Probability in the Engineering and Informational Sciences, **29**, 2015, 461–471.
doi:10.1017/S0269964815000091

OPTIMIZATION OF OVERFLOW POLICIES IN CALL CENTERS

G.M. KOOLE
Department of Mathematics, VU University Amsterdam
De Boelelaan 1081, 1081 HV Amsterdam the Netherlands
E-mail: ger.koole@vu.nl

B.F. NIELSEN AND T.B. NIELSEN
Department of Informatics and Mathematical Modelling
Technical University of Denmark
Richard Petersens Plads, 2800 Kgs. Lyngby Denmark
E-mail: bfni@dtu.dk

We examine how overflow policies in a multi-skill call center should be designed to accommodate performance measures that depend on waiting time percentiles such as service level. This is done using a discrete Markovian approximation of the waiting time of the first customers waiting in line. A Markov decision chain is used to determine the optimal policy. This policy outperforms considerably the ones used most often in practice, which use a fixed threshold. The present method can be used also for other call-center models and other situations where performance is based on actual waiting times and customers are treated in a FCFS order.

1. INTRODUCTION

In call centers a commonly used practice is to allow overflow of calls between different agent pools. The purpose of this can be to reduce the risk of having calls waiting an excessive amount of time in one queue while servers assigned to another queue may be vacant. There can obviously be restrictions on the overflow between groups, for example, due to one group of agents not having the competence to answer all types of calls or simply based on a wish of giving higher priority to one customer class.

The call routing structure of real call centers can often be very complicated with multiple customer classes and agent/server groups. In order to analyze these setups an often used practice is to look at different canonical structures, such as the "V", "N", "X", and "W" designs [3]. The system analyzed here is the one referred to as the N-design. Figure 1 shows an illustration of this system. Two customer classes, a and b, are each served by their own server group. Furthermore a-calls can be routed to b-servers according to decisions based on the number of free servers and/or the waiting times of the first customers in line in the two queues. b-calls can never go to a-servers. Also, a-calls cannot preempt b-calls already being served. It should be noted that a-calls are not necessarily of higher priority than b-calls, the fundamental difference lies only in the routing scheme. For example, if a-calls are of high

© Cambridge University Press 2015 0269-9648/15 $25.00

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

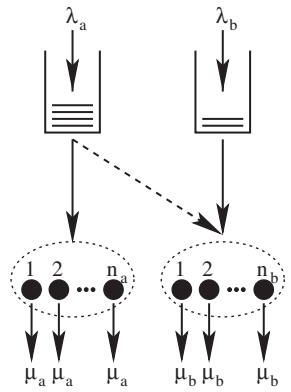


FIGURE 1. The N-design. a-customers can be sent to b-servers according to decisions based on the number of free servers and/or the waiting times of the first customers in line in the two queues.

priority, the overflow to b-servers could be used to reduce the risk that important customers wait too long in the queue. In the case that b-calls are of higher priority, the overflow from the a-queue could allow any excess capacity in the b-server group to be utilized. The prioritization of calls will be based on cost functions discussed later.

An often used policy is to allow overflow after the waiting time of the first customer in line has reached a given fixed threshold, thus not taking into account if servers are vacant in the server group the overflow is directed to. This policy is examined in [1,2,8]. The idea behind keeping b-servers vacant even though a-customers are waiting, is to have b-servers ready to answer incoming b-calls. However, the approach of letting the threshold where overflow is allowed to be independent of the number of free servers, is intuitively unattractive as it might very well be better to allow overflow earlier if many b-servers are vacant and later if few b-servers are vacant. In this paper we examine how overflow policies can be designed to accommodate different performance measures in the best way and thus avoid a waste of resources.

Different implementations of routing and call selection based on the waiting time of customers have been examined in the literature, especially for the V-design where two customer classes share a server group. One simple approach is to assign an ageing factor that grows proportionally with a customer's waiting time and choose the customer with the highest proportional waiting time. This approach was introduced in [7] and later also examined in [9]. A similar approach is to add a constant to the waiting time for each customer class and thus prioritize calls based on the values of the constant. This is often referred to as scheduling according to due dates or dynamic prioritization and was first examined in [5,6].

