**Call Center Routing Policy Using
Call Waiting and Agent Idle Times**

W. Chan, G. Koole,
P. L'Ecuyer

G–2011–82

December 2011

Call Center Routing Policy Using
Call Waiting and Agent Idle Times

Wyean Chan

GERAD & DIRO
Université de Montréal
Montréal (Québec) Canada, H3C 3J7
chanwyea@iro.umontreal.ca

Ger Koole

Department of Mathematics
VU University Amsterdam
1081 HV Amsterdam, The Netherlands
ole@few.vu.nl

Pierre L'Ecuyer

GERAD & DIRO
Université de Montréal
Montréal (Québec) Canada, H3C 3J7
lecuyer@iro.umontreal.ca

December 2011

Abstract

We study call routing policies commonly used in call centers with multiple call types and multiple agent groups. We propose a new weight-based routing policy where each pair (call type, agent group) is given a matching priority defined as an affine combination of the longest waiting time for that call type and the longest idle time in that agent group. The coefficients in this combination are parameters to be optimized. This type of policy is more flexible than traditional ones found in practice, and it performs better in many situations. We consider objective functions that account for the service levels, the abandonment ratios, and the fairness of occupancy across agent groups. We select the parameters of all considered policies via simulation-based optimization heuristics. This only requires the availablity of a simulation model of the call center, which can be much more detailed and realistic than the models used elsewhere in the literature to study the optimality of certain types of routing rules. We offer a first numerical study of realistic routing rules that takes into account the complexity of real-life call centers.

Key Words: multi-skill call centers; contact centers; call routing policies; simulation; stochastic optimization.

Résumé

Nous étudions des politiques de routage des appels couramment utilisées dans les centres d'appels recevant plusieurs types d'appels et disposant de plusieurs groupes d'agents ayant des habiletés différentes. Nous proposons une nouvelle politique de routage à l'aide de poids où la priorité d'assignation pour chaque paire (type d'appel, groupe d'agents) est calculée à partir d'une combinaison affine du plus long temps d'attente de ce type d'appel et du plus long temps d'inactivité de ce groupe d'agents. Les cœfficients de cette combinaison sont des paramètres à optimiser. Ce type de politique est plus souple que ceux utilisés traditionnellement dans la pratique, et il offre une meilleure performance dans plusieurs situations. Nous considérons des fonctions objectif basées sur les mesures des niveaux de service, des taux d'abandons et de l'équité des charges de travail entre les groupes d'agents. Pour toutes les politiques considérées, les paramètres de routage sont choisis à l'aide d'algorithmes d'optimisations heuristiques basés sur la simulation. Cette approche ne requiert que la disponibilité d'un modèle de simulation d'un centre d'appels, qui peut être beaucoup plus détaillé et réaliste que les modèles considérés dans la littérature sur l'étude d'optimalité de certaines politiques de routage. Nous présentons une première étude numérique sur des règles de routages réalistes qui tient compte de la complexité réelle des centres d'appels.

1   Introduction

Call centers have a significant economic importance in today's world, as explained in Gans et al. (2003) and Akşin et al. (2007), for example. Managers face complex optimization problems where the goal is to meet various constraints on quality of service (QoS) at the least possible cost, or optimize a given performance measure for a given budget. Most of these costs are actually the salaries of the people who answer the calls.

In this paper, we are considering an inbound multi-skill call center, where arriving calls initiated by the customers are categorized by the type of service that they require, called the *call type*. Call centers with several dozen call types are not uncommon. *Agents*, or *customer sale representatives*, are trained to answer and serve these calls. They are partitioned into *agent groups*, where all the agents in each group can answer the same subset of call types, called their *skill set*. It is not economical and sometimes even impossible to train all agents to have all skills; in practice most agents will have only a few skills.

Managing a multi-skill call center means, among other things, to decide how many agents of each skill set should be assigned to work in each time period (the *staffing* problem), establish admissible work schedules for agents with the right skill sets to cover those staffing requirements (the *scheduling* problem), and select rules (*routing* policies) to match the calls to agents with the right skills. A routing choice arises each time a new call arrives and there are idle agents able to serve it: "Which agent should serve this call?" A decision also has to be taken when an agent becomes free and there are calls waiting to be served: "Which waiting call should this agent serve next?" Ideally, the skill sets, staffing, scheduling, and routing should all be optimized simultaneously, but for large real-life call centers, this leads to excessively difficult optimization problems. In practice, much effort is often put into the staffing and scheduling of agents, while the skill sets and routing policy are often selected ad hoc rather than being systematically optimized. However, the routing policy usually plays an important role in the performance of a multi-skill call center.

In this paper, we study routing policies and their optimization for fixed staffing levels. We also propose a new routing policy based on weights, call waiting times, and agent idle times. This policy often provides better performance than conventional routing policies.

The most rudimentary routing rule is to assign the longest waiting call to the longest idle agent that has the right skill to serve it. This is a combination of the well-known *first-come first-served* (FCFS) and *longest idle server first* (LISF) rules. With the emergence of multi-skill call centers, more complex routing policies have been designed. A popular strategy is to assign different routing priorities between agents and calls. An agent will serve a call of higher priority even if some calls of lower priorities have waited longer. Sometimes, when other policies do not give satisfactory performance, the skill sets of certain agents are restricted by hand for a certain time interval. We will show in our examples that better performance can be achieved by allowing more flexibility by using the policies we introduce in this paper.

Optimization of the routing policy is a control problem that can be solved in principle with dynamic programming (DP). This has been done in the literature for call centers with a small number of call types, and under simplifying assumptions in the model such as Poisson arrivals and exponential service times (Koole and Pot 2005, Koole et al. 2009). These call centers are generally modeled as continuous-time Markov chains. But for real-life multi-skill call centers, the number of states in the DP model is usually much too large (it grows exponentially with the number of call types) for an optimal solution to be practically computable. Furthermore, certain types of performance measures are hard to approximate using DP. For example, if the performance depends on all waiting times, then the system state in the DP formulation must keep track of all those waiting times. Finally, it would be difficult to implement the optimal control policies in the routers of real-life call centers, because they usually turn out to be much too complex.

For large call centers, routing algorithms that are asymptotically optimal in a heavy-traffic regime, when agent occupancies converge to 100%, have been proposed and studied by various authors. The asymptotic optimality is proven under simplifying assumptions, for example, that the service time distribution depends only on the agent and not on the call type, or that the call center has a single call type, and for a given (fixed) staffing. Consequently, the asymptotically optimal policy often has a simple form that ignores certain features of the real system. For example, see van Mieghem (1995, 2003), Mandelbaum and Stolyar (2004), Atar (2005), Gurvich and Whitt (2010, 2009), Milner and Olsen (2008), Armony and Ward (2010b,a). Other

studies propose frameworks that simultaneously optimize the routing and the scheduling or staffing of a call center. Examples are Wallace and Whitt (2005), Sisselman and Whitt (2007), Harrison and Zeevi (2004) and Bassamboo et al. (2006). They all use heavy traffic limits to derive their results or to design heuristics.

In contrast to the studies described above, which are all based on approximations, we only assume the availability of a detailed simulation model of the call center and we use simulation-based optimization heuristics that allow more diversified objective functions than the approximations. Simulation is the only way to study richer objectives and controls that depend on the waiting and agent availability times in realistic systems. We consider several types of routing policies, for which we optimize the parameters for a given staffing and selected objective functions, and we compare their performance. In addition to certain routing rules found in practice, we propose routing policies that uses additive and multiplicative weights on the waiting time of a call and the idle time of an agent to compute a "reward" for each (call, agent) pair, and use this to determine the routing assignments. These routing policies are more flexible than the traditional ones found in practice. They also provide better performances in many cases. Their numbers of parameters are relatively small, so they are much easier to implement in a real router than optimal control policies obtained from dynamic programming. The parameters are optimized by simulation-based algorithms or heuristics. The call center model itself can incorporate arbitrary probability distributions and stochastic process, so there are much less limitations on the model than with analytical approaches. What we present here is the first study of realistic routing rules that takes into account the complexity of real-life call centers. The optimization problem we consider is formulated in terms of penalty functions only, rather than imposing hard constraints on long-term expectations. Penalties can be imposed as functions of standard measures such as abandonment ratios, service level, and average waiting times. They can also account for other important considerations such as agent occupancy and fairness between different agent groups, for example. We find that our weight-based routing policies are competitive with all other routing policies considered and often provide better performance.

The remainder is organized as follows. In Section 2, we define our model and the performance measures that we look at. In Section 3, we define the routing rules that we consider, we introduce our new weight-based routing policy, and we discuss the choice of weights. In Section A, we summarize the two optimization methods that we have used to optimize routing rule parameters. One is a stochastic gradient method and the other is a modified genetic algorithm. We also describe how we have optimized the parameters for each type of routing policy. In Section 5, we report our numerical experiments that compare policies, first in small two-call-type systems such as V, X, and M models, then on a larger model with 8 call types and 10 agent groups, and finally for a small example where the arrival rate is stochastic. A conclusion follows in Section 6. In the online supplement, we provide more details on our optimization methods and on the solutions obtained for some of the examples. An additional online appendix, available from the web site of the third author, gives the complete solutions to all examples and some (empirical) illustrations of the convergence of our optimization methods.

2   The model

2.1   A multi-skill center

We consider a model of a multi-skill call center where arriving calls are categorized in K types, named 1 to K. These calls are served by agents that are divided into G groups. Group $g \in \{1, \ldots, G\}$ is staffed with y_g agents and each of them has the skill set $\mathcal{S}_g \subseteq \{1, \ldots, K\}$, of cardinality $h_g = |\mathcal{S}_g|$, which gives the subset of call types that this agent can serve. An agent with very few skills (like one or two) is often called a *specialist* and an agent with many skills is a *generalist*. We assume that all agents in the same group are homogeneous. We define $\mathcal{I}_k = \{g : k \in \mathcal{S}_g\}$, the set of agent groups that can serve call type k. We do not assume any particular arrival process, service time distribution or patience time distribution in our model, we only need to be able to simulate them. If a call cannot be served immediately on arrival, it is put at the back of a waiting queue associated with this call type. We assume that each customer requires only one type of service and exits the system either at the completion of service or when his waiting time exceeds its (random) patience time. The work is non-preemptive: once an agent starts serving a call, there can be no

interruption until service completion. The assignment between a call and an agent is decided by the router, according to a routing policy such as those described in Section 3. When we optimize, all model parameters are fixed, except for the routing policy.

2.2   Performance Measures

The objective functions considered in this paper are defined in terms of the three performance measures defined below, namely the service level, abandonment ratio, and agent occupancy. These performance measures can be measured per time period, per call type, per agent group, or in aggregated form (for example, globally for all call types). In some cases, there could be variations in the definition; we give the definition used in our experiments. Other performance measures can also be considered; for example the average waiting time, the average excess waiting time above a given threshold, the delay ratio (fraction of calls that have to wait), and the matching rates between selected pairs (call type, agent group) (Sisselman and Whitt 2007). The optimization methods for the routing rules remain the same when the objective function involves those performance measures. In practice, the goal (and the choice of performance measure) can vary greatly across call centers. Henceforth, π denotes a routing policy and $\mathbb{E}$ the mathematical expectation operator.

The first performance measure we consider is a popular one named the *service level* (SL). It is defined as the fraction of calls that are answered within a given time threshold τ called the *acceptable waiting time* (AWT). That is, the SL over a given time period, under policy π, is

$$S(\pi, \tau) = \frac{\mathbb{E}[X(\pi, \tau)]}{\mathbb{E}[N - B(\pi, \tau)]}, \tag{1}$$

where $X(\pi, \tau)$ is the number of served calls that have waited no more than than τ, N is the total number of calls that arrived at the center during the period, and $B(\pi, \tau)$ is the number of calls that abandoned after waiting at most τ. We used the SL definition (1) in our numerical experiments, but there are other possible definitions, based on different ways of accounting for the abandonments. When these measures are for a given call type k, we add a k subscript in the notation; for example τ_k and $S_k(\pi, \tau_k)$ denote the AWT and the SL for call type k.

The second performance measure we consider is the *abandonment ratio*, defined in our simulator as

$$A(\pi) = \frac{\mathbb{E}[Z(\pi)]}{\mathbb{E}[N]}, \tag{2}$$

where $Z(\pi)$ is the number of calls that abandoned during the period considered. Again, we add a subscript k when the measure concerns a given call type k.

The third measure is the *occupancy ratio* of agent groups. It will be used to measure fairness between agent groups. We define the occupancy ratio of group g as

$$O_g(\pi) = \frac{1}{y_g T} \mathbb{E}\left[\int_0^T G_g(\pi, t)dt\right], \tag{3}$$

where T is the time horizon and $G_g(\pi, t)$ is the number of busy agents in group g at time t.

