures: (1) the overall blocking probability in case of no queues and (2) the fraction of customers who are served within 20 seconds waiting in the presence of queues. The first measure is approximated by using the Hayward-Fredericks method, and the second measure is approximated by means of simulation.

We consider an arbitrarily chosen model. There are three types of jobs, $\mathcal{M} = \{1, 2, 3\}$. Jobs arrive with intensities $\lambda_1 = 1$, $\lambda_2 = 1.5$, and $\lambda_3 = 2$. The groups are $S_1 = \{1\}$, $S_2 = \{2\}$, $S_3 = \{3\}$, $S_4 = \{1, 2\}$, $S_5 = \{2, 3\}$, and $S_6 = \{1, 2, 3\}$. The routing policy is $(1: 1 \rightarrow 4 \rightarrow 6)$, $(2: 2 \rightarrow 4 \rightarrow 5 \rightarrow 6)$, and $(3: 3 \rightarrow 5 \rightarrow 6)$. The service rates are $\mu_{1,1} = 0.5$, $\mu_{2,2} = 0.55$, $\mu_{3,3} = 0.6$, $\mu_{1,4} = 0.47$, $\mu_{2,4} = 0.50$, $\mu_{2,5} = 0.51$, $\mu_{3,5} = 0.56$, $\mu_{1,6} = 0.45$, $\mu_{2,6} = 0.45$, and $\mu_{3,6} = 0.52$. The initial staffing levels are $s_1 = 2$, $s_2 = 2$, $s_3 = 2$, $s_4 = 2$, $s_5 = 2$, and $s_6 = 2$.

By means of experiments, we measure and compare the relative differences of both performance measures when agents are moved from one group to another, for all possible combinations. Under the initial staffing levels, we obtain a service level of $\text{SL}^\pi(s) = 0.7922$ in case of queues, and a service level of $\widetilde{\text{SL}}^\pi(s) = 0.8896$ in case of no queues. Figure 1 relates the changes

Figure 1 **Blocking vs. Waiting Time Measures (in Percentages)**
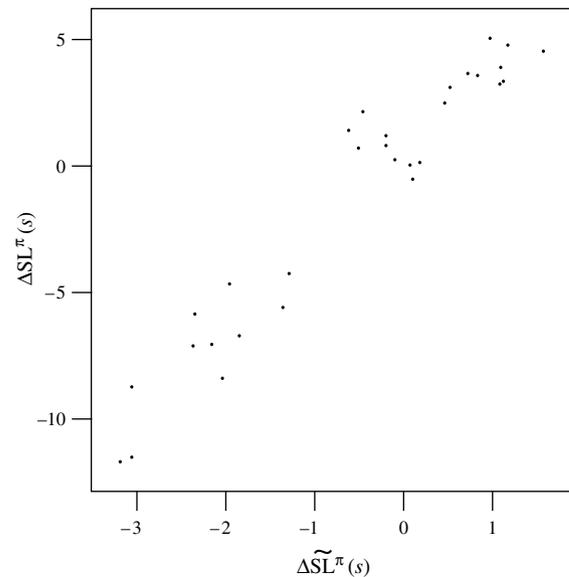
between the two types of service levels when the staffing levels are adjusted. It is a scatter plot of the relative changes of both measures compared to the initial staffing levels, with the change in blocking probability marked on the x-axis and the change in the quantile of the waiting time distribution plotted on the y-axis. We conclude that the ordering of these differences for both models are roughly identical.

Next, we execute another experiment to compare the staffing levels of the two methods that result from optimization. A local search method is applied to maximize the service levels with the same total number of agents, i.e., 12 persons. In case of queues, the optimal service level is 84% and the corresponding staffing levels are $(0, 0, 2, 4, 4, 2)$. Without queues, we obtain a service level of 83% and the staffing vector $(0, 0, 0, 2, 4, 6)$. This shows that approximations by means of blocking probabilities yield nearly optimal staffing levels. However, this example also reveals that the optimal staffing vectors can differ between both methods. In our opinion, there are three important observations of this experiment:

(1) Blocking probabilities are less sensitive to adjustments of the staffing vector.

