VALUE-BASED ROUTING AND PREFERENCE-BASED ROUTING IN CUSTOMER CONTACT CENTERS

by

Michael E. Sisselman
New York, NY
917-837-7223
mikesisselman@att.net

Ward Whitt
Columbia University
212-854-7255
ww2040@columbia.edu

Abstract

Telephone call centers and their generalizations - customer contact centers - usually handle several types of customer service requests (calls). Since customer service representatives (agents) have different call-handling abilities and are typically cross-trained in multiple skills, contact centers exploit skill-based routing (SBR) to assign calls to appropriate agents, aiming to respond properly as well as promptly. Established agent-staffing and SBR algorithms ensure that agents have the required call-handling skills and that call routing is performed so that constraints are met for standard congestion measures, such as the percentage of calls of each type that abandon before starting service and the percentage of answered calls of each type that are delayed more than a specified number of seconds. We propose going beyond traditional congestion measures to focus on the expected value derived from having particular agents handle various calls. Expected value might represent expected revenue or the likelihood of first-call resolution. Value might also reflect agent call-handling preferences. We show how value-based routing (VBR) and preference-based routing (PBR) can be introduced in the context of an existing SBR framework, based on static-priority routing using a highly-structured priority matrix, so that constraints are still met for standard congestion measures. Since an existing SBR framework is used to implement VBR and PBR, it is not necessary to replace the automatic call distributor (ACD). We show how mathematical programming can be used, with established staffing requirements, to find a desirable priority matrix. We select the priority matrix to use during a specified time interval (e.g., 30-minute period) by maximizing the total expected value over that time interval, subject to staffing constraints.

Keywords: customer contact centers, telephone call centers, skill-based routing, value-based routing, preference-based routing, indirect value-based routing, priorities, mathematical programming, assignment problem, turnover, churn.

February 24, 2005; Revision: June 13, 2006

1. Introduction

Many services - from emergency to retail - are largely teleservices, in that the people who provide the service and the people who receive the service, herein called *customers*, are remote from each other, at least when the service is initiated. With a teleservice, the delivery of service is provided or enabled by a customer *contact center*; e.g., see Cleveland and Mayben (1997) and Gans, Koole and Mandelbaum (2003).

A primary resource in a contact center is the group of people who respond to these service requests - the customer service representatives, herein called *agents*. The classical contact center is the (telephone) call center, wherein the interactions are telephone calls, but the telephone is no longer the only means of interaction. Alternative media such as email, fax, web pages and instant messaging are on the rise. Contact centers are supported by quite elaborate information-and-communication-technology (ICT) equipment, such as a private branch exchange (PBX), an automatic call distributor (ACD), personal computers (PC's) and assorted database systems. The ICT equipment and the technology of voice-over-Internet protocol (VoIP) reduce the requirement of spatial proximity: This technology makes it possible to have virtual contact centers of various kinds, ranging from a small connected group of large call centers on different continents to a large number of individual agents working out of their own homes.

Contact centers usually handle several different kinds of service requests, herein referred to as *calls*. For example, telephone callers may speak different languages or may call about different promotions. Thus, it is rarely possible or cost-effective to have every agent be able to handle every type of call. Consequently, the agents tend to have different call-handling skills, in different combinations. Fortunately, contact centers can handle diverse calls with agents having diverse skill sets, because modern ACD's have the capability of assigning calls to appropriate agents.

Contact centers maintain elaborate management-support systems to address important aspects of contact-center performance. First, there are *customer-relationship-management* (CRM) systems, that help ensure that customers are being served appropriately and business objectives are being met. The CRM systems keep information about customers, including a history of previous interactions. By analyzing the data (via "business analytics"), CRM systems enable retail businesses to know the potential sales value of each interaction. Retail businesses can also ensure that important customer information is immediately available on

the agent's computer screen just prior to establishing contact. The CRM systems also provide reports on how well business objectives are being met, by individual agents as well as by the contact center as a whole.

There also are elaborate *workforce-management* (WFM) systems, that aim to ensure that the right number of agents with the right skills are in the right place at the right time. The WFM systems have sophisticated algorithms and computer software to perform forecasting, staffing and scheduling. These three functions aim to determine the required number of agents with specified skills over different time periods (e.g. weeks, days, and half-hours). The WFM systems also help manage agent performance and compensation.

Within this context, contact centers have gone beyond the traditional load-based routing to perform *skill-based routing* (SBR). Traditional load-based routing is designed to ensure that calls are handled *promptly*; skill-based routing is designed to ensure that calls are not only handled promptly but are also resolved *properly*. Promptness is typically defined in terms of constraints upon standard congestions measures. For example, the percentage of type-j calls abandoning before starting service should be at most $a_j\%$; the average delay before starting service of type-j calls should be at most d_j seconds; and $x_j\%$ of all answered type-j calls should be answered within y_j seconds. Proper call handling typically means that the agent should possess, a priori, the required call-handling skills. There usually are only two alternatives: the agent is either judged to have the required skills or is not.

We propose going beyond these traditional congestion measures and zero-one skill requirements to focus on the *expected value accrued from having the agent handle the call*; i.e., we propose going beyond traditional SBR to obtain *value-based routing* (VBR). Expected value might represent expected revenue or the likelihood of first-call resolution. With modern CRM systems, such information is actually available.

Value might also reflect *satisfying agent call-handling preferences*. Indeed, we regard the first main contribution of this paper as being a *new paradigm for call routing* that responds to agent call-handling preferences. We propose a new *preference-based-routing* (PBR) algorithm, which aims to respond to agent call-handling preferences and thereby help empower the agents, so as to improve agent jobs satisfaction, thereby reducing agent turnover and absenteeism, and hopefully improving customer service as well.

We see two major challenges in advancing this idea. First, it remains to show that those desired benefits of PBR can actually be realized. Second, for both VBR and PBR, it remains to show that an algorithm can be developed that is workable, in the sense that traditional

congestion constraints can still be met, efficiently, without excessive staff. In this paper, we only address the second challenge. In future work we hope to address the first challenge as well.

The second main contribution of this paper is our response to the second challenge. We develop VBR and PBR algorithms that can efficiently satisfy traditional congestion constraints. Our approach relies on previous staffing and routing algorithms developed in Wallace and Whitt (2005), which have been shown to be effective through extensive simulation experiments. We construct versions of VBR and PBR in the context studied by Wallace and Whitt (2005). We use mathematical programming to construct a special VBR algorithm we call *indirect VBR*. We then construct our PBR algorithm as a special case of indirect VBR, by expressing agent preferences in the form of values. Since VBR applies more broadly, we first discuss it. Afterwards, we discuss PBR.

We assume that the staffing requirements for the contact center have already been determined. That is commonly done by the WFM system. Because of the time-varying nature of demand, those staffing requirements typically vary across the day, but are treated as constant over individual *staffing intervals*, e.g., often 30 minutes long. There are established procedures for setting time-varying staffing requirements in response to time-varying demand; see Green et al. (2005). We will assume that we are considering a single staffing interval, and not discuss staffing intervals any further.

Thus, in this paper we are concerned with call routing in a specified time interval, such as a half hour. We assume that there are m call types and n agents. We assume that a *value* $v_{i,j}$ (a real number) has been assigned for agent i handling a type-j call for each i and j for which an assignment is allowed. In a *sales environment*, the value $v_{i,j}$ might be the *expected dollar value* of having agent i handle the type-j call. On the other hand, these values $v_{i,j}$ might represent quantitative expressions of the strategic "value" management attributes to having agent i handle a call of type j. For example, those value scores might be based on management estimation of agent skill levels. It is natural to go beyond recognizing whether or not an agent has the skill to speak French; we might try to assess how well the agent can handle French-speaking calls.

Unlike expected revenue, values apply in any contact-center environment, e.g., in emergency services as well as in retail. We believe that "value" is good terminology, because it encompasses many different dimensions of performance we might want to consider. The important point is to go beyond the traditional congestion measures. But whatever definition of value is used, we

need an algorithm that ensures that traditional congestion constraints are met while achieving high value.

To meet that goal, we introduce indirect VBR, which uses static routing based on highly-structured priority matrices, as in Wallace and Whitt (2005), which in turn is patterned after algorithms in current ACD's. We apply mathematical programming to construct a priority matrix that maximizes the total value subject to constraints on the priority matrix obtained from the staffing algorithm. Since we are able to apply an existing SBR framework, we can work with current ACD's.

To appreciate our proposal, it is good to consider the call-routing problem we are facing. We have multiple types of calls with uncertain call-handling requirements arriving randomly over time. With call routing, we are thus faced with with a very complex dynamic stochastic control problem. Full optimality is beyond the state of the art of current analysis techniques. Moreover, the call routing is intimately linked to the agent staffing.