In practice, the constraints on performance measures are often imposed on averages over a medium to long time horizon, say from one day to one month. For long time horizons, performance measures based on expectations as above make sense. For shorter time horizons, such as one day or less, the (random) realization of the performance over the given horizon can be highly variable, due to the high uncertainty in arrival rates, agent absenteeism, etc. In that context, it may be more appropriate to consider objective functions that account for the robustness of solutions to important variations of arrival volumes, for example the probability that a given performance target is achieved over the day (Liao et al. 2010, Gurvich et al. 2010), instead of long-term expectations as above. At the end of the paper, we report on a preliminary experiment where the routing policy is optimized but then fixed, in a setting where the Poisson arrival rate is multiplied by a random factor of mean 1 for the entire day. We find that our weight-based policy does

better than the others in that situation. The routing parameters could also be reoptimized adaptively during the day to account for new information. We plan to study this in the future, together with more extensive experiments.

2.3 Objective Functions

In the optimization problems considered here, instead of imposing hard constraints on certain performance measures as is often done in the literature (Atlason et al. 2004, Avramidis et al. 2010, Cezik and L'Ecuyer 2008), we use penalty costs defined in terms of the performance measures. The objective is then to find routing-rule parameters that minimize the total cost (penalty). Given that only noisy estimates of the performance measures are available (via simulation), using penalties appears more appropriate and sensible than imposing hard constraints.

The penalty costs used in our examples are (truncated) polynomial functions of the performance measures introduced above. We express them as functions of the routing policy π, which contains all decision variables in the optimization problems studied in this paper. They are defined as follows.

1. The penalty cost for violating the SL targets is

$$F_{\mathrm{S}}(\pi) = \sum_{k=1}^{K} a_k \max(t_k - S_k(\pi, \tau_k), 0)^{e_{\mathrm{S},k}}, \tag{4}$$

   where for each call type k, τ_k is the AWT, S_k is the SL, t_k is the corresponding target, $e_{\mathrm{S},k} \geq 0$ is the polynomial degree, and a_k is the penalty weight. In practice, missing the SL target by a small percentage is often deemed acceptable, while missing it by a large percentage is highly unacceptable. This can motivate the choice of exponents $e_{\mathrm{S},k}$ larger than 1. Moreover, fairness between call types can also be an important criteria; for example, it is usually preferable to have two call types at $p\%$ below their SL targets than to have one call type right on target and the other call type $2p\%$ below its target. By taking large values of $e_{\mathrm{S},k}$, one can penalize this form of unfairness between SL violations across call types. The weights a_k may be selected to take into account the volumes of the different call types (for example, a_k could be proportional to the fraction of calls that are of type k) and they may also reflect other considerations (for example, certain call types deemed more important than others can have a larger a_k, or a larger t_k, or both). We could also include the aggregate SL target.

2. The penalty for abandonments is

$$F_{\mathrm{A}}(\pi) = \sum_{k=1}^{K} b_k \max(A_k(\pi) - u_k, 0)^{e_{\mathrm{A},k}}, \tag{5}$$

   where A_k is the abandonment ratio, u_k is the abandonment threshold, $e_{\mathrm{A},k} \geq 0$ is the polynomial degree, and b_k is the penalty weight, for call type k. As in the SL penalty function, more important call types may be given larger weights b_k, or smaller abandonment thresholds u_k, for example.

3. The penalty for unfairness between agent group occupancies is

$$F_{\mathrm{O}}(\pi) = \sum_{g=1}^{G} c_g \left| O_g(\pi) - \bar{O} \right|^{e_{\mathrm{O},g}}, \tag{6}$$

   where for each agent group g, O_g is the occupancy ratio, $\bar{O} = (1/G)\sum_{g=1}^{G} O_g(\pi)$ is the average of group occupancies, $e_{\mathrm{O},g}$ is the polynomial degree, and c_g is the penalty weight. We could also weight the groups by the number of agents in each group, to obtain an average per agent, but we did not do that in our experiments. Instead of penalizing the unfairness as we do here, there could also be a target occupancy ratio for each agent group, which would replace $\bar{O}$ in the formula. This type of unfairness penalty function could also be considered for the performance measures of call types.

Our overall objective function is simply the sum of the penalty functions in (4), (5), (6):

$$F_{\mathrm{SAO}}(\pi) = F_{\mathrm{S}}(\pi) + F_{\mathrm{A}}(\pi) + F_{\mathrm{O}}(\pi), \tag{7}$$

Note that the weights a_k, b_k, and c_g can be selected to give more or less importance to any of the three terms in (7). In particular, any of these terms can be removed by taking these weights equal to 0. For the special case where $c_g = 0$ for each g (so $F_O(\pi)$ is removed), which we use in our experiments, we will denote the objective function (7) by $F_{SA}(\pi)$. It is interesting to note that in the case where $F_{SAO}(\pi) = F_S(\pi)$ (the weights b_k and c_g are all 0), there is no incentive to serve calls whose waiting time has already exceeded the AWT τ_k, and one could easily argue that an optimal routing policy would never serve those calls (if this decision is allowed). This shows that if a call center is subject exclusively to SL constraints of this type, then the FCFS order is definitely not optimal.

We emphasize that the simulation-based stochastic optimization methods used in this paper are independent of the choice of objective function. That is, (7) can easily be replaced by a more general penalty function. This function is taken as a *black box* function by the optimization methods, in the sense that we only assume the possibility of performing (noisy) function evaluations (via simulation) for any policy π that we might want to consider. We also assume the absence of hard performance constraints in the problem formulation. A major advantage of this black-box scheme is that the methods usually require minimal modifications when changing the objective function.

3 Routing Rules

We describe in this section the various routing policies considered in this paper. Most of them are based on routing rules commonly found in industry. We also introduce a new type of policy in which each pair (idle agent, waiting call) is given an index of priority based on a weighted affine combination of the agent's idle time and the call's waiting time, where the weights depend on the agent group and call type. All the routing policies considered satisfy two basic fairness rules: (1) for any given call type, the calls are always served in FCFS order, and (2) for agents of the same group, the idle agent that finished his last service first is always the next agent to work (LISF). It is however acceptable for a call to be served before a call of another type that has waited longer, and for an agent who has been idle for a shorter time than another one to be assigned his next call earlier if they are in different groups. We now describe the routing policies. Their short acronyms given in the headers will be used when we present our numerical results later.

3.1 Global FCFS (G)

The simplest routing policy is when all call types have equal priorities. This policy can be implemented via a single FCFS queue, where an idle agent would scan the queue from the head and pick the first call that it can serve. When a call arrives, it will choose the idle agent that finished his last service first (LISF) among those that can serve it, independently of the group.

3.2 Priorities Lists (P)

This policy is also called *overflow routing* and is available in many major routing equipments, such as those produced by Cisco and Avaya. The call-to-agent and agent-to-call assignments are decided by priority lists, as follows. When an agent becomes idle and searches for the next waiting call to serve, a *group-to-type* list for its agent group determines the order in which the queues of the different call types are examined. Similarly, when a new call arrives and searches for an idle agent to answer it, a *type-to-group* priority list for that call type determines the order in which the agent groups are searched.

The group-to-type priority list for group g with skill set $\mathcal{S}_g$ can be written as $\mathcal{L}_g = (\mathcal{L}_g^{(1)}, \ldots, \mathcal{L}_g^{(m_g)})$, where $m_g \leq h_g = |\mathcal{S}_g|$ is the number of priority levels, and $\mathcal{L}_g^{(1)}, \ldots, \mathcal{L}_g^{(m_g)}$ form a partition of $\mathcal{S}_g$. Each subset $\mathcal{L}_g^{(i)}$ contains the call types having the same priority. These call types are served in FCFS order by agents of the group g. When an agent of group g becomes idle, he first scans the waiting queues of all call types in $\mathcal{L}_g^{(1)}$, and serves the call whose waiting time is the largest. If all these queues are empty, he continues with the call types in $\mathcal{L}_g^{(2)}$, and so on.

Similarly, the type-to-group priority list for call type k can be written as $\mathcal{G}_k = (\mathcal{G}_k^{(1)}, \ldots, \mathcal{G}_k^{(\ell_k)})$, where $\mathcal{G}_k^{(1)}, \ldots, \mathcal{G}_k^{(\ell_k)}$ form a partition of $\mathcal{I}_k = \{g : k \in \mathcal{S}_g\}$. An arriving call will first look for an idle agent whose group is in $\mathcal{G}_k^{(1)}$, and pick one using LISF order. If none is idle, he tries the groups in $\mathcal{G}_k^{(2)}$, and so on.

For the special case where all priorities are equal (all the lists have a single subset), we recover the global FCFS policy. A common situation in practice is when an agent group is given a primary skill and secondary skills. An agent from this group will first try to serve a call corresponding to his primary skill before serving any call corresponding to one of its secondary skills.

3.3 Priorities with Delays (PD)

One refinement of policies with priority lists consists in adding delay conditions based on the waiting time, as follows. For each pair of group g and call type $k \in \mathcal{S}_g$, we select a *time delay* $d_{k,g} \geq 0$. An agent of group g can answer a call of type k only when this call has waited at least $d_{k,g}$. This implies that if $d_{k,g} > 0$, then an agent of group g can never answer a call of type k immediately on arrival. Adding those time delays increases the flexibility and can improve the performance over policy P.

The main motivation for using such a delay can be to increase the match rate between the agents and the call types from their primary skill. In practice, the time delay is usually set below the AWT τ_k, so the call still has good service if answered when the delay expires. But the idle agent may also find and answer a call with smaller delay (that better matches his skills) in the meantime. There are also situations where it can be optimal (depending on the objective function) to set a time delay higher than the AWT. This will be illustrated in our numerical examples.

3.4 Priorities with Idle Agent Threshold (PT)

We have observed that in real-life call centers, managers sometimes restrict temporarily, in ad hoc fashion, the skill set of an agent group when the number of idle agents in that group becomes too low. They do this to favor more important call types, or to prevent the shortage of some skills, for example. This idea is captured by the following modification of the policy with priority lists, which uses *priorities with idle agent thresholds*. For this policy, in addition to the priority lists, there is a threshold $m_{k,g}$ on the number of idle agents for each pair of group g and call type $k \in \mathcal{S}_g$. The new condition is that an idle agent of group g can answer a call of type k only if the number of idle agents in this group is above $m_{k,g}$. Because of the non-preemptive assumption, this condition has no effect on the agents that are already serving calls: A working agent of group g will not stop serving a call of type k even if the number of idle agents in group g becomes smaller or equal to $m_{k,g}$. Unlike the time delay which is a continuous parameter, the idle agent threshold here is an integer, so there is much less room for fine tuning its value (which is also typically small). But it can give good performance in some settings.

3.5 Priorities with Delays and Idle Agent Threshold (PDT)

By combining the PD and PT policies, we obtain a further generalization where we have priority lists, time delays, and idle agent thresholds. In principle, because of its increased flexibility, this type of policy can always provide equal or better performance than all the policies discussed previously. But the larger number of parameters also increases the difficulty of the optimization. Choosing the priority lists is a combinatorial problem, the delay parameters are continuous variables, and the idle agent thresholds are integer variables. Optimizing all of these together can be very difficult.

3.6 Weight-based Routing (WR)

We now introduce our routing policy based on weights, which are defined as parameterized functions of call waiting times and agent idle times. Each pair of group g and call type $k \in \mathcal{S}_g$ is given a *weight* $c_{k,g} \in \mathbb{R}$, which can be interpreted as an index of priority, defined in this paper by the following affine function:

$$c_{k,g} = q_{k,g} + a_{k,g} w_k + b_{k,g} v_g, \tag{8}$$

where w_k is the waiting time of the oldest call of type k waiting in queue, v_g is the current longest idle time of an agent of group g, and $q_{k,g}, a_{k,g}, b_{k,g} \in \mathbb{R}$ are real-valued parameters. When there are either no idle agents in group g or no call of type k waiting in queue, we put $c_{k,g} = -\infty$. There is a total of $3 \sum_{g=1}^{G} y_g$ parameters to specify for this routing policy. This formulation assumes implicitly that all calls enter the waiting queue, and exit the queue with zero waiting time when they are served immediately.

In our implementation of the weight-based routing policy, the weights $c_{k,g}$ are updated periodically at relatively high frequency, for example every second in a large and busy call center, and perhaps at a lower frequency in a small call center. If we restrict the parameters $q_{k,g}, a_{k,g}$ and $b_{k,g}$ to be non-negative, then the router can update the $c_{k,g}$ only on the events of a call arrival or a completion of service by an agent. When all $c_{k,g} < 0$, the router does nothing (no call is assigned to an agent). If there is at least one $c_{k,g} \geq 0$, then the router selects a call type k^* and a group g^* for which $c_{k^*,g^*} = \max_{k,g}\{c_{k,g}\}$, and it assigns the call of type k^* with the longest waiting time to the longest idle agent of group g^*.