(2) The usage of blocking models yields solutions with relatively many generalists compared to delay models. This is intuitively clear because the service of a client in blocking models requires that an agent

Table 1   Output of the Staffing Algorithm

				n				
16			17			18		
$\tilde{\beta}$	s^*	SL$^\pi$ (%)	$\tilde{\beta}$	s^*	SL$^\pi$ (%)	$\tilde{\beta}$	s^*	SL$^\pi$ (%)
2.5e3	(6, 4, 6)	80	2.5e3	(7, 5, 5)	87	2.5e3	(8, 5, 5)	93
6.3e2	(7, 4, 5)	80	6.3e2	(8, 4, 5)	88	6.3e2	(9, 4, 5)	93
1.6e2	(7, 4, 5)	80	1.6e2	(8, 4, 5)	88	1.6e2	(9, 4, 5)	93
3.9e1	(9, 5, 2)	81	3.9e1	(10, 5, 2)	87	3.9e1	(10, 6, 2)	91
9.8e0	(11, 5, 0)	76	9.8e0	(11, 6, 0)	84	9.8e0	(12, 6, 0)	89
1.7e1	(10, 6, 0)	76	2.4e0	(14, 3, 0)	41	2.4e0	(15, 3, 0)	42
2.3e1	(10, 6, 0)	76	4.3e0	(12, 5, 0)	81	4.3e0	(13, 5, 0)	83
2.7e1	(10, 6, 0)	76	—	—	—	—	—	—
3.6e1	(9, 5, 2)	81	6.7e0	(12, 5, 0)	81	4.3e0	(13, 5, 0)	83
	9.8			9.5			10	

must be available upon arrival, which pleads for agents with many skills. In delay systems, however, customers can also be served after the arrival epoch because they can wait. Then, the higher service speed of specialists becomes more beneficial.

(3) The number of nearly optimal staffing vectors is huge, and the vectors are diverse. This holds for both types of models. For example, the previous example gave two solutions with almost the same service level. A reason for the latter two observations is that we did not differentiate between the staffing costs of the different agent groups. However, the staffing algorithm does differentiate between costs by means of the Lagrange multiplier. Hence the difference in staffing vectors between delay and blocking models becomes much smaller, which can be concluded from the example in §4.2. Moreover, §4 provides numerical examples of the algorithm of this paper. These allow us to conclude that the staffing algorithm works well and produces nearly optimal results in reasonable cases.

4. Numerical Experiments

In this section, we provide numerical examples illustrating the method from §3. The examples use models having infinite waiting queues, customers having infinite patience for service, and service according to a first-in-first-out service discipline per job type. Upon arrival, calls are assigned to employees according to overflow policies (see, e.g., Franx et al. 2006) and in such a way that specialists have the highest priority,

agents with two skills the second highest priority, and agents with three skills the third highest priority. At a service completion, among the queues with jobs for which the agent has the right skill, a job from the longest queue is taken.

The computations were executed on a PC with a 2800 AMD Athlon^{TM} processor. The RAM usage was always low.

4.1. A Call Center with Two Skills
Consider a call center with two skills and three agent groups, i.e., $G = 3$ and $\mathcal{M} = \{1, 2\}$. The arrival rates are $\lambda_1 = 1.5$ and $\lambda_2 = 2$. A group of specialists is included for each call type, $S_1 = \{1\}$, $S_2 = \{2\}$, and additional flexibility is assured by generalists, $S_3 = \{1, 2\}$. Upon arrival, the routing policy is $(1: 1 \to 3)$ and $(2: 2 \to 3)$. This means that calls of type 1 are assigned to agents of group 1, unless all type 1 agents are unavailable. In that case, agent group 3 is considered. The sequence for type 2 calls is interpreted similarly. The costs c_g of staffing an agent in the gth group is 0.5 if $g = 1$, 0.7 if $g = 2$, and 0.9 in case $g = 3$. Specialists are more efficient than generalists, thus $\mu_{11} = 0.18$, $\mu_{22} = 0.6$, $\mu_{13} = 0.16$, and $\mu_{23} = 0.5$.

