Probability in the Engineering and Informational Sciences, **23**, 2009, 101–119. Printed in the U.S.A.
doi:10.1017/S0269964809000096

DYNAMIC ROUTING POLICIES FOR MULTISKILL CALL CENTERS

SANDJAI BHULAI
VU University Amsterdam
Faculty of Sciences
1081 HV Amsterdam,
The Netherlands
E-mail: sbhulai@few.vu.nl

We consider the problem of routing calls dynamically in a multiskill call center. Calls from different skill classes are offered to the call center according to a Poisson process. The agents in the center are grouped according to their heterogeneous skill sets that determine the classes of calls they can serve. Each agent group serves calls with independent exponentially distributed service times. We consider two scenarios. The first scenario deals with a call center with no buffers in the system, so that every arriving call either has to be routed immediately or has to be blocked and is lost. The objective in the system is to minimize the average number of blocked calls. The second scenario deals with call centers consisting of only agents that have one skill and fully cross-trained agents, where calls are pooled in common queues. The objective in this system is to minimize the average number of calls in the system. We obtain nearly optimal dynamic routing policies that are scalable with the problem instance and can be computed online. The algorithm is based on one-step policy improvement using the relative value functions of simpler queuing systems. Numerical experiments demonstrate the good performance of the routing policies. Finally, we discuss how the algorithm can be used to handle more general cases with the techniques described in this article.

1. INTRODUCTION

A call center is a collection of resources providing a telephony interface between a service provider and its customers. The resources consist, among others, of agents (i.e., people who talk to customers over the telephone) and information and communication technology (ICT) equipment. The complexity in the design and management of call centers stems from the fact that each incoming call requires an agent with

 0269-9648/09 $25.00

specific skills, such as knowledge of a particular language or knowledge of specific products and services. In order to establish efficient workforce management, one needs to address the problem of effective routing of incoming calls to the agents with the right skill.

One strategy to deal with the different skills is to establish a single pool of agents, each of whom is cross-trained in all of the skills. This system can then be viewed as a classical call center with a single skill. The personnel costs for the multiskilled agents may be very high, however, especially if the number of skills supported is large. A different approach is to staff a separate pool of agents for each skill. This case corresponds to several smaller, independent call centers operating in parallel. However, there is a cost for the loss of economies of scale. In between, one might partition the skills into separate subsets. This approach still excludes the possibility of hiring agents who do not have an entire subset, and it does not fully use agents who have skills beyond a subset.

The solution is to use the more flexible skill-based routing. In this scheme, each agent is acknowledged for his or her subset of skills. Arriving calls are then identified by the skill they require and are routed to a qualified agent. With the ICT equipment it is possible to obtain relevant data for making the routing decision, but so far only rudimentary algorithms are used. The state of research into solving skill-based routing problems is still in its infancy. A special case in which two classes of calls are served by a single pool of cross-trained agents is studied in Bhulai and Koole [3], Gans and Zhou [8], Perry and Nilsson [18], and by Gurvich, Armony, and Mandelbaum [9], who studied multiple classes of calls. Shumsky [21], Stanford and Grassmann [22], and Wallace and Whitt [24] considered fixed, static priority policies. A similar approach was adopted by Koole and Talim [14] and Franx, Koole and Pot [6], who provided an approximate analysis of the overflow behavior from one pool of agents to another. For a literature survey on asymptotic heavy-traffic regimes, we refer to Koole and Mandelbaum [12] and Gans, Koole and Mandelbaum [7].

Borst and Seri [4] studied skill-based routing in a dynamic setting. For a fixed set of agents, they proposed a dynamic scheme in which the number of calls of each class that actually has been served is compared to the number that, nominally, should have been served under a long-run average allocation scheme. The actual number of services that are farther behind are given a higher resulting priority. The results of Borst and Seri can be improved upon by using dynamic state-dependent routing policies (e.g., by taking the state of the agents and the state of the queues into account).

A standard approach for deriving effective state-dependent routing policies is via dynamic programming. This technique results in dynamic policies that depend on the current state of the call center (i.e., the number of calls requiring a particular skill currently in service and the number of calls in the queue). The identification of effective routing policies via dynamic programming is often impractical. For a call center with many types of skills, the dimensionality of the state space becomes very large, making the derivation of effective policies difficult, both analytically and numerically. Hence, standard algorithms, such as value iteration or policy iteration, for computing optimal policies break down.

A possible way of circumventing the computation with high-dimensional state spaces is a one-step policy improvement—a simple approximation method that is distilled from the policy iteration algorithm. In policy iteration, one starts with an arbitrary policy. For this policy, the expected average costs are determined and also the total difference in costs between starting in a particular state and some fixed reference state. The function describing the differences for all states is known as the relative value function. It can be obtained by solving a set of equations, called the Poisson equations, induced by the initial policy. Next, using these expressions, one can improve the policy by doing one policy improvement step. The procedure can now be repeated with the improved policy, generating a sequence of policies converging to the optimal policy.

Instead of repeatedly solving the Poisson equations for the relative value function (which suffers from the large dimensionality of the state space), a one-step policy improvement consists of executing the policy improvement step once only. In this case, the algorithm starts with an approximation of the relative function, which might be motivated by a heuristic or a reasonable policy. The resulting improved policy is then used as an approximation for the optimal policy. This method has proven to be close to optimal in a variety of routing and assignment problems (see, e.g., Bhulai and Koole [2], Koole and Nain [13], Ott and Krishnan [17], and Sassen, Tijms, and Nobel [20]).

In this article we study the problem of routing calls dynamically in a multiskill call center. Calls from different skill classes are offered to the call center according to a Poisson process. The agents in the center are grouped according to their heterogeneous skill sets that determine the classes of calls that they can serve. Each agent group serves calls with independent exponentially distributed service times.

We consider two scenarios. The first scenario deals with a call center with no buffers in the system, so that every arriving call either has to be routed immediately or has to be blocked and is lost. The objective in the system is to minimize the average number of blocked calls. The second scenario deals with call centers consisting of only agents that have one skill and fully cross-trained agents, where calls are pooled in common queues. The objective in this system is to minimize the average number of calls in the system. In Section 2 the problems of both scenarios are described in greater detail.

We present an algorithm to obtain nearly optimal dynamic routing policies that is scalable with the problem instance and can be executed online. The algorithm is based on a one-step policy improvement using the relative value functions of simpler queuing systems. This is explained in Section 3. In Section 4 we demonstrate by numerical experiments that the routing policies have a good performance. Finally, in Section 5 we conclude the article by discussing how the algorithm can be used to handle more general cases with the techniques described in this article.