In [4], call selection is optimized and average costs are given for a single server and two customer classes with switching costs and preemptive-resume service discipline using one-step optimization. Non-preemptive systems with multiple customer classes are normally hard to analyze using Markov Decision Processes as knowledge of the customers currently being served is required. This requirement is relaxed here by modelling the waiting times of the first customers in line. As a result, the optimal policies will depend on the waiting time of the longest waiting call. This makes sense both from a theoretical and a practical point of view, especially for objectives that are functions of the waiting time such as the percentage of calls that have waited shorter than a specific threshold (in call centers referred to as

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

service level (SL)). One can imagine, and it is indeed shown in this paper, that a policy that uses actual waiting time information performs much better for this type of objective.

The model presented in this paper is also in other aspects more versatile than those examined in the literature. The cost functions can be chosen freely, and we consider the N-design instead of the more commonly examined V-design. The N-design can be seen as a more complicated version of call selection in a V-design system. When a b-server becomes vacant it has to decide whether to take an a- or b-customer into service, but with the complication of taking the number of working a-servers into account.

In Section 2, the model is introduced and the optimization process is described. Results for different cases are given in Section 3 and compared to the fixed threshold policy. Lastly in Section 4 we conclude on our work and give topics for further research.

2. MODEL DESCRIPTION

We consider the system illustrated in Figure 1. Two customer classes, a and b, have their own queue and server group with n_a and n_b servers, respectively. Overflow can be allowed from the a-queue to b-servers. Calls are taken from each of the two queues according to the First In, First Out (FIFO) principle. Arrivals to each of the queues happen according to Poisson processes with rates λ_a and λ_b. Exponential service times are assumed and individual servers in group a and b complete jobs with rates μ_a and μ_b, respectively. Service rates are thus associated with the servers, not the customers.

The underlying model we use is similar to the one used in [8], but with different overflow criteria. The system is modeled using a two-dimensional continuous-time Markov chain where the two dimensions represent a- and b-calls, respectively. For a-calls we let states $i \leq 0$ represent the number of vacant a-servers, that is, $i = -n_a$ corresponds to all a-servers being vacant and $i = 0$ corresponds to all being occupied. We use the second dimension to model b-calls according to the same principle.

When an arrival occurs and all servers are busy, the customer enters the queue and we start modelling the waiting time of the first customer in line. In this way, arrivals are not explicitly handled by the model for states $i > 0$ as arrivals do not affect the waiting time of the first in line when customers are queued. States $i > 0$ thus represent the waiting time of the first customer in line, w^{FIL}, in a discrete manner. We let transitions with rate γ between state i and $i + 1$ represent elapsing time. Service completions are represented by a geometric distributed set of transitions from state i to states $\{0, 1, ..., i\}$ with total rate $n_a\mu_a$. The weights of the transitions are given in Lemma 2.1, where t_- and t_+ denote the instants just before and after a transition. Again b-calls are modelled in the same way.

LEMMA 2.1.

$$\mathbb{P}_a(I_{t_+} = i_{t_+} | I_{t_-} = i_{t_-}) = \begin{cases} 1 - \sum_{h=0}^{i_{t_-}-1} \left(\frac{\lambda_a}{\lambda_a+\gamma}\right)\left(\frac{\gamma}{\lambda_a+\gamma}\right)^h, & \text{for } i_{t_+} = 0; \\ \left(\frac{\lambda_a}{\lambda_a+\gamma}\right)\left(\frac{\gamma}{\lambda_a+\gamma}\right)^{i_{t_-}-i_{t_+}}, & \text{for } i_{t_+} \in \{1, .., i_{t_-}\}. \end{cases}$$

See [8] for proof of Lemma 2.1. We introduce the following notations: $p_{a,ij} = \mathbb{P}_a(I_{t_+} = j | I_{t_-} = i)$ and $p_{b,ij} = \mathbb{P}_b(J_{t_+} = j | J_{t_-} = i)$.