Of course, the affine form of the weights in (8) is only one possibility. We could consider more general polynomials or other types of functions. In particular, one may think of replacing $a_{k,g}w_k$ in (8) by $a_{k,g}w_k\mathbb{I}[w_k < \tau_k] - A_{k,g}\mathbb{I}[w_k \geq \tau_k]$ where $A_{k,g}$ is another real-valued constant. In the situation where the objective function includes only the SL penalty F_S in (4), one may consider to take $a_{k,g} > 0$ and a large $A_{k,g} > 0$, so that a call whose waiting time has already passed their AWT τ_k are given a very low priority. When the weights are forced to have the affine form in (8), this can be somewhat approximated (very crudely) by taking $a_{k,g} < 0$, so calls are given lower priority when they have already waited too long. Setting $q_{k,g} > 0$ and $a_{k,g} < 0$ means that a call has higher priority when it arrives and its priority is reduced as it waits in the queue, unless an agent has been idle long enough and $b_{k,g} > |a_{k,g}|$. In our experiments, we found several cases where the optimal choice of $a_{k,g}$ was indeed negative. This often occurs when call type k is a secondary skill for agent group g. On the other hand, a manager might impose a priori that $a_{k,g} \geq 0$, for better fairness. Alternatively, this can be taken care of by further penalizing long waiting times in the objective function. Solutions with $b_{k,g} < 0$ do not seem justified and were returned extremely rarely by our optimization algorithms. We could have imposed $b_{k,g} \geq 0$ as a constraint, but did not do it. In our modified genetic algorithm, however, we disallowed all solutions for which $a_{k,g} < 0$ and $b_{k,g} < 0$ for any pair (k,g). Note that if all parameters $q_{k,g}, a_{k,g}, b_{k,g}$ are negative for a given pair (k,g), group g will never serve a call type k.

One could also define weights $c_{k,g}$ that depend on the state of the call center in different ways; for example, they may depend on the number of idle agents in group g, on the SL of call type k, or the occupancy ratio of agents in group g so far during the day, etc. However, for more complex definitions of the weights, the implementation becomes more complex and the optimization process may become much more difficult.

To better understand the relationship between the WR policy and the more traditional priority lists, and convince ourselves that the WR policy is more general and flexible than the G and P policies, we can examine how these policies can be approximated arbitrary closely by WR policies. The global FCFS routing can be approximated by setting $q_{k,g} = 0$, $a_{k,g} = 1$, and $b_{k,g} = \epsilon > 0$, for all pairs (k,g), where ϵ is arbitrarily small.

If we have a policy P that uses only the type-to-group priorities, or only the group-to-type priorities, or if the priority lists are symmetric (the primary call type of a group and the primary group of a call type are the same pair, which is often the case in practice), then it suffices either to set the $q_{k,g} \geq 0$ according to the priorities with $a_{k,g} = 1$, or to set $a_{k,g} > 0$ accordingly to the priorities with $q_{k,g} = 0$, with $b_{k,g} = \epsilon$ in both cases.

If the priority lists are not symmetric, then we can use $q_{k,g}$ to reproduce the type-to-group priorities and $a_{k,g}$ for the group-to-type priorities, and $b_{k,g} = \epsilon$. The parameters $q_{k,g}$ must be set by small increments while $a_{k,g}$ must be set by large increments. When a call arrives, its waiting time is 0, so $a_{k,g}$ has no influence in the selection of the group. But when an idle agent chooses a call in the queue, $q_{k,g}$ must be negligible compared to $a_{k,g}$. The same setting applies to approximate the policy PD with time delays. One can use the $q_{k,g}$ for the type-to-group priorities and for the delays, since both cannot be applicable simultaneously. If a call is delayed by a group, then the type-to-group priority for that group is meaningless and vice versa. Similarly, one can set the $a_{k,g}$ according to the group-to-type priorities and the delays.

Typically, the group-to-type priorities dominate when the call center is overloaded (most calls have to wait) and the type-to-group priorities dominate when the call center is underloaded (most calls do not wait). In these cases, one may still obtain a good approximation by converting only the group-to-type priorities in the high-traffic situation and the type-to-group priorities in the low-traffic case.

If we take $q_{k,g}, a_{k,g}, b_{k,g} = 0$ for all k and g, then the routing decisions are random.

Since it is more flexible, we can expect the WR policy to perform at least as well as the common routing policies G, P and PD.

4   Routing Optimization

We now discuss how we have optimized the parameters of the routing policies in our experiments. This optimization process is difficult because of the following reasons: the complexity of the routing mechanisms, the stochastic nature of the model (only noisy observations of the performance can be obtained for any choice of parameters, via simulation), the mixture of combinatorial, integer-valued, and real-values parameters, the dimension of the parameter space, and the possibly large number of local minima of the objective function. No efficient and foolproof optimization algorithm is available to optimize the routing policies with the *black box*-type objective functions considered here. Therefore, we have to rely on heuristics.

We have adapted and implemented different heuristic algorithms based on known metaheuristic ideas. Because of the difficulty of the optimization problems, we tried more than one algorithm for some of the routing policies. In the remainder of this section, we outline two types of methods that we have retained as the best performers in our experiments. The first one is a stochastic gradient descent method that can be used for continuous parameters. The second one is a modified genetic algorithm applied to a probability distribution over the space of solutions, which we have used for both discrete and continuous parameters. The detailed descriptions of those methods can be found in the online supplement. Finally, we give the optimization approach that we have selected for each type of routing policy. It generally depends on the size of the problem. Our work here offers the first numerical study of optimization heuristics for realistic routing policies in multi-skill call centers.

4.1   Stochastic Gradient Descent (SGD)

For the minimization of continuous variables, we implemented well-known descent methods based on a stochastic gradient obtained by simulation (Fu 1994, L'Ecuyer et al. 1994, Fu 2006). At each step, the gradient at the current point is estimated by a stochastic finite difference, using a finite-difference step size that decreases with the iteration number.

Suppose we are optimizing simultaneously d continuous parameters of a policy, $\mathbf{e}_i$ denotes the d-dimensional unit vector, and f is the objective function to minimize. The d parameters are the decision variables in the optimization problem. If the current solution is $\mathbf{x} \in \mathbb{R}^d$ and the step size is $\delta > 0$, the ith component of the gradient $\mathbf{g}(\mathbf{x})$ at the current solution can be approximated by the central finite difference

$$g(\mathbf{x})_i = \frac{f(\mathbf{x} + \delta \cdot \mathbf{e}_i) - f(\mathbf{x} - \delta \cdot \mathbf{e}_i)}{2\delta},$$

for $i = 1, \ldots, d$. The values of f at these two points are unknown, but f can be replaced by an estimate $\hat{f}$ obtained by simulation from the black box. We always use common random numbers (CRNs) (Asmussen and Glynn 2007, L'Ecuyer and Buist 2006) across all solutions $\mathbf{x} \in \mathbb{R}^d$ and across all iterations, for these evaluations. That is, for all simulations over the same time horizon, we had exactly the same call arrivals at the same times, the same service times (if served), and the same patience times for each call. Other random streams, such as for tie-breaking by the router, were also identical, for all simulations. This means that in the algorithm, the function f is replaced by a single sample function $\hat{f}$ which is optimized. Note that we could also have used CRNs for each gradient estimation, and different random numbers for the different SGD iterations, as in Buist et al. (2008), but we did not do that. Since we want to compare different routing policies in the numerical section, we also use CRNs across policies.

Note that here, we use the same step size δ for all coordinates. This is not a problem if we rescale the parameters (the coordinates of $\mathbf{x}$) so that they are on a similar scale (otherwise the algorithm may perform poorly). In our experiments, we have applied scaling factors to some of the parameters. These factors were found manually, in an ad hoc fashion.

Once a gradient estimator has been computed, at each step of the SGD method, the moving distance opposite to the gradient is determined by a line search such as the golden section search. This requires additional simulations.

When the current solution appears to have (approximately) converged, we reset the step size δ to a large value and restart the SGD algorithm from the current solution. This restart process is repeated a given number of times or until we observe no significant improvement on the cost of the best solution found for several consecutive restarts. The goal of this restart process is to explore a larger number of local minima. If f was unimodal, there would be no need to do this.

We also implemented a *quasi-Newton* (QN) method, which uses a second-order derivative approximation (Fletcher 1987, Kao et al. 1997). Instead of directly using the gradient as the search direction, the gradient is used to approximate the inverse of the Hessian matrix using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula. The golden section search is again used for the line search. The QN method does not always perform better than the gradient descent method because it is more sensible to the noise of the simulations and the structure of the problem. In our examples, we try both methods and retain the best solution found. The SGD and QN methods are used to optimize the weights of the WR policy and the time delays of the policies PD and PDT. The weights are unbounded, whereas the delays must remain non-negative.

4.2 A Modified Genetic Algorithm (MGA)

Classic genetic algorithms (Goldberg 1989) typically use crossover and mutation operators to generate new populations of solutions. Their performance in applications depends very much on how these operators are defined, and this is highly problem-dependent. For this study, we ended up using a form of genetic algorithm that operates on a parameterized probability distribution $\Phi = \Phi(\boldsymbol{\theta})$ over the set of solutions, by changing the parameter vector $\boldsymbol{\theta}$ rather than by making direct changes to a population of solutions, at each iteration. It can be seen as a simplified version of the *estimation of distribution algorithms* (EDA) (Mühlenbein and Paaß 1996, Larrañaga et al. 1999) and of the *cross-entropy* (CE) method for optimization (Rubinstein and Kroese 2004, de Boer et al. 2005).

This modified genetic algorithm (MGA) starts with an initial parameter vector $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0)}$ for the distribution, generates a *population* of P admissible solutions from this distribution, estimates the costs of these P solutions by simulation, and keeps the $\hat{P} \leq P$ solutions having the smallest estimated costs. This is the *elite population*. Then a new parameter vector $\tilde{\boldsymbol{\theta}}$ is estimated by maximum likelihood from the elite population, and the parameter $\boldsymbol{\theta}$ is reset to a convex combination of its previous value and the new $\tilde{\boldsymbol{\theta}}$. This smoothing helps convergence of the algorithm. Then, a new population of P solutions is generated from the corresponding distribution $\Phi(\boldsymbol{\theta})$, and so on. The idea is that by re-estimating the parameter vector from the elite sample at each iteration, the density (or mass) of the distribution should concentrate progressively around an optimal solution. We stop when the trace (or the maximum element) of the covariance matrix of the current probability distribution is small enough, or if we have reached a given maximum number of iterations. The algorithm returns the best solution found, $\mathbf{x}^*$, and its estimated value f^*.

In our implementation, to simplify the algorithm, for d routing decision parameters to optimize, the distribution $\Phi(\boldsymbol{\theta})$ was assumed to be a product of d univariate distributions, one for each decision parameter. Each of these univariate distributions can have one or more parameters (coordinates of $\boldsymbol{\theta}$), so $\boldsymbol{\theta}$ has dimension larger or equal to d. We could of course use a more general multivariate distribution, but this would increase the number of parameters (to take the dependence into account) and would make things more complicated.

The initial parameter value $\boldsymbol{\theta}^{(0)}$ was selected so that the initial density would be (hopefully) large enough around the (unknown) optimal value. Roughly, the initial distribution has relatively large variance, and the variance generally decreases with the iterations.

In the online supplement, we describe different ways of improving the MGA and how we have adapted this algorithm to optimize the routing policies presented in this paper.

4.3   Routing policy optimization

We now summarize the different optimization methods used for each routing policy.

1. **Priority lists (P).** For call centers with very few call types and agent groups, it is possible to do an exhaustive search by evaluating all the possible combinations of priority rules for call types and agent groups, and this is what we do. However, as the number of call types and agent groups increases, we quickly have a huge combinatorial problem. For example, for each group g, there are $h_g!$ possible permutations, and for each permutation there are 2^{h_g-1} ways of selecting "equal" or "higher than" between successive elements. The number of solutions is the product of all these numbers of possibilities, over all groups g and call types k. When this number is too large for an exhaustive search, we use the MGA.

2. **Priorities with delays (PD).** When the number of solutions for the priority rules is small, these rules can be optimized by an exhaustive search, and the delays optimized by the SGD. Typically, the priority rules have more influence on the performance measures than the delays. Delays can be useful to balance call types with very good and bad SLs. The delays are bounded to non-negative values and we do not impose upper limits. We start by optimizing the priority rules, then we follow up with the delay optimization, which is generally simple enough to require only a single run of the SGD.

   For larger call centers, we use the MGA to optimize both the priority rules and delays simultaneously. We can also implement an iterative method to optimize them individually or together. A two-step approach is to optimize the priority rules with the MGA, and then optimize the delays with the SGD. We did this for the large example in Section 5.5, because the solution space is too large for the MGA to optimize efficiently both the priorities and delays at the same time.

3. **Priorities with idle agent thresholds (PT).** The thresholds on the number of idle agents generally have a significant impact on the performance, because removing (or idling) one agent can make a substantial difference. For the special case of the V-model, we perform an exhaustive search for the priority rules and for each priority rule we perform a golden section search to determine each threshold. In the other cases, we use the MGA to optimize both the priority rules and the thresholds simultaneously.

4. **Priorities with delays and idle agent thresholds (PDT).** For the special case where the center has only one group with a maximum of two skills, we use the same method as for the PT policy to optimize the priority rules and the thresholds. Then, we optimize the delays with the SGD. This two-step method generally gives good results. In the other cases, we use the MGA, but it does not perform well if we try to optimize all the parameters (priorities, delays, and thresholds) simultaneously. Instead, we optimize subsets of parameters iteratively with the MGA. We always start the first iteration with the priority rules, since they tend to have a bigger impact on the performance measures. Then, for a fixed number of iterations, we optimize subsets chosen randomly from: (1) priorities, (2) priorities and delays, (3) priorities and thresholds, and (4) delays. A tabu list is used to store the subsets that give no improvement and these subsets cannot be chosen while they are in the list. This algorithm is very time-consuming, because of the multiple executions of the MGA.