2. PROBLEM FORMULATION

Consider a multiskill call center in which agents handle calls that require different skills. We represent the skill set $\mathcal{S}$ by $\mathcal{S} = \{1, \ldots, N\}$. We assume that calls that require

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

skill $s \in \mathcal{S}$ arrive at the call center according to a Poisson process with rate λ_s. Let $\mathcal{G} = \mathcal{P}(\mathcal{S})$ denote the groups with different skill sets defined by the power set of all the skills. Let $\mathcal{G}_s = \{G \in \mathcal{G} \mid s \in G\}$ denote all skill groups that include skill $s \in \mathcal{S}$. The agents are grouped according to their skills and can therefore be indexed by the elements of $\mathcal{G}$. Each group $G \in \mathcal{G}$ consists of S_G agents and serves a call that requires a skill within group G with independent exponentially distributed times with parameter μ_G.

2.1. Scenario 1: A Call Center With No Waiting Room

Suppose that we are dealing with a loss system; that is, there are no buffers at all in our model. Hence, there is no common queue for calls to wait, so every arriving call has either to be routed to one of the agent groups immediately or has to be blocked and is lost. In addition, when a call is routed to a group that has no idle servers left, the call is lost. The objective in this system is to minimize the average number of lost calls.

Let us formulate the problem as a Markov decision problem. Fix an order of the elements of $\mathcal{G}$, say $\mathcal{G} = (G_1, \ldots, G_{2^N-1})$, and define the state space of the Markov decision problem by $\mathcal{X} = \prod_{i=1}^{2^N-1} \{0, \ldots, S_{G_i}\}$. Then the $|\mathcal{G}|$-dimensional vector $\vec{x} \in \mathcal{X}$ denotes the state of the system according to the fixed order; that is, x_G is the number of calls in service at group $G \in \mathcal{G}$. In addition, represent by the $|\mathcal{G}|$-dimensional unit vector e_G the vector with 0 at every entry, except for a 1 at the entry corresponding to G according to the fixed order. When a call that requires skill $s \in \mathcal{S}$ arrives, the decision-maker can choose to block the call or to route the call to any agent group $G \in \mathcal{G}_s$ for which $x_G < S_G$. Hence, the action set is defined by $\mathcal{A}_s = \{G \in \mathcal{G}_s \mid x_G < S_G\} \cup \{b\}$, where action b stands for blocking the call. Therefore, for an arriving call that requires skill $s \in \mathcal{S}$, the transition rates $p(\vec{x}, a, \vec{y})$ of going from $\vec{x} \in \mathcal{X}$ to $\vec{y} \in \mathcal{X}$ after taking decision $a \in \mathcal{A}_s$ are given by $p(\vec{x}, b, \vec{x}) = \lambda_s$ and $p(\vec{x}, a, \vec{x} + e_a) = \lambda_s$ when $a \neq b$, and zero otherwise. The transition rates for the service completions are given by $p(\vec{x}, a, \vec{x} - e_G) = x_G \mu_G$ for $\vec{x} \in \mathcal{X}$, $G \in \mathcal{G}$, and any action a. The objective is modeled by the cost function $c(\vec{x}, a) = 1$ for any $\vec{x} \in \mathcal{X}$ and $a = b$.

The tuple $(\mathcal{X}, \{\mathcal{A}_s\}_{s \in \mathcal{S}}, p, c)$ defines the Markov decision problem. After uniformizing the system (see Puterman [19, Sect. 11.5]), we obtain the dynamic programming optimality equation for the system given by

$$g + \left[\sum_{s \in \mathcal{S}} \lambda_s + \sum_{G \in \mathcal{G}} S_G \mu_G \right] h(\vec{x}) = \sum_{s \in \mathcal{S}} \lambda_s \min\{1 + h(\vec{x}), h(\vec{x} + e_G) \mid G \in \mathcal{G}_s, x_G < S_G\}$$
$$+ \sum_{G \in \mathcal{G}} x_G \mu_G \, h(\vec{x} - e_G) + \sum_{G \in \mathcal{G}} (S_G - x_G) \mu_G \, h(\vec{x}), \qquad (1)$$

where g is the long-term expected average costs and h is the relative value function.

Note the unusual place of the minimization in (1): The actions in our case depend on the type of arriving call. It is easy to rewrite the optimality equation in

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

standard formulation (see, e.g., Koole [10]), but this would complicate the notation considerably.

A policy π is defined as the set of decision rules $(\pi_1, \ldots, \pi_N)$, where the mapping $\pi_s(\vec{x}) \in \mathcal{A}_s$ describes a feasible action to take in state $\vec{x} \in \mathcal{X}$ when a call that requires skill $s \in \mathcal{S}$ arrives. The long-term average optimal policy π is a solution of the optimality equation. It can also be obtained by the policy iteration algorithm. In this case, one starts with an arbitrary policy $\hat{\pi}$. This policy is evaluated by solving for the average costs $\hat{g}$ and the relative value function $\hat{h}$ in (1), with the minimization replaced by the actions described by the policy $\hat{\pi}$. Once the relative value function $\hat{h}$ has been derived, the policy in state $\vec{x}$ can be improved by computing $\arg\min\{1 + \hat{h}(\vec{x}), \hat{h}(\vec{x} + e_G) \mid G \in \mathcal{G}_s, x_G < S_G\}$.

The policy iteration algorithm converges to an optimal policy in a finite number of steps. The identification of optimal policies via this algorithm, however, is often impractical. The algorithm, in the policy improvement step, requires storage of the relative value function for the complete state space, which is of order $\mathcal{O}\left(\prod_{i=1}^{2^N-1}(S_{G_i}+1)\right)$. For a moderately sized call center, the dimensionality of the state space becomes very large, prohibiting the derivation of optimal routing policies numerically. The policy improvement step can be carried out, though, if the relative value function $\hat{h}$ is given, but solving the Poisson equations becomes very difficult after one or two iteration steps, so that iterating analytically is not an option either.

One way to circumvent the computational problems associated with policy iteration is a one-step policy improvement. In this case, one does not start with an arbitrary policy, but with an approximation of the relative value function $\hat{h}$. This approximation might be motivated by a heuristic or a policy that although perhaps is not optimal is not unreasonable either. In many cases, this is sufficient if the approximation has the right structure (i.e., linear, quadratic, etc.) (see, e.g., Bertsekas and Tsitsiklis [1]). Once the (approximate) relative value function has been obtained, the policy improvement step can be executed to obtain a better policy. Note that this step will result in a deterministic dynamic policy that is dependent on the state of the system, even if the starting relative value function was motivated by a randomized policy. Because the resulting policy will be too complicated to repeat the policy evaluation step, the algorithm stops here. In practice, for a suitably chosen approximation, the resulting policy is nearly optimal (see, e.g., Bhulai and Koole [2], Koole and Nain [13], Ott and Krishnan [17], and Sassen et al. [20]).