However, Wallace and Whitt (2005) found that there exists a reasonable approach to this complex problem. They found that, with adequate agent cross-training, it is possible to staff in a relatively simple way, acting as if all agents had all skills, so that the full complexities of SBR need not be directly addressed. And, with an appropriate staffing algorithm, they found that it suffices to use a suboptimal static-priority routing algorithm. They used simulation to verify that the congestion constraints are met without excessive staff, and to make adjustments if that is not initially the case. Our idea is to exploit those results for VBR and PBR.

Here is how the rest of this paper is organized: In Section 2 we formulate a performance target, based on values, which we can use to evaluate alternative routing algorithms. In Section 3 we specify the assumed existing SBR algorithm, which is a static-priority routing algorithm based on a priority matrix P. That is intended to illustrate what is done today in most ACD's.

Next, in Section 4 we discuss a direct VBR algorithm, in which the priorities are simply set equal to the values. If such a simple procedure were effective, then there would be no need to do what we propose. To see the shortcomings of direct VBR, we present a simple two-class example. We show that the performance of direct VBR can be arbitrarily bad (compared to our proposed indirect VBR and to what is commonly done in practice).

In Section 5 we introduce our proposed *indirect VBR algorithm*, in which mathematical programming is employed to convert the values into a priority matrix of the required form. We impose additional structure on the priority matrix. With that additional structure, the constraints in the mathematical program ensure, first, that traditional congestion constraints

are still met and, second, that the new indirect VBR algorithm is put in the framework of the SBR algorithm in Section 3.

In Section 6 we consider ways to incorporate agent preferences in the routing. We do so by expressing agent preferences in the form of values, and then applying the indirect VBR algorithm. We also propose several extra measures to ensure that management can maintain control, if it is deemed necessary. We describe a specific implementation of the PBR algorithm in Excel Solver to study its performance on small examples. We briefly state conclusions in Section 7.

2. A Performance Target

Before defining any specific routing algorithm, it is appropriate to consider the contact-center goals. Thus we start by formulating a *performance target* that can be used to judge candidate routing algorithms, including the one we present.

Given that we are starting with values, it is natural to base the target directly on the values $v_{i,j}$, but we are also interested in traditional congestion measures. We still want relatively few calls to abandon and most calls to be answered promptly. Thus, we ascribe a cost (negative value) $c_{L,j}$ to each type-j call that is lost due to customer abandonment and a cost $c_{D,j}$ to each served type-j call that has to wait more than y_j seconds. (This is only meant to be illustrative. We might pay attention to exactly how long abandoning calls waited before they abandoned and exactly how long served calls had to wait.)

We now define a specific *total-value function*. To do so, we specify a time interval over which performance is to be judged, e.g., a typical half hour. Let $N_{i,j}$ be the number of type-j calls answered by agent i within y_j seconds during the specified time period. Let L_j be the number of type-j calls lost because of customer abandonment within the designated time period, and let D_j be the number of answered type-j calls delayed beyond y_j seconds within the designated time period. Clearly, the quantities $N_{i,j}$, L_j and D_j should be regarded as random variables, which depend on the unknown pattern of arrivals and service times as well as the routing policy used.

For each routing algorithm, let the *total value* gained from the routing algorithm under consideration be the expected total value, i.e.,

$$\mathbf{V} \equiv \sum_{i=1}^{n}\sum_{j=1}^{m}(E[N_{i,j}]\cdot v_{i,j}) - \sum_{j=1}^{m}([E[L_j]\cdot c_{L,j}] + [E[D_j]\cdot c_{D,j}]) \ . \tag{2.1}$$

We may then take as our overall goal the maximization of the total value expressed in

(2.1). We can estimate the expected values of the three random variables in (2.1) by first specifying a stochastic model for the arrivals of call types and their service-time distributions, and then applying computer simulation. For any given stochastic model, we can use computer simulation to evaluate the performance of each alternative routing algorithm, e.g., as in Anton et al. (1999), Brigandi et al. (1994) or Leamon (2004).

However, we may want to go beyond (2.1) to ensure that traditional constraints on congestion measures are satisfied. We might well judge it difficult to properly balance the costs and benefits via (2.1). Indeed, that is our position. Hence, we may wish to consider familiar performance constraints. For that purpose, let $N_j \equiv \sum_{i=1}^{n} N_{i,j}$ be the total number of type-j calls in the time period. In addition to (2.1), we may want to impose the traditional constraints

$$\frac{E[L_j]}{E[N_j]} \le a_j \quad \text{and} \quad \frac{E[D_j]}{E[N_j]} \le (1 - x_j) \quad \text{for all} \quad j \ , \tag{2.2}$$

for appropriate constants a_j and x_j.

We present a candidate routing algorithm below, but we should hasten to say that we do not attempt to find the optimal routing algorithm. That seems to be well beyond current analytical capabilities. Instead, we aim to find a *favorable routing policy*, which performs relatively well with respect to the criteria above. We take a *conservative approach*, trying to achieve high value, while still *guaranteeing that the standard performance constraints in (2.2) are met*. We rely on the (external) staffing algorithm to determine the required numbers of agents with different combinations of skills. We use a (suboptimal) static-priority routing algorithm based on a priority matrix. We use a mathematical program to choose the best priority matrix: We maximize the total value, subject to the staffing and performance constraints.

We optimize, but only over the priority matrices, not nearly over all possible routing algorithms. We design our algorithm not to allow traditional constraints to be violated. Thus, our algorithm can be viewed as assigning agent names, with specified values, to pre-determined positions with specified call-handling behavior (skills and priorities).

And we do this in the context of an existing SBR algorithm. We believe it is very important to develop the new VBR algorithm in the framework of an existing SBR algorithm, so that it will be unnecessary to change the routing algorithm in existing ACD's. We do this for a specific SBR algorithm, but we believe our approach can also be adapted to other existing SBR algorithms as well. We specify a specific SBR algorithm framework next.

3. A Priority-Based SBR Algorithm

In this section we introduce a specific SBR algorithm, which we regard as illustrative of current SBR algorithms. Indeed, the SBR algorithm here is a minor modification of the routing algorithm in Wallace and Whitt (2005), which in turn is a variant of SBR algorithms commonly in practice. In various ways, the SBR algorithm here can be made more elaborate, with additional controls, e.g., see Section 2 of Sisselman and Whitt (2005a), but we deliberately keep things simple here in order to focus on the main ideas. The SBR algorithm here is intended to provide a framework for our indirect VBR algorithm, which we introduce in Section 5.

3.1. The Priority-Matrix Framework

We consider a contact center with n agents, who respond to m types of inbound calls. Routing decisions are based on an $n \times m$ *priority matrix* P. The matrix element $P_{i,j}$ gives the priority level for agent i to handle call type j. The i^{th} *row* of the matrix P determines the routing for agent i, while the j^{th} *column* of the matrix P determines the routing for any call of type j. We assume that $P_{i,j}$ is a nonnegative real number for each call type j that agent i has the required skill to handle. We do not assign numerical values to $P_{i,j}$ for all call types j, if any, that agent i cannot respond to under any condition.

Here we take the individual values $P_{i,j}$ as given. At this point we are concerned with routing for given staffing and given priority matrix P.

3.2. The Routing Algorithm

Given the priority matrix P, we can specify the routing algorithm. There are two situations in which action should be taken: First, we must decide what to do *when a new call arrives*; second, we must decide what to do *when an agent becomes free* after completing a call.

1. When a new type-j call arrives, route (assign) the call to the available agent i with the highest positive priority level $P_{i,j}$. Break ties with the *longest-idle-agent-routing* (LIAR) policy: route the call to that agent, among all those available agents with that best priority number, that has been idle the longest. If there is no available agent i, then put the arriving call at the end of a queue of waiting calls of type j. Let there be a separate queue for each of the m call types, with each queue served in the FIFO order.

2. When agent i becomes free, after completing a call, look for waiting calls to assign to agent i. Let the candidate calls to assign to agent i be from the front of the m call-type queues (the calls of each type that have been waiting the longest). Assign the waiting call of type j (in the front of its queue) with the highest priority. Break ties by assigning the call that has been waiting the longest.

Unfortunately, the algorithm as specified so far may not be effective, but we can make it effective by imposing additional structure, which we do in Section 5.1.

4. Direct VBR and Its Limitations

We can obtain a direct VBR algorithm by simply identifying the required priorities with the given values, i.e., by letting $P_{i,j} = v_{i,j}$ for all i and j, where $v_{i,j}$ are the given values. Unfortunately, however, such a simple procedure does not perform well consistently.

In this section we present a simple example to show why we need to go beyond direct VBR. We have chosen the example so that system performance and its deficiencies are easy to understand, without having to do any detailed simulation or involved mathematical analysis. It should be evident that similar difficulties can occur in more realistic systems. For additional discussion about the complexities of SBR, see Garnett and Mandelbaum (2000) and Gans et al. (2003).