5. **Weight-based routing (WR).** We use either the SGD and QN methods or the MGA. The gradient methods sometimes find better solutions than the MGA, and sometimes it is the opposite. What we have done is to execute the three types of algorithms and take the best solution.

5   Numerical Experiments

5.1   Models, Objective Functions, and Experimental Setting

We report our numerical experiments to compare the routing policies, first for a series of simple canonical models with two call types and 1 to 3 agent groups, then for a larger model with 8 call types and 10 agent

groups. The simple models with two call types are illustrated in Figure 1; they are known as the V, X, and M models in the literature (Gans et al. 2003).
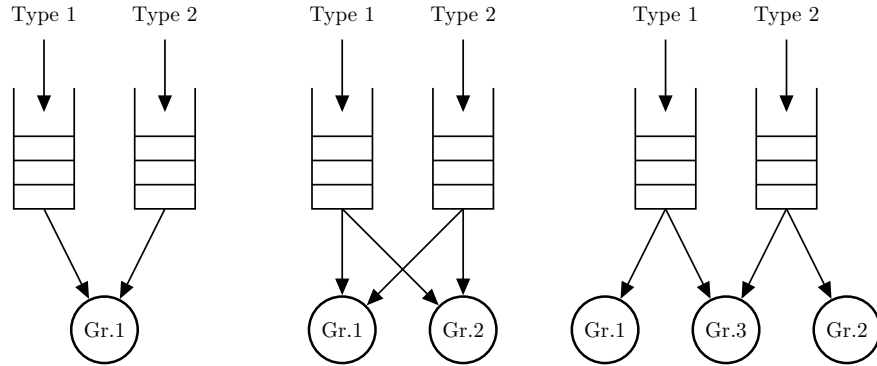

Figure 1: The V-model, X-model and M-model. Each model has 2 call types and 1 to 3 agent groups. The arrows between the call types and the agent groups define the skill sets.

For all the examples considered, for each call type k, we assume a stationary Poisson arrival process of rate λ_k, exponential service times with mean $1/\mu_k$ (independent of the agent group), exponential patience times with mean $1/\nu_k$, and all these random variables are independent. In the canonical examples, we use the same average service times $\mu_1^{-1} = \mu_2^{-1} = 0.1$ hour and average patience times $\nu_1^{-1} = \nu_2^{-1} = 0.1$ hour for call types 1 and 2. All the rates are given per hour. The time horizon is taken to be 1000 hours (so the system is almost in steady-state). The AWT for all call types is 20 seconds. We assume no preemption of service and no work sharing, so that each non-abandoning call is served without interruption by exactly one agent. The number of agents and the sizes of the agent groups are fixed a priori. Although our examples use highly simplified models (for simplicity), our methodology would apply in the same way for more complicated arrival and abandonment processes, more general service time distributions, etc.

We consider the following objective functions:

$$F_{\mathrm{S}}^1(\pi) = \sum_{k=1}^{K} \max(t_k - S_k(\pi, 20), 0)^2, \tag{9}$$

$$F_{\mathrm{S}}^2(\pi) = \max(t_1 - S_1(\pi, 20), 0)^2 + 3 \max(t_2 - S_2(\pi, 20), 0)^2, \tag{10}$$

$$F_{\mathrm{S}}^3(\pi) = 3 \max(t_1 - S_1(\pi, 20), 0)^2 + \max(t_2 - S_2(\pi, 20), 0)^2, \tag{11}$$

$$F_{\mathrm{SA}}(\pi) = \sum_{k=1}^{K} \max(t_k - S_k(\pi, 20), 0)^2 + \sum_{k=1}^{K} A_k(\pi)^2, \tag{12}$$

$$F_{\mathrm{SO}}(\pi) = \sum_{k=1}^{K} \max(t_k - S_k(\pi, 20), 0)^2 + 10 \sum_{g=1}^{G} \left| O_g(\pi) - \bar{O} \right|^2. \tag{13}$$

The functions $F_{\mathrm{S}}^2(\pi)$ and $F_{\mathrm{S}}^3(\pi)$ are used only for some of the two-call-type examples. We could easily add penalties for the aggregate performance over all call types in these objective functions, although we did not do it. Our penalty costs are not proportional to the call volumes per call type or the number of agents per group. We could make them proportional by setting $a_k = \lambda_k$, $b_k = \lambda_k$, and $c_g = y_g$. One drawback of doing this is that call types with very small volumes may have virtually no influence on the total cost. A reasonable compromise could be to make those penalties partially proportional to the call volumes.

As mentioned earlier, the objective function is replaced by an estimate $\hat{f}$, which we optimize via the algorithms described earlier. This sample function $\hat{f}$ is defined as the sample objective function obtained by simulating 20 independent replications of 1000 hours for the canonical examples, and 10 independent replications for the larger example. That is, we perform those simulations at each point $\mathbf{x}$ where we need to evaluate the objective function, using CRNs across those points. To focus on the comparison between routing policies and reduce the simulation noise in this comparison, we also use CRNs across all policies. That is,

we generate the same history of calls, with the same times of arrival, service durations and patience times, across all $\mathbf{x}$ and all policies. For each objective function considered and each type of policy, we apply the optimization methods described in Section A and report the best result found. The reported best solution and its value f^* are always in terms of this $\hat{f}$, so f^* represents the minimal value of $\hat{f}$ that we found. All SLs, abandonment ratios, and agent occupancies are reported in percentage unit. The simulations are precise enough to give 95% confidence interval widths of less than 1% for the SL and less than 0.3% for the abandonment ratios and the agent occupancies, at any given $\mathbf{x}$.

The examples have been constructed so that it would be difficult to obtain a perfect score of 0 for the objective function. The optimization times vary from a few minutes to several hours. We do not report these timings in detail, because our goal in this paper is not to compare the efficiency and speed of the optimization algorithms, but rather to compare policies when their parameters are optimized, in terms of the objective functions. Because the SGD and MGA converge slowly, it happens frequently that the second half of the execution improves the solution by less than 1%, so the algorithms can easily be stopped earlier. A discussion and empirical illustrations on the convergence of the algorithms are presented in the additional online appendix available on the third author's web site. All simulations were performed using the Java library ContactCenters (Buist and L'Ecuyer 2005). Except for the routing policy P for the canonical models and the policy PT for the V model, there is no guarantee that the returned solutions are optimal.

For the canonical models, we show the best results found for each choice of objective function. For these solutions, we report $\Delta S_k = S_k(\pi, \tau_k) - t_k$, the SL target violation for each call type k, the abandonment ratio A_k for each call type k when the objective function is F_{SA}, the occupancy O_g of each agent group g when the objective function is F_{SO}, and the estimated cost f^* for the best solution found. The smallest f^* for each example is put in boldface in the tables. A lower bound on the cost is 0, attained only if all SL targets are satisfied, if there are also no abandonments in the case of F_{SA}, and if all agent occupancies are equal in the case of F_{SO}.

The actual solutions for the selected examples of the V, X and M models with the objective function F_{S}^1 are included in the online supplement. The complete solutions for all the examples are available in the extra online appendix on the third author's web site.

5.2   Experiments with the V-model

We start with the simplest multi-skill call center model, the V-model, for which there are 2 call types and a single agent group; see Figure 1. We consider three parameter sets for this model:

1. Arrival rates $\lambda_1 = \lambda_2 = 50$, SL targets $t_1 = t_2 = 80\%$ and 12 agents. This is the symmetric case.
2. Arrival rates $\lambda_1 = \lambda_2 = 50$, SL targets $t_1 = 70\%$, $t_2 = 90\%$, and 12 agents. Calls of type 2 have a higher SL target.
3. Arrival rates $\lambda_1 = 100$, $\lambda_2 = 10$, SL targets $t_1 = 70\%$, $t_2 = 90\%$, and 13 agents. Here, the low-volume call type has a higher SL target.

Table 1 shows the results for the objective function F_{S}^1, which penalizes only the SL target violation, equally for the two call types. We see that the more flexible policies give lower-cost solutions in general, and that this lower cost often comes with higher abandonment rates, which are not penalized here. In particular, WR is the best performer in all three cases. The PDT routing also performs better than the individual policies P, PD, and PT. The policy G (a single global queue, which involves no optimization) performs the worst, although it is not so bad in the symmetric case 1, where merging the two call types in a single queue is reasonable because of the symmetry.

In the solutions (given in the online supplement), for policies P and PD, call type 2 always has priority over call type 1. However, for PD, in case 1, there is a delay of 6 seconds for call type 2. This reduces the penalty cost compared with policy P by balancing the SLs. In cases 2 and 3, the PD policy sets a delay of 19.9 seconds for call type 1, and ΔS_1 is still well above 0. This shows the limitation of the PD policy. Note that a delay larger than the AWT of 20 seconds would imply a SL of 0% for call type 1. For PT, in cases 2 and 3, calls of type 1 have priority, but are answered only when there is more than one idle agent. For PDT

Table 1: Results for the V-model examples under the objective function F_S^1. For case 1, WR* refers to WR optimized manually.

Routing	Case 1					Case 2					Case 3				
policy	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*
G	-3.4	-3.5	5.3	5.3	24	6.6	-13.5	5.3	5.3	182	6.0	-14.0	5.3	5.3	195
P	-4.0	0.9	7.2	3.4	16	6.0	-9.1	7.2	3.4	82	5.9	-5.3	5.6	2.4	28
PD	-3.2	-1.2	6.9	4.8	12	2.7	-7.1	11.4	3.1	51	4.4	-2.4	9.5	1.9	6
PT	1.0	-4.0	3.4	7.3	16	-4.5	-0.3	11.6	1.9	20	-4.0	8.4	8.6	0.2	16
PDT	-1.1	-3.2	4.8	7.0	11	-2.8	-2.3	10.8	4.4	13	4.4	-2.5	9.5	1.9	6
WR	-0.9	-0.9	5.5	5.8	2	-0.9	-1.6	16.7	2.1	**3**	3.6	0.1	7.7	1.5	**0**
WR*	0.3	0.3	7.1	7.1	**0**		-					-			

Table 2: Results for the V-model examples under the objective function F_S^2.

Routing	Case 1					Case 2					Case 3				
policy	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*
G	-3.4	-3.5	5.3	5.3	49	6.6	-13.5	5.3	5.3	547	6.0	-14.0	5.3	5.3	585
P	-4.0	0.9	7.2	3.4	16	6.0	-9.1	7.2	3.4	247	5.9	-5.3	5.6	2.4	84
PD	-3.5	-0.5	7.0	4.4	13	2.7	-7.1	11.4	3.0	152	4.4	-2.5	9.5	1.9	18
PT	-4.0	0.9	7.2	3.4	16	-4.5	-0.4	11.5	1.9	21	-4.0	8.4	8.6	0.2	16
PDT	-3.5	-0.5	7.0	4.4	13	-3.9	-1.0	11.3	2.9	18	-3.4	7.0	8.4	5.6	12
WR	-1.1	-0.4	6.0	6.7	2	-1.7	-1.3	17.6	2.0	**8**	3.6	0.4	8.1	1.5	**0**
WR*	0.3	0.3	7.1	7.1	**0**		-					-			

in case 2, both call types have the same priority, but there is a delay of about 10 seconds for calls of type 2, and calls of type 1 are answered only when there is more than one idle agent. For WR, it is interesting to observe that the two coefficients $a_{k,1}$ in case 1, and $a_{1,1}$ in the other two cases, take large negative values. This may look strange at first sight, but this is due to choice of objective function, which only accounts for the fraction of calls answered within 20 seconds, together will the linear form of the weights. It turns out that to optimize this objective, it is better to give very low priority to calls that have already waited more than 20 seconds, and this is what explains the negative coefficients $a_{k,1}$; a newly available agent will prefer to serve a call type for which the longest waiting call has waited less. This suggests that using the SL alone in the objective function is questionable.

There is no guarantee that our heuristic optimization methods always find the optimal solution. As an illustration, for case 1, by optimizing manually the parameters of the WR policy (denoted by WR* in the table) we found the solution $q_{1,1} = q_{2,1} = 1$, $a_{1,1} = a_{2,1} = -180$, and $b_{1,1} = b_{2,1} = 627$, which gives a perfect score of 0, although there are more abandonments. For comparison, the solution WR found by the algorithm has a score of 2 and is $q_{1,1} = 10$, $q_{2,1} = 10$, $a_{1,1} = -114$, $a_{2,1} = -128$, $b_{1,1} = 220$, $b_{2,1} = 62$. Cases 2 and 3 are more complex and we did not optimize them manually.

Table 2 gives the results for the objective function F_S^2, which gives a higher penalty for missing the SL target of call type 2. The comparison between the routing policies is similar to what we have seen for F_S^1. Again, the WR policy gives the best results in all three cases. As expected, the SL violations ΔS_2 are larger or equal in Table 2 than the corresponding ones in Table 1, because call type 2 has a higher SL penalty factor. The SL of call type 1 is also lower or equal. Policy P is too inflexible to improve the SL for call type 2. For cases 2 and 3, the policy PD could not improve the SL of call type 2 either, because the delay on call type 1 was already set to its maximum value of 20 seconds with F_S^1. For WR*, we still obtain $f^* = 0$ with the same parameters as in Table 1 (the coefficients in the objective function have changed, but the penalty is still zero).