For the problem of skill-based routing, we will base the approximate relative value function on a static randomized policy for the initial routing policy. Let $\mathcal{G}^{(n)} = \{G \in \mathcal{G} \mid |G| = n\}$ be all agent groups that have exactly n skills for $n = 1, \ldots, N$. Define, accordingly, $\mathcal{G}_s^{(n)} = \{G \in \mathcal{G}^{(n)} \mid s \in G\}$ to be all agent groups that have n skills, including skill $s \in \mathcal{S}$. For a given skill s, this creates a hierarchical structure $\mathcal{G}_s^{(1)}, \ldots, \mathcal{G}_s^{(N)}$ from specialized agents having only one skill to cross-trained generalists having all skills. For a call center with three skills, we would have three levels in the hierarchy: the specialists (groups $\{1\}$, $\{2\}$, and $\{3\}$), the agents with two skills (groups $\{1, 2\}$,

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

{1, 3}, and {2, 3}), and the generalists (group {1, 2, 3}). In the following subsections we will describe the steps in the one-step policy improvement algorithm in more detail using this call center with three skills as an illustration.

2.1.1. Initial policy. The initial policy $\hat{\pi}$, on which we will base the approximate relative value function, tries to route a call requiring skill s through the hierarchy; that is, it tries $\mathcal{G}_s^{(1)}$ first and moves to $\mathcal{G}_s^{(2)}$ if there are no idle agents in $\mathcal{G}_s^{(1)}$. In $\mathcal{G}_s^{(2)}$ there might be more than one group to which one can route. The initial policy routes the call according to fixed probabilities to the groups in $\mathcal{G}_s^{(2)}$; that is, it splits the overflow stream from $\mathcal{G}_s^{(1)}$ into fixed fractions over the groups in $\mathcal{G}_s^{(2)}$. The call progresses through the hierarchy whenever it is routed to a group with no idle agents until it is served at one of the groups or it is blocked at $\mathcal{G}_s^{(N)}$ eventually. The rationale behind the policy that sends calls to the agents with the fewest number of skills first has been rigorously justified in Örmeci [16] and Chevalier, Tabordon, and Shumsky [5]. To illustrate this for three skills, the overflow of calls from group {1} are split by fixed fractions α and $\overline{\alpha} = 1 - \alpha$ in order to be routed to groups {1, 2} and {1, 3}, respectively. The overflow process from groups {2} and {3} are treated accordingly by the fractions β and γ, respectively.

We have yet to define how to choose the splitting probabilities in the initial routing policy. In order to define these, we ignore the fact that the overflow process is not a Poisson process and we consider all overflows to be independent. Thus, we procede as if the overflow process at group $G \in \mathcal{G}_s^{(i)}$ is a Poisson process with rate λ_G times the blocking probability. Together with the splitting probabilities, one can compute the rate of the Poisson arrivals at each station in $\mathcal{G}^{(i+1)}$ composed of the assumed independent Poisson processes. The splitting probabilities are then chosen such that the load at every group in $\mathcal{G}^{(i+1)}$ is balanced. To illustrate this for the call center with three skills, recall the Erlang loss formula $B(\lambda, \mu, S)$ for an $M/M/S/S$ queue with arrival rate λ and service rate μ:

$$B(\lambda, \mu, S) = \frac{(\lambda/\mu)^S/S!}{\sum_{i=0}^{S}(\lambda/\mu)^i/i!}. \tag{2}$$

The overflow rate at group $\{i\}$ for $i = 1, 2, 3$ is then given by $\lambda_i\, B(\lambda_i, \mu_{\{i\}}, S_{\{i\}})$. Hence, the arrival rates at the groups in $\mathcal{G}^{(2)}$ are given by

$$\begin{aligned}
\lambda_{\{1,2\}} &= \lambda_1\, B(\lambda_1, \mu_{\{1\}}, S_{\{1\}})\, \alpha + \lambda_2\, B(\lambda_2, \mu_{\{2\}}, S_{\{2\}})\, \beta,\\
\lambda_{\{1,3\}} &= \lambda_1\, B(\lambda_1, \mu_{\{1\}}, S_{\{1\}})\, (1-\alpha) + \lambda_3\, B(\lambda_3, \mu_{\{3\}}, S_{\{3\}})\, \gamma,\\
\lambda_{\{2,3\}} &= \lambda_2\, B(\lambda_2, \mu_{\{2\}}, S_{\{2\}})\, (1-\beta) + \lambda_3\, B(\lambda_3, \mu_{\{3\}}, S_{\{3\}})\, (1-\gamma).
\end{aligned}$$

The splitting probabilities can be determined by minimizing the average cost in the system. Because the average cost under this policy is the sum of the blocking probabilities of each queue in the system, the optimal splitting probability could create asymmetry in the system; that is, one queue has a very low blocking probability,

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

whereas another has a high blocking probability. Numerical experiments show that a situation with a more balanced blocking probability over all queues leads to better one-step improved policies. Therefore, we choose the splitting probabilities such that

$$\frac{\lambda_{\{1,2\}}}{S_{\{1,2\}}\mu_{\{1,2\}}} = \frac{\lambda_{\{1,3\}}}{S_{\{1,3\}}\mu_{\{1,3\}}} = \frac{\lambda_{\{2,3\}}}{S_{\{2,3\}}\mu_{\{2,3\}}}.$$

In case this equation does not have a feasible solution, we choose the splitting probabilities such that we minimize

$$\left|\frac{\lambda_{\{1,2\}}}{S_{\{1,2\}}\mu_{\{1,2\}}} - \frac{\lambda_{\{1,3\}}}{S_{\{1,3\}}\mu_{\{1,3\}}}\right| + \left|\frac{\lambda_{\{1,2\}}}{S_{\{1,2\}}\mu_{\{1,2\}}} - \frac{\lambda_{\{2,3\}}}{S_{\{2,3\}}\mu_{\{2,3\}}}\right| + \left|\frac{\lambda_{\{1,3\}}}{S_{\{1,3\}}\mu_{\{1,3\}}} - \frac{\lambda_{\{2,3\}}}{S_{\{2,3\}}\mu_{\{2,3\}}}\right|.$$

Finally, the arrival rate at the last group with all skills is given by

$$\lambda_{\{1,2,3\}} = \sum_{G \in \mathcal{G}^{(2)}} \lambda_G B(\lambda_G, \mu_G, S_G).$$

2.1.2. Policy evaluation. In the policy evaluation step, one is required to solve the long-run expected average costs and the relative value function from the Poisson equations for the policy $\hat{\pi}$. Note that for our purpose of deriving an improved policy, it suffices to have the relative value function only. Under the initial policy, this leads to solving the relative value function of a series of multiserver queues in tandem. This is a difficult problem that is yet unsolved even for a tandem series of single-server queues. Therefore, we choose to approximate the relative value function based on the assumptions made in defining the initial policy.