In our example there are two call types and two agent pools, as depicted in Figure 1. To reduce the importance of stochastic fluctuations, we make the model large. The large size makes it possible to apply deterministic fluid approximations, as in Whitt (2004, 2006a,b), which ignore all stochastic fluctuations. The fluid approximations justify the approximate performance descriptions given below, but the approximations should be convincing directly.

There are 2000 agents in each agent pool. All call service times are independent and identically distributed (i.i.d.) exponential random variables with mean 1. (We think of the mean service time being about 10 minutes, but we measure time in units of mean service times. That makes the arrival rate coincide with the offered load.) Calls of the two types arrive according to independent Poisson processes with arrival rates $\lambda_1 = \lambda_2 = 1800$ (calls per mean service time). Thus type-i calls have offered load $R_i = \lambda_i = 1800$, so that the total offered load is $R = 3600$. We also assume that waiting customers may abandon. Let the times to abandon be i.i.d. exponential random variables with mean 1.

Thus the total system is an Erlang-A model ($M/M/s+M$); e.g., see Garnett et al. (2002). Since the individual customer abandonment rate equals the individual service rate, the stochas-

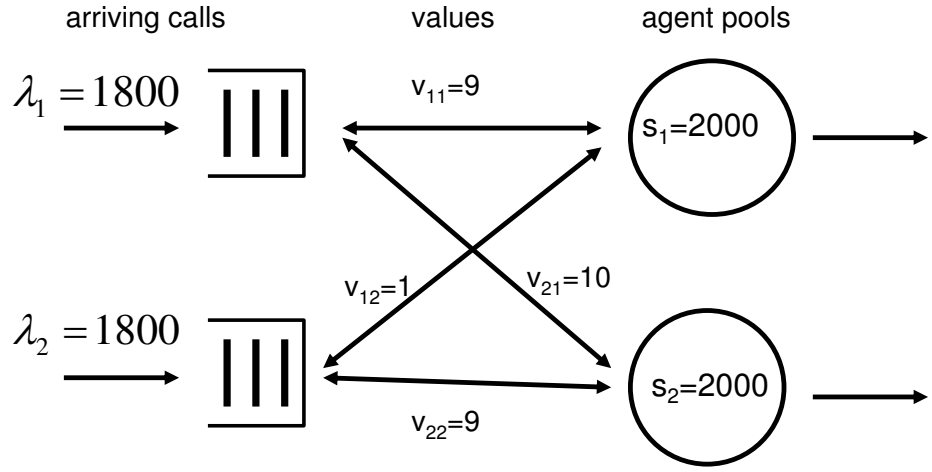

Figure 1: A customer contact center with two call types served by two agent pools.

tic process representing the number of customers in the system, either waiting or being served, in the $M/M/s + M$ model has the same probability law (finite-dimensional distributions) as the stochastic process representing the number of busy servers in the associated infinite-server ($M/M/\infty$) model (with same arrival process and service-time distribution). The steady-state number of customers in the system, either waiting or being served, thus has a Poisson distribution with mean 3600; p. 304 of Ross (2003). Since a Poisson distribution with a large mean is approximately normally distributed, the steady-state number of customers in the $M/M/s+M$ system is approximately normally distributed with mean 3600. Since the variance of a Poisson distribution equals its mean, the standard deviation is 60. Since 4000 is more than 6 standard deviations above the mean, the steady-state probability a customer is delayed is negligible. We deliberately construct the example in this way to ensure that congestion is not a factor. Thus, for the performance target in (2.1), only the first value term is relevant.

Now let us consider possible values. Recall that $v_{i,j}$ is the value (or expected value) accrued from agent i serving call-type j. For simplicity, suppose that all agents in the same pool have the same values. We can then simplify notation. Here let $v_{i,j}$ be the value accrued from an

agent in service pool i serving a type-j call. All values are thus specified by a 2×2 matrix $v \equiv (v_{i,j})$. For this example, let the 2×2 value matrix be

$$v = \begin{pmatrix} 9 & 1 \\ 10 & 9 \end{pmatrix} . \tag{4.1}$$

Note that the value matrix v in (4.1) is a great simplification. In general, for this example, we should have a 4000×2 value matrix, with a separate row for each agent. We also observe that the value matrix in (4.1) is realistic. From (4.1), we see that higher value is accrued for all agents from handling type-1 calls than type-2 calls, but the difference is greater for the agents in pool 1. Such values are representative of a *bilingual call center*, as in Green (1985) and Stanford and Grassman (1993, 2000). In that setting, all agents might do better with type-1 calls, because it is the dominant language, but the agents from pool 1, who speak only one language, do much worse when serving type-2 calls. The higher value 10 for agents in pool 2 serving type-1 calls might arise because the bilingual agents tend to be more skilled overall. Service skill might be correlated with language ability.

So, what happens in this example? Since calls are only delayed very rarely, the vast majority of all calls are answered immediately upon arrival. With direct VBR using the value matrix in (4.1), calls of both types first attempt to find an agent from agent pool 2. They take a pool-2 agent if one is available; otherwise they go to pool 1. Since the total number of agents is 4000, while the total offered load is 3600, there typically will be $300 - 500$ free agents at pool 1.

From time to time, working agents will complete their service and become free. However, at such moments, it is extremely unlikely that there will be any customers waiting, so they will have to wait in agent-pool queues. Since the arrival rate at pool 2 is so high, those agents will be snapped up quickly. Thus agent pool 2 is nearly fully busy all the time, processing calls at nearly the maximum possible rate of 2000 (per mean service time). Since the type-1 and type-2 call arrival rates are the same, and since Poisson arrival processes see time averages (PASTA) - see p. 293 of Wolff (1989) - the two classes tend to be using equal number of agents from pool 2. (That is evident as well from symmetry.)

Let $\lambda^D_{i,j}$ be the rate (again per mean service time) that agents from pool i are processing type-j calls. Then in this example we have approximately the processing rate matrix λ^D (D for direct), where

$$\lambda^D = \begin{pmatrix} 800 & 800 \\ 1000 & 1000 \end{pmatrix} , \tag{4.2}$$

which yields an approximate total value of

$$V^D = \sum_{i=1}^{2}\sum_{j=1}^{2} \lambda^D_{i,j} v_{i,j} = 27,000 \quad \text{(rate per mean service time)} \ . \tag{4.3}$$

So, what might we do instead of that? Following common practice, as a first step, we might create an agent pool dedicated to each call type. Then we would assign highest priority to having agents from agent pool j serve calls of type j, for each j. (Of course, that leaves open the question of which agents are assigned to each agent pool, which is precisely the problem we address in this paper.) As we do in the next section, we assume that the staffing has already been specified. Consistent with Figure 1, we might stipulate that the total number of agents is 4000, the number of agents handling type-1 calls with high priority and type-2 calls with low priority is $n_{1,2} = 2000$, and the number of agents handling type-2 calls with high priority and type-1 calls with low priority is $n_{2,1} = 2000$. We would then assign the agents to the agent pools to maximize the value. In this case, instead of using the value matrix v in (4.1) as our priority matrix, we are led to assign calls to agents according to the priority matrix

$$P = \quad \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \ , \tag{4.4}$$

where again the priorities $P_{i,j}$ in this 2×2 priority matrix apply to all agents in agent pool i serving a type-j call. Indeed, our indirect VBR algorithm in the next section will produce exactly this priority matrix, given the specified staffing. In this example, since there is excess service capacity, the priority matrix P in (4.4) just means that each call of type j would be handled by an agent in pool j. Of course, that is commonly done in contact centers, but direct VBR often fails to do it.

As indicated above, in this context it remains to assign the agents to the agent pools. The notation for the value matrix in (4.1) suggests a direct match, but that is nor required. However, given the value matrix in (4.1), that direct match is preferred, because $9+9 > 10+1$.

If we use the priority matrix P in (4.4) for routing, then we instead have approximately the processing rare matrix λ^I (I for indirect, as explained above, and to be explained more fully in the next section), where

$$\lambda^I = \quad \begin{pmatrix} 1800 & 0 \\ 0 & 1800 \end{pmatrix} \ , \tag{4.5}$$

which yields an approximate total value of

$$V^I = \sum_{i=1}^{2}\sum_{j=1}^{2} \lambda^I_{i,j} v_{i,j} = 32,400 \quad \text{(rate per mean service time)} \ . \tag{4.6}$$

Note that V^I is 20% greater than V^D.

