On the Scheduling of Operations in a Chat Contact Center

Benjamin Legros[a] • Oualid Jouini (Corresponding author)[b]

[a] *Ecole de Management de Normandie, Laboratoire Métis, 64 Rue du Ranelagh, 75016 Paris, France*
[b] *Laboratoire Genie Industriel, CentraleSupélec, Université Paris-Saclay, 9 rue Joliot Curie, 91190 Gif-sur-Yvette, France*

benjamin.legros@centraliens.net • oualid.jouini@centralesupelec.fr

European Journal of Operational Research. To appear, 2018.

Abstract

Unlike calls, the chat channel allows a contact center agent to simultaneously work with many customers. The benefit of this flexibility can be however challenged by a new abandonment feature during service. In this context, new flow routing questions arise. How many chats should an agent serve? Which agent should be selected? Or may a chat be served by more than one agent? We aim to answer these questions so as to find the best trade-off between time spent in service, queueing delay and abandonment. For this purpose, we determine conditions under which the traditional Least Busy First or Most Busy First policies are optimal for agent selection. Using a dynamic programming analysis, we prove that a state-dependent threshold reservation policy is optimal when a chat can be handed to other agents without loss of efficiency. This analysis reveals that reservation is useful as it allows to maintain of sufficient agent productivity and abandonment from the queue which in turn reduce the system's congestion. In addition, we show the closeness between the optimal policy and the easier-to-implement fixed capacity policy if this fixed capacity is optimized. When a chat cannot be handed to other agents, we develop a dispatching policy improvement algorithm based on the explicit computation of the relative value function for an initial policy with fixed probabilities. The dispatching policy significantly outperforms non state-dependent ones, as it partly compensates the agents' inability to share a chat service.

Keywords: OR in service industries; contact centers; queueing systems; Markov decision process; chat messages; routing; multitasking; abandonment.

1 Introduction

Context and motivation. New advances in telecommunication technology are revolutionizing the way call centers interact with customers. Customer preferences are also evolving rapidly toward the use of new technology. In this context, there has been, in recent years, a surge of interest in new software products that make it possible for contact centers to offer assistance to online users via the *chat channel*. Chat systems allow customers to access an instant messaging system built into the call center website to interact with agents online.

From an operational point of view, chats offer more interactivity compared to emails and this leads to more efficient demand resolution and less waste of time on already solved problems. They also allow for less working time per customer than call conversations because of the multitasking possibility. Chats thus somehow combine the benefits of call and email channels. Other advantages of chat systems are the features such as screen sharing and the ability to share files and data, which are particularly useful to computer companies, software companies, and e-retailers (Cui and Tezcan, 2016). Despite its prevalence in practice (40% of contact centers use chats as reported in ICMI (2013)), very few related papers have been published. Due to their unique features, we believe that chat service systems are and will be an important way to interact with customers. Organizations that run contact centers will take better care of this alternative communication channel to improve efficiency and reduce costs relative to the cost of servicing phone calls.

In this paper, we consider a chat contact center and address the problem of routing chats to agents. The optimization problem consists of a trade-off between waiting times and abandonments. Interactions between agents and customers in a chat contact center can be modeled as a queueing system. The characteristics of this queueing model are that agents can handle more than one chat at once, customers can abandon the system not only during the wait but also during the service due to what is perceived as a long wait during the discussion, and a customer can be served by more than one agent successively because the conversation is written and agents are anonymous. This possibility is nevertheless not implemented in all chat contact centers.

The idea of handing a chat to another agent during its initial service is appealing. Inefficient situations where an agent is too busy compared to another one can thus be avoided. Yet in practice, the evident operational value of this flexibility is sometimes counterbalanced by the risk of agents' disempowerment or loss of efficiency. By sharing the amount of work, agents may not feel fully responsible for the quality of service provided. They may believe that the system will always be able to compensate for the difficulties that they may encounter. Moreover, there is also the possibility of a new agent losing time when reading and understanding the conversation history. That is why many contact centers prefer not to let chats be served by more than one agent. In this paper, we consider the two cases where chats can or cannot be handed to other agents during their service.

In contrast to the existing studies, our modeling allows for two important practical features. First, existing studies usually assume, for simplicity's sake, that, for a given agent, the service rate per chat decreases with the number of simultaneously handled chats. However, this is not always the case in practice. A statistical study by Tan and Netessine (2014) shows that the agents' speed-up behavior is observed and may be explained by a rushing effect. Here, we allow the service rate per chat to be an arbitrary function of the number of simultaneously handled chats. As a consequence, to obtain better efficiency, we also relax the assumption of the traditional lightest-load-first rule which says that the least busy agent should be given priority for the service of a new customer. Second, we do not restrict the routing policy of chats to agents to the traditional fixed threshold policy, where chats are automatically assigned to a given agent until a predefined maximum limit is reached. We instead allow for idling policies. A chat is not automatically routed to an agent because this may deteriorate the system's performance. The decision then depends on the system's state. The use of a state-dependent policy is illustrated in Figure 1 for a 24/7 Brazilian chat contact center. This contact center belongs to a service provider that offers customers technical support. The staffing is modified every two hours in order to match demand variation. The points in the figure correspond to a change in the agent's state due to a chat service completion or a chat entering service. The figure shows the number of simultaneously handled chats for a given agent as a function of time. We observe from the figure that the number of simultaneously handled chats increases with the system workload (the threshold is the highest during peak hours).

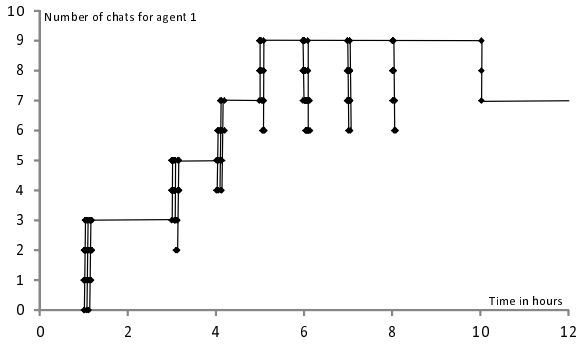


Figure 1: Data from a Brazilian chat contact center

Contributions. In this paper, we focus on a chat routing problem where the contact center manager seeks a trade-off between various conflicting objectives, namely, the chat waiting time in the queue, the service time duration per chat, and the proportion of lost customers due to abandonment. We consider two cases; the *shared work case* where chats can be handed to other agents during service and the *non-shared work case* where this flexibility is forbidden.

In the shared work case, we provide the required conditions under which the traditional Least Busy First (LBF) or the Most Busy First (MBF) policies are optimal for agent selection. In addition, we provide a

method to optimally distribute chats among agents. Next, we consider the chat initiation problem. The idea is to determine whether or not a chat should be sent to service. Using a Markov decision process approach, we prove in a context without abandonment that a state-dependent threshold reservation policy based on the number of chats in the queue and in service is optimal. With abandonment, using a smoothed rate truncation approach, we derive the required conditions for the optimality of this policy. Our analysis reveals that partial idling is beneficial for the system's performance, as it allows agents to have higher productivity and customers to abandon more from the queue in order to reduce the system congestion. In practice, the agents' capacity is used at its maximal value. We show that this may lead to poor performance. Yet, by optimizing the maximum number of chats per agent, the contact center can implement a simple and close-to-optimal policy. As the contact center size increases, the gap between the optimal and the fixed threshold policy reduces.

In the non-shared work case, we develop a one-step policy improvement algorithm to obtain an efficient dispatching policy. This method is based on the explicit computation of the relative value function for an initial policy with fixed probabilities for admission control. An important value of this method is that it avoids the curse of dimensionality problem. The method developed in the non-shared work case is generic and goes beyond the analysis of the chat queue in this paper. It provides near-optimal routing policies for the classical problem of routing jobs to parallel queues with impatient customers. In the chat context, we show that our dispatching policy significantly outperforms non state-dependent policies (i.e., policies with fixed probabilities). Compared to the latter, it reduces the abandonment from the queue without necessarily increasing the abandonment from service. The value of our policy is that it partly compensates the negative constraint of not sharing work between agents. The policy almost achieves the same performance as in the shared work case in small or large contact centers.

The remainder of this paper is structured as follows. Section 2 reviews the related literature. Section 3 describes the setting and the optimization question for the chat contact center. Section 4 focuses on the agent selection problem in the shared work case. Section 5 is devoted to the optimization of the service admission policy in the shared work case. Section 6 develops an algorithm to derive an efficient routing policy in the non-shared work case. Section 7 concludes the paper. The mathematical proofs and additional results are given in an online supplement.

2 Literature review

The literature on the operations management in call centers is rich (Akşin and Harker, 2003; Avramidis et al., 2010; Jouini et al., 2010; Aktekin, 2014; Ibrahim et al., 2016; Barrow and Kourentzes, 2018). For some background on this literature, we refer the reader to the two surveys by Gans et al. (2003) and Akşin et al.

(2007). Although the use of chats is growing, there are very few related papers in the literature. Tezcan and Zhang (2014) consider the objective of minimizing the staffing while providing a certain service level measured in terms of the proportion of customers who abandon the system in the long run. They propose effective routing policies based on a static planning linear programming for the two cases of observable or unobservable arrival rates. The modeling is however restricted to the case of decreasing service rates per chat and a fixed threshold for the maximum number of chats per agent. Luo and Zhang (2013) in turn fix the routing policy and focus on staffing optimization by investigating a fluid limit approximation assuming infinitely patient customer chats. Other solutions to the staffing problem are provided by Cui and Tezcan (2016) using diffusion limits.

Another stream of literature related to this paper is the analysis of *processor sharing* queues. Processor sharing at a server is a setting where all arriving customers enter service immediately (there is no queue), but the service rate they receive is proportional to the number of customers in service. Web servers are a good example where the processor sharing discipline is applied. This discipline is usually opposed to the first-come-first-served discipline where tasks are handled one by one. Chow et al. (1979) and Altman et al. (2011) consider the problem of routing jobs to different processor sharing servers. In the case of homogeneous servers, it is shown that an equal load balancing policy is optimal. This may explain the assumption of the Least Busy First policy used in most articles on the chat channel although this policy is not optimal. Some articles on processor sharing also focus on performance evaluation. The framework is to use a measure-valued process. For instance, Gromoll et al. (2008) use a measure-valued descriptor for the analysis of processor sharing queues that are not overloaded. Zhang et al. (2009) propose a measure-valued fluid model and show that there is a unique associated fluid model solution. Further references on processor sharing include Avi-Itzhak and Halfin (1988); Jean-Marie and Robert (1994); Puha et al. (2006); Haviv and Van der Wal (2008); Ravner et al. (2016); Vanlerberghe et al. (2018). The chat channel studied in this article differs from a processor sharing queue in the sense that all customers are not immediately admitted in service. They may wait in a queue before entering service. Moreover, the service rate received is not necessarily proportional to the number of chats in service in our modeling.