For the approximation, we assume that the overflow process is indeed a Poison process and that it is independent of all other overflow processes. We approximate the relative value function by the sum of the relative value functions for each agent group. More formally, let $h_G(x_G, \lambda_G, \mu_G, S_G)$ be the relative value function for an $M/M/S_G/S_G$ system with arrival rate λ_G and service rate μ_G, evaluated when there are x_G calls present. The value function for the whole system is then approximated by

$$\hat{h}(\vec{x}) = \sum_{G \in \mathcal{G}} h_G(x_G, \lambda_G, \mu_G, S_G). \tag{3}$$

2.1.3. Policy improvement. By substitution of the relative value function determined in the policy evaluation step into the optimality equations, one can derive

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

a better policy by evaluating

$$\min\{1 + \hat{h}(\vec{x}), \hat{h}(\vec{x} + e_G) \mid G \in \mathcal{G}_s, x_G < S_G\}$$
$$= \min\{1, h_G(x_G + 1, \lambda_G, \mu_G, S_G) - h_G(x_G, \lambda_G, \mu_G, S_G) \mid G \in \mathcal{G}_s, x_G < S_G\}.$$

The last term follows by subtracting $h(\vec{x})$ from all terms in the minimization and using the linear structure of $h(\vec{x})$.

2.2. Scenario 2: A Call Center With Specialists and Generalists Only

Suppose that we are dealing with a call center having agents with one skill (specialists) and fully cross-trained agents (generalists) only; that is, $\mathcal{G} = (G_1, \ldots, G_{|\mathcal{S}|}, G_{|\mathcal{S}|+1}) = (\{1\}, \ldots, \{N\}, \{1, \ldots, N\})$. In this scenario we assume that calls that require skill $s \in \mathcal{S}$ are pooled in a common infinite buffer queue, after which they are assigned to a specialist or a generalist in a first-come first-served order. The objective in this system is to minimize the average number of calls in the system, which in its turn is related to the average waiting time of a call.

In addition to the state of the agents groups, we also have to include the state of the queues in the state space. For this purpose, let the $|\mathcal{S}|$-dimensional vector $\vec{q}$ denote the state of the queues; that is, q_s is number of calls in queue s that require skill $s \in \mathcal{S}$. Moreover, the system also has to address the agent-selection problem and the call-selection problem. The first problem occurs when one has to assign an agent to an arriving call. The second problem occurs when an agent becomes idle and a potential call in the queue can be assigned. Following the same approach as in scenario 1, we obtain the dynamic programming optimality equation given by

$$g + \left[\sum_{s\in\mathcal{S}} \lambda_s + \sum_{G\in\mathcal{G}} S_G \mu_G\right] h(\vec{q}, \vec{x}) = \sum_{s\in\mathcal{S}} q_s + \sum_{G\in\mathcal{G}} x_G + \sum_{s\in\mathcal{S}} \lambda_s \tilde{h}(\vec{q} + e_s, \vec{x})$$
$$+ \sum_{G\in\mathcal{G}} x_G \mu_G \tilde{h}(\vec{q}, \vec{x} - e_G)$$
$$+ \sum_{G\in\mathcal{G}} (S_G - x_G)\mu_G h(\vec{q}, \vec{x}), \qquad \textbf{(4)}$$

where $\tilde{h}(\vec{q}, \vec{x}) = \min\{h(\vec{q} - e_s, \vec{x} + e_G) \mid s \in \mathcal{S}, G \in \mathcal{G}_s, q_s > 0, x_G < S_G\} \cup \{h(\vec{q}, \vec{x})\}$. The first two terms on the right-hand side represent the cost structure (i.e., the number of calls in the system). The third and fourth terms represent the agent-selection and the call-selection decisions, respectively.

2.2.1. Initial policy. In principle, we could take as initial policy $\hat{\pi}$ a policy similar to that used in scenario 1: A fraction α_s of type s calls are assigned to the corresponding specialists, and a fraction $1 - \alpha_s$ are assigned to the generalists. Instead of using the relative value function for the $M/M/S/S$ queue, we could use the relative

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

value function for the $M/M/S$ queue. However, this approximation would then assume a dedicated queue in front of the group of generalists. Consequently, a customer that is sent to the group of generalists that are all busy would still have to wait until a specialist of that call type is idle. Therefore, this Bernoulli policy does not efficiently use the resources in the system and leads to an inefficient one-step improved policy. To overcome the inefficiency of the Bernoulli policy, we instead use a policy that uses the specialists first and assigns a call to a generalist only if a generalist is available when all specialists who can handle that call type are busy. The effectiveness of this policy has been rigorously justified in Örmeci [16] and Chevalier et al. [5].

2.2.2. Policy evaluation. The relative value function for the policy $\hat{\pi}$, which uses specialists first and then generalists, is complicated. The policy $\hat{\pi}$ creates a dependence between the different agent groups that prohibits the derivation of a closed-form expression for the relative value function. Therefore, we approximate the relative value function by $\hat{h}$ as follows. Let $h_W(x, \lambda, \mu, S)$ be the relative value function of an $M/M/S$ queuing system. The approximation $\hat{h}$ for the policy $\hat{\pi}$ is then given by

$$\hat{h}(\vec{q}, \vec{x}) = \sum_{s \in \mathcal{S}} h_W(q_s + x_s, \tilde{\lambda}_s, \mu_{\{s\}}, S_{\{s\}}) + h_W\left((q_1 + x_1 - S_1)^+ + \cdots + (q_N + x_N - S_N)^+ + x_{N+1}, \lambda_{G_{N+1}}, \mu_{G_{N+1}}, S_{G_{N+1}}\right),$$

with $\tilde{\lambda}_s = \lambda_s\big(1 - B(\lambda_s, \mu_{\{s\}}, S_{\{s\}})\big)$ the approximate rate to the specialists of call type s and with $\lambda_{G_{N+1}} = \sum_{s \in \mathcal{S}} \lambda_s B(\lambda_s, \mu_{\{s\}}, S_{\{s\}})$ the approximate rate to the generalists. Note that this approximation follows the idea of the motivating initial policy in that it ensures that all idle specialists of type s, given by $S_s - x_s$, are used for all calls in the queue of the same type. This results in $(q_s - [S_s - x_s])^+ = \max\{q_s + x_s - S_s, 0\}$ calls that cannot be served. These calls are therefore waiting for a generalist to become idle. The approximation does take into account also the fact that a specialist can become idle before a generalist, and it immediately assigns a call to the specialist.

The idea behind the approximation is that it roughly estimates the different flows to the different agent groups and then computes the value function as if the calls are waiting simultaneously at the two queues where they can be served. Note that strictly hierarchical architectures in which agents groups are structured so that no overflow of calls has to be split between two possible subsequent agent pools can be dealt with similarly. Observe that it might be possible that the overflow to the generalists is larger than their service rate. However, the system will still be stable because the actual number of calls that will be served by the specialists will be higher, unless the system load is quite close to unity.

2.2.3. Policy improvement. In the policy improvement step, the initial policy $\hat{\pi}$ is improved by evaluating

$$\min\{\hat{h}(\vec{q} - e_s, \vec{x} + e_G) \mid s \in \mathcal{S}, G \in \mathcal{G}_s, q_s > 0, x_G < S_G\} \cup \{\hat{h}(\vec{q}, \vec{x})\}.$$

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

Note that in this case the policy has to be determined for both the agent-assignment and the call-selection decisions.