Table 3 gives the results for the objective function F_{SA}, where the abandonments account for a significant fraction of the penalties, in addition to the SL target violations. Again, in all cases, the best policy is WR and the worst is G. The solutions for the policies PD, PT, and PDT use no delay and no idle agent threshold, because this would increase the overall cost by increasing the number of abandonments. For case 1, policy G is almost as good as the other ones, except for WR. By manually optimizing WR (shown as WR*) for case 1,

Table 3: Results for the V-model examples under the objective function F_{SA}. WR* is obtained by manual optimization.

Routing	Case 1					Case 2					Case 3				
policy	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*
G	-3.4	-3.5	5.3	5.3	81	6.6	-13.5	5.3	5.3	239	6.0	-14.0	5.3	5.3	251
P	-4.0	0.9	7.2	3.4	80	6.0	-9.1	7.2	3.4	146	5.9	-5.3	5.6	2.4	65
PD	-3.9	0.6	7.2	3.6	80	6.0	-9.1	7.2	3.4	146	5.9	-5.3	5.6	2.4	65
PT	-4.0	0.9	7.2	3.4	80	6.0	-9.1	7.2	3.4	146	5.9	-5.3	5.6	2.4	65
PDT	-3.9	0.6	7.2	3.6	80	6.0	-9.1	7.2	3.4	146	5.9	-5.3	5.6	2.4	65
WR	-1.4	-1.2	4.5	6.2	62	1.3	-5.7	9.2	2.8	**125**	3.2	-1.8	6.5	1.8	**49**
WR*	-1.0	-1.1	5.3	5.3	**59**			-					-		

Table 4: Results for the X-model examples under the objective function F_{S}^1. WR* is obtained by manual optimization.

Routing	Case 1					Case 2					Case 3				
policy	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*
G	-5.8	-5.8	4.9	4.9	67	4.2	-15.8	4.9	4.9	249	-0.9	-20.8	4.9	4.9	435
P	-4.1	-4.0	4.9	4.9	33	2.8	-6.7	7.3	2.5	44	-1.2	-6.2	5.3	1.7	40
PD	-1.9	-5.4	4.3	6.7	33	-0.8	-4.6	10.7	2.2	22	-2.4	-3.9	8.1	1.3	21
PT	-4.1	-4.1	4.9	4.9	33	-2.0	-2.2	8.9	1.8	9	-1.2	-6.2	5.3	1.7	40
PDT	-1.7	-5.4	4.5	6.7	32	-2.0	-2.2	8.9	1.8	9	-2.5	-4.0	8.3	1.3	22
WR	-1.0	-0.7	5.4	5.5	1	-0.2	-0.2	12.7	1.6	**0**	-2.2	-1.6	7.3	1.0	**7**
WR*	0.1	0.2	6.6	6.6	**0**			-					-		

we found the solution $q_{k,1} = q_{k,2} = 1$, $a_{k,1} = a_{k,2} = -10$, and $b_{k,1} = b_{k,2} = 500$, for which $f^* = 59$, a slight improvement over what was found by the automatic algorithms SGD and MGA.

5.3   Experiments with the X-model

The X-model has 2 call types and 2 agent groups that can serve both call types, so $\mathcal{S}_1 = \mathcal{S}_2 = \{1, 2\}$, as shown in Figure 1. In practice, each group usually has a primary skill and a secondary skill. We test three sets of parameters:

1. Arrival rates $\lambda_1 = \lambda_2 = 100$, SL targets $t_1 = t_2 = 80\%$, and $y_1 = y_2 = 11$ agents. This is the symmetric example.
2. Arrival rates $\lambda_1 = \lambda_2 = 100$, SL targets $t_1 = 70\%$, $t_2 = 90\%$, and $y_1 = y_2 = 11$ agents. Call type 2 requires a higher SL.
3. Arrival rates $\lambda_1 = 180$, $\lambda_2 = 20$, SL targets $t_1 = 75\%$, $t_2 = 95\%$, and $y_1 = 16$, $y_2 = 11$ agents. Call type 2 is more important but has a lower volume of calls.

Table 4 gives the results for the objective function F_{S}^1. The WR policy gives the best results in all cases. For case 1, the solution for P gives opposite group-to-type priorities to the two agent groups, and the PT solution is exactly the same. This improves the SL by 1.8% compared with policy G, while the abandonment ratios remain identical. The type-to-group priorities have a negligible impact in this example. This suggests that we can potentially improve the SLs with policy P by dividing a call type into artifical sub-call types, or dividing an agent group into subgroups, or even giving to each agent its own routing parameters. This could make sense because the routing policy is more flexible (but has more parameters) when there are more agent groups. For policies PD and PDT in case 1, both agent groups give priority to call type 1, but with a delay between 2 and 13 seconds. This can be seen as a strategy to first serve the calls that have waited less than the AWT. This is similar to the solutions for policies PD and PDT for case 1 in the V-model. Reserving idle agents (PT and PDT) is useful in case 2, but not in cases 1 and 3. The WR policy has positive parameters $q_{k,g}$ and $b_{k,g}$, but mostly negative $a_{k,g}$'s, so higher priority is often given to calls that have waited less. By optimizing manually the parameters of the WR policy, we found the solution $q_{k,g} = 1$, $a_{k,g} = -180$, and $b_{k,g} = 700$, for $g = 1, 2$, for which $f^* = 0$, although there are more abandonments (see the WR* entry).

Table 5: Results for the X-model examples under the objective function F_S^3.

Routing	Case 1					Case 2					Case 3				
policy	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*
G	-5.8	-5.8	4.9	4.9	135	4.2	-15.8	4.9	4.9	249	-0.9	-20.8	4.9	4.9	437
P	3.3	-7.2	2.5	7.3	52	2.8	-6.7	7.3	2.5	44	-1.2	-6.2	5.3	1.7	43
PD	-0.7	-5.7	4.1	6.8	34	-0.5	-4.7	10.5	2.2	23	-1.8	-4.5	6.5	1.4	30
PT	3.3	-7.2	2.5	7.3	52	-2.0	-2.2	8.9	1.8	17	-1.2	-6.2	5.3	1.7	43
PDT	-0.5	-5.8	3.8	6.9	34	-1.5	-3.1	8.7	2.4	16	-1.9	-4.5	6.6	1.4	31
WR	-0.3	-3.8	4.9	5.0	**15**	-0.1	-1.1	11.4	1.7	**1**	-1.7	-2.8	6.7	1.2	**17**

Table 6: Results for the X-model examples under the objective function F_{SA}.

Routing	Case 1					Case 2					Case 3				
policy	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*
G	-5.8	-5.8	4.9	4.9	115	4.2	-15.8	4.9	4.9	297	-0.9	-20.8	4.9	4.9	484
P	-4.1	-4.0	4.9	4.9	81	2.8	-6.7	7.3	2.5	104	-1.2	-6.2	5.3	1.7	71
PD	-4.1	-4.0	4.9	4.9	81	1.1	-5.6	8.1	2.3	101	-1.7	-4.8	6.0	1.5	64
PT	-4.1	-4.0	4.9	4.9	81	-2.0	-2.2	8.9	1.8	91	-1.2	-6.2	5.3	1.7	71
PDT	-4.1	-4.0	4.9	4.9	81	-1.9	-2.4	8.8	1.9	91	-1.7	-4.7	6.1	1.5	64
WR	-1.5	-1.4	4.8	5.0	**52**	0.3	-3.9	8.4	2.1	**90**	-1.5	-3.3	6.1	1.2	**51**

Table 7: Results for the X-model examples under the objective function F_{SO}.

Routing	Case 1					Case 2					Case 3				
policy	ΔS_1	ΔS_2	O_1	O_2	f^*	ΔS_1	ΔS_2	O_1	O_2	f^*	ΔS_1	ΔS_2	O_1	O_2	f^*
G	-5.8	-5.8	86	86	67	4.2	-15.8	86	86	249	-0.9	-20.8	86	86	435
P	-4.1	-4.0	86	86	33	2.8	-6.7	86	86	44	-1.2	-6.2	86	86	40
PD	-4.1	-4.0	86	86	33	-0.6	-4.7	85	85	23	-1.2	-6.2	86	86	40
PT	-4.1	-4.0	86	86	33	-2.6	-2.0	85	87	21	-1.2	-6.2	86	86	40
PDT	-4.1	-4.0	86	86	33	-2.3	-3.1	86	85	18	-1.2	-6.2	86	86	40
WR	-4.2	-1.7	86	86	**20**	-0.5	-1.2	84	84	**2**	-2.1	-2.0	85	85	**8**

Table 5 shows the results with the objective function F_S^3, where the penalty for the SL violation for call type 1 is higher. As expected, the SL for call type 1 has increased and the SL of call type 2 has decreased for most policies. The comparison between the routing policies is similar to Table 4, with the WR policy providing the best results.

Table 6 presents the results with the objective function F_{SA}, which penalizes both the SL violation and the abandonments. Overall, the high abandonment ratios have been reduced, but there is no significant change on the low abandonment ratios. As in Table 4, the WR policy has the best performance in all cases. A good solution for the objective function F_S^1 is not necessarily good for F_{SA}. For example, the WR policy has a near optimal solution for F_S^1 in case 2, but it gives a high abandonment ratio of 12.7% for call type 1, and this solution has a cost of 163 with F_{SA}, which is higher than for all other policies except for G.

Table 7 shows the results with the objective function F_{SO}, where the unfairness in agent occupancies is penalized. Occupancy fairness seems easy to satisfy in the X-model, because all the agents are generalists. The policies P, PD, PT and PDT give the same results in cases 1 and 3.

5.4   Experiments with the M-model

The M-model has 2 call types and 3 agent groups, where agents in groups 1 and 2 are specialists, and those in group 3 are generalists, as shown in Figure 1. This can be seen as a V-model with 2 additional groups of specialists. We test three sets of parameters:

1. Arrival rates $\lambda_1 = \lambda_2 = 100$, SL targets $t_1 = t_2 = 80\%$, $y_1 = y_2 = 10$, and $y_3 = 3$ agents. This is the symmetric case.
2. Arrival rates $\lambda_1 = 25$, $\lambda_2 = 100$, SL targets $t_1 = t_2 = 80\%$, $y_1 = 2$, $y_2 = 10$, and $y_3 = 3$ agents.

Table 8: Results for the M-model examples under the objective function F_S^1.

Routing		Case 1					Case 2					Case 3			
policy	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*
G	-5.3	-5.3	5.4	5.4	56	-12.4	-2.1	9.5	4.8	159	-0.3	-10.3	5.4	5.4	107
P	-5.6	-3.2	5.8	4.9	41	-3.2	-5.3	6.6	5.6	38	-2.9	-7.3	6.3	4.7	62
PD	-3.6	-5.0	5.3	6.0	38	-3.8	-3.4	6.7	5.7	26	-1.6	-7.1	6.4	4.6	52
PT	-5.6	-3.2	5.8	4.9	41	-3.2	-5.3	6.6	5.6	38	-2.9	-7.3	6.3	4.7	62
PDT	-3.7	-4.9	5.3	5.9	38	-3.9	-3.4	6.7	5.6	27	-1.6	-7.0	6.6	4.6	52
WR	-4.1	-3.9	5.6	5.6	**32**	-2.9	-3.7	7.3	6.3	**22**	-3.1	-5.8	7.6	4.4	**43**

Table 9: Results for the M-model examples under the objective function F_SA.

Routing		Case 1					Case 2					Case 3			
policy	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*	ΔS_1	ΔS_2	A_1	A_2	f^*
G	-5.3	-5.3	5.4	5.4	116	-12.4	-2.1	9.5	4.8	273	-0.3	-10.3	5.4	5.4	166
P	-5.6	-3.2	5.8	4.9	98	-3.2	-5.3	6.6	5.6	113	-2.9	-7.3	6.3	4.7	123
PD	-5.0	-3.7	5.7	5.1	97	-2.7	-4.5	6.4	5.8	103	-1.4	-7.2	6.3	4.7	115
PT	-5.6	-3.2	5.8	4.9	98	-3.2	-5.3	6.6	5.6	113	-2.9	-7.3	6.3	4.7	123
PDT	-3.8	-4.9	5.1	5.7	97	-3.8	-3.4	6.7	5.7	103	-1.5	-7.1	6.4	4.6	**114**
WR	-4.1	-4.2	5.3	5.4	**92**	-2.7	-4.1	6.4	6.0	**101**	-1.9	-6.7	6.7	4.6	**114**

Table 10: Results for the M-model examples under the objective function F_SO.