Our study is also related to the historical problem of *routing to parallel queues*. Winston (1977) is the first to show that the intuitive "Join the shortest queue" policy is optimal to minimize the expected sojourn time with identical exponential servers. Using a dynamic programming approach, Hordijk and Koole (1992) extend the analysis to different service rates and show that the "Shorter Queue Faster Server Policy" is optimal. When the exponential assumption for services is relaxed, the above intuitive policies are no longer necessarily optimal. Counterexamples can be found in Whitt (1986) and Koole et al. (1999). Several related articles can be found with focus on routing optimization, performance evaluation, for individual or social optimization, with observable queues or not (Guo et al., 2004; Hordijk and van der Laan, 2004; Anselmi

and Gaujal, 2011; Anselmi and Casale, 2013; Legros, 2018). The method developed in this article for the non-shared work case contributes to the methodology for the routing to parallel queues. Unlike the existing studies, here we allow for state-dependent departure rates.

A last stream of literature related to this paper is the analysis of *reservation strategy* in call centers. In most studies, reservation strategies have been considered when two different job types, inbound and outbound calls, have to be handled by a unique group of agents. This consists in analyzing *call blended policies*. Some papers focus on performance evaluation, and others address the analysis of blending policies or staffing decisions. Deslauriers et al. (2007) develop various continuous Markov chain models for a call center with inbound and outbound calls. The authors consider a threshold policy and characterize the rate of outbounds and the waiting time distribution of inbounds. Further references include Bernett et al. (2002); Pichitlamken et al. (2003); Kim et al. (2012); Legros et al. (2018). Other call center papers address the analysis of blending policies. Gans and Zhou (2003) and Bhulai and Koole (2003b) prove that a threshold policy on the number of idle agents is optimal to maximize the outbound throughput under a service level constraint on the inbound waiting time. Similar results are also found in Legros et al. (2015), for a non-stationary model where inbound calls arrive according to a non-homogeneous Poisson process. Pang and Perry (2014) consider a large call center blending model and propose a logarithmic safety staffing rule, combined with a threshold control policy to ensure that agents' utilization is close to one with always idle agents present. For a call center model with a callback option, Legros et al. (2016) examine the effect of the callback offer on the system's performance and show that a state-dependent reservation policy is optimal. Considering a single job type, Legros (2017) shows that reservation can also be efficiently used to reduce balking and abandonment. The particularity of our model compared to the models with the call channel in the literature is that we allow the simultaneous handling of chats per agent. Moreover, unlike the call channel in the above studies, an inbound chat is not necessarily served when an agent is available. Depending on the system congestion, one has to decide whether to serve a chat immediately or to maintain it in the queue. This enlarges the range of possibilities for improving the system's performance but also leads to a more complex optimal policy than a fixed threshold policy.

3 Setting

We first describe the queueing chat model and then formulate the routing optimization problem of the system manager.

Model description. Chats arrive at an infinite capacity first-come-first-served queue according to a Poisson process with rate λ. The system capacity consists of a homogeneous pool of s agents. Given the chat channel nature (written conversation with less required responsiveness than for calls), an agent has the abil-

ity to serve more than one customer simultaneously. An agent is said to be at *level* i if she is serving i customers ($i \geq 0$) at once. We assume that the service time of a customer receiving service from an agent at level i is exponentially distributed with rate μ_i, for $i \geq 1$.

The traditional ordering assumption is $\mu_{i+1} \leq \mu_i$, for $i \geq 1$. It means that the more tasks an agent is working on, the slower she will be on each task. In this paper, we do not restrict the modeling to a specific ordering of the service rates. For instance, the traditional ordering $\mu_{i+1} \leq \mu_i$, for $i \geq 1$, is not appropriate for the agents' behavior when rushing. It is observed in practice that a large number of simultaneous tasks may push the agents to shorten the conversation duration. This often corresponds to real-time instructions given by the management as is the case for the voice channel. Service rates may therefore increase in the number of simultaneous chats. The increasing speed per served chat due to a rushing effect is usually followed by a decreasing speed per served chat when the number of chats in service is high. This behavior is due to loss of concentration or excessive workload (Tezcan and Zhang, 2014).

Because of the text messaging interaction, a customer may abandon not only while waiting in the queue, but also while being in service if she estimates that the quality of the interaction is poor. We assume that the abandonment time of chats from the queue is exponentially distributed with rate γ_q, and that of chats during the service is exponentially distributed with rate γ_s. Since the conversation history is available, a new agent may take over the remaining service without informing the customer. This flexibility can be used as a tool to improve performance. In summary, the main characteristics of the chat channel studied in this article compared to the call channel are: (i) abandonment during service, (ii) simultaneity in chat handling, and (iii) a chat can be handed to other agents during service.

The optimization objective. We use $E(W_q)$ and $E(S)$ to denote the stationary expected waiting time in the queue, and the customers' expected service time, respectively. More specifically, $E(W_q)$ is the expected waiting time of both served and lost customers. If a chat does not wait in the queue, the value 0 is counted for its waiting time. The expected service time, $E(S)$, corresponds to the expected time spent in service, irrespective of whether the service is completed or not or if a customer enters service or not. If a customer does not enter service, the value 0 is associated to its service time. We use $P_{a,q}$ to denote the stationary proportion of abandonment from the queue, and $P_{a,s}$ to denote the stationary proportion of abandonment from service. The overall proportion of abandonment is denoted by P_a; $P_a = P_{a,q} + P_{a,s}$.

In a chat service system, the performance of the queue is in conflict with the performance of the service. The first conflict is between $E(W_q)$ and $E(S)$. As in a traditional voice call center, the manager worries about the waiting time in the queue. Thanks to the agents' ability to simultaneously work on different chats, the manager may decide to assign many waiting chats into service. Although this may improve the expected waiting time in the queue, chat handling may take too much time due to an excessive number of chats in service. Therefore, the chat contact center manager wants to strike a good balance between two conflicting

objectives, namely, the expected waiting time in the queue and the expected service time.

The system manager is also worried about losing customers due to abandonment. Abandonment may happen either from the queue or from service. If too many customers are sent to service, service speed may significantly slow down. This may also lead to a high proportion of abandonment from service. Again, a good balance has to be found, so as to minimize the overall proportion of abandonment.

The expected waiting time in the queue (respectively, the expected service time) and the proportion of abandonment from the queue (respectively, the proportion of abandonment from service) increase with the congestion in the queue (respectively, in service). Hence, by appropriately controling the congestion in the queue and in service, one may appropriately control the two aforementioned performance measures at the same time. Yet, due to the difference in the importance perceived by the manager of the two performance measures (abandonment and wait), we cannot merge these two objectives into a simple congestion control. For instance, if the manager does not care about abandonment but cares more about waiting time, then it may be beneficial to let customers abandon in order to reduce the waiting time instead of sending them into service so as to reduce the congestion of the queue.

In summary, the manager's goal can be formulated as minimizing the following weighted cost summation, denoted by M,

$$M = c_1 E(W_q) + c_2 E(S) + c_3 P_a, \tag{1}$$

where the coefficients c_1, c_2 and c_3 are the cost parameters that translate the relative importance between the three cost components of the objective function M. The objective is to optimize the chat routing policy in order to minimize M. The routing problem in chat service systems has two connected parts: i) Agent selection problem: at any point in time we optimize the choice of the best agent to serve a chat, if any; ii) Chat initiation problem: at any point in time we optimize the chat service initiation, i.e., the choice of serving a chat or not. This means that we may force a chat to stay in the queue while an agent has the capacity to serve it. This is useful since a new customer entering service could harm the performance for the customers already in service.

Determining the optimal routing policy consists in finding the best action at each state of the system. Yet, simultaneity in chat handling leads to a very high dimensional problem which could not be tackled because of the curse of dimensionality. This may even be a problem with space state truncation. For instance, if the queue is truncated and has n waiting spaces ($n \geq 1$) and if each agent can serve up to u chats simultaneously ($u \geq 1$), the state space contains exactly $n \times u^s$ states. The routing problem therefore has an exponential complexity. This problem has already been described in Bellman (1961). Approximation methods or simplifications are therefore often developed.

In order to avoid this problem of dimensionality, we assume that chats can be handed to other agents

during their service without loss of efficiency. Under this assumption, agents are seen as a team which can be optimized at any point of time in order to maximize its efficiency. This team approach leads to a two-dimensional problem: the number of customers in the queue and the number of customers in service. The corresponding optimal policy is given in Sections 4 and 5. We call this case the *Shared Work Case* (SWC).

In practice, there is also the possibility of a new agent losing time when reading and understanding the conversation history. In addition, removing a task from an agent can be wrongly interpreted by giving a signal to the agent that the service is not efficiently provided. That is why the possibility that a chat can be handed to another agent during its service is not always used in practice. We then propose to also study the case where chats cannot be handed to other agents. We call this case the *Non-shared Work Case* (NWC). We assume that each agent has its own queue. Depending on the system's state, a dispatcher has to decide the queue to which a chat should be routed. Chats are dispatched upon arrival but we also allow for jockeying, i.e., moving a chat from one queue to the other at any point of time. By moving one chat from one queue to the other, the situations where one chat is waiting while an agent could efficiently serve this chat can be avoided. The benefits of having one common queue are therefore maintained.

Under the NWC, we develop a *one-step policy improvement* method in Section 6 to obtain near-optimal solutions for chats' dispatching. The value of this method is to reduce the complexity of the problem to a linear one. By using the results of Section 5 in the single server case, a chat dispatching solution in the NWC can be derived. Figure 2 depicts the methodological approach for the analysis.

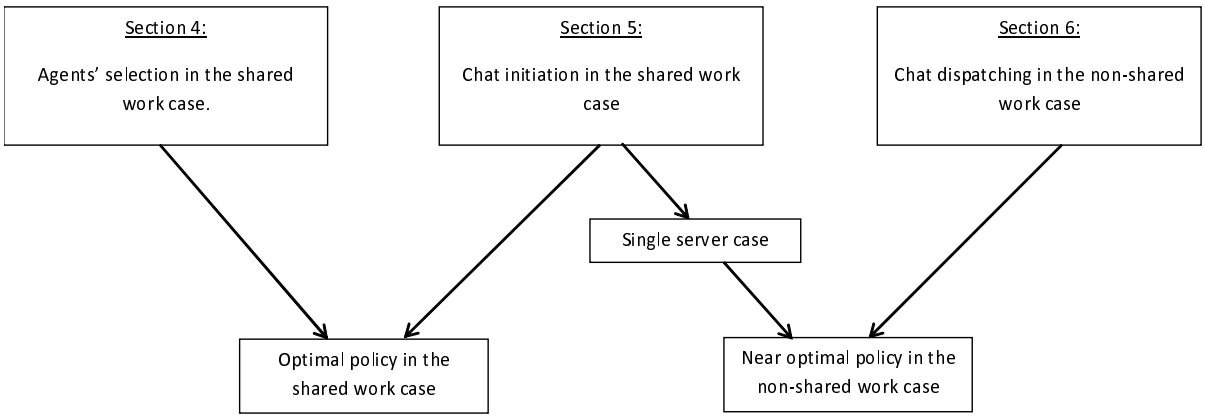


Figure 2: Methodological approach

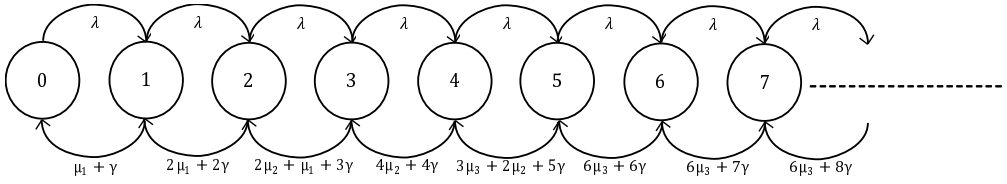
4 The agent selection problem in the SWC