3. RELATIVE VALUE FUNCTIONS

In this section we will study relative value functions for multiserver queuing systems with Poisson arrivals and identical independent servers with exponentially distributed service times. We will derive a closed-form expression for the long-run expected average costs and the relative value function by solving the Poisson equations (in particular, we will derive expressions for h_G and h_W, which were discussed in the previous section). Typically, the Poisson equations of these models give rise to linear difference equations. Due to the nature of the Poisson equations, the difference equations have a lot of structure when the state description is one dimensional. Therefore, it is worthwhile to study difference equations prior to the analysis of the multiserver queue. We will restrict our attention to second-order difference equations, as this is general enough to model a birth–death queuing process that has the multiserver queue as a special case.

Let $f(x)$ be an arbitrary function defined on $\mathbb{N}_0$. Define the backward difference operator Δ by

$$\Delta f(x) = f(x) - f(x-1)$$

for $x \in \mathbb{N}$. Note that the value of $f(x)$ can be expressed as

$$f(x) = f(k-1) + \sum_{i=k}^{x} \Delta f(i) \tag{5}$$

for every $k \in \mathbb{N}$ such that $k \leq x$. This observation is the key to solving first-order difference equations and to solving second-order difference equations when one solution to the homogeneous equation is known. We first state the result for first-order difference equations. The result can be found in Mickens [15, Chap. 2], but is stated less precisely there.

LEMMA 3.1: *Let $f(x)$ be a function defined on $\mathbb{N}$ satisfying the relation*

$$f(x+1) - \gamma(x) f(x) = r(x), \tag{6}$$

with γ and r arbitrary functions defined on $\mathbb{N}$ such that $\gamma(x) \neq 0$ for all $x \in \mathbb{N}$. With the convention that an empty product equals 1, $f(x)$ is given by

$$f(x) = f(1)\, Q(x) + Q(x) \sum_{i=1}^{x-1} \frac{r(i)}{Q(i+1)} \quad \text{with } Q(x) = \prod_{i=1}^{x-1} \gamma(i).$$

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

PROOF: Let the function $Q(x)$ be as stated in the theorem. By assumption we have that $Q(x) \neq 0$ for all $x \in \mathbb{N}$. Dividing (6) by $Q(x+1)$ yields

$$\Delta\left[\frac{f(x+1)}{Q(x+1)}\right] = \frac{f(x+1)}{Q(x+1)} - \frac{f(x)}{Q(x)} = \frac{r(x)}{Q(x+1)}.$$

From (5) with $k = 2$, it follows that

$$f(x) = Q(x)\frac{f(1)}{Q(1)} + Q(x)\sum_{i=2}^{x}\frac{r(i-1)}{Q(i)} = f(1)\,Q(x) + Q(x)\sum_{i=1}^{x-1}\frac{r(i)}{Q(i+1)}.$$

■

Note that the condition $\gamma(x) \neq 0$ for all $x \in \mathbb{N}$ is not very restrictive in practice. If there is a state $y \in \mathbb{N}$ for which $\gamma(y) = 0$, then the analysis can be reduced to two other first-order difference equations, namely the part for states $x < y$ and the part for states $x > y$ for which $\gamma(y) = 0$ is the boundary condition.

The solution to first-order difference equations plays an important part in solving second-order difference equations when one solution to the homogeneous equation is known. In that case, the second-order difference equation can be reduced to a first-order difference equation expressed in the homogeneous solution. Application of Lemma 3.1 then gives the solution to the second-order difference equation. The following theorem summarizes this result.

THEOREM 3.2: *Let $f(x)$ be a function defined on $\mathbb{N}_0$ satisfying the relation*

$$f(x+1) + \alpha(x)f(x) + \beta(x)f(x-1) = q(x), \tag{7}$$

with α, β, and q arbitrary functions defined on $\mathbb{N}$ such that $\beta(x) \neq 0$ for all $x \in \mathbb{N}$. Suppose that one homogeneous solution is known, say $f_1^h(x)$, such that $f_1^h(x) \neq 0$ for all $x \in \mathbb{N}_0$. Then, with the convention that an empty product equals 1, $f(x)$ is given by

$$\frac{f(x)}{f_1^h(x)} = \frac{f(0)}{f_1^h(0)} + \left[\Delta\left[\frac{f(1)}{f_1^h(1)}\right]\right]\sum_{i=1}^{x} Q(i) + \sum_{i=1}^{x} Q(i)\sum_{j=1}^{i-1}\frac{q(j)}{f_1^h(j+1)\,Q(j+1)},$$

where $Q(x) = \prod_{i=1}^{x-1}\beta(i)f_1^h(i-1)/f_1^h(i+1)$.

PROOF: Note that f_1^h always exists, since a second-order difference equation has exactly two linearly independent homogeneous solutions (see Mickens

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

[15, Thm. 3.11]). The known homogeneous solution $f_1^h(x)$ satisfies

$$f_1^h(x+1) + \alpha(x) f_1^h(x) + \beta(x) f_1^h(x-1) = 0. \tag{8}$$

Set $f(x) = f_1^h(x)\, u(x)$ for an arbitrary function u defined on $\mathbb{N}_0$. Substitution into (7) yields

$$f_1^h(x+1)\, u(x+1) + \alpha(x) f_1^h(x)\, u(x) + \beta(x) f_1^h(x-1)\, u(x-1) = q(x).$$

By subtracting (8) $u(x)$ times from this expression and rearranging the terms, we derive

$$\Delta u(x+1) - \gamma(x) \Delta u(x) = r(x),$$

with

$$\gamma(x) = \frac{f_1^h(x-1)}{f_1^h(x+1)}\, \beta(x) \quad \text{and} \quad r(x) = \frac{q(x)}{f_1^h(x+1)}.$$

From Lemma 3.1 it follows that

$$\Delta u(x) = [\Delta u(1)]\, Q(x) + Q(x) \sum_{j=1}^{x-1} \frac{r(j)}{Q(j+1)} \quad \text{with} \quad Q(x) = \prod_{i=1}^{x-1} \gamma(i).$$

From (5) it finally follows that

$$u(x) = u(0) + [\Delta u(1)] \sum_{i=1}^{x} Q(i) + \sum_{i=1}^{x} Q(i) \sum_{j=1}^{i-1} \frac{r(j)}{Q(j+1)}.$$

Since $f(x) = f_1^h(x)\, u(x)$, it follows that $u(x) = f(x)/f_1^h(x)$ for $x = 1, 2$. ■

The result of Theorem 3.2 is very useful for solving relative value functions from Poisson equations expressed as (7) for queuing systems with a one-dimensional state description. Note that the general form of the Poisson equations is given by $g \cdot e + f = c + Pf$ (in vector form) after uniformization and rescaling the uniformization factor to 1. Here, c are the direct costs, P is the transition probability matrix, $g \cdot e$ is the vector with long-run expected average costs, and f is the relative value function, where $f(x)$ describes the asymptotic difference in total costs that results from starting the process in state x instead of some fixed reference state y. Without loss of generality, we can take the reference state to be $y = 0$, so that for the relative value function, we have $f(0) = 0$. Moreover, note that $f_1^h(x) = 1$ is always a solution to the homogeneous Poisson equations. Hence, we have the following corollary.