On the other hand, it is easy to see that we can do even better than the indirect routing algorithm in this example by serving 200 type-1 offered load by agent pool 2, but that requires we deviate from the specified staffing. (That is, we take the agent pool sizes as given, but we do not specify $n_{1,2}$ and $n_{2,1}$.) We obtain the improved total expected value without assigning any agents from pool-1 to customer class 2, and accruing any of the low value 1. In this way we obtain the optimal (approximate) processing rate matrix λ^{opt} (*opt* for optimal), where

$$\lambda^{opt} = \begin{pmatrix} 1600 & 0 \\ 200 & 1800 \end{pmatrix}, \tag{4.7}$$

which yields an approximate total value of $V^{opt} = 32,600$ (rate per mean service time).

However, the main point of this example is to show the weakness of direct VBR. As a byproduct, this example also demonstrates that our proposed indirect VBR routing algorithm is not optimal, if we allow deviations in the staffing, because $V^I < V^{opt}$, but it is close. The primary reason that $V^I < V^{opt}$ is that we have over-staffed. Proper staffing would reduce the total number of agents down much closer to 3600, and we would lose the opportunity to do better than V^I.

There is a further problem with direct VBR. Both the direct VBR and the optimal routing in this example have the disadvantage that one class is under-utilized, while the other is over-utilized, potentially causing agent stress. In contrast, the utilization of all servers with indirect VBR is 90%. With direct VBR, we would need to increase the staffing level at service pool 2 to 3600 before we would see any slack there. The over-staffing is prevented if we rely on previously (properly) determined staff levels.

To further demonstrate the deficiencies of direct VBR, we construct a variant of the example above in which direct VBR performs even worse. Indeed, in this example the ratio V^I/V^D can be made arbitrarily large by a proper choice of the parameter ϵ. This example is somewhat contrived, but it shows that direct VBR can perform very poorly by allowing too much sharing.

Example 4.1. (*a more extreme example.*) We close this section by giving an example in which the ratio of the value rate received from indirect VBR (assigning all type-j calls to server group j) to the value rate received from direct VBR can be made arbitrarily large. To achieve that bad performance of direct VBR, we let (1) the differences in the relative values be greater, and (2) we make the arrival rates (and offered loads) of the two call types unbalanced. Let the service times and times to abandon be i.i.d exponential random variables with mean 1, just as before. Overall, we thus have the same $M/M/s + M$ model.

Let the arrival rate (and offered load) for call-type 1 be λ/ϵ, and for class 2 be λ. Let the server-pool capacity exactly match the offered loads, as shown in Figure 2.

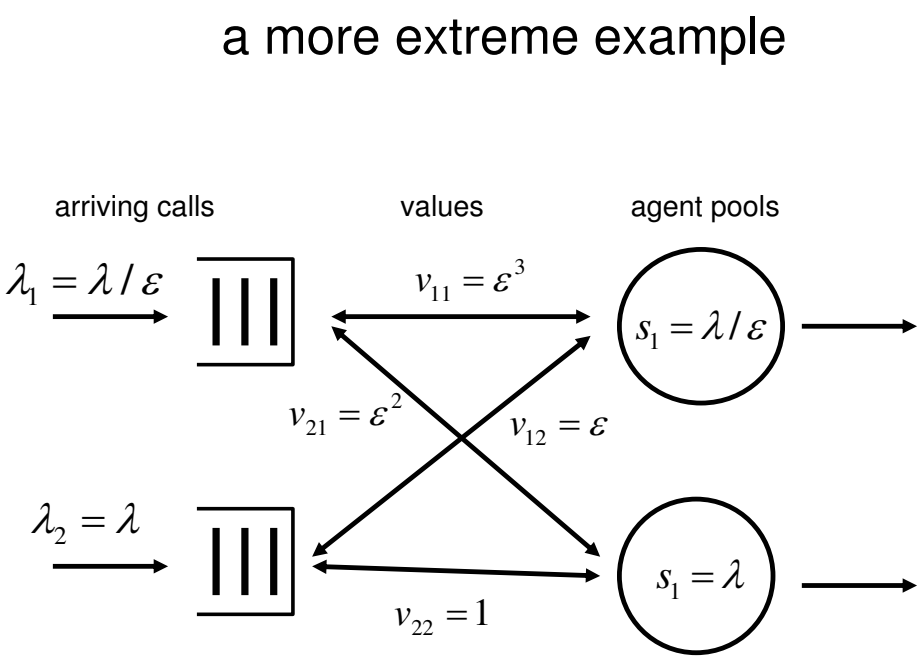

Figure 2: A more extreme example, in which V^I/V^D becomes arbitrarily large as $\epsilon \downarrow 0$.

We assume that the parameter λ is large, so that the simple deterministic approximations are again appropriate. Let the value matrix be

$$v = \begin{pmatrix} \epsilon^2 & \epsilon^3 \\ \epsilon & 1 \end{pmatrix}, \tag{4.8}$$

where ϵ is a small number. With the value matrix (4.8), all calls initially try to go to agent pool 2, just as before.

Let $\lambda^D_{i,j}$ be the processing rate of type-j calls by agents from pool i with direct VBR. Then we have approximately the processing rate matrix λ^D (D for direct), where

$$\lambda^D = \begin{pmatrix} \frac{\lambda}{\epsilon(1+\epsilon)} & \frac{\lambda}{1+\epsilon} \\ \frac{\lambda}{1+\epsilon} & \frac{\lambda\epsilon}{1+\epsilon} \end{pmatrix}, \tag{4.9}$$

which yields an approximate total value of $V^D = \epsilon(3+\epsilon^2)/(1+\epsilon)$ per mean service time.

In contrast, if we use the priority matrix P in (4.4), and simply serve each call type by its own agent pool, we would instead have approximately the processing rate matrix λ^I (I for

indirect, to be explained in the next section), where

$$\lambda^I = \begin{pmatrix} \frac{\lambda}{\epsilon} & 0 \\ 0 & \lambda \end{pmatrix} , \tag{4.10}$$

which yields an approximate total value of $V^I = \lambda(1+\epsilon)$ per mean service time. The ratio V^I/V^D goes to infinity as $\epsilon \downarrow 0$; indeed,

$$\frac{\epsilon V^I}{V^D} \to \frac{\lambda}{3} \quad \text{as} \quad \epsilon \downarrow 0 . \tag{4.11}$$

5. Indirect VBR

In this section we introduce our proposed indirect VBR algorithm. We start with general values, just as in Section 4, but now we convert those values into a priority matrix P with special structure. We first specify the additional structure for the priority matrix.

5.1. Additional Structure for the Priority Matrix

To ensure that we produce an effective priority matrix and thus an effective SBR algorithm, we follow Wallace and Whitt (2005). In doing so, we are being conservative, adding more restrictions than necessary, to be on the safe side.

First, we assume that there is an agent pool for each call type, where call-type j has highest priority for all agents in agent pool j. We assume that the system has been designed so that type-j calls are primarily served by agents in agent pool j. (Of course, it remains to determine which agents are in each agent pool.) Beyond this, the priority matrix is used to determine appropriate sharing, where agents from one pool handle calls of another type. This design is consistent with common practice.

We now go even further to severely limit the structure of the priority matrix. We find that the loss in flexibility is compensated for by the gain in control. We will here assume that there are only *two priority levels*: the possible values of $P_{i,j}$ are 1 and 2, with 1 being low priority and 2 high priority. We assume that each agent has one and only one call type at priority 2. We assume that each agent has at most two active skills, necessarily with at most one at each priority level. Here we will go further and assign each agent two skills, but that is not necessary, and may not be possible in some contact centers. Again, this is illustrative, but it follows Wallace and Whitt (2005). And it is consistent with the priority matrix P in (4.4) used in the examples in Section 4. Given that each agent has at most two skills, we can refer to them as the primary skill (high priority) and the secondary skill (low priority).

However, in practice agents may be allowed additional active skills. Such extra flexibility is especially important if the system needs to be able to respond to loads very different than were anticipated. However, adding extra skills is not automatically beneficial, because routing anomalies such as in Example 4.1 can result.

A key strategy in this paper is to rely on the staffing algorithm to determine the constraints on the numbers of required agents with different combinations of skills. With the structure above, the staffing algorithm needs to specify the numbers $n_{j,k}$ of agents that have primary skill j and secondary skill k, where $1 \leq j \leq m$, $1 \leq k \leq m$ and $j \neq k$.

To describe one feasible way to obtain those required staffing levels, we review the staffing algorithm in Wallace and Whitt (2005). In doing so, we assume that there is sufficient cross training, so that we can approximately size the contact center by acting as if all agents had all skills. Wallace and Whitt (2005) found that a little flexibility goes a long way: Assuming that the service-time distributions do not differ greatly, they found that it suffices for each agent to have at most two skills, provided there is appropriate sharing across the agent pools, to ensure economies of scale. Surprisingly, perhaps, they found that the required number of agents when each agent has two skills is within a single agent of the required number when all agents have all skills. Moreover, it suffices to use a suboptimal routing algorithm, such as our indirect VBR. Supporting theoretical results have been established by Armony et al. (2006) and Chevalier et al. (2004).