We address the agent selection problem in the SWC. The objective is to develop a solution to balance the load among agents when a given quantity of chats is in service. More precisely, we consider a situation with k chats in service. We denote by x_i the number of chats currently handled by agent i, for $0 \leq x_i \leq k$ and

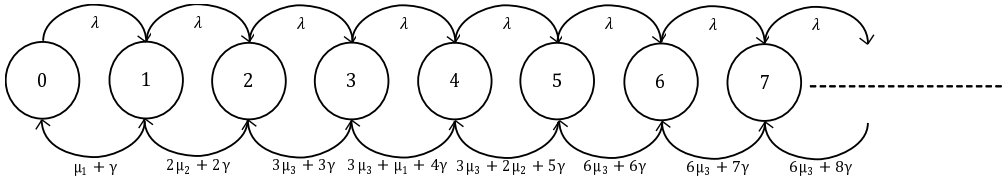
$1 \leq i \leq s$. We want to maximize the average service rate per chat defined as $\frac{x_1\mu(x_1)+x_2\mu(x_2)+\cdots+x_s\mu(x_s)}{s}$ subject to the constraint $x_1 + x_2 + \cdots + x_s = k$. By maximizing the average service rate per chat for a given quantity of chats in service, the system also minimizes the expected time spent in service and maximizes the flow from service. This in turn minimizes the expected waiting time and the proportion of abandonment from the queue by allowing customers to enter service quicker than under another policy. Moreover, since the proportion of abandonment from service increases with the expected time spent in service, by maximizing the average service rate per chat, one also minimizes the proportion of abandonment from service. In conclusion, for a given quantity of chats in service, a policy which maximizes the average service rate per chat optimizes the rate selection and provides a first step for minimizing M, as given in Equation (1).

The most common routing rule, in practice as well as in the existing literature, is the Least Busy First (LBF) policy. It consists in equiprobably sending a chat to one of the agents currently serving the smallest number of chats. This rule is simple and fair between agents. Yet, it may not be optimal. Surprisingly, the opposite mechanism under the Most Busy First (MBF) policy could outperform LBF in certain situations. In what follows, we give a counterexample which shows that the known LBF policy is not optimal.

Counterexample to show that LBF is not optimal. Consider a chat queueing model with two agents where each agent has the ability to serve up to 3 chats. In addition, we assume the same abandonment rate in the queue and in service; $\gamma_s = \gamma_q = \gamma$. The Markov chains of this model under LBF and MBF are depicted in Figure 3. From the transition rates, one may already observe that the most efficient load balancing rule is not necessarily LBF. First, due to the rushing effect we could have $\mu_3 > \mu_2 > \mu_1$, which makes the transition rates from state 2 to 1 and from state 3 to 2 higher under MBF than LBF. Second, if $\mu_2 < \frac{3\mu_3+\mu_1}{4}$ which reflects a type of convexity, the transition rate from state 4 to 3 is higher under MBF than LBF. Figure 4 shows that LBF is not always the optimal policy. In our illustration, the performance



(a) LBF



(b) MBF

Figure 3: Markov chains (a state is defined by the number of customers in the system).

of LBF is more sensitive to the increase in μ_1 than the performance of MBF, which makes the former policy

worse than the latter one for low values of μ_1.

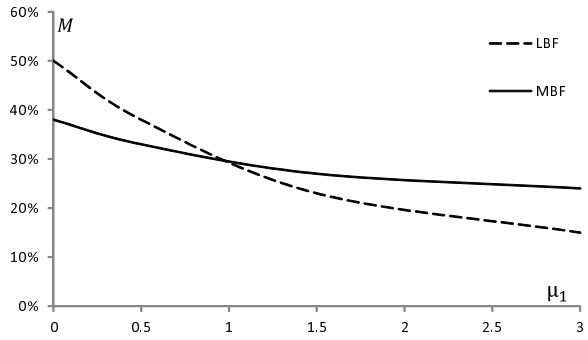


Figure 4: M as a function of μ_1 ($s = 2$, $\mu_2 = 1.5$, $\mu_3 = 1$, $\lambda = 2$, $\gamma = 0.5$, $c_1 = c_2 = 0$, and $c_3 = 1$)

LBF and MBF are extreme load balancing solutions for agent selection. Although these policies are not necessarily optimal, they are simple to implement. It is therefore interesting to determine the conditions which make one of these policies optimal. These are given in Proposition 1 and Corollary 1. We use a myopic approach to determine how the chats in service should be distributed among the agents to maximize the average service rate per chat. Consider a situation with k chats in service. The function μ_i is defined for *integer* values of i for $1 \leq i \leq k$. We consider the Lagrange interpolation polynomial with a degree at most equal to $k-1$ which coincides with μ_i for $i = 1, 2, \cdots, k$. This polynomial, denoted by $\mu(x)$, in the variable $x \geq 0$ is unique (see, e.g., Weisstein (2004)). It is given by $\mu(x) = \sum_{i=1}^{k} \mu_i \left(\prod_{j=1, j \neq i}^{k} \frac{x-j}{i-j} \right)$. One can then compute derivatives of $\mu(x)$, since $\mu(x)$ is a polynomial in x. In particular, we define $\mu'(x)$ by $\mu'(x) = \frac{\partial \mu(x)}{\partial x}$ and $\mu''(x)$ by $\mu''(x) = \frac{\partial^2 \mu(x)}{\partial x^2}$. We now refer to μ_x or to $\mu(x)$ whether we consider the values of the service rates defined for $x \in \mathbb{N}$ or $x \in \mathbb{R}$, respectively.

Proposition 1 *Consider a state with k chats in service. For $1 \leq x \leq k$, if the function $\mu(x) + x\mu'(x)$ is strictly increasing (respectively decreasing), then MBF (respectively LBF) maximizes the average service rate per chat.*

Corollary 1 *For $1 \leq x \leq k$, if the function μ_x is strictly increasing and convex (respectively decreasing and concave), then MBF (respectively LBF) maximizes the average service rate per chat.*

The proofs of Proposition 1 and Corollary 1 are given in Section 1 of the online supplement. The point here is that the commonly used LBF policy may not lead to optimized performance measures. In particular, if the service rate per chat is increasing and convex, then the optimal policy is MBF in some states. In other words, if a rushing effect is observed (agents are more efficient when they have more work to do), then it may be better not to be fair in the distribution of work among the agents. Note however from Corollary 1 that MBF cannot be optimal irrespective of the number of chats in service. If μ_x is strictly increasing and

convex in x for $x \geq 1$ then μ_x is an unbounded function of x. This condition is unrealistic for a human agent. Moreover, having MBF optimal irrespective of the number of chats in service and without a limit in the number of chats per agent would mean that it is optimal to only have one agent in the contact center. This is also unlikely to happen.

From now on, we assume that in the SWC when k chats are admitted into service, they are distributed so as to maximize the service rate per chat. We use μ_k to denote the optimal service rate per chat when k chats are in service. The overall service rate is then $k\mu_k$. This overall service rate is the so-called agent productivity. Consequently, instead of considering the number of chats per agent, we consider the overall number of chats in the agents' team. The agents' team is said to be at level k if k customers ($k \geq 0$) are in service. In the next section, we address the routing optimization problem of serving a chat or keeping it in the queue.

5 The service initiation problem in the SWC

Since the agent productivity may decrease in the number of simultaneous tasks, it could be interesting to apply partial idling. We apply a Markov decision process approach to characterize the optimal routing policy. The value of this type of approach is that they can lead to exact optimal policies in the long run under a stochastic context (Pandelis, 2010; Zhuang and Li, 2012; Fianu and Davis, 2018). As mentioned above, chats are distributed among the agents so as to maximize the average service rate per chat. One can therefore see the number of chats in service as only one dimension of the problem. It is therefore equivalent to considering the multi-server setting as a single server one.

Let us denote the number of chats in the queue by x ($x \in \mathbb{N}$), and the number of chats in service by y ($y \in \mathbb{N}$). We then describe the possible transitions from a given state (x, y), for $x, y \geq 0$.

1. An arrival in the queue with rate λ. The number of chats in the queue is increased by 1, which changes the state to $(x + 1, y)$.

2. An abandonment from the queue with rate $x\gamma_q$. The number of chats in the queue is reduced by 1. This changes the state to $(x - 1, y)$.

3. A service completion with rate $y\mu_y$. The number of chats in service is reduced by 1. This changes the state to $(x, y - 1)$.

4. An abandonment from service with rate $y\gamma_s$. The number of chats in service is reduced by 1. This changes the state to $(x, y - 1)$.

Since an abandonment from service and a service completion have the same effect on the system's state, we consider the departure rate per chat, denoted by d_y, when y chats are in service, $d_y = \mu_y + \gamma_s$, for $y \geq 1$.

At any point of time if the queue is not empty, the contact center operator has to decide whether to send a new chat into service or not. A function which associates one of these decisions to each state of the system is a so-called *service initiation policy*.

We want to determine the form of the optimal service initiation policy. One major difficulty of the analysis is that the overall event rate is an unbounded function of actions and states because of abandonments. This prevents the uniformization technique from being applied when defining the dynamic programming operators. To overcome this difficulty, we propose the following approach. In Section 5.1, we restrict the analysis to the case with no abandonments. This allows us to prove a set of structural properties from which we deduce the form of the optimal policy. In addition, we investigate the impact of the chat arrival rate on the parameters of the optimal policy. In Section 5.2, we extend the analysis to the case with abandonments using the *smoothed rate truncation* method in order to compute the optimal policy and solve the problem of unbounded rates. Finally, in Section 5.3, we evaluate the optimal policy in comparison with the commonly used *fixed threshold policy* where each agent receive chats for service until a predefined capacity is reached.