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

COROLLARY 3.3: *Let $f(x)$ represent the relative value function in the Poisson equations of a queuing system defined on $\mathbb{N}_0$ satisfying the relation*

$$f(x+1) + \alpha(x)f(x) + \beta(x)f(x-1) = q(x). \tag{9}$$

The relative value function is then given by

$$f(x) = \sum_{i=1}^{x} Q(i)\left[f(1) + \sum_{j=1}^{i-1} \frac{q(j)}{Q(j+1)}\right] \quad \textit{where} \quad Q(x) = \prod_{i=1}^{x-1} \beta(i). \tag{10}$$

We are now ready to study the relative value function of the $M/M/s/s$ and the $M/M/s$ queue.

3.1. The Blocking Costs in an *M/M/s/s* Queue

Consider a queuing system with s identical independent servers and no buffers for customers to wait. The arrivals are determined by a Poisson process with parameter λ. The service times are exponentially distributed with parameter μ. An arriving customer that finds no idle server is blocked and generates a cost of one monetary unit. Let state x denote the number of customers in the system. The Poisson equations for this $M/M/s/s$ queue are then given by

$$\begin{aligned} g + \lambda\, h(0) &= \lambda\, h(1), \\ g + (\lambda + x\mu)\, h(x) &= \lambda\, h(x+1) + x\mu\, h(x-1), \qquad x = 1, \ldots, s-1, \\ g + s\mu\, h(s) &= \lambda + s\mu\, h(s-1). \end{aligned}$$

From the first equation we can deduce that $h(1) = g/\lambda$. The second equation can be written as (9) with $\alpha(x) = -(\lambda + x\mu)/\lambda$, $\beta(x) = x\mu/\lambda$, and $q(x) = g/\lambda$. From (10) we have that

$$\begin{aligned} h(x) &= \frac{g}{\lambda} \sum_{i=1}^{x} \sum_{k=0}^{i-1} \frac{(i-1)!}{(i-k-1)!} \left(\frac{\lambda}{\mu}\right)^{-k} \\ &= \frac{(\lambda/\mu)^s/s!}{\sum_{i=0}^{s} (\lambda/\mu)^i/i!} \sum_{i=1}^{x} \sum_{k=0}^{i-1} \frac{(i-1)!}{(i-k-1)!} \left(\frac{\lambda}{\mu}\right)^{-k}. \end{aligned} \tag{11}$$

The last term follows from the last equation, from which we derive that

$$g = \frac{(\lambda/\mu)^s/s!}{\sum_{i=0}^{s} (\lambda/\mu)^i/i!}\, \lambda.$$

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

3.2. The Average Number of Customers in an *M/M/s* Queue

Consider the previous queuing system with s identical independent servers. Let state x again denote the number of customers in the system. Suppose that a buffer of infinite size is added to the system so that blocking does not occur anymore. Instead, we focus on the average number of customers in the system represented by generating a cost of x monetary units when x customers are present in the system. The Poisson equations for this $M/M/s$ queue are then given by

$$\begin{aligned} g + \lambda\, h(0) &= \lambda\, h(1), \\ g + (\lambda + x\mu)\, h(x) &= x + \lambda\, h(x+1) + x\mu\, h(x-1), \qquad x = 1, \ldots, s-1, \\ g + (\lambda + s\mu)\, h(x) &= x + \lambda\, h(x+1) + s\mu\, h(x-1), \qquad x = s, s+1, \ldots \end{aligned}$$

In order to obtain the relative value function h, we repeat the argument as in the case of the $M/M/s/s$ queue. In this case, we use Theorem 3.2 instead of Corollary 3.3 to match the boundary conditions, and we then shift from x to $x - s$. Let $\rho = \lambda/(s\mu)$; then for $x = 0, \ldots, s$, the relative value function is given by

$$h(x) = \frac{g}{\lambda} \sum_{i=1}^{x} \sum_{k=0}^{i-1} \frac{(i-1)!}{(i-k-1)!} \left(\frac{\lambda}{\mu}\right)^{-k} - \frac{1}{\lambda} \sum_{i=1}^{x} (i-1) \sum_{k=0}^{i-2} \frac{(i-2)!}{(i-k-2)!} \left(\frac{\lambda}{\mu}\right)^{-k}.$$

For $x = s, s+1, \ldots$, we have

$$\begin{aligned} h(x) = h(s) - \frac{(x-s)\rho}{1-\rho}\frac{g}{\lambda} + \left[\frac{(x-s)(x-s+1)\rho}{2(1-\rho)} + \frac{(x-s)(\rho + s(1-\rho))\rho}{(1-\rho)^2}\right]\frac{1}{\lambda} \\ + \frac{(1/\rho)^{x-s} - 1}{1-\rho}\left[\frac{\rho}{1-\rho}\frac{g}{\lambda} + h(s) - h(s-1) - \frac{(\rho + s(1-\rho))\rho}{\lambda(1-\rho)^2}\right]. \end{aligned}$$

The long-term expected average costs, which represents the average number of customers in the system in this case, is given by

$$g = \frac{(s\rho)^s \rho}{s!(1-\rho)^2}\left[\sum_{n=0}^{s-1} \frac{(s\rho)^n}{n!} + \frac{(s\rho)^s}{s!(1-\rho)}\right]^{-1} + s\rho.$$

4. NUMERICAL EXPERIMENTS

In this section we illustrate the one-step policy improvement method by studying the routing problem for the call center under scenario 1 and scenario 2 described in Section 2. The examples are chosen small enough so that the optimal policies can be computed and compared to the one-step improved policies. We first start with a call center with three skills under scenario 1.

The algorithm is started with the overflow policy described in Section 2 as the initial policy. We denote the long-run expected average costs (divided by $\lambda_1 + \lambda_2 + \lambda_3$) for this policy by g. The average costs g are computed using (1) with the decisions

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

chosen according to the initial policy. The reason for dividing by $\lambda_1 + \lambda_2 + \lambda_3$ is to get the blocking probability instead of the average number of blocked calls. In this way we can compare our results with the results stated in Tabordon [23], who used the same policy in her work. The approximation to the relative value function of this policy is given by (3), as described in Section 2, where h_G is given by (11). The relative value function is used to get an improved policy. The blocking probability obtained by this policy is denoted by g'. Note that g' is also computed using (1), where the decisions have been chosen according to the improved policy. After this, we also compute the optimal blocking probability g^* under the optimal policy and the relative error $\Delta = 100(g' - g^*)/g^*$.