An approach for approximating the waiting time distribution for a system with a fixed overflow policy, is described in [8]. This approach can also be applied here. It involves the steady-state distribution, $\boldsymbol{\pi}$, which is found by setting up a generator matrix, $\mathbf{G}$, according

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

to the system description above, and solving the equation system $\boldsymbol{\pi}\mathbf{G} = \mathbf{0}$. Next, the state probabilities at the instants before service initiations, $\alpha(i,j)$, are found by considering the embedded Markov chain at service initiations. From this, the waiting time distribution for a-customers is found as

$$P(W_a \le t) = \sum_{i=-n_b}^{D_b} \sum_{j=-n_a}^{-1} \alpha_a(i,j) + \sum_{i=1}^{D_a} \left[F_\Gamma(t;i,\gamma) \sum_{j=-n_b}^{D_b} \alpha_a(i,j) \right],$$

where $F_\Gamma(t;i,\gamma) = 1 - \sum_{h=0}^{i-1} \frac{(\gamma t)^h}{h!} e^{-\gamma t}$ is the cdf of an Erlang-distribution with shape parameter $i \in \mathbb{N}$ and scale parameter $\gamma \in \mathbb{R}_+$. The waiting time distribution for b-customers is found in a similar way.

2.1. Overflow Optimization

By introducing decisions in the individual states, we can choose when to allow overflow from the a-queue to b-servers and thus optimize the system according to some chosen cost functions. For this purpose Markov decision processes are used. We let V_n be the total expected cost n steps from the horizon and use backwards recursion to determine the optimal policy [10,11]. We base the costs on the waiting time experienced by customers, that is, when a customer is taken into service after having waited a given time, this event is assigned a cost. As the waiting times of customers are represented by states with positive indices we introduce cost functions of the form $c_a(i)$ and $c_b(j)$. We also introduce the possibility of imposing a cost for allowing overflow, that is, whenever an a-customer is sent to a b-server. This is denoted with c_s.

In the case where a-customers are queued and b-servers are free, that is, states where $i > 0,\ j < 0$, we have the possibility of sending an a-customer to a b-server. We introduce decisions whenever there has been a transition to one of these states. These decisions are represented by the W's given in Eq. (1). The first part of the minimization term corresponds to doing nothing and staying in the state in which the system just arrived. The second part corresponds to letting one of the vacant b-servers take an a-customer into service, which then happens immediately after the decision has been taken.

$$W_n(i,j) = \begin{cases} \min\left\{ V_{n-1}(i,j),\ c_s + c_a(i) + \sum_{h=0}^{i} p_{a,ih} V_{n-1}(h,j+1) \right\} & \text{for } i > 0,\ j < 0, \\ V_{n-1}(i,j) & \text{else.} \end{cases} \tag{1}$$

A slightly different approach to decisions is taken for the case where customers are queued in both queues, that is, states $i > 0, j > 0$. Here all servers are working and decisions are only allowed whenever a b-server finishes a job, that is, when a μ_b-transition occurs. We introduce decisions represented by $U(i,j)$ when a b-server finishes a job in state (i,j). The option here is whether to take an a- or b-customer into service, as represented by U in (2).

$$U_n(i,j) = \min\left\{ c_b(j) + \sum_{h=0}^{j} p_{b,jh} V_{n-1}(i,h),\ c_s + c_a(i) + \sum_{h=0}^{i} p_{a,ih} V_{n-1}(h,j) \right\}. \tag{2}$$

For $i \le 0$, we have no decisions to make as no a-calls are queued. For states $i > 0,\ j = 0$ there is also no decision as no b-servers are free and if one becomes free a transition is made to a state $j = -1$, which the next decision will be based on.

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

Having defined the decisions, we can now set up the equations for the backward recursion. Five different equations are needed due to the structure of the state space. These are given in (3)–(7). In order to have the same average time between transitions from each individual state we introduce uniformization terms. These are given in the last line in each of the equations giving a uniformization rate of $\gamma + \lambda_a + \lambda_b + n_a\mu_a + n_b\mu_b$.