We do not insist on precisely this staffing method, but it produces an appropriate total staffing level plus an appropriate staffing level for each server pool j. When the total offered load is relatively large, but not extremely large (e.g., about 100), so that the total number of agents is of that magnitude, and none of the individual agent pools is too small, then most type-j calls will be served by agents in agent pool j, but the sharing plays a critical role to provide good performance without excess staff. The simulation experiments in Wallace and Whitt (2005) provide good illustrations.

To define the staffing algorithm from Wallace and Whitt (2005), we introduce additional notation: Let λ_j be the arrival rate for call-type j, let μ_j^{-1} be the mean service time for call-type j (assumed not to depend on the agent), and let $R_j \equiv \lambda_j/\mu_j$ be the offered load for type j. (We assume that the mean service times $1/\mu_j$ do not vary greatly.) Let $R \equiv R_1 + \cdots + R_m$ be the total offered load.

Given the total offered load R, then we can let the total number n of agents be the least integer greater than or equal to $R + \gamma\sqrt{R}$ for some positive γ. The parameter γ broadly

characterizes the quality of service, with higher values yielding higher quality of service, but at the expense of additional staff. We then assign n_j agents to agent pool j (who will assign highest priority to call type j), where n_j is the least integer greater than or equal to x_j, with

$$x_j = R_j + \frac{\gamma\sqrt{RR_j}}{\sum_{j=1}^{m}\sqrt{R_j}} \ . \tag{5.1}$$

Let $n_{j,k}$ denote the number of agents in agent pool j (having primary skill j) that will be assigned secondary skill k, i.e., the number of agents that will be assigned skill-pair (j,k). Let $x_{j,k}$ be an initial estimate of $n_{j,k}$, which may not be integer. For any j and k with $k \neq j$, we let

$$x_{j,k} = \frac{n_j n_k}{n - n_j} \ . \tag{5.2}$$

The sum of $x_{j,k}$ over k is n_j. Finally, we round to obtain the desired numbers $n_{j,k}$. Aside from the rounding, this rule lets the number $n_{j,k}$ of agents in agent pool j having secondary skill k be directly proportional to both n_j and n_k. The $n_{j,k}$ agents with primary skill j and secondary skill k have identical rows in the priority matrix P: Those rows have a 2 in column j and a 1 in column k.

However, agent skills may not allow sharing with all other call types, as dictated by (5.2). It is significant that full sharing via (5.2) is not actually required. Indirect sharing will suffice, provided that there is appropriate *chaining* throughout the agent pools, as in Jordan and Graves (1995) and Jordan et al. (2003). For example, with four agent pools, we might have $n_{1,2} > 0$, $n_{2,3} > 0$, $n_{3,4} > 0$ and $n_{4,1} > 0$, but $n_{j,k} = 0$ for all other pairs (j,k). Wallace and Whitt (2005) found that sharing with chaining suffices when agents have only two skills.

Since each agent is given two skills, we have fully specified the priority matrix in our framework by this procedure. We then apply simulation to test for feasibility with regard to the constraints in (2.2) and then iteratively refine the numbers $n_{j,k}$ as needed to make the staff as small as possible, subject to the performance constraints being met. See Wallace and Whitt (2005) for further discussion, including simulation examples. As stated before, we assume that this staffing step is complete when we consider routing. But Wallace and Whitt (2005) provides an effective staffing algorithm.

5.2. Using the Values to Construct the Priority Matrix

We now consider how to use the values to construct a priority matrix P with the structure specified above. We make no special assumptions about the values, but it is natural to require that they be approximately in the same scale, because we are going to maximize the total

value over all assignments, as in the objective function (2.1). So we will implicitly be making an interpersonal comparison of utilities.

Our goal here, then, is to develop an algorithm to create the priority matrix P, aiming to determine the highest two call-handling priorities for each agent. For the single staffing interval we are considering, we assume that the staffing requirements have already been specified, perhaps by drawing upon Wallace and Whitt (2005). Hence we assume that the required number of agents with each priority pair has been determined. Let $n_{j,k}$ be the number of agents needed for priority pair (j,k), where j is the high-priority call type and k is the low-priority call type. We may have $n_{j,k} = 0$ for some pairs (j,k).

What is not yet determined is which agents will be assigned each priority pair. We are assuming that there is further choice to be made. Clearly, contact centers will differ in their ability to take advantage of VBR. We are assuming that agents have a range of skills, so that there is flexibility in how the priority pairs can be assigned. That flexibility is crucial to allow us to pay attention to values, while still meeting the constraints on traditional performance measures, as in (2.2).

To allocate priority pairs to agents, we will use values $v_{i,j,k}$ for agent i handling priority-pair (j,k) for all i and (j,k), $1 \le i \le n$, $1 \le j \le m$ and $1 \le k \le m$, again with $j \ne k$. (There are n agents and $m(m-1)$ distinct call-type pairs.) However, we are initially given only values $v_{i,j}$ for agent i handling a call of type j. We could of course plan to start by assigning values directly to priority pairs, which would give us $v_{i,j,k}$.

However, we think it is natural to start with values in the form $v_{i,j}$, as we have done so far. So we next convert initial values $v_{i,j}$ for agent i handling call-type j into corresponding values $v_{i,j,k}$ for agent i being assigned priority-pair (j,k). We propose taking a weighted average of the values assigned to individual call types, but placing less weight on the lower-priority call types. Specifically, we propose letting

$$v_{i,j,k} = (1 - p_{i,j,k})v_{i,j} + p_{i,j,k}v_{i,k} \ , \tag{5.3}$$

where $p_{i,j,k}$ in (5.3) is a weight satisfying $0 < p_{i,j,k} < 0.5$.

Of course it remains to specify the weights $p_{i,j,k}$ in (5.3); we discuss that now. It is natural to think of the weight $p_{i,j,k}$ being defined such that $p_{i,j,k}$ is approximately the proportion of type-k calls handled by agent i when agent i is assigned priority pair (j,k), while $1 - p_{i,j,k}$ is approximately the proportion of type-j calls handled by agent i. That goal is hard to achieve directly, because the proportions of calls of different types handled by the agents depends on all

the priority-pair assignments and the random traffic that appears during that staffing period, but it should suffice to use rough estimates. A simple approach is to assume that $p_{i,j,k} = p$ for all i, j and k; e.g., we might let $p = 0.3$, and then later make adjustments to make the weights correspond approximately to the proportion of calls of call-type k handled by agent i. A rough estimate should suffice.

It is natural to consider an iterative simulation-optimization algorithm. We set $p_{i,j,k}$ at some initial reasonable value for each triple (i, j, k), and then (i) optimize in the manner of the next section to determine the priority-pair assignments and the full priority matrix P, and then (ii) simulate the contact center with that priority matrix P and the forecasted call traffic. In the simulation, we observe the actual proportions of low-priority calls handled by the agents. We then update the weights $p_{i,j,k}$ and repeat the steps above. We anticipate that only a few iterations should be required to obtain reasonable weights $p_{i,j,k}$, but that needs to be verified.

As we go forward to consider the optimization to determine the priority-pair assignments and the priority matrix P, we assume that the values $v_{i,j,k}$ have been determined.

5.3. Mathematical Programs

Given the specified staffing needs - the numbers $n_{j,k}$ of priority pair (j, k) needed - and the values - $v_{i,j,k}$ for agent i being assigned priority pair (j, k), we can determine the agent priority-pair assignments by solving a mathematical program, which in this case is an *assignment problem*; e.g., see Chapter 7 of Bertsimas and Tsitsiklis (1997) or Chapter 8 of Hillier and Lieberman (2001). An assignment problem can be solved by a special highly efficient linear programming algorithm, such as the Hungarian algorithm. Fortunately, it automatically has integer solutions.

Our approach to routing is similar to commonly applied applications of linear programs to *schedule* agents in contact centers; see Segal (1974). The optimization problem here is actually far easier, because we are concerned with a single staffing interval, not shifts throughout the day, with constraints on breaks, etc.

We can specify the priority pairs the agents are assigned with *decision variables* $x_{i,j,k}$. We let $x_{i,j,k} = 1$ if agent i is assigned priority-pair (j, k); we let $x_{i,j,k} = 0$ if agent i is not assigned priority-pair (j, k), where we understand that $j \neq k$. We will obtain decision variables with $x_{i,j,k} = 1$ for precisely one pair (j, k). Given the $0 - 1$ decision variables $x_{i,j,k}$, we can define the priority matrix P by letting $P_{i,j} = 2$ and $P_{i,k} = 1$ if $x_{i,j,k} = 1$.