Routing		Case 1						Case 2						Case 3				
policy	ΔS_1	ΔS_2	O_1	O_2	O_3	f^*	ΔS_1	ΔS_2	O_1	O_2	O_3	f^*	ΔS_1	ΔS_2	O_1	O_2	O_3	f^*
G	-5.3	-5.3	81	81	89	416	-12.4	-2.1	68	79	84	1386	-0.3	-10.3	81	81	89	466
P	-4.6	-5.6	80	83	86	231	-15.0	-0.6	72	78	83	805	-0.6	-9.6	83	80	86	275
PD	-7.2	-3.8	83	79	85	218	-17.8	-0.2	73	78	82	717	-3.7	-8.1	82	79	84	211
PT	-4.6	-5.6	80	83	86	231	-14.5	-10.3	73	77	77	467	-7.8	-5.9	82	82	78	223
PDT	-3.6	-7.7	79	82	85	217	-1.7	-10.3	70	78	73	425	-7.9	-6.6	80	82	78	203
WR	-4.3	-4.4	82	82	82	**39**	-9.0	-7.4	72	73	73	**145**	-2.2	-7.1	82	82	82	**58**

3. Arrival rates $\lambda_1 = \lambda_2 = 100$, SL targets $t_1 = 75\%$, $t_2 = 85\%$, $y_1 = y_2 = 10$, and $y_3 = 3$ agents. Call type 2 requires a higher SL.

Table 8 gives the results for the objective function F_S^1. The WR policy has the lowest penalty costs in all cases. The solutions (given in the online supplement) show that it is useless to reserve idle agents in any of the 3 cases, so the policies PT and PDT do not perform better than P and PD, respectively. For case 1, policy P has a lower cost than the policy G, with the generalists giving higher priority to call type 2, because the gain in the SL of call type 2 is greater than the loss in the SL of call type 1. The PD policy improves over P by adding delays of 18.2 and 19.6 seconds to call types 1 and 2, respectively, for the generalist group. For the WR policy, the specialists serve their respective call types in FCFS order, but the generalists are used more aggressively to increase the SLs by favoring the queue that has waited the least, unless the agent's idle time is high. We have $q_{k,3} > 0$, $b_{k,3} > 0$, and $a_{k,3} < 0$ in all 3 cases.

Table 9 gives the results for the objective function F_SA. The relative differences in cost are much smaller when abandonment penalties are added to the SL penalties. The WR policy wins in case 1, because it better balances the penalties. For cases 2 and 3, PD and PDT perform almost as well as WR.

Table 10 presents the results for the objective function F_SO. The WR policy provides the best results by far, for all cases. It is the only policy that gives very good occupancy fairness.

5.5   Experiments with a larger model

We now consider a larger example, with 8 call types and 10 agent groups. The arrival rates are $(\lambda_1, \ldots, \lambda_8) = (250, 200, 100, 80, 50, 20, 15, 10)$, the service rates are $(\mu_1, \ldots, \mu_8) = (10, 6, 6, 10, 6, 6, 8, 10)$, and the patience rates are $(\nu_1, \ldots, \nu_8) = (10, 8, 10, 12, 6, 10, 12, 10)$. The staffing vector is $(y_1, \ldots, y_8) = (21, 12, 14, 8, 16, 5, 3, 7, 8, 9)$. The skill sets are $\mathcal{S}_1 = \{1, 4\}$, $\mathcal{S}_2 = \{2, 5\}$, $\mathcal{S}_3 = \{3, 4, 7\}$, $\mathcal{S}_4 = \{4, 6, 8\}$, $\mathcal{S}_5 = \{2, 5\}$, $\mathcal{S}_6 = \{6, 7, 8\}$,

Table 11: Results for the large example under the objective functions F_{S}^1.

Routing	ΔS			S	A			A	f^*
policy	max	med	min	agg	max	med	min	agg	
G	16.8	-0.2	-17.1	73	5.3	3.0	0.5	4.4	638
P	16.0	3.1	-11.7	75	5.9	2.0	0.6	4.3	267
PD	16.2	1.7	-13.2	74	6.6	4.1	0.7	4.9	380
PD*	15.4	-0.4	-10.9	75	5.5	3.3	0.8	4.3	219
PT	9.9	-0.6	-22.5	72	8.2	4.3	1.5	5.1	684
PDT	17.5	3.8	-11.6	75	5.8	1.9	0.4	4.3	271
WR	-0.5	-1.8	-5.0	77	11.9	6.5	4.1	5.1	**58**

Table 12: Results for the large example under the objective functions F_{SA}.

Routing	ΔS			S	A			A	f^*
policy	max	med	min	agg	max	med	min	agg	
G	16.8	2.0	-17.1	73	5.4	3.0	0.5	4.4	745
P	16.0	3.3	-11.5	75	5.6	2.0	0.7	4.4	350
PD	16.2	1.7	-13.2	74	6.6	4.1	0.7	4.9	538
PD*	14.1	2.3	-10.8	75	5.5	2.5	1.1	4.4	303
PT	9.8	0.0	-22.1	72	8.1	4.3	1.9	5.1	831
PDT	15.9	3.7	-10.8	75	5.4	1.9	0.6	4.4	361
WR	12.4	-1.0	-8.9	76	5.0	4.0	1.4	4.4	**233**

$\mathcal{S}_7 = \{1,3,7\}$, $\mathcal{S}_8 = \{2,4,8\}$, $\mathcal{S}_9 = \{1,3,4,8\}$, and $\mathcal{S}_{10} = \{2,7,8\}$. The SL targets are $t_k = 80\%$ with an AWT of 20 seconds for all call types.

Tables 11 and 12 show the results for the objective functions F_{S}^1 and F_{SA}. The SL gaps ΔS_k and the abandonment ratios A_k are summarized by their maximum, median and minimum values. The tables also include the aggregate SL (S agg) and the aggregate abandonment ratios (A agg) over all call types. The WR policy is the best performer by far, and it gives more balanced SLs and abandonment ratios between the call types than the other policies. As was the case in smaller models, the SL improvement with the WR policy and F_{S}^1 comes at the expense of more abandonments. Using the WR solution obtained for F_{S}^1 would cost 499 with the objective function F_{SA}, which would be even worse than the P policy. The WR policy has the highest aggregate SL for both objective functions F_{S}^1 and F_{SA}, even though it has the worst median values for the ΔS_k. The reason is that the aggregate SL is weighted by the number of arrivals of each call type and the other policies have a high ΔS_k mostly on the low volume call types.

It is very difficult to optimize simultaneously the priorities, delays, and idle agent thresholds; this explains the higher costs for policies PD, PT, and PDT than for policy P. Applying the MGA effectively to simultaneously optimize all the parameters for those policies requires a very large population size in this example, and this is too time consuming. A better approach in this case, for PD and PDT, is to first optimize the priorities with the MGA (that is, start with the best policy P), then optimize the delays with the SGD. This is policy PD* in the tables.

5.6 Poisson-gamma arrival process

For our last example, we consider a case where the arrival process is doubly stochastic. Calls of type i arrive according to a Poisson process with constant rate $\Lambda_i = \beta_i \lambda_i$ over the current day, where β_i is a random factor which can be interpreted as the "busyness" level for the day. As in Avramidis et al. (2004), we model β_i as a gamma random variable with mean 1 and variance $1/(\lambda_i \theta_i)$, whose shape and scale parameters are both equal to $\lambda_i \theta_i$. We use the parameter θ_i to control the variance of the arrival rate in our experiments. In this example, we use $\theta_i = \theta$ for all call types and the arrival rate has mean $\mathbb{E}[\Lambda_i] = \lambda_i$ and variance $\mathrm{Var}[\Lambda_i] = \lambda_i/\theta$. We assume independent arrivals across call types.

We evaluate the robustness of the policies to the uncertainty of the arrival rate for a small illustrative example considered earlier, namely the X-model in case 1, with objective function F_{S}^1. For each type of routing policy, we took the solution corresponding to the result in Table 4 and used it in a simulation of 1000

independent "days" of 20 hours each, with a random busyness factor β_i for each day, and we estimated the expected value of the objective function. We repeated this for several values of θ ranging from 0.1 to 10. The estimated performance, as a function of θ, is given in Figure 2 for three policies, namely WR (which turns out to be the best by far), PDT (which is the second best), and G (which is the worst performer). In the figure, the horizontal lines (labeled WR*, PDT*, and G*) represent the performance when the arrival rate is deterministic. They correspond to the case where $\theta \to \infty$ (or $\text{Var}[\beta_i] = 0$). We see that for all policies, the expected cost increases when $\text{Var}[\beta_i]$ increases, and the cost with the WR policy always remains lower than that of the other policies. We observed a similar type of behavior for other examples as well. This suggests that with the frequently-used approach that optimizes the parameters under the assumptions of a Poisson process with deterministic rate, when the arrival process is in fact doubly stochastic, we typically underestimate the cost.
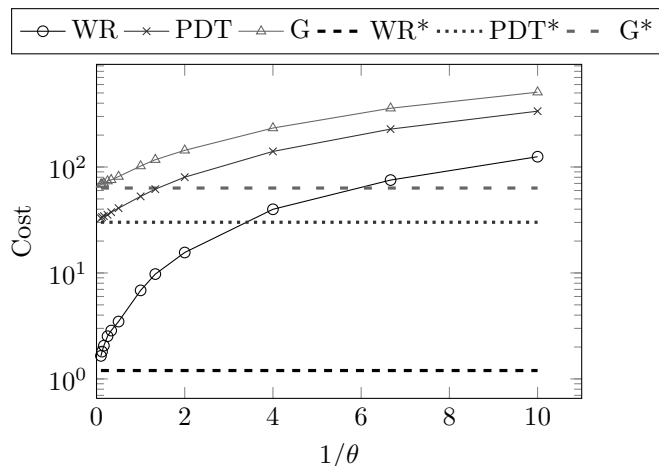
Figure 2: In the X-model, case 1, with objective function F_S^1, we replace the Poisson arrival process by a Poisson-gamma process and we compare the expected cost for policies G, PDT and WR. The cost axis is in logarithmic scale.

6   Conclusion

We have proposed a routing policy based on weights, expressed as linear functions of the call waiting times and agent idle times. To optimize the parameters of this policy and of other parameterized routing policies, we have adapted two simulation-based optimization methods as heuristics, a SGD and a MGA, in a setting where performance constraints are incorporated into the objective function via penalty costs. These methods evaluate the objective function as a black box that returns noisy values. Important advantages of our simulation-based optimization approach over other analytical-based or numerical methods (for example, fluid networks and dynamic programming) are that it permits one to optimize practical call center problems without relying on asymptotic-type approximations, to easily change the objective function, and use basically any performance measure implementable in a simulator (such as the SL, abandonment rate, the AWT, agent occupancy, etc.). We compared the performance of our policy to routing policies commonly used in practice, on various examples. The flexibility of using weights instead of fixed priority rules or thresholds was often reflected in the solutions. Our WR policy gave far better results than other policies commonly used in practice.

Possibilities for further research include improving the WR policy by considering weights that are nonlinear functions of the waiting and idle times, and that may depend on the state of the queues. Another possibility, very relevant to practice, is extending the study into the robustness of routing policies. Our preliminary experiments with random arrival rates suggest that the WR policy is more robust than the other commonly-used routing policies. We plan to do further research on this topic, including the possibility of dynamically changing the parameters of the routing policy when the arrival rates depart significantly from the forecasts.

APPENDIX

In this appendix, we give additional descriptions of the implementation of the optimization algorithms presented in the paper. Then, we give the solutions that we obtained for each routing policy for the V, X and M models with objective function F_S^1.

A   Routing Optimization

A.1   Stochastic Gradient Descent (SGD)

We present the SGD framework in Algorithm 1. The inputs are the initial solution $\mathbf{x}^{(0)}$, the initial step size δ_0, the maximum number of iterations maxIt, and the number of restarts maxRestart. Each variable is allowed to be bounded to a given interval. A variable is truncated to the nearest bound when the next iteration attempts to send it outside of its interval. The parameters $\epsilon_1, \epsilon_2, \epsilon_3$ are used to define a stopping criterion in the inside loop of the algorithm.

Input: $\mathbf{x}^{(0)}$, δ_0, maxIt, maxRestart
Output: the best solution found, $\mathbf{x}^*$
for $m = 1$ **to** maxRestart **do**
    for $k = 1$ **to** maxIt **do**
        $\delta \leftarrow \delta_0/k$
        $\mathbf{g}^{(k-1)} \leftarrow$ a finite-difference gradient estimate at $\mathbf{x}^{(k-1)}$, with step size δ
        $\bar{\lambda} \leftarrow \arg\min_{\lambda \geq 0} \hat{f}(\mathbf{x}^{(k-1)} - \lambda \mathbf{g}^{(k-1)})$, computed by a golden section search
        $\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k-1)} - \bar{\lambda} \mathbf{g}^{(k-1)}$
        if $\hat{f}(\boldsymbol{x}^{(k)}) < \hat{f}(\boldsymbol{x}^*)$ **then**
            $\mathbf{x}^* \leftarrow \mathbf{x}^{(k)}$
        end
        if $\left(\left| \frac{\boldsymbol{x}_i^{(k)} - \boldsymbol{x}_i^{(k-1)}}{\boldsymbol{x}_i^{(k-1)}} \right| < \epsilon_1 \text{for all } i \text{ and } |\boldsymbol{g}| < \epsilon_2 \text{ and } d < \epsilon_3 \right)$ **then**
            $k \leftarrow$ maxIt $+ 1$ // exit the loop
        end
    end
    $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^*$ // prepare for restart
end

Algorithm 1: The SGD algorithm used in this paper.