$$\begin{aligned} V_n(i,j) &= \lambda_a W_n(i+1,j) + \lambda_b V_{n-1}(i,j+1) \\ &\quad + \mu_a(n_a+i)V_{n-1}(i-1,j) + \mu_b(n_b+j)V_{n-1}(i,j-1) \\ &\quad + (-i\mu_a - j\mu_b + \gamma)V_{n-1}(i,j), \qquad \text{for } i \le 0, j \le 0, \end{aligned} \tag{3}$$

$$\begin{aligned} V_n(i,j) &= \gamma V_{n-1}(i,j+1) + \lambda_a V_{n-1}(i+1,j) \\ &\quad + \mu_a(n_a+i)V_{n-1}(i-1,j) + \mu_b n_b \left(c_b(j) + \sum_{h=0}^{j} p_{b,jh} V_{n-1}(i,h) \right) \\ &\quad + (\lambda_b - i\mu_a)V_{n-1}(i,j), \qquad \text{for } i \le 0, j > 0, \end{aligned} \tag{4}$$

$$\begin{aligned} V_n(i,j) &= \gamma W_n(i+1,j) + \lambda_b W_n(i,j+1) \\ &\quad + n_a\mu_a \left(c_a(i) + \sum_{h=0}^{i} p_{a,ih} W_n(h,j) \right) + (n_b+j)\mu_b W_n(i,j-1) \\ &\quad + (\lambda_a - j\mu_b)V_{n-1}(i,j), \qquad \text{for } i > 0, j < 0, \end{aligned} \tag{5}$$

$$\begin{aligned} V_n(i,j) &= \gamma V_{n-1}(i+1,j) + \lambda_b V_{n-1}(i,j+1) \\ &\quad + n_a\mu_a \left(c_a(i) + \sum_{h=0}^{i} p_{a,ih} V_{n-1}(h,j) \right) + n_b\mu_b W_n(i,j-1) \\ &\quad + \lambda_a V_{n-1}(i,j), \qquad \text{for } i > 0, j = 0, \end{aligned} \tag{6}$$

$$\begin{aligned} V_n(i,j) &= \gamma V_{n-1}(i+1,j+1) \\ &\quad + n_a\mu_a \left(c_a(i) + \sum_{h=0}^{i} p_{a,ih} V_{n-1}(h,j) \right) + n_b\mu_b U_n(i,j) \\ &\quad + (\lambda_a + \lambda_b)V_{n-1}(i,j), \qquad \text{for } i > 0, j > 0. \end{aligned} \tag{7}$$

We let the value iteration algorithm run until $V_n - V_{n-1}$ converges as follows, where ϵ is some small number, for example, 10^{-9}:

$$\max_{(i,j)} \{V_n(i,j) - V_{n-1}(i,j)\} - \min_{(i,j)} \{V_n(i,j) - V_{n-1}(i,j)\} < \epsilon.$$

When the average optimal value function has thus been determined, we can easily find the optimal actions, one for each of the states $i > 0,\ j < 0$ and $i > 0,\ j > 0$. If required we can determine the stationary waiting time distribution the way described earlier in this section. Performance measures such as the SL can readily be extracted from the waiting time distributions.

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

2.2. Performance Measures

The prevalent way of measuring performance in call centers is to use SL or average waiting time (called *average speed of answer*, ASA). The cost functions for a and b-calls can be designed to emulate either, or both of these. Furthermore, it may be appropriate to penalize overflow as it might be desired that a-calls are assigned to a-servers and b-calls to b-servers. The actual form and value of the cost functions $c_a(i)$ and $c_b(j)$ together with the overflow/switching cost c_s will always be a managerial decision based on often subjective priorities.

One might choose $c_a(i)$ and $c_b(j)$ as step functions being 0 below a certain threshold and some non-zero value above this threshold to represent a given SL such as 80/20 (percentage/SL target), where we introduce SLT_a and SLT_b as the SL targets for a- and b-customers, respectively. Another possibility is to let the cost functions increase linearly, which would then correspond to having a performance measure based on ASA. A combination of the two performance measures is of course also possible. A call-center manager might choose to value a-calls twice as much as b-calls and thus let the cost function for a-calls increase twice as fast. However, a-calls need not necessarily be the high priority customers as discussed earlier.

The optimization of the routing policy is done under a given set of parameters, thus our target is not to dimension the system to fulfill a given SL but rather doing as good as possible under given conditions. That is, serving as many as possible within the time limit in the SL of the two customer classes with the possibility of having more weight on serving one class. Or, in the case where the cost functions are shaped to emulate an emphasis on ASA, trying to keep this as low as possible.

The way to relate states in the model to actual waiting times is to use the mean time spent in a state before a γ-transition happens, that is, $1/\gamma$. For example, if $\gamma = 30$ min^{-1} then being in state $i > 0$ will correspond to the first customer in line having waited approximately $2i$ s.