Here is the assignment problem:

$$\text{Maximize} \quad \text{Total Value} \quad = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1,k\neq j}^{m} v_{i,j,k} x_{i,j,k} \tag{5.4}$$

over real-valued decision variables $x_{i,j,k}$, subject to the constraints

$$\begin{aligned} \sum_{j=1}^{m} \sum_{k=1,k\neq j}^{m} x_{i,j,k} &= 1 \quad \text{for all} \quad i \\ \sum_{i=1}^{n} x_{i,j,k} &= n_{j,k} \quad \text{for all} \quad (j,k) \quad \text{with} \quad j \neq k \ . \\ x_{i,j,k} &\geq 0 \qquad \text{for all} \quad i,j \quad \text{and} \quad k \ . \end{aligned} \tag{5.5}$$

The linear program in (5.4) and (5.5) is not written directly in the form of an assignment problem, but it is equivalent to an assignment problem. We can think of n agents and n possible assignments, with the n assignments being an enumeration of the n specified priority pairs (j,k), with the value of assigning agent i to job (agent pair) (j,k) being $v_{i,j,k}$. We do not need to constrain the decision variables $x_{i,j,k}$ in advance to be one of the integers 0 or 1. That is implied by the first and third constraints, because this is an assignment problem.

However, we might want to consider alternative more complicated mathematical programs with more flexibility. In particular, it might be advantageous to relax the equality constraints and replace them by lower and upper bounds. For that purpose, we may introduce lower and upper bounds n_j^L and n_j^U on the number of agents having primary skill j, and $n_{j,k}^L$ and $n_{j,k}^U$ lower and upper bounds $n_{j,k}^L$ and $n_{j,k}^U$ on the number of agents having priority-pair (j,k). The resulting mathematical program becomes an *integer program*, because the decision variables $x_{i,j,k}$ now must be forced to be integers. For background on integer programs, see the mathematical programming books cited before and Wolsey (1998). Integer programs are substantially more difficult to solve than linear programs, which in turn tend to more difficult than assignment problems. Fortunately, however, some modified mathematical programs can still be solved by the network simplex method.

6. Preference-Based Routing

In this section we show how PBR can be implemented within the framework of indirect VBR, but first we discuss the motivation for considering PBR. For additional discussion, see Sisselman and Whitt (2005a, b) and Whitt (2005, 2006c).

6.1. Performance Problems

Unfortunately, despite the complex systems supporting contact centers, and perhaps partly because of those complex systems, the performance of contact centers frequently falls far short of expectations. Contact centers often are not able to meet business objectives; e.g., see Cleveland and Hash (2004). First and foremost, customers are often dissatisfied with their contact-center experience. Even though there are elaborate WFM systems, contact centers often are not staffed efficiently. Even though there is the capability of performing skill-based routing, often the routing does not achieve the desired results. There often are too many agents with some skills, but not enough agents with other skills, so that some agents are unproductive, while others are overworked. Despite all the management-support systems, service levels and other performance targets often are not met. As a consequence, some service requests are handled adequately, but too many others are not. Moreover, costs often are higher than expected, while revenues are lower. Despite all the sophistication, contact-center performance frequently does not nearly reach its potential.

There are clear signs that contact centers are not achieving their desired goals. There is also strong evidence that it is currently difficult to maintain an energized, effective workforce; e.g., see Deery and Kinnie (2004) and Holtgrewe et al. (2002). Indeed, there is strong evidence of agent job dissatisfaction, based on frequent reports of *four telling conditions*: (i) high turnover or churn, (ii) chronic absenteeism, (iii) frequent deviation from work schedules, and (iv) agent fatigue toward the end of the day.

Indeed, many contact centers report agent-retention problems, expressed as high *churn*; see Gans et al. (2003) and Batt et al. (2005). There are significant costs associated with high turnover, which are often not fully appreciated; see Bliss (2004). When considering the costs of employee turnover, it is natural to classify the costs, dividing them into two types: (i) transition costs and (ii) productivity costs.

Transition costs account for the per-agent cost of terminating the departing agent, recruiting and training the new agent to replace the departing one, and disruption costs associated with the change, such as the cost of hiring a temporary employee, and the costs of managers coping with the change, such as the cost of performing exit interviews, the administrative cost of stopping benefit deductions and starting benefit enrollments, and so forth.

Since new agents typically must undergo a significant start-up learning period in order to perform effectively, high turnover also tends to produce significant *productivity costs*. High

turnover implies that too many agents are in the start-up learning period. More importantly, however, high turnover strongly indicates that many agents are dissatisfied with their job, and that job dissatisfaction is likely to make the agent a less effective worker; see Batt (2002). (Whitt (2006c) develops models to help analyze the productivity benefits of increased retention.)

6.2. Root Causes

When these agent-performance problems are discussed by contact-center management, we often hear direct or indirect criticism of the agents. The agents are criticized for their poor work ethic. We hear management express their frustration. They complain that their employees are not motivated and do not work efficiently. Frequently, the management response is to put even greater pressure on the agents. For example, in an attempt to achieve better adherence to work schedules, contact-center management might adopt a pervasive call monitoring system, recording every call.

However, we believe that such responses may be a mistake. To put the issue in perspective, it is helpful to recall the long history of the quality movement in manufacturing, led by such pioneers as W. Edwards Deming and Joseph M. Juran; see Deming (2000) and Juran and Godfrey (1999). Similar failures in manufacturing were blamed on inadequacies of the workers, but experience and statistical evidence showed that the key to improved performance is usually improved processes. That is highlighted by the 85/15 rule of Deming and Juran, which says that at least 85% of the problems are system problems, while less than 15% of the problems are due to the workers.

It is important to recognize that the fundamental problem in contact centers may be with contact-center processes, not the agents themselves. The easily measurable quantities - turnover, absenteeism, schedule deviation and fatigue - are all indications that agents are dissatisfied with their job. Indeed, research studies show that contact-center agents frequently do not have a sense of wellbeing in the workplace; see Holman (2002, 2003) and Witt et al. (2004). When agents are required to respond to telephone calls by carefully following prepared scripts, the agents inevitably feel a loss of control, a lack of autonomy. When contact-center management adopts a pervasive monitoring system, recording every call, the agents inevitably experience anxiety and stress; see Witt et al. (2004).

6.3. Empowering the Agents

Indeed, it is widely recognized that agent job dissatisfaction is a serious contact-center problem. The importance of agent motivation and compensation is frequently emphasized; see Cleveland and Hash (2004). Others have said that contact-center managers need to empower the agents; e.g., see Spraetz (2004). However, the traditional responses may not be enough. Radical changes may be needed.

In this paper we aim to contribute to significant work redesign in contact centers. Our thesis is that, in many circumstances, contact-center performance can be improved by enhancing agent participation. That follows the general management principle that employee participation in decision making and problem solving usually improves the quality of work; again see Deming (2000) and Juran and Godfrey (1999); also see Hackman and Oldham (1976) and Karasek and Theorell (1990)

Thus we seek to *empower* the agents. We suggest considering structural changes in the way contact centers operate. Clearly, call routing is a major decision process in contact centers. We aim to improve agent participation by dynamically soliciting and responding to agent preferences about call handling. We thus propose going beyond skill-based routing to achieve *preference-based routing*. In addition to responding to calls promptly and properly, we increase agent satisfaction from the contact experience, and hopefully thereby increase customer satisfaction as well.

However, in this paper we do not attempt to prove our hypothesis about PBR; we do not try to demonstrate that PBR can actually empower contact center agents. And we do not claim that other factors may empower contact center agents even more than PBR. We leave that for future work and actual implementation. Instead, here we confine ourselves to showing how PBR can be deployed within existing routing algorithms in ACD's, without sacrificing traditional contact center performance. We do that by expressing PBR as a form of Indirect VBR, for which we have already made the case.

6.4. Expressing PBR as a Form of Indirect VBR

Following current practice, it is natural to assume that management assigns both the values and the additional constraining parameters (the staffing constraints in the mathematical programs). However, we obtain PBR instead if we let the agents participate in assigning the values $v_{i,j}$, leaving management to assign the additional constraining parameters. We might even let the values be assigned entirely by the agents.

We envision the agents expressing their call-handling preferences in real time through toolbars on their computer screens. The call routing would then be updated every half hour, in the manner described above, but it might be possible to allow even more rapid management response (which we do not discuss here). The agent values might simply represent their preferences for handling different call types, but agent values could also be generated in other ways. Expressed agent values might represent their personal assessment of their ability to handle different types of calls. Alternatively, agent values might be determined indirectly via agent bids in an interactive game, allowing dynamic interactions with management. A routing game would allow management to offer incentives, and agents to win rewards such as "loyalty points" for their bids and call handling.