5.1 Without abandonment ($\gamma_q = \gamma_s = 0$)

Without abandonment the overall event rate is bounded (otherwise the agent productivity could be infinite). Our model is therefore uniformizable (Puterman, 1994). We therefore assume without loss of generality that $\lambda + \max(y\mu_y) = 1$. We define a 2-step value function as follows: $U_0(x, y) = V_0(x, y) = 0$, $V_n(x, -1) = 0$ for $x, n \geq 0$ and

$$U_{n+1}(x, y) = \frac{c_1}{\lambda}x + \frac{c_2}{\lambda}y + \lambda V_n(x + 1, y) + y\mu_y V_n(x, y - 1) + (1 - \lambda - y\mu_y)\, V_n(x, y), \tag{2}$$

with

$$V_{n+1}(x, y) = \begin{cases} U_{n+1}(0, y), & \text{if } x = 0, \\ \min(U_{n+1}(x, y), U_{n+1}(x - 1, y + 1)), & \text{if } x > 0, \end{cases} \tag{3}$$

for $n, x, y \geq 0$. The cost parameters c_1 and c_2 are divided by λ in order to measure the expected waiting time and the expected time spent in service for a given customer according to Little's law. The long-run average optimal actions can be obtained through value iteration, by recursively evaluating V_n using Equations (2) and (3), for $n \geq 0$. As n tends to ∞, the minimizing actions converge to the optimal ones (Puterman, 1994). For $x > 0$, the minimizing action is chosen between keeping a chat in the queue or starting the service of this chat. Theorem 1 proves for increasing and concave cases of agent productivity that if some structural properties defining the state-dependent threshold structure of the optimal policy are satisfied for V_n, then these properties are also satisfied for V_{n+1}. They therefore hold for every n. As n tends to infinity, the optimal policy converges to the unique average optimal policy. This convergence result is ensured by

Theorem 8.10.1 in Puterman (1994) since our problem satisfies the conditions of the theorem (countable state set, finite set of actions and uniformizable system). More explicitly, a state-dependent threshold policy is characterized by the fact that if keeping a customer in the queue is optimal in x, then keeping a customer is also optimal in $x+1$. A sufficient condition is

$$V_n(x, y+1) - V_n(x+1, y) \leq 0 \Longrightarrow V_n(x+1, y+1) - V_n(x+2, y) \leq 0.$$

This condition is satisfied if

$$V_n(x+2, y) + V_n(x, y+1) - V_n(x+1, y+1) - V_n(x+1, y) \geq 0.$$

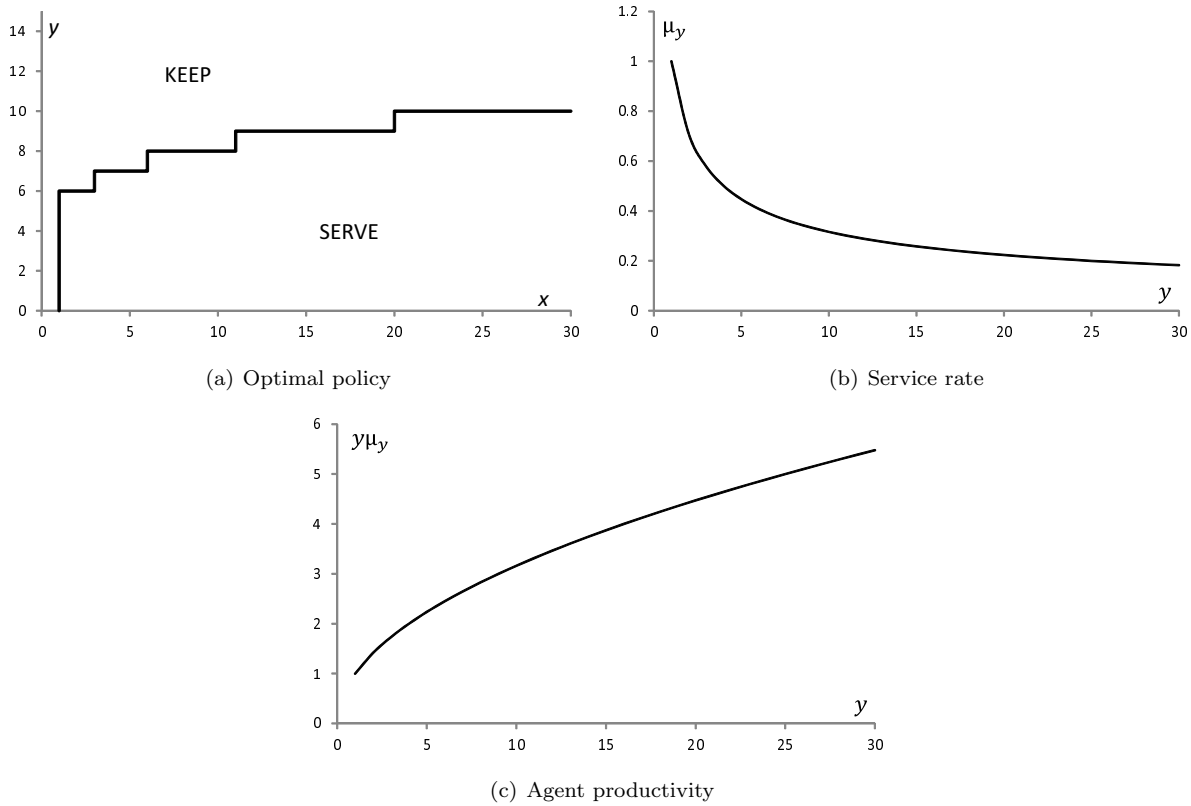
Theorem 1 *If $y\mu_y$ is increasing and concave in y, then the optimal policy is a state-dependent threshold policy. This threshold is increasing in the variables x and y. More precisely, one can define a threshold function, $y = u(x)$ for $x > 0$, such that the optimal action in state (x, y) is to route a chat in service if and only if $y \leq u(x)$.*

The proof of Theorem 1 is given in Section 2 of the online supplement.

Numerical illustration. Figure 5 illustrates Theorem 1. We choose $\mu_y = \frac{1}{\sqrt{y}}$ for $1 \leq y \leq 50$, and $\mu_y = 0$ for $y > 50$. The function $y\mu_y$ is therefore increasing and concave in y (Figure 5(c)). Figure 5(a) gives the optimal policy. It is defined by a threshold function, $y = u(x)$, which separates the states where it is optimal to serve a chat (on or below the curve) from those where it is optimal to keep a chat in the queue (strictly above the curve). As proven in Theorem 1, we observe that the threshold function is increasing in x. In addition, we observe that the function $u(x)$ is characterized by a first important jump at $x = 1$ followed by smaller ones. This means that an agent should serve up to 6 chats irrespective of the system's state. Yet, under low demand situations, an agent cannot necessarily have at least 6 chats at all times. One way to avoid the inefficient use of the contact center resources is to allow agents to initiate outbound chats. This possibility is investigated in Section 3 of the online supplement.

Impact of the arrival rate. We now want to investigate the impact of the arrival rate on the reservation policy. Proposition 2 states that reservation decreases with the arrival rate. In other words, at a state (x, y) $(x, y \geq 0)$, if the optimal action is to keep a chat in the queue for a given arrival rate λ_1, then with $\lambda_2 \leq \lambda_1$ it is also optimal to keep a chat in the queue. This is a confirmation of the observation made on the real data in the introduction. The more the system is congested, the more likely the decision will be to have more chats per agent.

Proposition 2 *Consider two identical situations with increasing and concave agent productivity except that*



(a) Optimal policy

(b) Service rate

(c) Agent productivity

Figure 5: Numerical illustration ($\lambda = 5$, $\gamma_q = \gamma_s = 0$, $c_1 = 0.1$, $c_2 = 1$ and $c_3 = 0$)

the first has arrival rate λ_1 and the second has λ_2. In each situation, the optimal policy is a state-dependent threshold policy defined by the functions $y = u_1(x)$ and $y = u_2(x)$. If $\lambda_1 \geq \lambda_2$, then $u_1(x) \geq u_2(x)$, for $x > 0$.

The proof is given in Section 4 of the online supplement. The idea is to consider two systems with two different arrival rates and to prove that if it is optimal to serve a customer at a given state (x, y) in the system with the smallest arrival rate, then it is also optimal to serve a customer at state (x, y) in the other system.

5.2 With abandonment ($\gamma_s, \gamma_q \geq 0$)

We now include the feature of abandonment. In order to transform the initial continuous-time Markov decision process into a discrete-time one, the Markov decision process has to be uniformizable. In other words, the jump rates must be uniformly bounded as a function of actions and states. Yet, with abandonment, the jump rates are unbounded functions of states. One solution to overcome this problem is to truncate the state space. However, a simple truncation does not preserve the structural properties of the initial model and may not allow us to derive the optimal policy. In particular, a simple truncation breaks the convexity properties of the value in the number of chats in the queue (variable x) at the truncated state.

To overcome this difficulty, Bhulai et al. (2014) develop a new method called the smoothed rate truncation

method. This method uses linearly smoothed transition rates, such that the system retains the structural properties of the infinite system. The approximation is obtained by linearly decreasing the relevant transition rates as functions of the state variables until these rates equal zero. The method provides a naturally finite state system. More precisely, the transition structure is modified in the smoothed rate system using the control parameter N such that the transition rate from state (x, y) to state (x', y'), denoted by $t_{(x,y),(x',y')}$, is given by

$$t_{(x,y),(x',y')} = \begin{cases} \left(\lambda - x\frac{\lambda}{N}\right)^+, \text{ for } x' = x+1, y' = y, \text{ and } x, y \geq 0, \\ \left(\gamma_q - y\frac{\gamma_q}{N}\right)^+ x, \text{ for } x' = x-1, y' = y, \text{ and } x > 0, y \geq 0, \\ \left(d_y - x\frac{d_y}{N}\right)^+ y, \text{ for } x' = x, y' = y-1, \text{ and } x \geq 0, y > 0, \\ 0, \text{ otherwise.} \end{cases}$$

As N tends to infinity, the smoothed rate system converges to the original one. This allows to determine the optimal policy and the relevant performance measures. The convergence result is proven in Theorem 3.1 of Bhulai et al. (2014). Since the smoothed rate system has a finite number of states, it is uniformizable. We can therefore redefine the relative value function associated with our model assuming $\lambda+\gamma_q N+\gamma_s N+\mu_{max} N = 1$ (uniformization). Again, we choose to formulate a 2-step value function. Let us use $V_n(x, y)$ to denote the expected costs over n steps, for $n, x, y \geq 0$. We have

$$\begin{aligned} U_{n+1}(x,y) = {} & \frac{c_1}{\lambda}x + \frac{c_2}{\lambda}y + \left(\lambda - x\frac{\lambda}{N}\right)^+ V_n(x+1,y) + \left(\gamma_q - y\frac{\gamma_q}{N}\right)^+ x\left(V_n(x-1,y) + \frac{c_3}{\lambda}\right) \\ & + \left(d_y - x\frac{d_y}{N}\right)^+ y V_n(x,y-1) + \left(\gamma_s - x\frac{\gamma_s}{N}\right)^+ y\frac{c_3}{\lambda} \\ & + \left(1 - \left(\lambda - x\frac{\lambda}{N}\right)^+ - \left(\gamma_q - y\frac{\gamma_q}{N}\right)^+ x - \left(d_y - x\frac{d_y}{N}\right)^+ y\right) V_n(x,y), \end{aligned} \tag{4}$$

with

$$V_{n+1}(x,y) = \begin{cases} U_{n+1}(0,y) \text{ if } x = 0 \\ \min(U_{n+1}(x,y), U_{n+1}(x-1,y+1)) \text{ if } x > 0 \end{cases}, \tag{5}$$

for $n, x, y \geq 0$ and an arbitrary $V_0(x, y)$ for $x, y \geq 0$. We choose $V_0(x, y) = U_0(x, y) = 0$ for $x, y \geq 0$. We divide c_3 by λ because we are interested in the proportion of abandonment.

Not all structural properties exhibited in Theorem 1 can be proven under a value iteration step. However in Theorem 2, we prove that the main structural properties (first and second order monotonicities in x and y and the supermodularity) are maintained in the induction step from V_n to U_{n+1} when yd_y is increasing and concave. In a context with abandonment, this partly proves that the optimal policy is a state-dependent threshold policy as in Theorem 1. The condition on the departure rates in Theorem 2 is a natural extension of the condition on the service rates in Theorem 1. The proof is given in Section 5 of the online supplement.