Table 1 summarizes the main results that are obtained by applying the algorithm to this example. The table consists of three columns for different numbers of agents n, determined by the optimization procedure. For each n, it shows in each consecutive row the $\tilde{\beta}$ obtained by bisection, together with the optimal staffing level obtained by local search, and the corresponding service level. The variable $\tilde{\beta}$ converges over

Table 2   Cases with Three Skills

										Parameters										
ID	λ_1	λ_2	λ_3	μ_{11}	μ_{22}	μ_{31}	μ_{32}	μ_{41}	μ_{43}	μ_{52}	μ_{53}	μ_{61}	μ_{62}	μ_{63}	c_1	c_2	c_3	c_4	c_5	c_6
1	1.0	1.5	2.0	0.20	0.18	0.19	0.17	0.19	0.16	0.17	0.16	0.18	0.16	0.15	1.0	1.0	1.1	1.1	1.1	1.2
2	1.0	1.5	2.0	0.20	0.18	0.19	0.17	0.19	0.16	0.17	0.16	0.18	0.16	0.15	1.3	1.0	1.4	1.4	1.1	1.5
3	1.0	1.5	2.0	0.20	0.18	0.19	0.17	0.19	0.16	0.17	0.16	0.18	0.16	0.15	1.0	1.0	1.1	1.3	1.3	1.4
4	1.0	1.5	2.0	0.20	0.18	0.19	0.17	0.19	0.16	0.17	0.16	0.18	0.16	0.15	1.0	1.2	1.3	1.3	1.3	1.4
5	1.0	1.5	2.0	0.15	0.18	0.14	0.17	0.14	0.16	0.17	0.16	0.13	0.16	0.15	1.0	1.0	1.1	1.1	1.1	1.2
6	1.0	1.5	2.0	0.20	0.15	0.19	0.14	0.19	0.16	0.14	0.16	0.18	0.13	0.15	1.0	1.0	1.1	1.1	1.1	1.2
7	1.0	1.5	2.0	0.15	0.15	0.14	0.14	0.14	0.16	0.14	0.16	0.13	0.13	0.15	1.0	1.0	1.1	1.1	1.1	1.2
8	2.0	1.5	2.0	0.15	0.18	0.14	0.17	0.14	0.16	0.17	0.16	0.13	0.16	0.15	1.0	1.0	1.1	1.1	1.1	1.2
9	2.0	1.5	2.0	0.20	0.15	0.19	0.14	0.19	0.16	0.14	0.16	0.18	0.13	0.15	1.0	1.0	1.1	1.1	1.1	1.2
10	2.0	1.5	2.0	0.15	0.15	0.14	0.14	0.14	0.16	0.14	0.16	0.13	0.13	0.15	1.0	1.0	1.1	1.1	1.1	1.2

the different rows to the optimal value. The integer after the "e" in the column of $\tilde{\beta}$, say k, denotes that the value should be multiplied by 10^k. The table only presents data of the first nine iterations of the bisection algorithm because they show the most significant improvements.

The last two rows in Table 1 show the optimal outcomes for each value of n; the first row presents the optimal $\tilde{\beta}$, s^*, and SL^π. The second row contains the staffing costs of the optimal vector. In addition, the staffing costs $K(s^*)$ are 9.8, 9.5, and 10 for 16, 17, and 18 agents, respectively. Moreover, taking $n = 19$ yields as optimum $K(s^*) = 10.5$. The objective value of 9.5 for $n = 17$ is the best solution found. It also appears to be optimal; no better solution was found by enumerating the whole search space and evaluating each staffing vector.

The algorithm requires simulating a few more than 17 different staffing vectors, taking a few seconds computation time each. Moreover, the optimization procedure performed by local search hardly takes any time.

4.2. Comparison to Cutting Planes Methods

In this section, we make a comparison to the method from Cezik and L'Ecuyer (2006), which is the best available alternative, so it should be used as a benchmark. We are not able to compare our results to those reported in the paper by Cezik and L'Ecuyer (2006), because they differentiate between the importance of the different call types. Priorities between call types are incorporated by the way that calls are selected

when an agent finishes a call. Agents give the highest priority to type 1, and the higher the index of the call type, the lower the priority is. In contrast, no priorities are supported in our model. An agent serves the longest waiting customer. Moreover, we impose only one constraint on the service level, while Cezik and L'Ecuyer (2006) consider a constraint for each customer type. Hence we composed other instances with this type of call selection policy.