We now describe our implementation of the *quasi-Newton* (QN) method. We use the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula to approximate the inverse of the Hessian matrix. Let $\mathbf{x}^{(k)}$ be the solution at iteration k, $\mathbf{g}^{(k)}$ the gradient at $\mathbf{x}^{(k)}$, $\mathbf{Z}^{(k)}$ the inverse of the Hessian matrix, $\alpha^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}$ and $\gamma^{(k)} = \mathbf{g}^{(k)} - \mathbf{g}^{(k-1)}$. The BFGS formula is

$$\mathbf{Z}^{(k)} = \mathbf{Z}^{(k-1)} + \left(1 + \frac{(\gamma^{(k)})^{\mathrm{T}} \mathbf{Z}^{(k-1)} \gamma^{(k)}}{(\alpha^{(k)})^{\mathrm{T}} \gamma^{(k)}} \right) \frac{\alpha^{(k)} (\alpha^{(k)})^{\mathrm{T}}}{(\alpha^{(k)})^{\mathrm{T}} \gamma^{(k)}} - \frac{\alpha^{(k)} (\gamma^{(k)})^{\mathrm{T}} \mathbf{Z}^{(k-1)} + \mathbf{Z}^{(k-1)} \gamma^{(k)} (\alpha^{(k)})^{\mathrm{T}}}{(\alpha^{(k)})^{\mathrm{T}} \gamma^{(k)}}.$$

The search direction is $\mathbf{s}^{(k)} = -\mathbf{Z}^{(k-1)} \mathbf{g}^{(k-1)}$. In our implementation, $\mathbf{g}^{(k)}$ is actually a noisy estimate of the gradient rather than its exact value, as in the previous algorithm. The golden section search is again used for the line search.

Algorithm 2 presents the modifications to Algorithm 1 to implement the QN method. For the initialization, we assume that $\mathbf{x}^{(-1)}$ and $\mathbf{g}^{(-1)}$ are zero vectors, and $\mathbf{Z}^{(-1)} = \mathbf{I}$, the identity matrix.

---

for $k = 1$ **to** maxIt **do**

    ...

    $\mathbf{g}^{(k-1)} \leftarrow$ finite-difference gradient estimate at $\mathbf{x}^{(k-1)}$, with step δ

    $\mathbf{Z}^{(k-1)} \leftarrow$ update of the inverse Hessian, based on the BFGS formula

    $\mathbf{s}^{(k)} \leftarrow -\mathbf{Z}^{(k-1)}\mathbf{g}^{(k-1)}$ `// the new search direction`

    $\bar{\lambda} \leftarrow \arg\min_{\lambda \geq 0} \hat{f}(\mathbf{x}^{(k-1)} + \lambda\mathbf{s}^{(k)})$, computed by a golden section search

    $\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k-1)} + \bar{\lambda}\mathbf{s}^{(k)}$

    ...

end

Algorithm 2: Quasi-Newton method, showing the changes to Algorithm 1.

A.2 A Modified Genetic Algorithm (MGA)

In Algorithm 3, we present the framework of the MGA described in the paper. The parameter maxIt determines the maximum number of iterations, γ is used for the smoothing of the parameters of the distributions and the other parameters are described in the main text.

Input: maxIt, P, $\hat{P}$, γ, ϵ, $\boldsymbol{\theta}^{(0)}$

Output: the best solution found, $\mathbf{x}^*$, and its estimated cost f^*

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$

for $i = 1$ **to** maxIt **do**

    for $p = 1$ **to** P **do**

        generate a solution $\mathbf{x}^{(p)}$ from the distribution $\Phi(\boldsymbol{\theta})$

        $f^{(p)} \leftarrow \hat{f}(\mathbf{x}^{(p)})$, estimated by simulation

    end

    sort and relabel $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(P)}$ by order of estimated cost $f^{(1)}, \ldots, f^{(P)}$

    if $f^{(1)} < f^*$ **then**

        $\mathbf{x}^* \leftarrow \mathbf{x}^{(1)}$ **and** $f^* \leftarrow f^{(1)}$

    end

    compute estimator $\tilde{\boldsymbol{\theta}}$ (e.g., by max. likelihood) from the set of elite solutions $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\hat{P})}$

    $\boldsymbol{\theta} \leftarrow \gamma\tilde{\boldsymbol{\theta}} + (1-\gamma)\boldsymbol{\theta}$

    if $\Phi(\boldsymbol{\theta})$ has small enough generalized variance, say $< \epsilon$, **then**

        stop

    end

end

Algorithm 3: A modified genetic algorithm.

The choice of population sizes P and $\hat{P}$ can have a significant impact on the convergence speed and the quality of the final solution. When P is large and $\hat{P}$ is small, the MGA generally converges faster to a local minimum, whereas if $\hat{P}$ is large, convergence is slower but often yields better solutions. For our experiments, we made preliminary tests to select good values.

Various heuristics can be used to further improve the performance of the MGA. For example, the simulation time can be reduced by rejecting early (without doing all the planned simulation runs) the solutions whose estimated cost already appears too high after a few runs. In our implementation, we used the following simple procedure. We started by simulating the P solutions with a small number of runs, say r_0, then we simulated the $M = \min\{2\hat{P}, P\}$ solutions with the smallest empirical cost for another r_1 runs, and we finally selected the $\hat{P}$ (empirically) best solutions among those M to form the elite population. More sophisticated *ranking and selection* procedures (Goldsman et al. 2002, Benson et al. 2006) could have been used at this stage, but here the probability of correct selection does not need to be high, so these procedures

are not necessarily useful. Another idea that we used is to give more weight to the best solutions in the elite population when computing the maximum likelihood estimator. This often speeds up the convergence to a local minimum, and can be useful especially when the computing budget is limited.

In our implementation, for the weight parameters in the WR policy, we used the normal distribution. These parameters can then take any real value, and the maximum likelihood estimator is easy to compute. For each of those weight parameters, there are two coordinates in $\boldsymbol{\theta}$: the mean and the variance of the normal distribution. Any solution for which $a_{k,g} < 0$ and $b_{k,g} < 0$ for any pair (k,g) is rejected immediately, and a new one is generated. At each iteration of the MGA, random solutions are generated from $\Phi(\boldsymbol{\theta})$ until there are P admissible solutions. This procedure is particularly useful at the beginning of the algorithm, where many bad candidate solutions are rejected.

We also use the normal distribution to generate the delays for the policies PD and PDT, and the idle agent thresholds for the policies PT and PDT. The negative values are replaced by 0, and the thresholds are rounded to the nearest integer.

Optimizing the priority lists with the MGA is more complicated, because it is a combinatorial problem. Here, we assume that the skill sets $\mathcal{S}_g$ and their cardinalities are fixed, and we only decide on the priorities. To generate a solution, we generate the type-to-group and the group-to-type priority lists independently. We describe how to generate the preference list $\mathcal{L}_g$ of group g, with skill set $\mathcal{S}_g$. The type-to-group priority list $\mathcal{G}_k$ of a call type k is generated in a similar way. Note that even in very large call centers, although there may be many priority lists, these lists are usually small.

If $h_g = 1$, then the preference list is trivial and there is nothing to do, so we now assume that $h_g > 1$. The distribution used to generate the list $\mathcal{L}_g$ has parameters $m_{g,k} > 0$ for $k \in \mathcal{S}_g$ and $q_{g,n} \in [0.05, 0.95]$ for $n = 1, \ldots, h_g - 1$. We initialize $m_{g,k} = 1$ for all k and $q_{g,n} = 0.5$ for all n. The priority list is generated in two independent steps:

1. We generate a permutation $\mathcal{P} = (p_1, p_2, \ldots, p_{h_g})$ such that $p_i \in \mathcal{S}_g$, for all i and $p_i \neq p_j$ for $i \neq j$. To do this, we generate a random variate $z_{g,k}$ uniformly distributed over the interval $[0, m_{g,k}]$, for each $k \in \mathcal{S}_g$, and the permutation is obtained by sorting these $z_{g,k}$'s in increasing order. Note that the $h_g!$ permutations do not have all equal probabilities. Call types k with a smaller $m_{g,k}$ have more chance to appear earlier in the list and to have higher priorities.

2. Given the permutation, we now have to decide on the subsets $\mathcal{L}_g^{(\cdot)}$ of call types having equal priorities in this list. That is, we must decide between "equal" and "higher than" for the priority relationship between any two successive positions in the permutation $\mathcal{P}$. There are $h_g - 1$ relationships to determine, and the nth one is chosen to be "equal" with probability $q_{g,n}$, for $n = 1, \ldots, h_g - 1$. We bound $q_{g,n}$ away from 0 and 1 to allow for a minimum of diversification. Call types p_n and p_{n+1} have the same priority if $q_{g,n} = 1$, otherwise call type p_n has higher priority.

The distribution parameters $m_{g,k}$ and $q_{g,n}$ are updated at each iteration as follows. For $m_{g,k}$, we simply take the average rank of call type k in the priority lists of group g over the $\hat{P}$ elite solutions. Call types with equal priorities have the same rank. Those with the highest priority have rank 1, those with the next priority have rank 2, and so on. To update $q_{g,n}$, we compute the proportion of call types at position n having the same priority as the call types at position $n+1$ in the $\hat{P}$ elite solutions. For the stopping criterion based on the variance of $\Phi(\boldsymbol{\theta})$, we only look at the variance of the ranks used to update $m_{g,k}$, from the elite sample, for each g and k.

B   Solutions to the numerical examples

We report the actual solutions for the V, X and M models with the objective function F_S^1 from the numerical section, to give an idea of how the policy parameters look like. The complete solutions for all the examples are available on the third author's web site, in "publications". The delays for the policies PD and PDT are given in seconds. For the WR policy, the call waiting times w_k and agent idle times v_g are measured in hours. To save space, we do not report the trivial cases where the type-to-group and group-to-type priority lists have a single group or a single call type. We also do not report the delay when $d_{k,g} = 0$, nor the idle

Table 13: Solutions for the V-model examples under the objective function F_S^1.

Case	Routing solutions
1	**P:** $\mathcal{L}_1 = (\{2\}, \{1\})$. **PD:** $\mathcal{L}_1 = (\{2\}, \{1\}), d_{2,1} = 5.9$. **PT:** $\mathcal{L}_1 = (\{1\}, \{2\})$, No threshold. **PDT:** $\mathcal{L}_1 = (\{1\}, \{2\}), d_{1,1} = 5.8$, No threshold. **WR:** $q_{1,1} = 10, q_{2,1} = 10, a_{1,1} = -114, a_{2,1} = -128, b_{1,1} = 220, b_{2,1} = 62$. **WR*:** $q_{1,1} = q_{2,1} = 1, a_{1,1} = a_{2,1} = -180, b_{1,1} = b_{2,1} = 627$.
2	**P:** $\mathcal{L}_1 = (\{2\}, \{1\})$. **PD:** $\mathcal{L}_1 = (\{2\}, \{1\}), d_{1,1} = 19.9$. **PT:** $\mathcal{L}_1 = (\{1\}, \{2\}), m_{1,1} = 1$. **PDT:** $\mathcal{L}_1 = (\{1, 2\}), d_{2,1} = 10.4, m_{1,1} = 1$. **WR:** $q_{1,1} = -0.09, q_{2,1} = 13, a_{1,1} = -81, a_{2,1} = 125, b_{1,1} = 189, b_{2,1} = 105$.
3	**P:** $\mathcal{L}_1 = (\{2\}, \{1\})$. **PD:** $\mathcal{L}_1 = (\{2\}, \{1\}), d_{1,1} = 19.9$. **PT:** $\mathcal{L}_1 = (\{1\}, \{2\}), m_{1,1} = 1$. **PDT:** $\mathcal{L}_1 = (\{2\}, \{1\}), d_{1,1} = 19.9$, No threshold. **WR:** $q_{1,1} = -0.013, q_{2,1} = 1.0, a_{1,1} = -493, a_{2,1} = 10, b_{1,1} = 1710, b_{2,1} = 10$.

Table 14: Solutions for the X-model examples under the objective function F_S^1.