Theorem 2 *Consider the class of functions $\mathcal{G}$ from $\mathbb{N}^2$ to $\mathbb{R}$ defined as follows: $f \in \mathcal{G}$ if for $x, y \geq 0$, we have*

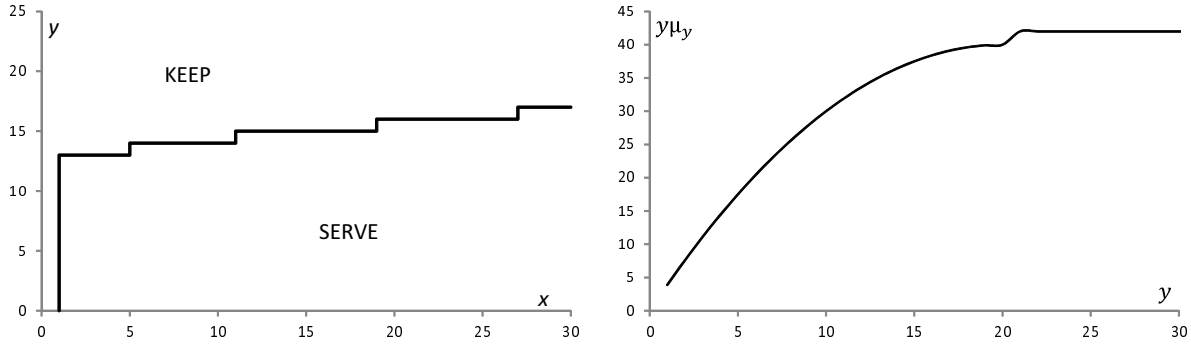
$$f(x+1,y) \geq f(x,y),$$
$$f(x,y+1) \geq f(x,y),$$
$$f(x+2,y) + f(x,y) \geq 2f(x+1,y),$$
$$f(x,y+2) + f(x,y) \geq 2f(x,y+1),$$
$$f(x,y) + f(x+1,y+1) \geq f(x+1,y) + f(x,y+1).$$

If yd_y is increasing and concave in y, and if $V_n \in \mathcal{G}$ then $U_{n+1} \in \mathcal{G}$.

Although the optimal policy cannot be completely proven, we numerically observe that it is a state-dependent threshold one even when the conditions of Theorem 2 are not satisfied. As illustrations, we consider the following three cases for the change in the service rates.

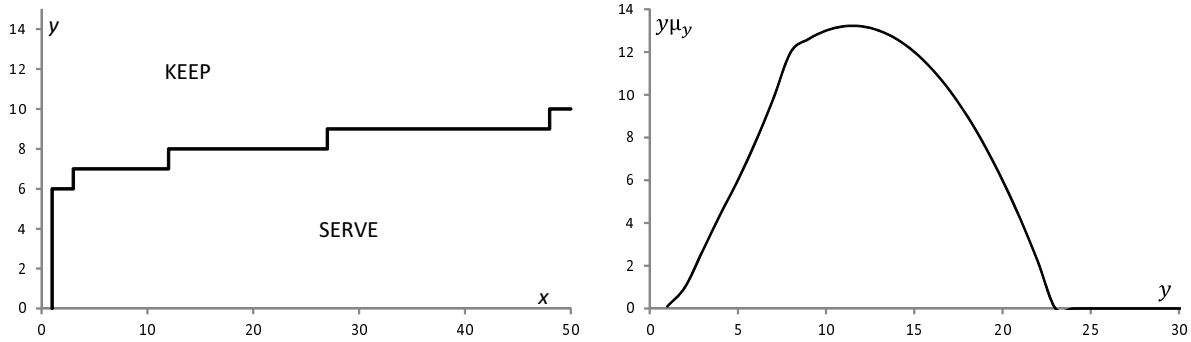
- Case 1: $\lambda = 30$, $\mu_y = 4 - 0.1y$, for $1 \leq y \leq 20$ and $\mu_y = 42/y$, for $y \geq 21$.
- Case 2: $\lambda = 10$, $\mu_1 = 0.1$, $\mu_2 = 0.5$, $\mu_3 = 0.9$, $\mu_4 = 1.1$, $\mu_5 = 1.2$, $\mu_6 = 1.3$, $\mu_7 = 1.4$, $\mu_{y+8} = 1.5 - 0.1y$, for $0 \leq y \leq 15$, and $\mu_{15+y} = 0$, for $y \geq 0$.
- Case 3: $\lambda = 6$, $\mu_y = 0.1y$, for $1 \leq y \leq 8$ and $\mu_y = 0$, for $y \geq 9$.

To obtain the optimal policy, we choose N high enough such that any further increase in N would not modify the state-dependent threshold. Case 1 corresponds to the condition of Theorem 1 where $y\mu_y$ is increasing and concave (Figure 6(b)). This case illustrates a situation where the overall throughput of served chats increases with the number of chats in service but the service time per chat decreases with the number of chats in service. Cases 2 and 3 are likely to correspond to realistic situations for the change in the service rate which are not represented in the condition of Theorem 1. Case 2 corresponds to a case where the productivity of the agents' team is first increasing then decreasing (Figure 7(b)). The increasing part is the gain of productivity obtained due to simultaneity when the number of simultaneously served chats is limited. The decreasing part illustrates the inability of agents to serve an excessive number of chats simultaneously which may reduce their productivity. Case 3 corresponds in turn to a case where the productivity is increasing and convex but limited to a capacity of 8 chats (Figure 8(b)). This case is an illustration of the rushing effect mentioned in the introduction section. For all cases, even for those where yd_y is not increasing and concave (Cases 2 and 3), the optimal policy is an increasing state-dependent threshold policy as illustrated in Figures 6(a), 7(a), and 8(a). Examples of service rate forms which do not lead to an increasing state-dependent threshold may nevertheless exist. This may happen for instance when the system is blocked for a given quantity of chats in service and only an extra chat in service could restart the agents' work. We provide an illustration of such



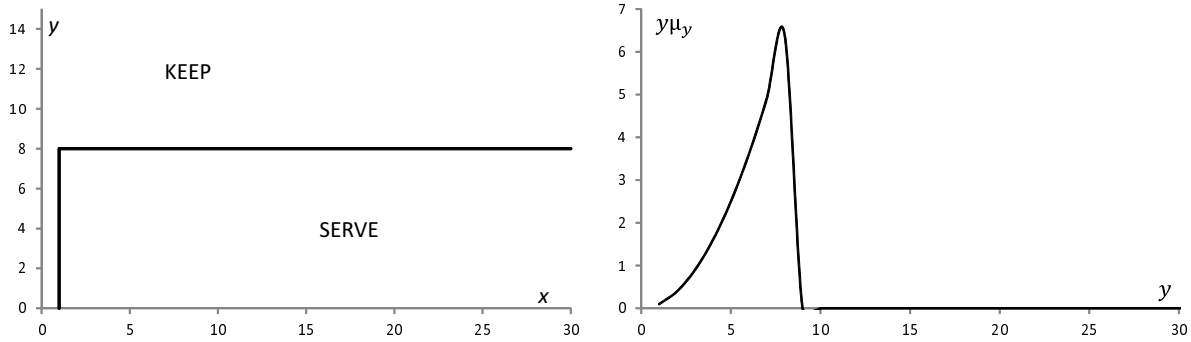
(a) Optimal policy (b) Agent productivity

Figure 6: Case 1 ($\lambda = 30$, $\gamma_q = \gamma_s = 1$, $c_1 = 0.1$, $c_2 = 1$ and $c_3 = 0.01$)



(a) Optimal policy (b) Agent productivity

Figure 7: Case 2 ($\lambda = 10$, $\gamma_q = \gamma_s = 1$, $c_1 = 0.1$, $c_2 = 1$ and $c_3 = 0.01$)



(a) Optimal policy (b) Agent productivity

Figure 8: Case 3 ($\lambda = 6$, $\gamma_q = \gamma_s = 1$, $c_1 = 0.1$, $c_2 = 1$ and $c_3 = 0.01$)

a case in Section 6 of the online supplement. Note however that such examples are not likely to happen in real-life.

5.3 Comparison with the fixed threshold policy

We now evaluate the optimal state-dependent threshold policy for chat initiation in comparison with the *fixed threshold policy*. The fixed threshold policy is controlled by a parameter u such that the number of chats

allowed in service per agent is at most equal to u. The parameter u determines the service capacity per agent. More precisely, a chat is equiprobably routed to one of the least busy agents if there is at least one agent with strictly less than u chats in service. If all agents have u chats in service, then an incoming chat waits in the queue. In other words, the fixed threshold policy is a special case of the state-dependent threshold policy where the threshold function $u(x)$ defined in Theorem 1 is a constant and where the LBF policy is applied. The fixed threshold policy is used in practice because it is easy to implement. Yet, the servers' capacity is used to its maximum most of the time (Tezcan and Zhang, 2014) instead of being optimized. In Figure 9, we show that using agent capacity at its maximal value may lead to poor performance. For instance, we observe that for $\lambda = 5$, we have 22.5% more abandonment for $u = 5$ (maximal capacity) than for $u = 2$ (optimal fixed threshold).

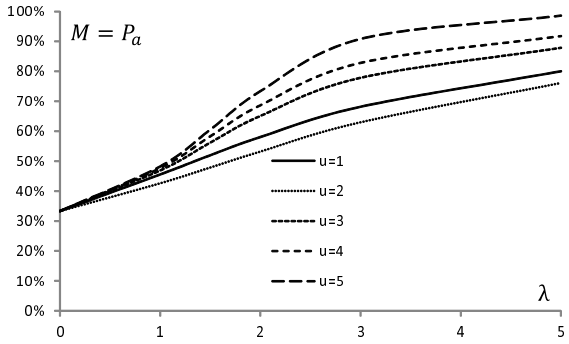


Figure 9: Impact of the fixed threshold ($\gamma_s = \gamma_q = 0.5$, $\mu_1 = 1$, $\mu_2 = 0.6$, $\mu_3 = 0.2$, $\mu_4 = 0.1$, $\mu_5 = 0.01$, $\mu_x = 0$, for $x > 5$)

Table 1 gives the performance measures under the optimal fixed threshold policy, the maximal fixed threshold policy ($u = 7$ in our illustrations) and the optimal state-dependent policy. The last column gives the differences, defined as $d_1 = M(\text{Optimal fixed threshold policy}) - M(\text{state-dependent policy})$ and $d_2 = M(\text{Maximal fixed threshold policy}) - M(\text{state-dependent policy})$. We choose $c_1 = c_2 = 0$, and $c_3 = 1$ such that the objective is to minimize the overall proportion of abandonment, $M = P_{a,q} + P_{a,s}$. The performance measures are obtained via a Markov chain analysis. The closed-form expressions of the performance measures under the optimal routing policy are derived in Section 7 of the online supplement.

The optimal fixed threshold is not necessarily equal to the agents' capacity. We only allow 1 or 2 chats per agent in Table 1 when at most 7 chats could be served per agent. The reason is related to two phenomena. First, increasing the number of chats in service may reduce the agent productivity. This in turn may lead to more abandonment from service. Second, the abandonment from the queue may help to reduce the congestion of the contact center.