3. RESULTS

In this section, we illustrate the optimal policy for a number of different cases. The policies are illustrated by plots of the state space with colors representing the optimal decision in each state. In all cases $a = 1$ (blue) indicates that no action should be taken and $a = 2$ (green) indicates that a call should be taken from the a-queue and assigned to a b-server. The white areas indicate that no decision is possible. The policy of allowing overflow after a given fixed time would in this case correspond to the state space being divided in a blue section to the left and green section to the right where the border would be a vertical line situated at the threshold value. No units are given on the parameters or results as these can be chosen as desired. However, the values have been chosen such that they can realistically be assumed to be in minutes (or min^{-1} for the intensities) in a call-center context.

For all results, a larger than shown state space has been used in order to avoid the influence of the state-space truncation. Typically a state space of size 120×120 has been used in the calculations to produce valid results for a state space of size 70×70. The results have been verified by setting $c_s = \infty$, which leads to overflow never being allowed. This corresponds to having two independent $M/M/n$ queues and the resulting waiting time distribution from the model can thus be compared to analytical results. The model has also been compared to the simulations used for verification of the fixed threshold case in [8], and again the results agree.

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

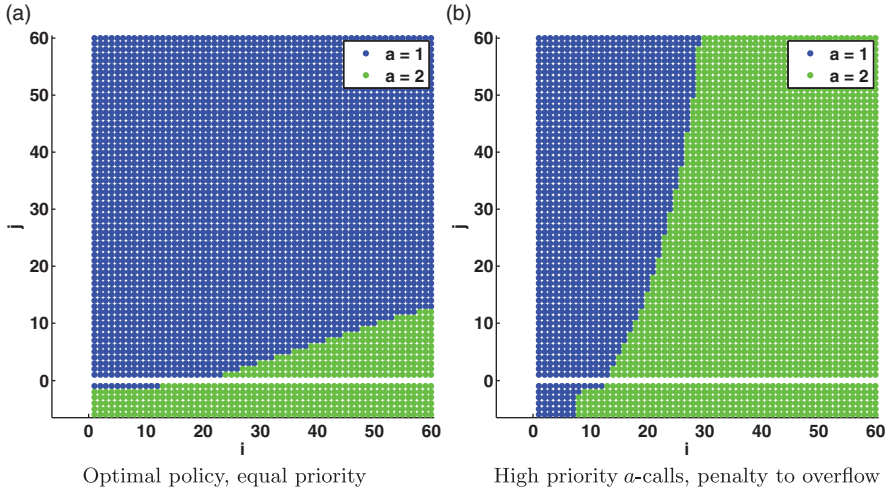


FIGURE 2. Decisions based on optimization of ASA. Green shows where overflow is allowed. Unique parameters for Figure 2a are: $c_a(i) = i$, $c_s = 0$, and for Figure 2b $c_a(i) = 3i$, $c_s = 120$. Shared parameters for the two figures are $c_b(j) = j$, $\lambda_a = 5$; $\lambda_b = 4$; $\mu_a = 1$; $\mu_b = 1$; $n_a = 6$; $n_b = 6$; $\gamma = 30$; $\epsilon = 10^{-9}$. (a) Optimal policy, equal priority. (b) High priority a-calls, penalty to overflow.

Figure 2a shows an optimal policy when the cost functions are based on minimizing ASA and both types of calls are assigned the same weight or value. It is almost always optimal to route a-calls to b-servers when some of these are available, which makes sense as this minimizes the time servers are free while calls are waiting. When calls are waiting in both queues it is in general optimal to take the call that has waited the longest with a small bias toward b-customers. This is due to that fact that it is not desirable to end up in a states $i < 0$ where some servers will not be working.

Figure 2b shows what happens when a penalty is given when calls are allowed to overflow and the mean waiting time of a-calls are given triple weight compared to the mean for b-customers. It is seen that the decision to allow overflow is only taken when the waiting time of a-customers has reached a certain value. In this case, the optimal policy actually bears some resemblance to the often used policy of allowing overflow after a fixed amount of time, with the difference that the border separating the areas with different decisions is not totally vertical.