Here we assume that agent values have been expressed, and we are concerned with how to make those values affect call routing. We first propose possible methods for modifying the values before applying the mathematical program. We propose value-determination using input from *both* management and the agents. We could let the values be directly specified by the agents, but in many cases it will be desirable to compromise between management priorities and agent preferences, as expressed via their separate value functions. It may not be enough to let management specify the additional constraining parameters.

Thus we show how we can combine *management-specified values* $v^m_{i,j}$ with associated *agent-specified values* $v^a_{i,j}$ in order to obtain *composite values* $v^c_{i,j}$. To do so, we define a function mapping the pair of functions (v^m, v^a) into the single function v^c. A simple specific way to do that is via the *product mapping*, yielding the *value product*:

$$v^c_{i,j} = v^a_{i,j} \cdot v^m_{i,j} \quad \text{for all} \quad i, j \ . \tag{6.1}$$

In addition, we provide management with *controls* to continually adjust between these three kinds of values. Management can choose *weights* $w^m_{i,j}$, $w^a_{i,j}$ and $w^c_{i,j}$, which in generally we allow to depend on the agent i and the call type j, with the property that, for each pair (i, j), $w^m_{i,j} \geq 0$, $w^a_{i,j} \geq 0$, $w^c_{i,j} \geq 0$ and $w^m_{i,j} + w^a_{i,j} + w^c_{i,j} = 1$. Then the final *weighted values* are

$$v^w_{i,j} = w^m_{i,j} \cdot v^m_{i,j} + w^a_{i,j} \cdot v^a_{i,j} + w^c_{i,j} \cdot v^c_{i,j} \ . \tag{6.2}$$

Clearly, PBR does not guarantee that management is able to respond to all agent work preferences. Some agents will have their preferences met by the assignments much better than others. That should be acknowledged to agents. Further measures should be provided to ensure fairness over time. Furthermore, if management exercises control with too much

force, then the priority assignment may not reflect agent preferences well. To benefit from this scheme, we need strong agent preferences, with adequate variation in preferences among all available agents, and the flexibility to respond to those preferences. It would be better if management did not have conflicting values, but we provided a framework to cope with possible value conflicts. The flexible controls allow management to place more weight on agent values over time as the agent demonstrates good performance. In that way, the agent can acquire *earned empowerment*, as is being investigated by Bamberger and Borin (2005).

6.5. Additional Value Structure

For implementation, we still need additional structure. We show one that might be provided. As before, we start with n agents responding to m types of calls. We assume that management determines its values $v^m_{i,j}$ for all nm pairs (i, j), assuming that they each take one of the four values in the set $\{0, 1, 2, 3\}$. Management then communicates these values along with appropriate incentives to the agents. In response, agent i for each i individually determines his values $v^a_{i,j}$ for each call type j, again assuming that they take one of the four values in the same set $\{0, 1, 2, 3\}$. By design, we use a symmetric structure. We then use the product function in (6.1) to form composite values $v^c_{i,j}$, which necessarily take one of the seven values in the set $\{0, 1, 2, 3, 4, 6, 9\}$. For the weighted value function in (6.2), we assume that $w_c = 1$, so that $v^w = v^c$; i.e., we work directly with the composite value function.

6.6. A Small Example

We implemented a prototype of PBR, using *Excel Solver* to solve the linear program; see Whitt (2005d). We describe a typical small example that we considered. For that example, there are $n = 10$ agents and $m = 8$ call types.

We classify calls by considering two *variables*: language and purpose. For each of these variables, we consider two *variable types*: (E)nglish and (F)rench for language, and (T)echnical and (S)ervice for purpose. That leads to $2 \times 2 = 4$ *call types*: ET, ES, FT and FS. That in turn leads to $4 \times 3 = 12$ possible *priority pairs*, of which we considered 8, as depicted in Table 1. In Table 1 we also display assumed numbers $n_{j,k}$ of required agent assignments for priority-pair (j, k).

We then assigned real-valued values ranging from 1 to 3 for each agent handling each call type. We then applied (5.3) with $p_{i,j,k} = 0.3$ to obtain corresponding values for the priority pairs. Finally, we applied Solver to determine the optimal assignment satisfying all

Table 1: The eight priority pairs with the required numbers of each.

Priority Pair j	Primary skill	Secondary skill	Required Number
$1 = (1, 2)$	ET	ES	2
$2 = (1, 3)$	ET	FT	1
$3 = (2, 1)$	ES	ET	1
$4 = (2, 4)$	ES	FS	1
$5 = (3, 1)$	FT	ET	1
$6 = (3, 4)$	FT	FS	1
$7 = (4, 2)$	FS	ES	2
$8 = (4, 3)$	FS	FT	1

the constraints. This small linear programming problem has 98 decision variables $x_{i,j}$. For this small example, Excel Solver found an optimal (all-integer) solution in 80 iterations, virtually instantaneously, in less than a second. By comparing the optimal solutions with minimum and maximum objective functions, we can see the range of possible total values that we can achieve. Clearly, depending on the way values are assigned, the optimization may provide more or less value benefit. In favorable circumstances, there is significant benefit.

6.7. Alternative Constraints

In this section we discuss alternative constraints for the linear program. It is significant that, even though we lose the structure of the assignment problem, many of these revised formulations are linear programs solvable by the network-simplex algorithm; see Chapter 7 of Bertsimas and Tsitsiklis (1997) and Chapter 9 of Hillier and Lieberman (2003).

We first modified the simple example in Section 6.6 to set equality constraints only for the numbers of agents with each primary skill. This linear program is easily solved too, but sometimes the solutions are inappropriate, having insufficient sharing. To do better, we imposed lower-bound constraints on the priority-pair assignments. In particular, we imposed four lower bounds to enforce a minimum amount of *chaining*, as discussed at the end of Section 5.1. We want to establish the sharing chain:

$$ET \to ES \to FS \to FT \to ET \ ; \tag{6.3}$$

i.e., some agents with primary skill ET will have secondary skill ES; some agents with primary skill ES will have secondary skill FS; some agents with primary skill FS will have secondary skill FT; and some agents with primary skill FT will have secondary skill ET. That will establish a full sharing chain throughout the four primary-skill agent pools.

The chaining structure ensures that there is sharing across the entire contact center, even

if indirectly. To appreciate the benefit of chaining, it is useful to consider a non-chaining alternative. Instead of the sharing graph in (6.3), we might have the sharing graph

$$ET \leftrightarrow ES \quad FS \leftrightarrow FT \ . \tag{6.4}$$

In (6.4), ES is the only secondary skill for an agent with primary skill ET, while ET is the only secondary skill for an agent with primary skill ES. Similarly, FS is the only secondary skill for an agent with primary skill FT, while FT is the only secondary skill for an agent with primary skill FS. It is as if all agents could only speak a single language. With the sharing structure in (6.4), the English-speaking agents and the French-speaking agents act as two separate contact centers, and the overall efficiency is reduced. There is sharing within each of these two groups, but not between them. This sharing scheme without full chaining is less efficient. For a given number of agents, the performance will be worse. For specified performance targets, more agents will be required.

To guarantee sufficient chaining, we place lower bounds b_1, b_2, b_3, and b_4, respectively, on the priority pairs (ET, ES), (ES, FS), (FS, FT), and (FT, ET). Of course, these lower bounds must be less than the numbers of agents with that primary skill; otherwise the constraints would be infeasible. But we do not insist that all agents necessarily share according to this scheme; otherwise there would be only 4 allowed priority-pair assignments. We use the bounds to ensure that there is sufficient sharing in a chaining structure, but we allow some slack to achieve a higher reward (and thereby better respond to agent preferences).

For the example in Section 6.6, we assumed that the numbers of agents with primary skills ET, ES, FS and FT are, respectively, 3, 2, 3 and 2. With those numbers, we let the 4 lower bounds be $b_1 = 2$, $b_2 = 1$, $b_3 = 2$ and $b_4 = 1$. That allows one agent with each primary skill to have a different secondary skill.

Having made this generalization, linear programming still applies. For larger problems, it is important that the network simplex method applies to networks with lower bound constraints added to the arcs. That is the case, as discussed on p. 432 of Hillier and Lieberman (2001).

7. Conclusions

We have shown how to go beyond traditional skill-based routing (SBR) in contact centers to obtain value-based routing (VBR), which aims to assign calls to agents to achieve the maximum expected value, subject to constraints ensuring that traditional congestion constraints are still met. Expected value might represent expected revenue or the likelihood of first-call resolution.

Value might also reflect satisfying agent call-handling preferences. Indeed, there might be some composite value that reflects all these factors, but the main focus here is on ensuring that traditional congestion constraints are still met.