As expected, the state-dependent threshold policy outperforms the fixed threshold policy. Yet, the difference between the two policies is not significant. This can be understood from the figures in Section 5.2. First, in many cases (as in Case 3, Figure 8) the fixed threshold policy is optimal. Second, Figure 6 and

Table 1: Performance comparison ($\gamma_s = 1$, $\gamma_q = 0.1$, $\mu_1 = 1$, $\mu_2 = 0.7$, $\mu_3 = 0.6$, $\mu_4 = 0.5$, $\mu_5 = 0.4$, $\mu_6 = 0.3$, $\mu_7 = 0.2$, and $\mu_x = 0$ for $x > 7$)

			Optimal fixed threshold			Maximal fixed threshold			Optimal state-dependent				
λ	s	u	$P_{a,q}$	$P_{a,s}$	M	$P_{a,q}$	$P_{a,s}$	M	$P_{a,q}$	$P_{a,s}$	M	d_1	d_2
1	1	1	3.866%	48.067%	51.933%	0.000%	54.094%	54.094%	3.829%	48.103%	51.932%	0.001%	2.163%
2	1	2	2.321%	54.799%	57.121%	0.001%	58.073%	58.074%	4.582%	51.379%	55.961%	1.159%	2.112%
3	1	2	7.269%	53.556%	60.825%	0.013%	61.346%	61.358%	2.349%	56.990%	59.339%	1.486%	2.019%
5	5	1	0.243%	49.879%	50.121%	0.000%	50.431%	50.431%	0.243%	49.879%	50.121%	0.000%	0.310%
10	5	1	6.499%	46.751%	53.249%	0.000%	53.437%	53.437%	2.397%	50.322%	52.719%	0.530%	0.718%
15	5	2	1.783%	56.190%	57.973%	0.000%	58.622%	58.622%	0.068%	57.472%	57.540%	0.433%	1.082%
10	10	1	0.035%	49.983%	50.017%	0.000%	50.074%	50.074%	0.035%	49.983%	50.017%	0.000%	0.057%
20	10	1	4.603%	47.699%	52.301%	0.000%	52.446%	52.446%	1.842%	50.090%	51.932%	0.370%	0.514%
30	10	2	0.806%	56.658%	57.464%	0.000%	58.257%	58.257%	0.000%	57.331%	57.331%	0.134%	0.926%
100	100	1	0.000%	50.000%	50.000%	0.000%	50.000%	50.000%	0.000%	50.000%	50.000%	0.000%	0.000%
200	100	1	1.458%	49.271%	50.729%	0.000%	50.778%	50.778%	0.588%	50.027%	50.615%	0.114%	0.164%
300	100	2	0.000%	57.143%	57.143%	0.000%	57.143%	57.143%	0.000%	57.143%	57.143%	0.000%	0.000%

Figure 7 reveal that the optimal policy defined by the function $u(x)$ is characterized by a first important jump of $u(x)$ at $x = 1$ followed by smaller jumps which are more and more spread out as the number of chats in the queue increases. This first jump of $u(x)$ has a major impact on the system performance compared to the other ones. In many cases this first jump corresponds to the optimal service capacity of the fixed threshold policy. The performance of the fixed threshold policy is therefore relatively close to that of the optimal one. This justifies its use in practice provided that the fixed threshold is optimized. We note however that the difference between the two policies is a bit greater for smaller values of s. In large contact centers, the beneficial effect of pooling reduces the additional benefits resulting from clever routing strategies. Finally, as an illustration of Proposition 2, we observe that the optimal threshold and the state-dependent threshold increase as λ increases.

6 The chat dispatching problem in the NWC

We assume here that a chat cannot be handed to another agent during a service and that each agent has its own queue. The problem consists in optimizing the routing of chats upon arrival to parallel queues and in dispatching chats from one queue to the other at other event epochs (service completions or abandonment times). The problem of routing jobs/customers upon arrival to a set of parallel queues to achieve some performance objectives has received a high interest in the research literature (Winston, 1977; Hordijk and Koole, 1992; Whitt, 1986; Koole et al., 1999; Guo et al., 2004; Hordijk and van der Laan, 2004; Anselmi and Gaujal, 2011; Anselmi and Casale, 2013). The motivation is the complexity of the theoretical problem together with its usefulness in practice. The main difficulty in finding the optimal routing policy is the high dimensionality of the system. Here, we propose using the one-step policy improvement method introduced by Ott and Krishnan (1992) and the developed the work by Bhulai and Koole (2003a) to obtain a near-optimal

policy. One-step improvement works for models for which the value function of a certain policy can be computed. It consists in performing the improvement step on the basis of this initial policy. The motivation for considering this method is that policy improvement gives the biggest improvement during the first step (Ott and Krishnan, 1992).

We propose the following approach to derive a dispatching policy in the NWC. In Section 6.1, we derive the explicit expression of the relative value function for a single server chat queue. The approach is explained under the optimal state-dependent policy and explicit expressions of the relative value function and the average cost are given with a fixed threshold policy. Next, in Section 6.2, we provide a one-step improvement algorithm based on the results of Section 6.1 to derive our policy for chat dispatching. Finally, in Section 6.3, we evaluate our improved policy in comparison with the optimal one in the SWC.

6.1 Relative value function for a single chat queue

We derive the relative value function of a single server chat queue under the optimal initiation policy. As proven in Section 5, a state-dependent threshold policy defined by the function $y = u(x)$ is optimal for the chat initiation problem in the SWC. The result also applies to a single server queue in the NWC (there is no question of sharing work in a single server queue). As in Section 5, a state of the system is defined by the couple (x, y), where x is the number of chats in the queue and y is the number of chats in service. Due to Proposition 8.2.1 of Puterman (1994), the long-run average cost is independent of the initial state. The dynamic programming optimality equation which gives the relative value function $V(x, y)$ of the single server chat queue is given by

$$V(x,y) + M = \frac{c_1}{\lambda}x + \frac{c_2}{\lambda}y + \lambda\left[\mathbb{1}_{y>u(x)}V(x+1,y) + \mathbb{1}_{y\leq u(x)}V(x,y+1)\right] + y(\mu_y+\gamma_s)\left[\mathbb{1}_{y>u(x)}V(x,y-1)\right. \tag{6}$$
$$\left. +\mathbb{1}_{y\leq u(x)}V(x-1,y)\right] + \frac{c_3}{\lambda}\gamma_s y + x\gamma_q V(x-1,y) + \frac{c_3}{\lambda}\gamma_q x + (1-\lambda-x\gamma_q - y(\mu_y+\gamma_s))V(x,y),$$

for $x, y \geq 0$, where $\mathbb{1}_{x\in A}$ is the indicator function of a given subset A, with $V(0,0) = 0$ (reference state). Equation (6) corresponds to an infinite set of equations. One solution to solve this system is to truncate the number of states in order to obtain a finite set of equations so as to approximate $V(x, y)$. Instead, we propose using another approach based on the properties of $u(x)$. Since $u(x)$ is an increasing and bounded function of x which takes values in $\mathbb{N}$, there exists $x^*, u \geq 0$ such that $u(x) = u$ for $x \geq x^*$. This means that if the number of chats in the system is higher than or equal to $x^* + u$, then u chats are in service and the remaining ones are in the queue. No other possibility can be encountered. This reduces the dimensionality of the problem. Instead of a 2-dimensional value function, we obtain a 1-dimensional one which can be computed explicitly as a function of $V(x^*, u)$. After this step, a finite set of equations remains which corresponds to the states

with strictly less than $x^* + u$ chats in the system. By solving the set of equations, one can compute $V(x, y)$ and M.

The difficult part of this approach is to explicitly derive the value function and the long run average cost if at least $x^* + u$ chats are in the system. For this purpose, we consider the case $x^* = 0$. The optimal policy is therefore a fixed threshold policy with $u(x) = u$, for $x \geq 0$. This particular case is interesting since the value function can be explicitly obtained at all states of the system. The case $x^* > 0$ can be obtained similarly. However, a remaining finite set of equations depending on $u(x)$ has to be solved to obtain $V(x, y)$. This in turn does not allow us to obtain $V(x, y)$ in closed-form if $x^* > 0$. If $x^* = 0$, the value function, $V(x)$, only depends on the number of chats in the system, x. The dynamic programming optimality equations which give the relative value function $V(x)$ are thus given by

$$V(0) + M = \lambda V(1) + (1 - \lambda)V(0), \tag{7}$$

$$V(x) + M = \frac{c_2}{\lambda}x + \lambda V(x+1) + d_x V(x-1) + x\gamma_s \frac{c_3}{\lambda} + (1 - \lambda - d_x)V(x), \text{ for } 0 < x \leq u - 1, \tag{8}$$

$$\begin{aligned} V(x) + M = \frac{c_1}{\lambda}(x-u) + \frac{c_2}{\lambda}u + \lambda V(x+1) + d_x V(x-1) + (x-u)\gamma_q \frac{c_3}{\lambda} + u\gamma_s \frac{c_3}{\lambda} \\ + (1 - \lambda - d_x)V(x), \text{ for } x \geq u, \end{aligned} \tag{9}$$

where $d_x = \min(x, u)(\gamma_s + \mu_{\min(x,u)}) + (x-u)^+\gamma_q$.

In Theorem 3, we give the closed-form expressions of the relative value function as a function of the polynomial in the variable z, $P_{x,k}(z)$ for $0 \leq k \leq x \leq n$, defined by

$$P_{x,k}(z) = \sum_{n=0}^{x-k} z^n \cdot \left(1 + \sum_{i=k+n+1}^{x} \prod_{j=k+n+1}^{i} t_j\right),$$

where $t_x = \frac{d_x}{\lambda}$. Because of linearity, the value function can be broken down into $V(x) = V_{c_1}(x) + V_{c_2}(x) + V_{c_3}(x)$, which are due to the different components of M. The idea of the proof is to introduce the difference $\Delta(x) = V(x+1) - V(x)$, for $x \geq 0$. The initial value of $\Delta(0)$ can be directly expressed as a function of M. Subtracting the expression of $V(x)$ from the expression of $V(x+1)$ using the optimality equations allows us to derive a linear relation for $\Delta(x)$. Using this relation, we obtain an expression of $\Delta(x)$ as a function of M. Finally, using the reference state, we obtain the unique expression of g which in turn leads to the expression of $V(x)$ via $V(x) = \sum_{k \leq x-1} \Delta(k)$. The proof is given in Section 8 of the online supplement.

Theorem 3 *We have for* $x \geq 0$,

$$V_{c_1}(x) = P_{x-1,0}(1) \cdot \frac{E(W_q)}{\lambda} - P'_{x-1,u}(1) \cdot \frac{c_1}{\lambda^2}, \text{ with } E(W_q) = \frac{c_1}{\lambda} \cdot \frac{\sum_{k=u}^{\infty} (k-u) \prod_{j=1}^{k} a_j}{\sum_{k=0}^{\infty} \prod_{j=1}^{k} a_j}$$

$$V_{c_2}(x) = P_{x-1,0}(1) \cdot \frac{E(S)}{\lambda} - (P'_{x-1,0}(1) - P'_{x-1,u}(1)) \cdot \frac{c_2}{\lambda^2}, \text{ with } E(S) = \frac{c_2}{\lambda} \cdot \frac{\sum_{k=1}^{s} k \prod_{j=1}^{k} a_j + u \sum_{k=u+1}^{\infty} \prod_{j=1}^{k} a_j}{\sum_{k=0}^{\infty} \prod_{j=1}^{k} a_j},$$

$$V_{c_3}(x) = P_{x-1,0}(1) \cdot \frac{P_a}{\lambda} - (P'_{x-1,0}(1) - P'_{x-1,u}(1)) \cdot \frac{c_3\gamma_s}{\lambda^2} - P'_{x-1,u}(1) \cdot \frac{c_3\gamma_q}{\lambda^2},$$

$$\text{with } P_a = \frac{c_3}{\lambda} \left(\gamma_s \cdot \frac{\sum_{k=1}^{s} k \prod_{j=1}^{k} a_j + u \sum_{k=u+1}^{\infty} \prod_{j=1}^{k} a_j}{\sum_{k=0}^{\infty} \prod_{j=1}^{k} a_j} + \gamma_q \cdot \frac{\sum_{k=u+1}^{\infty} (k-u) \prod_{j=1}^{k} a_j}{\sum_{k=0}^{\infty} \prod_{j=1}^{k} a_j} \right),$$

where $a_x = \frac{\lambda}{\min(x,u)(\mu_{\min(x,u)}+\gamma_s)+(x-u)^+\gamma_q}$.