The optimal policy when using SL as performance indicator can take a rather complicated form as seen in Figure 3a. If lines are overlaid at SLT_a, SLT_b, $i = 0$, and $j = 0$, as is done with red dotted lines in Figure 3a, the picture becomes more clear. Within each of these seven areas it can actually be seen that the decisions do indeed follow a monotone pattern.

The cost functions for Figure 3 have been chosen such that the b-customers are the ones requiring shorter waiting times (below 10 s) and a-customers are of lower priority and can wait up to 20 s before a penalty is given. In this case, it can clearly be seen that when a-customers have not waited for a long time, they should only be sent to b-servers if many of these are vacant. If customers are waiting in both queues, overflow should only be allowed if the first in line in the a-queue is just about to reach its SL threshold. In the case when both types of customers have exceeded their threshold, we have a diagonal division of the state space, roughly corresponding to taking the customer with the shortest waiting time of the

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

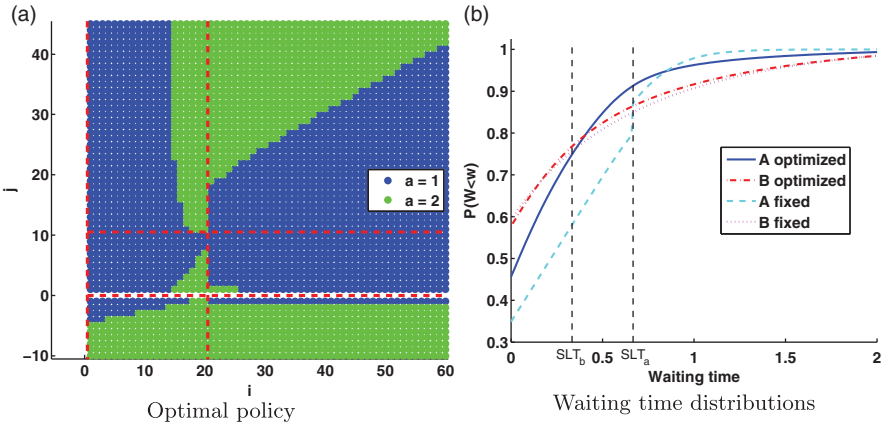


FIGURE 3. Optimal policy and waiting distributions when using SL as performance indicator. The used parameters are: $c_a(i) = 0$ for $i \leq 20$, $c_a(i) = 1$ for $i > 20$, $c_b(j) = 0$ for $j \leq 10$, $c_b(j) = 1$ for $j > 10$, $c_s = 0, \lambda_a = 5$; $\lambda_b = 7$; $\mu_a = 1$; $\mu_b = 1; n_a = 5$; $n_b = 10$; $\gamma = 30$; $\epsilon = 10^{-9}$. The vertical lines in panel b show the SL thresholds for a- and b-customers. (a) Optimal policy, (b) Waiting time distributions.

TABLE 1. Numerical results; values from Figure 3.

Policy	SL_a %	SL_b %	ASA_a	ASA_b
Fixed	87.25	75.60	0.3071	0.2274
Optimized	91.29	76.79	0.2074	0.2152