To accomplish that goal, we proposed indirect VBR. In Sections 3 and 5 we developed indirect VBR in the context of an existing SBR algorithm – a static-priority routing algorithm based on highly-structured priority matrices, as in Wallace and Whitt (2005). We used mathematical programming to convert the values into an appropriate priority matrix to use for call routing over a single staffing interval. Relatively large mathematical programs can be solved efficiently, because they they often have the structure of an assignment problem or a network flow problem. We took a conservative approach, guaranteeing that traditional constraints constraints are still met, provided that staffing has been determined appropriately. For a specific staffing algorithm, we drew upon Wallace and Whitt (2005). With this approach, we can apply existing SBR algorithms, without having to alter the ACD. To provide motivation for our mathematical-programming approach, in Section 4 we showed that a naive direct VBR algorithm can perform poorly.

In Section 6 we advocated a new strategy to empower contact-center agents via preference-based routing (PBR). We want agents to be able to participate more fully in contact-center decisions, and thereby experience more job satisfaction, which is intended to reduce agent turnover and absenteeism. Our intention is that PBR be used in conjunction with other management actions, such as providing better compensation and better career paths.

While we have not proven that PBR actually will improve agent job satisfaction, we have demonstrated that PBR can be implemented without sacrificing traditional contact center performance. It remains to explore the benefits of PBR. We are encouraged by preliminary laboratory experiments conducted by industrial psychologists Bamberger and Borin (2005), in which subjects act as agents responding to email service requests, indicating that PBR can indeed improve agent job satisfaction, but clearly more research and field testing are needed.

8. Acknowledgments

We thank our SeatLink colleague Mohammed Asif for his assistance and Peter Bamberger and Michal Borin of the Technion for their supporting work on preference-based routing and job satisfaction in contact centers, and for sharing their knowledge of the literature related to employee job satisfaction and performance.

References

[1] Anton, J., V. Bapat, B. Hall. 1999. *Call Center Performance Enhancement Using Simulation and Modelling*. Purdue University Press, Ashland, OH.

[2] Armony M, Gurvich I., A. Mandelbaum. 2006. Service level differentiation in call centeers with fully flexible servers. The Technion, Haifa, Israel.

[3] Bamberger, P. A., M. Biron. 2005. Earned empowerment: motivating employees on the basis of incremental increases in job-based control. Technion Working Paper, Technion, Haifa, Israel.

[4] Batt, R. 2002. Managing customer services: human resource practices, quit rates and sales growth. *Academy of Management Journal* 45 (3), 587–599.

[5] Batt, R., V. Doellgast, H. Kwon. 2005. A comparison of service management and employment systems in U. S. and Indian call centers. Working paper, Cornell University.

[6] Bertsimas, D., J. N. Tsitsiklis. 1997. *Introduction to Linear Optimization*, Athena Scientific, Nashua, NH.

[7] Bliss, W. G. 2004. Cost of employee turnover. *The Advisor*.
Available at: http://www.isquare.com/turnover.cfm

[8] Brigandi, A.,D. Dargon, M. Sheehan, T. Spencer, III. 1994. AT&T's call processing simulator (CAPS): operational design for inbound call centers. *Interfaces* 24 (1), 6–28.

[9] Chevalier, P., R. A. Shumsky, N. Tabordon. 2004. Routing and staffing in large call centers with specialized and fully flexible servers. Universite catholique de Louvain, University of Rochester and Belgacom Mobile/Proximus.

[10] Cleveland, B., S. Hash. 2004. (editors) *Call Center Agent Motivation and Compensation*, Call Center Press, ICMI, Annapolis, MD.

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

[12] Deery, S., N. Kinnie. 2004. *Call Centres and Human Resource Management*, Palgrave, Macmillan, New York, NY.

[13] Deming, W. E. 2000. *Out of the Crisis*, M.I.T. Press, paperback, Cambridge, MA.

[14] Gans, N., Koole, G., A. Mandelbaum. 2003. Telephone call centers: tutorial, review and research prospects. *Manufacturing and Service Opns. Mgmt.* 5 (2), 79–141.

[15] Garnett, O., A. Mandelbaum. 2000. An introduction to skills-based routing and its operational complexities. Technion, Israel. Available at: http://iew3.technion.ac.il/serveng

[16] Garnett, O., A. Mandelbaum, M. I. Reiman. 2002. Designing a call center with impatient customers. *Manufacturing and Service Opns. Mgmt.*, 4 (3), 208–227.

[17] Green, L. 1985. A queueing system with general-use and limited-use servers. *Operation Research* 33 (1), 168–182.

[18] Green, L. V., P. J. Kolesar, W. Whitt. 2005. Coping with time-varying demand when setting staffing requirements for a service ssytem. *Production and Operations Management*, forthcoming Available at http://columbia.edu/∼ww2040.

[19] Hackman, J. R., G. R. Oldham. 1976. Motivation through the design of work. *Organizational Behavior and Human Decision Processes* 16, 250–279.

[20] Hillier, F. S., G. J. Lieberman. 2003. *Introduction to Operations Research*, seventh edition, McGraw-Hill, New York, NY.

[21] Holman, D. 2002. Employee wellbeing in call centres. *Human Resource Management Journal* 12, 35–50.

[22] Holman, D. 2003. Call centres. Chapter 7 in *The New Workplace: A Guide to the Human Impact of Modern Work Practices*, D. J. Holman, T. D. Wall, C. W. Clegg, P. Sparrow and A. Howard (editors), Wiley.

[23] Holtgrewe, U., C. Kerst, K. Shire. 2002. *Re-organizing Service Work: Call Centres in Germany and Great Britain*, Aldershot, Burlington, VT.

[24] Jordan, W. C., S. C. Graves. 1995. Principles on the benefits of manufacturing process flexibility. *Management Science* 41 (4), 577–594.

[25] Jordan, W. C., R. R. Inman, D. E. Blumenfeld. 2003. Chained cross-training of workers for robust performance. General Motors Corporation.

[26] Juran, J. M., A. B. Godfrey. 1999. *Juran's Quality Handbook*, fifth edition, McGraw-Hill, New York, NY.

[27] Karasek, R. A., T. Theorell. 1990. *Healthy Work: Stress, Productivity and the Reconstruction of Working Life*, Basic Books, New York, NY.

[28] Leamon, P. 2004. Workforce management for skill-based routing: the need for integrated simulation. IEX White Paper, IEX Corporation, Richardson, Texas.

[29] Ross, S. M. 2003. *Introduction to Probability Models*, eighth ed., Academic Press, New York, NY.

[30] Segal, M. 1974. The operator scheduling problem: a network-flow approach. *Operations Research* 22 (4), 808–823.

[31] Sisselman, M., W. Whitt. 2005a. Preference-based routing. Working paper. Available at http://columbia.edu/∼ww2040.

[32] Sisselman, M., W. Whitt. 2005b. Empowering customer-contact-center agents via preference-based routing. SeatLink White Paper. Available at http://www.seatlink.net/

[33] Spraetz, B. 2004. Empowerment: a recipe for success; one part trust, one part technology. *InView*, IEX Corporation, Richardson, TX, November-December 2004.

[34] Stanford, D., W. K. Grassmann. 1993. The bilingual server systems: a queueing model featuring fully and partially qualified servers. *INFOR* 31, 261–277.

[35] Stanford D., W. K. Grassmann. 2000. Bilingual server call centers. *Analysis of Communication Networks: call centers, traffic and performance*, eds. D. McDonald and S. R. E. Turner (*Amer. Math. Soc.*, Providence), 31–47.

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

[37] Whitt, W. 2004. Efficiency-driven heavy-traffic approximations for many-server queues with abandonments. *Management Sci.* 50 (10), 1449–1461.

[38] Whitt, W. 2005. A preference-based-routing example solved by linear programming with Excel solver¡/b¿. Available at http://columbia.edu/∼ww2040.

[39] Whitt, W. 2006a Fluid models for multi-server queues with abandonments. *Operations Res.* 54 (1), 37–54.

[40] Whitt, W. 2006b. A multi-class fluid model for a contact center with skill-based routing. International Journal of Electronics and Communications (AEÜ) 60 (2), 95–102.

[41] Whitt, W. 2006c. The impact of increased employee retention upon performance in a customer contact center. *Manufacturing and Service Opns. Mgmt.* 8, forthcoming. Available at http://columbia.edu/∼ww2040.

[42] Witt, L. A., Andrews, M. C., D. S. Carlson. 2004. When conscientiousness isn't enough: emotional exhaustion and performance among call center customer service representatives. *Journal of Management* 30 (1), 149–160.

[43] Wolff, R. W. 1989. *Stochastic Models and the Theory of Queues*, Prentice-Hall, Englewood Cliffs, NJ.

[44] Wolsey, L. A. 1998. *Integer Programming*, Wiley, New York, NY.