Remark. The approach developed in this section to obtain the relative value function for the chat queue can be easily extended to a more general setting with state-dependent departure rates to find routing solutions for the historical problem of routing to parallel queues. The approach may be useful for a context with abandonment. This contribution is explained in Sections 8 and 9 of the online supplement. In particular, we show that our result meets the ones derived in the work by Bhulai and Koole (2003a) for the M/M/s, M/M/1 or M/M/∞ queues by choosing $u = s$, $u = 1$ or letting u tends to infinity in Theorem 3. In Section 9 of the online supplement, we show how to obtain the explicit relative value function in the M/M/s+M queue. To the best of our knowledge, this article is the first to provide the relative value function for this queue.

6.2 The chat dispatching algorithm

In what follows, we explain how the derivation of the relative value function can be used to obtain an efficient chat dispatching policy in the NWC.

The initial Bernoulli policy. Consider a situation with s identical agents and s parallel queues. As a starting point, we use the so-called Bernoulli policy with fixed probabilities p_1, p_2, $\cdots$, p_{s-1}, and $p_s = 1 - p_1 - p_2 - \cdots p_{s-1}$. This decouples the queues such that each chat queue behaves as a single independent chat queue with a Poisson arrival rate $p_1 \cdot \lambda$, $p_2 \cdot \lambda$, $\cdots$, and $p_s \cdot \lambda$. We can therefore derive the value function and the average cost in each queue using the results of Section 6.1. The average cost and the value function for the whole system is given by the sum of the individual average costs and the sum of the individual value

functions of the different queues. The parameters p_1, p_2, $\cdots$, and p_s are chosen in order to minimize the average cost of the system. With identical agents, the computation of the optimal p_i's leads to $p_i = \frac{\lambda}{s}$. Note that with heterogeneous agents, one could find situations where $p_i \neq \frac{\lambda}{s}$.

One-step policy improvement. Next, we improve this optimal Bernoulli policy from the minimizing action between sending a chat to one of the s chat queues. More precisely, assume that Queue i is in state (x_i, y_i), for $1 \leq i \leq s$. We use $V_i(x_i, y_i)$ to denote the value function at Queue i under the optimal Bernoulli policy. The value function of the set of the s chat queues is $V(x_1, y_1, x_2, y_2, \cdots, x_s, y_s) = V_1(x_1, y_1) + V_2(x_2, y_2) + \cdots + V_s(x_s, y_s)$. The action at customer arrival consists in changing the state of one queue to either one more chat in service or one more chat in the queue. If Queue i is chosen then the value function becomes $V_1(x_1, y_1) + V_2(x_2, y_2) + \cdots + V_{i-1}(x_{i-1}, y_{i-1}) + \min(V_i(x_i+1, y_i), V_i(x_i, y_i+1)) + V_{i+1}(x_{i+1}, y_{i+1}) + \cdots + V_s(x_s, y_s) = V(x_1, y_1, x_2, y_2, \cdots, x_s, y_s) + \min(V_i(x_i+1, y_i), V_i(x_i, y_i+1)) - V_i(x_i, y_i)$. We want to minimize this expression by determining the best chat queue to which to send an incoming chat. At customer arrival, if

$$j = \underset{1 \leq i \leq s}{\operatorname{argmin}} \ (\min(V_i(x_i+1, y_i), V_i(x_i, y_i+1)) - V_i(x_i, y_i)),$$

then it is optimal to send an incoming chat to Queue j. After a service departure or an abandonment, we consider the queue which maximizes $V_i(x_i, y_i)$. Assume that this queue is Queue m. If $x_m = 0$, no action is decided. Otherwise if $x_m > 0$, then we choose to move a chat from Queue m to Queue j if

$$j = \underset{1 \leq i \leq s}{\operatorname{argmin}} \ (\min(V_i(x_i+1, y_i), V_i(x_i, y_i+1)) - V_i(x_i, y_i) + V_m(x_m - 1, y_m) - V_m(x_m, y_m)).$$

If strictly more than one queue minimizes the above expressions, one should apply an equiprobably routing. This step of improvement determines the improved dispatching policy at event epochs. Algorithm 1 summarizes the steps to obtain this policy.

Algorithm 1: Computation of the improved policy.

1. *Situation at an event instant.* Consider an event instant (i.e., a chat arrival or a chat departure due to an abandonment or a service completion). Assume that Queue i is in state (x_i, y_i), for $1 \leq i \leq s$, and $x_i, y_i \geq 0$.

2. *Optimal Bernoulli policy.* Determine the optimal Bernoulli policy and the related value function at Queue i, $V_i(x_i, y_i)$ using the results of Section 6.1.

3. *Improvement step.*

   (a) If the event is an arrival, then send the chat to Queue j such that

   $$j = \underset{1 \leq i \leq s}{\operatorname{argmin}} \ (\min(V_i(x_i+1, y_i), V_i(x_i, y_i+1)) - V_i(x_i, y_i)),$$

(b) If the event is a departure, choose Queue m such that $m = \underset{1\leq i\leq s}{\text{argmax}}\ V_i(x_i, y_i)$. If $x_m = 0$, then no action is decided. If $x_m > 0$ then move a chat from Queue m to Queue j such that

$$j = \underset{1\leq i\leq s}{\text{argmin}}\ (\min(V_i(x_i+1, y_i), V_i(x_i, y_i+1)) - V_i(x_i, y_i) + V_m(x_m - 1, y_m) - V_m(x_m, y_m)).$$

Remark. Our method avoids the curse of dimensionality problem. For instance, with s queues each with capacity n and a fixed service capacity u, the optimal decision should normally be determined in $(u+n)^s$ states. Using the one-step policy improvement method, only $(u+n)\times s$ states have to be computed. We therefore have a linear complexity instead of an exponential one.

6.3 Evaluation of the dispatching policy

We now evaluate the performance of the dispatching policy in the NWC. Table 2 tabulates the performance of the optimal Bernoulli policy, the performance of the dispatching policy in the NWC and the performance of the optimal policy in the SWC which serves as a lower bound for the NWC. As in Section 5.3, we choose $c_1 = c_2 = 0$ and $c_3 = 1$ such that the objective is to minimize the overall proportion of abandonment. We also choose a decreasing and concave service rate per chat and per agent such that the LBF routing rule is optimal. Note in general that as in the SWC in Section 4, the LBF policy is not always optimal. In Section 10 of the online supplement we show an example where MBF is optimal in some states.

In our illustration, it is optimal to fully use the agents' capacity. Theorem 3 can therefore be directly applied with $u = 10$ to obtain the performance of the optimal Bernoulli policy found with an arrival rate per chat queue equal to λ/s. The performance measures of the improved dispatching policy are found using simulation and the result of Theorem 3. After each event (i.e., an arrival or a departure), the simulation is interrupted and a routing decision is taken based on Algorithm 1. The performance of the optimal policy in the SWC is obtained using the results of Section 5 together with the performance measures derived in Section 7 of the online supplement. The last column gives the differences, defined as $d_1 = M(\text{Optimal Bernoulli Policy}) - M(\text{Improved Dispatching Policy})$ and $d_2 = M(\text{Improved Dispatching Policy}) - M(\text{Optimal SWC Policy})$.

Table 2: Performance comparison ($\gamma_s = 0.1$, $\gamma_q = 1$, $\mu_x = 1/\sqrt{x}$ for $1 \leq x \leq 10$, and $\mu_x = 0$ for $x > 10$)

		Optimal Bernoulli policy			Improved dispatching policy			Optimal SWC policy				
λ	s	$P_{a,q}$	$P_{a,s}$	M	$P_{a,q}$	$P_{a,s}$	M	$P_{a,q}$	$P_{a,s}$	M	d_1	d_2
1	2	0.000%	10.704%	10.704%	0.000%	9.981%	9.981%	0.000%	9.734%	9.734%	0.723%	0.247%
5	2	1.089%	18.147%	19.236%	0.213%	18.046%	18.259%	0.105%	17.795%	17.900%	0.977%	0.359%
8	2	11.661%	19.499%	31.160%	9.152%	20.011%	29.163%	7.637%	20.770%	28.407%	1.997%	0.756%
5	10	0.000%	10.704%	10.704%	0.000%	9.468%	9.468%	0.000%	9.105%	9.105%	1.236%	0.363%
25	10	1.089%	18.147%	19.236%	0.000%	17.953%	17.953%	0.000%	17.308%	17.308%	1.283%	0.645%
40	10	11.661%	19.499%	31.160%	4.362%	22.018%	26.380%	2.425%	22.570%	24.996%	4.780%	1.384%
50	100	0.000%	10.704%	10.704%	0.000%	9.346%	9.346%	0.000%	9.091%	9.091%	1.358%	0.255%
250	100	1.089%	18.147%	19.236%	0.000%	17.634%	17.634%	0.000%	17.197%	17.197%	1.602%	0.437%
400	100	11.661%	19.499%	31.160%	1.243%	22.914%	24.157%	0.125%	23.409%	23.533%	7.003%	0.624%

Impact of the improvement step. The improvement step from the optimal Bernoulli policy to the improved dispatching policy, measured by d_1, can be significant (at most 7% difference). The reason is that the improved policy is state-dependent whereas the Bernoulli one is not. The step of improvement reduces the situations where a chat waits in a queue while an agent should serve this chat. This leads to significantly lower values for $P_{a,q}$ under the improved dispatching policy than under the optimal Bernoulli policy. With less abandonment from the queue there should be more congestion and on average more chats per agent in service. The chat service should therefore be slower and more abandonment from service should be observed. However, we observe that less abandonment from the queue does not necessarily mean more abandonment from service. With low arrival rates, we even observe an improvement of $P_{a,s}$ (see lines 1 and 2) whereas a deterioration is observed for higher arrival rates (see line 3). This means that the step of improvement can also have a beneficial effect on $P_{a,s}$. The improvement step creates a better balance in the number of chats in service per agent by sending incoming chats to the least busy agent in priority. This has the effect of increasing the average service rate per chat. Less abandonment from service can thus be observed thanks to the improvement step that one can observe for low arrival rates. Note that the chance of finding an available agent to operate the step of improvement increases with the contact center size. This explains why the impact of the improvement step increases with the system size (lines 3, 6 and 9 for instance).