We consider a call center with three skills. The agent groups are $S_1 = \{1\}$, $S_2 = \{2\}$, $S_3 = \{1, 2\}$, $S_4 = \{1, 3\}$, $S_5 = \{2, 3\}$, and $S_6 = \{1, 2, 3\}$. The routing policy is $(1: 1 \to 3 \to 4 \to 6)$, $(2: 2 \to 3 \to 5 \to 6)$, and $(3: 4 \to 5 \to 6)$. The instances are described in Table 2, which contains all the other required parameters. The time unit is one minute and the period of the simulation runs in both methods is 160 hours. Cases 2–10 are derived from Case 1. The differences compared to Case 1 are given next:

• Case 2 (3): cost of type 1 (3) is increased,
• Case 4: costs of types 2 and 3 are increased,
• Case 5 (6): service time of type 1 (2) is increased,
• Case 7: service times of types 1 and 2 are increased,
• Case 8: arrival rate of type 1 and the service time of type 1 are increased,
• Case 9: arrival rate of type 1 and the service time of type 2 are increased,
• Case 10: arrival rate of type 1 and the service times of types 1 and 2 are increased.

The results are presented in Table 3. The service level in all cases is close to 80% with a deviation of

Table 3   **Results with Three Skills**

	Instance IDs									
g	1	2	3	4	5	6	7	8	9	10
Lagrange relaxation										
$-f$	34.6	38.6	37.8	40.2	36.8	36.7	38.7	43.8	42.0	46.0
CPU	3	3	3	4	2	2	3	4	3	2
No. of sim.	31	32	30	40	18	19	21	34	22	17
Cutting planes										
$-f$	34.8	38.5	38.8	43.4	37.1	37.6	38.0	44.8	42.5	47.0
CPU	16	14	12	25	22	14	14	17	21	26
No. of sim	50	44	38	50	56	32	45	43	55	44

at most 2%. The service levels are not reported in the table because it is not to the advantage on one or both methods.

We read from Table 3 that the objective values of the two methods are close to each other for each instance. In most cases, the objective values are lower with the Lagrangean relaxation, as introduced in this paper. However, the service levels are also slightly lower in these cases. Hence we conclude that the accuracy is comparable to the cutting planes method.

5. Concluding Remarks

In this paper, we present a method for staffing in multiskill call centers. Two appealing properties are the short computation times and the high accuracy. Although our experiments deal with call centers having at most five skills, computations are still tractable for call centers with more skills. Another advantage is that the method is easy to implement. The results of the method can therefore be used for the shift scheduling problem to generate shifts and rosters.

References

Cezik, M. T., P. L'Ecuyer. 2006. Staffing multiskill call centers via linear programming and simulation. *Management Sci.* Forthcoming.

Chevalier, P., N. Tabordon. 2003. Overflow analysis and cross-trained servers. *Internat. J. Production Econom.* **85** 47–60.

Chevalier, P., J. Van den Schrieck. 2005. Optimizing the staffing and routing of small size hierarchical call-centers. *Production Oper. Management.* Forthcoming.

Chevalier, P., R. A. Shumsky, N. Tabordon. 2004. Routing and staffing in large call centers with specialized and fully flexible servers. Working paper, Simon School, University of Rochester, Rochester, NY.

Franx, G. J., G. M. Koole, S. A. Pot. 2006. Approximating multi-skill blocking systems by hyperexponential decomposition. *Performance Eval.* **63** 799–824.

Fredericks, A. 1980. Congestion in blocking systems—A simple approximation technique. *Bell System Tech. J.* **59**(6) 805–827.

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

Gomory, R. E. 1958. Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc.* **64** 275–278.

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

Koole, G. M., S. A. Pot, J. Talim. 2003. Routing heuristics for multiskill call centers. S. Chick, P. J. Sánchez, D. Ferrin, D. J. Morrice, eds. *Proc. 2003 Winter Simulation Conf.*, Institute of Electrical and Electronics Engineers, Washington, D.C., 1813–1816.

Land, A. H., A. G. Doig. 1960. An automated method for solving discrete programming problems. *Econometrica* **28** 497–520.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. 2002. *Numerical Recipes in C.* Cambridge University Press, Cambridge, UK.