In Table 1 we find the results of the one-step policy improvement algorithm for scenario 1. The parameters for λ_X, μ_X, and S_X in Table 1 are organized for groups $\{1\}$, $\{2\}$, and $\{3\}$. The parameters for μ_{XY} and S_{XY} are ordered for groups $\{1, 2\}$, $\{1, 3\}$, and $\{2, 3\}$, respectively. The greatest proportional extra cost $(g' - g^*)/g^*$ over all experiments in Table 1 is 12.9% (the last experiment). Other experiments show that the proportional extra costs lies within the range of 0% to 13%. Thus, we can see that the improved policy has a good performance already after one step of policy improvement.

For scenario 2 we consider a call center with three skills also. However, the call center only consists of agents with one skill (specialists) or all skills (generalists) only. Calls that require the same skill are pooled into a common queue. The initial policy adopted in this system is to use the specialists first and to route to generalists when there is a generalist idle and all specialists of the required call type are busy. Table 2 lists the results under this scenario with the same notation as under scenario 1.

The first line of Table 2 seems to suggest that no significant gains over the static policy can be obtained when the service rates of the specialists and the generalists are equal to each other. In Tables A1–A3 in the Appendix, we have varied the service rate of the generalists under different loads. The results show that as the system is offered higher loads, the gains by using the one-step improved policy are also higher compared to the static routing policy. Next, we scale the system such that the increase in the offered load was compensated by faster service rates or by an increase in the number of servers. Table A4 shows that when the service rates are scaled, the gains

TABLE 1. Numerical Results for Scenario 1 with $\mu_{\{1,2,3\}} = 1$ and $S_{\{1,2,3\}} = 2$

λ_X	μ_X	μ_{XY}	S_X	S_{XY}	g	g'	g^*	Δ
6 6 6	1.0 1.0 1.0	1.0 1.0 1.0	2 2 2	2 2 2	0.361	0.349	0.344	1.505
6 5 4	2.0 1.5 1.0	1.5 1.0 1.0	2 2 2	2 2 2	0.170	0.147	0.143	2.781
7 6 5	2.0 1.5 1.0	1.5 1.5 1.0	3 3 3	2 2 2	0.119	0.099	0.096	3.264
7 6 5	1.5 1.0 1.0	1.5 1.0 1.0	3 3 3	3 3 2	0.130	0.107	0.103	3.930
6 5 4	2.0 1.5 1.0	1.5 1.0 1.0	3 3 3	2 2 2	0.072	0.057	0.054	5.672
6 5 4	2.0 1.5 1.0	1.5 1.5 1.0	3 3 3	2 2 2	0.061	0.046	0.042	8.011
10 4 4	1.5 1.0 1.0	1.5 1.5 1.0	2 2 2	2 2 2	0.281	0.274	0.252	8.816
10 6 3	1.5 1.0 1.0	1.5 1.0 1.0	3 3 3	3 3 2	0.160	0.148	0.131	12.851

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

Table 2. Numerical Results for Scenario 2

λ_X			μ_X			S_X			$\mu_{\{1,2,3\}}$	$S_{\{1,2,3\}}$	g	g'	g^*	Δ
6	6	6	2.0	2.0	2.0	3	3	3	2.0	3	11.043	10.899	10.746	1.429
6	3	3	1.5	1.0	1.0	3	3	3	1.0	3	20.122	19.259	18.361	4.892
6	5	4	2.5	2.0	1.5	2	2	2	2.0	3	13.186	11.562	11.314	2.187
6	5	4	1.8	1.8	1.2	3	3	3	1.2	3	16.348	14.931	14.602	2.253
7	6	5	2.5	2.0	1.5	3	3	3	2.0	2	15.269	13.310	13.127	1.397
7	6	5	1.8	1.8	1.2	3	3	3	1.8	3	23.260	20.091	19.125	5.054
10	6	3	3.0	2.0	1.0	3	3	3	1.0	2	31.834	26.844	25.184	6.594
6	5	4	3.0	2.0	1.0	3	3	3	1.0	2	20.083	14.403	13.795	4.404

over the static policy remain roughly unaffected. However, when more servers are added, it slightly decreases. From the other lines in Table 2 we can conclude that the gain over the static policy is greater in asymmetric systems. In conclusion, we can observe that the improved policy has a good performance and that its performance is close to the performance of the optimal policy.

Note that our proposed method is scalable and can easily be used for larger call centers. In this section, however, we restricted ourselves to a call center with three skills, as the computation of optimal policies becomes numerically difficult for larger call centers. The optimal policy grows exponentially in the number of skills, whereas a single step of policy iteration has linear complexity.

5. DISCUSSION AND FURTHER RESEARCH

We have described nearly optimal dynamic routing policies for two call center settings. The settings differ in terms of whether blocking or queuing occurs in the call center. The dynamic policies are based on the one-step policy improvement algorithm using the relative value function of the $M/M/s/s$ queue (in case of blocking) and the $M/M/s$ queue (in case of queueing). The policies can be easily obtained even when the problem instance grows large and can be computed online as well. Numerical experiments suggest that the routing policies have a good performance. However, these results cannot be compared with the optimal policies for large systems due to computational difficulties.

In the problem, we have assumed that the service rates vary with the agent group rather than the call type. A further extension of the model could take the variation in the call content, rather than the agent type, into account. This can be done by deriving the relative value function for a multiclass multiserver queuing system. The current model, however, is still useful as an approximation to a call center in which the service rates are close to each other or to handle the special case when all service rates are equal to each other.

In scenario 2 we have chosen the average number of calls in the system as performance measure. Alternatively, one could study tail probabilities of the waiting time.

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press

The relative value function for this performance measure cannot be obtained from Theorem 3.2 and requires a different approach. Note that both performance measures might lead to poor service for some customer classes if discriminating against those classes helps the overall performance measure. Koole [11] suggested some new performance measures that do not suffer from this.

A different avenue for further research could be to include abandonments to the call center in case of queuing. In principle, Theorem 3.2 is general enough to compute the value function of a multiserver queue with abandonments. However, it requires insight into the call center to choose an initial policy such that the relative value function can be approximated sufficiently well. This extension would also enable one to study mixed call center architectures (i.e., queues with finite and infinite buffers, call types with finite and infinite patience, etc.). It would be interesting to see how our model can be extended along these dimensions.