Case	Routing solutions
1	**P:** $\mathcal{G}_1 = \mathcal{G}_2 = (\{1, 2\}), \mathcal{L}_1 = (\{2\}, \{1\}), \mathcal{L}_2 = (\{1\}, \{2\})$. **PD:** $\mathcal{G}_1 = (\{1, 2\}), \mathcal{G}_2 = (\{2\}, \{1\}), \mathcal{L}_1 = \mathcal{L}_2 = (\{1\}, \{2\}), d_{1,1} = 13, d_{1,2} = 2.1, d_{2,1} = 0.2$. **PT:** $\mathcal{G}_1 = \mathcal{G}_2 = (\{1\}, \{2\}), \mathcal{L}_1 = (\{2\}, \{1\}), \mathcal{L}_2 = (\{1\}, \{2\})$, No threshold. **PDT:** $\mathcal{G}_1 = \mathcal{G}_2 = (\{1, 2\}), \mathcal{L}_1 = \mathcal{L}_2 = (\{1\}, \{2\}), d_{1,1} = 7.8, d_{1,2} = 9.3$, No threshold. **WR:** $q_{1,1} = 5.0, q_{1,2} = 16, q_{2,1} = 4.1, q_{2,2} = 17, a_{1,1} = -209, a_{1,2} = -68, a_{2,1} = -122, a_{2,2} = -120, b_{1,1} = 125, b_{1,2} = 334, b_{2,1} = 137, b_{2,2} = 113$. **WR*:** $q_{1,1} = q_{1,2} = q_{2,1} = q_{2,2} = 1, a_{1,1} = a_{1,2} = a_{2,1} = a_{2,2} = -180, b_{1,1} = b_{1,2} = b_{2,1} = b_{2,2} = -700$.
2	**P:** $\mathcal{G}_1 = \mathcal{G}_2 = (\{1\}, \{2\}), \mathcal{L}_1 = \mathcal{L}_2 = (\{2\}, \{1\})$. **PD:** $\mathcal{G}_1 = \mathcal{G}_2 = (\{1\}, \{2\}), \mathcal{L}_1 = \mathcal{L}_2 = (\{2\}, \{1\}), d_{1,1} = 16, d_{1,2} = 19.9$. **PT:** $\mathcal{G}_1 = \mathcal{G}_2 = (\{1\}, \{2\}), \mathcal{L}_1 = (\{1\}, \{2\}), \mathcal{L}_2 = (\{2\}, \{1\}), m_{1,1} = 1$. **PDT:** $\mathcal{G}_1 = \mathcal{G}_2 = (\{1\}, \{2\}), \mathcal{L}_1 = (\{1, 2\}), \mathcal{L}_2 = (\{2\}, \{1\})$, No delay, $m_{1,1} = 1$. **WR:** $q_{1,1} = 0.34, q_{1,2} = 1.0, q_{2,1} = 16, q_{2,2} = 14, a_{1,1} = -59, a_{1,2} = -149, a_{2,1} = 115, a_{2,2} = 74, b_{1,1} = 266, b_{1,2} = 201, b_{2,1} = 11, b_{2,2} = 230$.
3	**P:** $\mathcal{G}_1 = (\{1, 2\}), \mathcal{G}_2 = (\{2\}, \{1\}), \mathcal{L}_1 = (\{2\}, \{1\}), \mathcal{L}_2 = (\{2\}, \{1\})$. **PD:** $\mathcal{G}_1 = (\{1, 2\}), \mathcal{G}_2 = (\{2\}, \{1\}), \mathcal{L}_1 = \mathcal{L}_2 = (\{2\}, \{1\}), d_{1,1} = 19.9, d_{1,2} = 1$. **PT:** $\mathcal{G}_1 = \mathcal{G}_2 = (\{1\}, \{2\}), \mathcal{L}_1 = \mathcal{L}_2 = (\{2\}, \{1\})$, No threshold. **PDT:** $\mathcal{G}_1 = (\{1, 2\}), \mathcal{G}_2 = (\{1\}, \{2\}), \mathcal{L}_1 = \mathcal{L}_2 = (\{2\}, \{1\}), d_{1,1} = 15, d_{1,2} = 19.7, d_{2,2} = 0.01$, No threshold. **WR:** $q_{1,1} = 0.071, q_{1,2} = 0.47, q_{2,1} = 0.33, q_{2,2} = 1.1, a_{1,1} = -9.0, a_{1,2} = -77, a_{2,1} = -9.1, a_{2,2} = 11, b_{1,1} = 42, b_{1,2} = 73, b_{2,1} = 10, b_{2,2} = 7.2$.

Table 15: Solutions for the M-model examples under the objective function F_S^1.

Case	Routing solutions
1	**P:** $\mathcal{G}_1 = (\{1\}, \{3\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{2\}, \{1\})$. **PD:** $\mathcal{G}_1 = (\{1\}, \{3\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{1\}, \{2\}), d_{1,3} = 18.2, d_{2,3} = 19.6$. **PT:** $\mathcal{G}_1 = (\{1\}, \{3\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{2\}, \{1\})$, No threshold. **PDT:** $\mathcal{G}_1 = (\{1\}, \{3\}), \mathcal{G}_2 = (\{3\}, \{2\}), \mathcal{L}_3 = (\{1\}, \{2\}), d_{1,3} = 18.2, d_{2,3} = 17.2$, No threshold. **WR:** $q_{1,1} = 13, q_{1,3} = 1.3, q_{2,2} = 12, q_{2,3} = 1.6, a_{1,1} = 1475, a_{1,3} = -49, a_{2,2} = 1465, a_{2,3} = -62, b_{1,1} = 2007, b_{1,3} = 60, b_{2,2} = 2008, b_{2,3} = 69$.
2	**P:** $\mathcal{G}_1 = (\{3\}, \{1\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{1\}, \{2\})$. **PD:** $\mathcal{G}_1 = (\{1\}, \{3\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{1\}, \{2\}), d_{1,3} = 0.2, d_{2,3} = 19.9$. **PT:** $\mathcal{G}_1 = (\{3\}, \{1\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{1\}, \{2\})$, No threshold. **PDT:** $\mathcal{G}_1 = (\{1\}, \{3\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{1\}, \{2\}), d_{1,3} = 0.3, d_{2,3} = 19.5$, No threshold. **WR:** $q_{1,1} = 16, q_{1,3} = 1.1, q_{2,2} = 23, q_{2,3} = 1, a_{1,1} = 91, a_{1,3} = -30, a_{2,2} = 46, a_{2,3} = -129, b_{1,1} = 130, b_{1,3} = 60, b_{2,2} = 491, b_{2,3} = 75$.
3	**P:** $\mathcal{G}_1 = (\{1\}, \{3\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{2\}, \{1\})$. **PD:** $\mathcal{G}_1 = (\{1\}, \{3\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{2\}, \{1\}), d_{1,3} = 19.9, d_{2,3} = 0.01$. **PT:** $\mathcal{G}_1 = (\{1\}, \{3\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{2\}, \{1\})$, No threshold. **PDT:** $\mathcal{G}_1 = (\{1\}, \{3\}), \mathcal{G}_2 = (\{2\}, \{3\}), \mathcal{L}_3 = (\{2\}, \{1\}), d_{1,1} = 2, d_{1,3} = 19.9$, No threshold. **WR:** $q_{1,1} = 12, q_{1,3} = 0.5, q_{2,2} = 15, q_{2,3} = 2.6, a_{1,1} = 1461, a_{1,3} = -80, a_{2,2} = 1743, a_{2,3} = -29, b_{1,1} = 2127, b_{1,3} = 25, b_{2,2} = 2359, b_{2,3} = 144$.

agent threshold when $m_{k,g} = 0$. We mention "No delay" if all $d_{k,g} = 0$ and "No threshold" if all $m_{k,g} = 0$. The parameters $q_{k,g}$, $a_{k,g}$ and $b_{k,g}$ for the WR policy are not reported when their values are 0. Refer to the descriptions of the routing policies for the definitions of the parameters.

References

Akşin, O. Z., M. Armony, V. Mehrotra. 2007. The modern call center: A multi-disciplinary perspective on operations management research. *Production and Operations Management* **16**(6) 665–688.

Armony, M., A. R. Ward. 2010a. Blind fair routing in large-scale service systems. Manuscript.

Armony, M., A. R. Ward. 2010b. Fair dynamic routing in large-scale heterogeneous-server systems. *Operations Research* **58**(3) 624–637.

Asmussen, S., P. W. Glynn. 2007. *Stochastic Simulation*. Springer-Verlag, New York.

Atar, R. 2005. Scheduling control for queueing systems with many servers: Asymptotic optimality in heavy traffic. *Annals of Applied Probability* **15**(4) 2606–2650.

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.

Avramidis, A. N., W. Chan, M. Gendreau, P. L'Ecuyer, O. Pisacane. 2010. Optimizing daily agent scheduling in a multiskill call centers. *European Journal of Operational Research* **200**(3) 822–832. http://dx.doi.org/10.1016/j.ejor.2009.01.042.

Avramidis, A. N., A. Deslauriers, P. L'Ecuyer. 2004. Modeling daily arrivals to a telephone call center. *Management Science* **50**(7) 896–908.

Bassamboo, A., J. M. Harrison, A. Zeevi. 2006. Design and control of a large call center: Asymptotic analysis of an LP-based method. *Operations Research* **54**(3) 419–435. doi:http://dx.doi.org/10.1287/opre.1060.0285.

Benson, K. C., D. Goldsman, Amy R. Pritchett. 2006. Ranking and selection procedures for simulation. *Proceedings of the 2006 Winter Simulation Conference*. IEEE Press, 179–185.

Buist, E., W. Chan, P. L'Ecuyer. 2008. Speeding up call center simulation and optimization by Markov chain uniformization. *Proceedings of the 2008 Winter Simulation Conference*. IEEE Press, Pistacaway, New-Jersey, 1652–1660.

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

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

de Boer, P.-T., D. P. Kroese, S. Mannor, R. Y. Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of Operations Research* **134**(1) 19–67.

Fletcher, R. 1987. *Practical Methods of Optimization*. 2nd ed. Wiley.

Fu, M. C. 1994. Optimization by simulation: A review. *Annals of Operations Research* **53** 199–247.

Fu, M. C. 2006. Gradient estimation. S. G. Henderson, B. L. Nelson, eds., *Simulation*. Handbooks in Operations Research and Management Science, Elsevier, Amsterdam, The Netherlands, 575–616. Chapter 19.

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

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st ed. Addison-Wesley Longman Publishing Co., Boston, MA, USA.

Goldsman, D., S.-H. Kim, W. S. Marshall, Barry L. Nelson. 2002. Ranking and selection for steady-state simulation: Procedures and perspectives. *INFORMS Journal on Computing* **14**(1) 2–19.

Gurvich, I., J. Luedtke, T. Tezcan. 2010. Staffing call centers with uncertain demand forecasts: A chance-constrained optimization approach. *Management Science* **56**(7) 1093–1115.

Gurvich, I., W. Whitt. 2009. Queue-and-idleness-ratio controls in many-server service systems. *Mathematics of Operations Research* **34**(2) 363–396.

Gurvich, I., W. Whitt. 2010. Service-level differentiation in many-server service systems via queue-ratio routing. *Operations Research* **58**(2) 316–328. doi:http://dx.doi.org/10.1287/opre.1090.0736.

Harrison, J. M., A. Zeevi. 2004. Dynamic scheduling of a multi-class queue in the Halfin-Whitt heavy-traffic regime. *Operations Research* **52** 243–257.

Kao, C., W. T. Song, S.-P. Chen. 1997. A modified quasi-newton method for optimization in simulation. *International Transactions in Operational Research* **4**(3) 223–233.

Koole, G., B. F. Nielsen, T. B. Nielsen. 2009. Optimization of overflow policies in call centers Working paper.

Koole, G., A. Pot. 2005. Approximate dynamic programming in multi-skill call centers. *Proceedings of the 2005 Winter Simulation Conference*. IEEE Press.

Larrañaga, P., R. Etxeberria, J. A. Lozano, B. Sierra, I. Inza, J. M. Peña. 1999. A Review of the Cooperation between Evolutionary Computation and Probabilistic Graphical Models. A. Ochoa, M. R. Soto, R. Santana, eds., *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*. Habana, Cuba, 314–324.

L'Ecuyer, P., E. Buist. 2006. Variance reduction in the simulation of call centers. *Proceedings of the 2006 Winter Simulation Conference*. IEEE Press, 604–613.

L'Ecuyer, P., N. Giroux, P. W. Glynn. 1994. Stochastic optimization by simulation: Numerical experiments with the $M/M/1$ queue in steady-state. *Management Science* **40**(10) 1245–1261.

Liao, S., C. Van Delft, G. Koole, O.Jouini. 2010. Shift-scheduling of call centers with uncertain arrival parameters. *MOSIM 2010*. Hammamet, Tunisia, 1–10. `http://www.enim.fr/mosim2010/program.php`.

Mandelbaum, A., A. L. Stolyar. 2004. Scheduling flexible servers with convex delay costs: Heavy-traffic optimality of the generalized c-mu rule. *Operations Research* **52** 836–855.

Milner, J. M., T. L. Olsen. 2008. Service-level agreements in call centers: Perils and prescriptions. *Management Science* **54**(2) 238–252.

Mühlenbein, H., G. Paaß. 1996. From recombination of genes to the estimation of distributions 1. binary parameters. Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, Hans-Paul Schwefel, eds., *Parallel Problem Solving from Nature PPSN IV*, *Lecture Notes in Computer Science*, vol. 1141. Springer Berlin / Heidelberg, 178–187.

Rubinstein, R. Y., D. P. Kroese. 2004. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer-Verlag.

Sisselman, M. E., Ward Whitt. 2007. Value-based routing and preference-based routing in customer contact centers. *Production and Operations Management* **16**(3) 277–291.

van Mieghem, J. A. 1995. Dynamic scheduling with convex delay costs: the generalized c*mu* rule. *Annals of Applied Probability* **5** 809–833.

van Mieghem, J. A. 2003. Due-date scheduling: Asymptotic optimality of generalized longest queue and generalized largest delay rules. *Operations Research* **51**(1) 113–122.

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