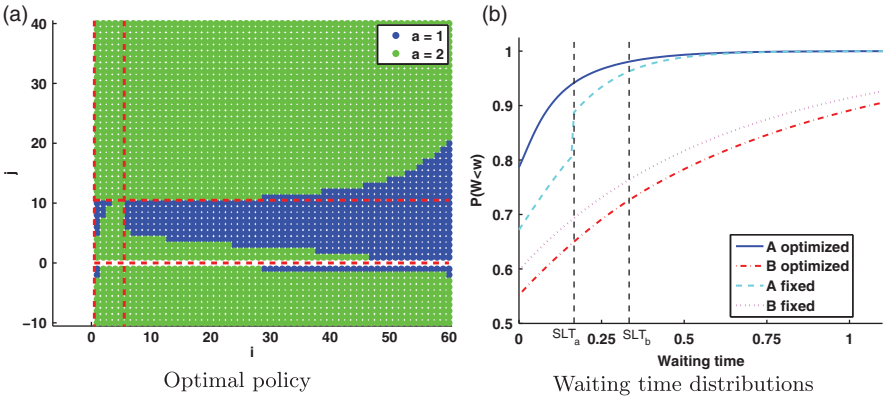


FIGURE 4. Optimal policy and waiting time distributions when using SL as performance indicator and a-customers are given high priority. The used parameters are: $c_a(i) = 0$ for $i \leq 5$, $c_a(i) = 2$ for $i > 5$, $c_b(j) = 0$ for $j \leq 10$, $c_b(j) = 1$ for $j > 10$, $c_s = 0$, $\lambda_a = 5$; $\lambda_b = 7$; $\mu_a = 1$; $\mu_b = 1$; $n_a = 6$; $n_b = 10$; $\gamma = 30$; $\epsilon = 10^{-9}$. The vertical lines in panel b show the SL thresholds for a- and b-customers. (a) Optimal policy. (b) Waiting time distributions.

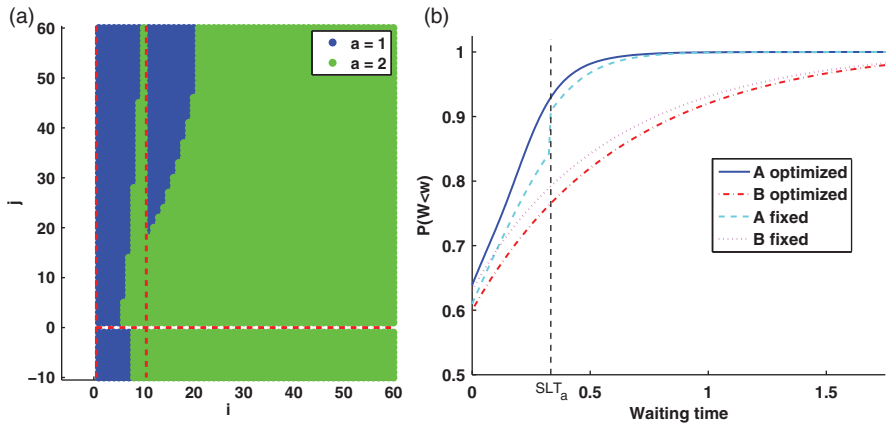
https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

TABLE 2. Numerical results; values from Figure 4.

Policy	SL_a (%)	SL_b (%)	ASA_a	ASA_b
Fixed	88.76	76.36	0.0603	0.2151
Optimized	94.18	72.63	0.0297	0.2504

TABLE 3. Numerical results; values from Figure 5.

Policy	SL_a (%)	SL_b (%)	ASA_a	ASA_b
Fixed	90.90	88.02	0.1033	0.1919
Optimized	92.97	86.23	0.0809	0.2157



Optimal policy Waiting time distributions

FIGURE 5. A combination of ASA and SL as performance indicator is used here. The used parameters are: $c_a(i) = 3i$ for $i \leq 10$, $c_a(i) = 3i + 40$ for $i > 10$, $c_b(j) = j$; $c_s = 120$, $\lambda_a = 5$; $\lambda_b = 7; \mu_a = 1$; $\mu_b = 1$; $n_a = 6$; $n_b = 10$; $\gamma = 30$; $\epsilon = 10^{-9}$. The vertical line in Figure 3b shows the SL threshold for a-customers. (a) Optimal policy. (b) Waiting time distributions.

two situated first in line into service. In this way, the probability that the next customer from that queue will be served within its threshold is maximized.

Figure 3b shows that the number of customers served within their threshold is increased for both a and b customers. The tail of the waiting time distribution for the optimized case of a-customers is longer than that of the fixed threshold case, but this does not matter when the focus is entirely on improving SL. Also less b-customers are served immediately using the optimized policy, but again SL is improved. Table 1 shows the actual numbers. ASA is also improved for both classes even though it is not taken into account during the optimization in this case.

Figure 4 shows a case where a-customers are given high priority: they are valued twice as much as b-customers. Furthermore, the SL targets have been set to 10 and 20 s for a- and b-customers, respectively. The results show that it is almost always advantageous to allow overflow when a-customers are given high priority. From the structure of the optimal policy shown in Figure 4a it appears that we should base the overflow decision on the waiting time

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

of the b-customers instead of on the a-customers. A simple threshold policy based on the waiting time of the first b-customer in line would resemble the picture seen in Figure 4a to a high degree. The optimal policy results in a significant improvement for a-customers at a minor cost to b-customers as seen in Figure 4b. The numbers are given in Table 2.