References

1. Bertsekas, D. & Tsitsiklis, J. (1996). *Neuro-dynamic programming*. Bellmont, MA: Athena Scientific.
2. Bhulai, S. & Koole, G. (2003). On the structure of value functions for threshold policies in queuing models. *Journal of Applied Probability* 40: 613–622.
3. Bhulai, S. & Koole, G. (2003). A queueing model for call blending in call centers. *IEEE Transactions on Automatic Control* 48: 1434–1438.
4. Borst, S. & Seri, P. (2000). Robust algorithms for sharing agents with multiple skills. Working paper, Bell Laboratories, Murray Hill, NJ.
5. Chevalier, P., Tabordon, N., & Shumsky, R. (2004). Routing and staffing in large call centers with specialized and fully flexible servers. Louvain-la-Neuve, Belgium: Université Catholique de Louvain.
6. Franx, G., Koole, G., & Pot, S. (2006). Approximating multi-skill blocking systems by hyperexponential decomposition. *Performance Evaluation* 63: 799–824.
7. Gans, N., Koole, G., & Mandelbaum, A. (2003). Telephone call centers: tutorial, review, and research prospects. *Manufacturing and Service Operations Management* 5: 79–141.
8. Gans, N. & Zhou, Y. (2003). A call-routing problem with service-level constraints. *Operations Research* 51: 255–271.
9. Gurvich, I., Armony, M., & Mandelbaum, A. (2008). Service level differentation in call centers with fully flexible servers. *Management Science* 54: 279–294.
10. Koole, G. (1995). *Stochastic scheduling and dynamic programming*. CWI Tract 113. Amsterdam: CWI.
11. Koole, G. (2003). Redefining the service level in call centers. Technical report, Vrije Universiteit, Amsterdam.
12. Koole, G. & Mandelbaum, A. (2002). Queueing models of call centers: an introduction. *Annals of Operations Research* 113: 41–59.
13. Koole, G. & Nain, P. (2000). On the value function of a priority queue with an application to a controlled polling model. *Queueing Systems* 34: 199–214.
14. Koole, G. & Talim, J. (2000). Exponential approximation of multi-skill call centers architecture. In *Proceedings of QNETs 2000*, pp. 23/1–23/10.
15. Mickens, R. (1990). *Difference equations: Theory and applications*. New York: Chapman & Hall.
16. Örmeci, E. (2004). Dynamic admission control in a call center with one shared and two dedicated service facilities. *IEEE Transactions on Automatic Control* 49: 1157–1161.
17. Ott, T. & Krishnan, K. (1992). Separable routing: A scheme for state-dependent routing of circuit switched telephone traffic. *Annals of Operations Research* 35: 43–68.
18. Perry, M. & Nilsson, A. (1992). Performance modeling of automatic call distributors: Assignable grade of service staffing. In *XIV International Switching Symposium*, pp. 294–298.

19. Puterman, M. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. New York: Wiley.
20. Sassen, S., Tijms, H., & Nobel, R. (1997). A heuristic rule for routing customers to parallel servers. *Statistica Neerlandica* 51: 107–121.
21. Shumsky, R. (2004). Approximation and analysis of a queueing system with flexible and specialized servers. *OR Spektrum* 26: 307–330.
22. Stanford, D. & Grassmann, W. (2000). Bilingual server call centres. In D. McDonald & S. Turner (eds.), *Call centres, traffic and performance*, Providence, RI: American Mathematical Society. Fields Institute Communications, Vol. 28, pp. 31–48.
23. Tabordon, N. (2002). Modeling and optimizing the management of operator training in a call center. Ph.D thesis, Université Catholique de Louvain.
24. Wallace, R. & Whitt, W. (2005). A staffing algorithm for call centers with skill-based routing. *Manufacturing and Service Operations Management* pp. 276–294.

APPENDIX

TABLE A1. Scenario 2 – $\lambda_X = 5$, $\mu_X = 2$, $S_X = 3$, and $S_{\{1,2,3\}} = 3$

$\mu_{\{1,2,3\}}$	g	g'	g^*	Δ
1.5	9.012	8.933	8.928	0.051
1.6	8.773	8.707	8.703	0.045
1.7	8.559	8.504	8.500	0.047
1.8	8.366	8.320	8.316	0.045
1.9	8.192	8.154	8.150	0.044
2.0	8.035	8.006	7.999	0.091
2.1	7.892	7.864	7.851	0.166
2.2	7.762	7.713	7.675	0.496
2.3	7.644	7.561	7.489	0.962
2.4	7.535	7.405	7.302	1.418
2.5	7.436	7.238	7.118	1.690

TABLE A2. Scenario 2 – $\lambda_X = 6$, $\mu_X = 2$, $S_X = 3$, and $S_{\{1,2,3\}} = 3$

$\mu_{\{1,2,3\}}$	g	g'	g^*	Δ
1.5	13.674	13.142	12.877	2.055
1.6	12.995	12.595	12.344	2.035
1.7	12.405	12.106	11.872	1.969
1.8	11.891	11.665	11.453	1.848
1.9	11.440	11.265	11.080	1.671
2.0	11.043	10.899	10.746	1.429
2.1	10.692	10.449	10.445	0.046
2.2	10.379	10.179	10.160	0.189
2.3	10.101	9.934	9.881	0.532
2.4	9.851	9.712	9.610	1.062
2.5	9.626	9.494	9.347	1.568

TABLE A3. Scenario 2 – $\lambda_X = 7$, $\mu_X = 2$, $S_X = 3$, and $S_{\{1,2,3\}} = 3$

$\mu_{\{1,2,3\}}$	g	g'	g^*	Δ
1.5	27.101	25.142	23.387	7.502
1.6	25.138	22.841	21.638	5.559
1.7	23.306	20.923	20.099	4.097
1.8	21.623	19.326	18.755	3.042
1.9	20.099	17.992	17.586	2.304
2.0	18.737	16.867	16.571	1.784
2.1	17.533	15.910	15.690	1.401
2.2	16.475	15.086	14.920	1.114
2.3	15.551	14.370	14.240	0.915
2.4	14.746	13.741	13.631	0.802
2.5	14.044	13.183	13.082	0.771

TABLE A4. Scenario 2: Scaling of the System

λ_X			μ_X			S_X			$\mu_{\{1,2,3\}}$	$S_{\{1,2,3\}}$	g	g'	g^*	Δ
6	5	4	2.5	2.0	1.5	2	2	2	2.0	3	13.186	11.562	11.314	2.187
12	10	8	5.0	4.0	3.0	2	2	2	4.0	3	13.187	11.564	11.313	2.215
24	20	16	10.0	8.0	6.0	2	2	2	8.0	3	13.190	11.567	11.310	2.273
12	10	8	2.5	2.0	1.5	4	4	4	2.0	6	18.625	17.789	17.562	1.290
24	20	16	5.0	4.0	3.0	4	4	4	4.0	6	18.628	17.792	17.559	1.326
24	20	16	2.5	2.0	1.5	8	8	8	2.0	12	31.925	31.380	30.991	1.253
30	20	10	2.5	2.0	1.5	15	15	15	2.0	15	28.924	27.996	27.256	2.714
30	20	20	2.5	2.0	1.5	20	20	20	2.0	20	35.290	34.829	32.015	8.788

https://doi.org/10.1017/S0269964809000096 Published online by Cambridge University Press