Comparison with the SWC. The difference between the improved policy in the NWC and the optimal one in the SWC, measured by d_2, is relatively small (+0.56% on average for the different cases). It means that the improved policy has the ability to partly compensate the impossibility of redistributing chats among agents as in the SWC. This shows that the one-step policy improvement method yields *nearly optimal* policies. Our observations are in line with those of Hwang et al. (2000) and Bhulai and Koole (2003a) who have shown the closeness between the improved and the optimal policy in other queueing contexts. As expected, the performance gap between the two policies increases with the arrival rate as the proportion of abandonment increases with λ. The effect of the contact center size is less clear. The larger the contact center, the higher the number of chats distributions among agents. This increases the possibilities of unproductive states where some agents have too many chats while others have too few. This tends to show the negative impact of the contact center size. Note, for instance, that in the extreme case with $s = 1$, the SWC and NWC are identical. However, if the contact center is very large, an incoming chat may always find an available agent. The distribution of chats among agents is therefore balanced with one chat per agent and there is no difference between the SWC and NWC. For instance, with an infinite number of agents the SWC and NWC are identical with $P_{a,s} = \frac{\gamma_s}{\gamma_s+\mu_1} = 9.09\%$ in our case as one can observe in line 7. This may explain why the difference between the two policies is maximized for intermediate contact center size ($s = 10$ in Table 2).

7 Concluding remarks

We showed how to efficiently route chats in a contact center using agent reservation and the possibility of sharing work or not among agents. In the shared work case, we gave the required conditions under which the LBF and MBF routing rules are optimal. We also showed that the optimal policy for agent reservation is a state-dependent threshold policy based on the number of chats in the queue and in service. The numerical investigations argued for the implementation of a simpler fixed threshold policy in large contact centers provided that this threshold is optimized. In the non-shared work case, we developed a policy improvement algorithm to efficiently route chats. This improved policy was shown to partly compensate the negative impact of not sharing work among the agents. Its value is to reduce the abandonment from the queue without necessarily increasing the abandonment from service.

Several interesting areas of future research arise. It would be interesting to question the exponential assumption for service times. Since the chat conversation is a succession of questions and answers, an Erlang or a Coxian distribution may more appropriately model the service duration. Another challenging and interesting direction is to develop and analyze a framework that combines chats and other channels such as calls or emails. The possibility of switching from textual chat to video chat may also be a challenging problem. Video chats do not allow the agent to handle different tasks simultaneously. She may be able to talk to more than one customer at a time but these customers should have the same interests. This is different from the textual chat channel as studied in this article. Optimizing the switch from one channel to another is an interesting future research subject where a good balance has to be found between a simultaneous handling (textual chats) with long service times per chat and successive handling (video chats) with shorter service times per chat.

Acknowledgments

This work was supported by INTERACTIV GROUP via the Chair *Call Centers* at CentraleSupélec. The authors would like to thank Sébastien Thorel from INTERACTIV GROUP for helpful discussions. The authors want also to express their gratitude to the Associate Editor and two anonymous referees for their useful comments that significantly improved this paper.

References

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

Akşin, Z. and Harker, P. (2003). Capacity sizing in the presence of a common shared resource: Dimensioning an inbound call center. *European Journal of Operational Research*, 147(3):464–483.

Aktekin, T. (2014). Call center service process analysis: Bayesian parametric and semi-parametric mixture modeling. *European Journal of Operational Research*, 234(3):709–719.

Altman, E., Ayesta, U., and Prabhu, B. (2011). Load balancing in processor sharing systems. *Telecommunication Systems*, 47(1-2):35–48.

Anselmi, J. and Casale, G. (2013). Heavy-traffic revenue maximization in parallel multiclass queues. *Performance Evaluation*, 70(10):806–821.

Anselmi, J. and Gaujal, B. (2011). The price of forgetting in parallel and non-observable queues. *Performance Evaluation*, 68(12):1291–1311.

Avi-Itzhak, B. and Halfin, S. (1988). Expected response times in a non-symmetric time sharing queue with a limited number of service positions. *Proceedings of ITC*, 12(5.4):1–2.

Avramidis, A., Chan, W., Gendreau, M., L'Ecuyer, P., and Pisacane, O. (2010). Optimizing daily agent scheduling in a multiskill call center. *European Journal of Operational Research*, 200(3):822–832.

Barrow, D. and Kourentzes, N. (2018). The impact of special days in call arrivals forecasting: A neural network approach to modelling special days. *European Journal of Operational Research*, 264(3):967–977.

Bellman, R. E. (1961). *Adaptive control processes: a guided tour*. Princeton university press.

Bernett, H., Fischer, M., and Masi, D. (2002). Blended call center performance analysis. *IT Professional*, 4(2):33–38.

Bhulai, S., Brooms, A., and Spieksma, F. (2014). On structural properties of the value function for an unbounded jump markov process with an application to a processor sharing retrial queue. *Queueing Systems*, 76(4):425–446.

Bhulai, S. and Koole, G. (2003a). On the structure of value functions for threshold policies in queueing models. *Journal of Applied Probability*, 40(3):613–622.

Bhulai, S. and Koole, G. (2003b). A queueing model for call blending in call centers. *IEEE Transactions on Automatic Control*, 48(8):1434–1438.

Chow, Y. et al. (1979). Models for dynamic load balancing in a heterogeneous multiple processor system. *IEEE Transactions on Computers*, 100(5):354–361.

Cui, L. and Tezcan, T. (2016). Approximations for chat service systems using many-server diffusion limits. *Mathematics of Operations Research*, 41(3):775–807.

Deslauriers, A., L'Ecuyer, P., Pichitlamken, J., Ingolfsson, A., and Avramidis, A. (2007). Markov chain models of a telephone call center with call blending. *Computers & operations research*, 34(6):1616–1645.

Fianu, S. and Davis, L. (2018). A Markov decision process model for equitable distribution of supplies under uncertainty. *European Journal of Operational Research*, 264(3):1101–1115.

Gans, N., Koole, G., and Mandelbaum, A. (2003). Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management*, 5(2):73–141.

Gans, N. and Zhou, Y.-P. (2003). A call-routing problem with service-level constraints. *Operations Research*, 51:255–271.

Gromoll, H. C., Robert, P., and Zwart, B. (2008). Fluid limits for processor-sharing queues with impatience. *Mathematics of Operations Research*, 33(2):375–402.

Guo, X., Lu, Y., and Squillante, M. (2004). Optimal probabilistic routing in distributed parallel queues. *SIGMETRICS Performance Evaluation Review*, 32(2):53–54.

Haviv, M. and Van der Wal, J. (2008). Mean sojourn times for phase-type discriminatory processor sharing systems. *European Journal of Operational Research*, 189(2):375–386.

Hordijk, A. and Koole, G. (1992). On the assignment of customers to parallel queues. *Probability in the Engineering and Informational Sciences*, 6(4):495–511.

Hordijk, A. and van der Laan, D. (2004). Periodic routing to parallel queues and billiard sequences. *Mathematical Methods of Operations Research*, 59(2):173–192.

Hwang, R., Kurose, J., and Towsley, D. (2000). MDP routing for multi-rate loss networks. *Computer Networks*, 34(2):241–261.

Ibrahim, R., L'Ecuyer, P., Shen, H., and Thiongane, M. (2016). Inter-dependent, heterogeneous, and time-varying service-time distributions in call centers. *European Journal of Operational Research*, 250(2):480–492.

ICMI (2013). Extreme engagement in the multichannel contact center: Leveraging the emerging channels research Report and best practices guide. ICMI Research Report.

Jean-Marie, A. and Robert, P. (1994). On the transient behavior of the processor sharing queue. *Queueing Systems*, 17(1-2):129–136.

Jouini, O., Pot, A., Koole, G., and Dallery, Y. (2010). Online scheduling policies for multiclass call centers with impatient customers. *European Journal of Operational Research*, 207(1):258–268.

Kim, C., Dudina, O., Dudin, A., and Dudin, S. (2012). Queueing system MAP/M/N as a model of call center with call-back option. *Chapter in Analytical and Stochastic Modeling Techniques and Applications, Series Lecture Notes in Computer Science, Springer Berlin Heidelberg, Editors: Al-Begain, K. and Fiems, D. and Vincent, J.M.*, 7314:1–15.

Koole, G., Sparaggis, P., and Towsley, D. (1999). Minimizing response times and queue lengths in systems of parallel queues. *Journal of Applied Probability*, 36(4):1185–1193.

Legros, B. (2017). Reservation, a tool to reduce the balking effect and the probability of delay. *Operations Research Letters*, 45(6):592–597.

Legros, B. (2018). Waiting time based routing policies to parallel queues with percentiles objectives. *Operations Research Letters*, 46(3):356–361.

Legros, B., Jouini, O., and Koole, G. (2015). Adaptive threshold policies for multi-channel call centers. *IIE Transactions*, 47(4):414–430.

Legros, B., Jouini, O., and Koole, G. (2016). Optimal scheduling in call centers with a callback option. *Performance Evaluation*, 95:1–40.

Legros, B., Jouini, O., and Koole, G. (2018). Blended call center with idling times during the call service. *IISE Transactions*, 50(4):279–297.

Luo, J. and Zhang, J. (2013). Staffing and control of instant messaging contact centers. *Operations Research*, 61(2):328–343.

Ott, T. and Krishnan, K. (1992). Separable routing: A scheme for state-dependent routing of circuit switched telephone traffic. *Annals of operations research*, 35(1):43–68.

Pandelis, D. (2010). Markov decision processes with multidimensional action spaces. *European Journal of Operational Research*, 200(2):625–628.

Pang, G. and Perry, O. (2014). A logarithmic safety staffing rule for contact centers with call blending. *Management Science*, 61(1):73–91.

Pichitlamken, J., A., D., P., L., and Avramidis, A. (2003). Modeling and simulation of a telephone call center. *Proceedings of the 37th Conference on Winter Simulation, New Orleans, LA*, pages 1805–1812.

Puha, A. L., Stolyar, A. L., and Williams, R. J. (2006). The fluid limit of an overloaded processor sharing queue. *Mathematics of Operations Research*, 31(2):316–350.

Puterman, M. (1994). *Markov Decision Processes*. John Wiley and Sons.

Ravner, L., Haviv, M., and Vu, H. (2016). A strategic timing of arrivals to a linear slowdown processor sharing system. *European Journal of Operational Research*, 255(2):496–504.

Tan, T. and Netessine, S. (2014). When does the devil make work? an empirical study of the impact of workload on worker productivity. *Management Science*, 60(6):1574–1593.

Tezcan, T. and Zhang, J. (2014). Routing and staffing in customer service chat systems with impatient customers. *Operations Research*, 62(4):943–956.

Vanlerberghe, J., Walraevens, J., Maertens, T., and Bruneel, H. (2018). Calculation of the performance region of an easy-to-optimize alternative for generalized processor sharing. *European Journal of Operational Research*.

Weisstein, E. (2004). Lagrange interpolating polynomial.

Whitt, W. (1986). Deciding which queue to join: Some counterexamples. *Operations research*, 34(1):55–62.

Winston, W. (1977). Optimality of the shortest line discipline. *Journal of Applied Probability*, 14(1):181–189.

Zhang, J., Dai, J., and Zwart, B. (2009). Law of large number limits of limited processor-sharing queues. *Mathematics of Operations Research*, 34(4):937–970.

Zhuang, W. and Li, M. (2012). Monotone optimal control for a class of Markov decision processes. *European Journal of Operational Research*, 217(2):342–350.