A realistic cost function would be a combination of SL and ASA as both are often used as performance measures in call centers. Figure 5 shows a case where the cost functions are based on a combination of the two with a penalty for overflow. Its numbers are given in Table 3. Again the optimal policy has some resemblance to the fixed threshold, especially if the blue area for $i > 10$, $j > 0$ is disregarded. This indicates that the use of a fixed threshold policy corresponds to having high-priority a-customers and a penalty for overflow. We see that if a fixed threshold is used, it should be set to a lower value than SLT_a.

4. CONCLUSION

In this paper, we have presented a method for optimizing the overflow in queueing systems based on a call-center scenario. We have shown that it is possible to obtain a significant performance improvement by using more complicated overflow policies compared to the simple fixed threshold.

The practice of not taking a-customers into service no matter how many free b-servers there are, is not recommendable. If a-customers are of high priority, it is almost always optimal to allow overflow to vacant b-servers, especially if no penalty is given to the overflow. However, in the case where b-customers are of higher priority it may indeed be better to reserve some servers for potential b-arrivals even if a-customers are waiting as illustrated in Figure 3. A general rule for when overflow should be allowed based on the number of free b-servers will most likely depend on many parameters.

The policy of using a fixed threshold for determining when to allow overflow has been shown to be a good approach when a penalty is associated with overflow and a-customers are of high priority. This result gives an idea of what the consequence is when this policy is used in call centers.

In all cases it appears that a fixed threshold should not be set to the same value as the SL threshold, but to a lower value. This makes sense, as some time is needed for b-servers to have a chance to become free before the threshold is reached, when calls are queued in both queues.

A natural and fairly straightforward extension of the model would be to allow overflow from the b-queue to a-servers and optimize both overflows simultaneously according to cost functions. This would constitute an X-design, which is one of the other canonical designs referred to in the Introduction.

A final subject for further research is including fairness between servers in the objective function, that is, trying to balance the occupation level of the servers. This is highly relevant for practical purposes, but it is hard to include in the dynamic programming formulation.

It is also important to note that the current approach can be used to many other queueing models, not necessarily within the area of call centers. Crucial to the analysis is that the queues are operated in a FIFO order and that the arrivals occur according to Poisson processes.

References

1. Barth, W., Manitz, M. & Stolletz, R. (2010). Analysis of two-level support systems with time-dependent overflow – a banking application. *Production and Operations Management* 19(6): 757–768.

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

2. Bekker, R., Koole, G., Nielsen, B.F. & Nielsen, T.B. (2011). Queues with waiting time dependent service. *Queueing Systems*, 68(1): 61–78.
3. Gans, N., Koole, G. & Mandelbaum, A. (2003). Telephone call centers: Tutorial, review, and research prospects. *Manufacturing and Service Operations Management* 5(2): 79–141.
4. Groenevelt, R., Koole, G. & Nain, P. (2002). On the bias vector of a two-class preemptive priority queue. *Mathematical Methods of Operations Research* 55(1): 107–120.
5. Jackson, J.R. (1960). Some problems in queueing with dynamic priorities. *Naval Research Logistics Quarterly* 7: 235–249.
6. Jackson, J.R. (1961). Queues with dynamic priority discipline. *Management Science* 8(1): 18–34.
7. Kleinrock, L. & Finkelstein, R.P. (1967). Time dependent priority queues. *Operations Research* 15(1): 104–116.
8. Koole, G., Nielsen, B.F. & Nielsen, T.B. (2012). First in line waiting times as a tool for analyzing queueing systems. *Operations Research* 60(5): 1258–1266.
9. Perry, M. & Nilsson, A. (1992). Performance modelling of automatic call distributors: assignable grade of service staffing. *International Switching Symposium 1992. "Diversification and Integration of Networks and Switching Technologies Towards the 21st Century" Proceedings*, Vol. 2, pp. 294–298.
10. Puterman, M.L. (2005). *Markov Decision processes. Discrete stochastic dynamic programming*. 2nd ed., New York: Wiley.
11. Tijms, H.C. (2003). *A first course in Stochastic models*. 2nd ed., New York: Wiley.

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press

https://doi.org/10.1017/S0269964815000091 Published online by Cambridge University Press