Erlang-S: A Data-Based Model of Servers in Queueing Networks

Journal:	*Management Science*
Manuscript ID	MS-14-01016.R1
Manuscript Type:	Stochastic Models and Simulation
Keywords:	Queues : Applications, Queues : Statistical inference, Forecasting : Time Series

Abstract:

Classical queueing theory has typically focused on customers, while server availability has been taken for granted. However, data accessibility and the emergence of complex service systems, for example call centers, revealed the need to stochastically model the complex behavior of servers. In this work, we propose a new model that accommodates such behavior; we call it Erlang-S, where ``S'' stands for Servers. Our model assumes a pool of present servers, some of whom are available to serve customers from the queue while others are not, and the process of becoming available or unavailable is modeled explicitly. Our focus here is on applying the model to real systems, specifically call centers.
Estimating the parameters of the new model from call center data is challenging since reality is observed discretely in time, as opposed to its continuous evolution. We thus use an EM algorithm that computes the expected number of relevant quantities given the discrete-time data.
Erlang-S differs from the Erlang-A model, which has become standard for modeling call centers: the latter assumes that all agents who are present are in fact available for service.
When comparing predictions of the two models against call center data, we find that Erlang-A generally overestimates queue length and consequently also the fraction of abandonment, while Erlang-S predicts reality more closely and usefully.
Our conclusion is that it is important to model explicitly server dynamics, in order to obtain accurate and valuable models of complex service systems.

Erlang-S: A Data-Based Model of Servers in Queueing Networks

November 15, 2015

Abstract

Classical queueing theory has typically focused on customers, while server availability has been taken for granted. However, data accessibility and the emergence of complex service systems, for example call centers, revealed the need to stochastically model the complex behavior of servers. In this work, we propose a new model that accommodates such behavior; we call it Erlang-S, where "S" stands for Servers. Our model assumes a pool of present servers, some of whom are available to serve customers from the queue while others are not, and the process of becoming available or unavailable is modeled explicitly. Our focus here is on applying the model to real systems, specifically call centers. Estimating the parameters of the new model from call center data is challenging since reality is observed discretely in time, as opposed to its continuous evolution. We thus use an EM algorithm that computes the expected number of relevant quantities given the discrete-time data. Erlang-S differs from the Erlang-A model, which has been commonly used for modeling call centers: the latter model assumes that all agents who are present are in fact available for service. When comparing predictions of the two models against call center data, we find that Erlang-A generally overestimates queue length and consequently also the fraction of abandonment, while Erlang-S predicts reality more closely and usefully. Our conclusion is that it is important to model explicitly server dynamics, in order to obtain accurate and valuable models of complex service systems.

1 Introduction

Operational models of service systems have been traditionally customer-centric. A case in point is Queueing Theory, which has been mainly a theory of customers: indeed, its "Queueing" refers to customers that queue up for service. The emergence of service systems with 100's of servers, and the accessibility to their data, has shifted some attention to the servers. A case in point here are many-server queues that are Quality- and Efficiency-Driven (QED), the study of which was inspired by telephone call centers: Quality indicates that customers enjoy short waits for servers, and Efficiency corresponds to servers who do not wait long for customers.

The symmetry between the operational roles of customers and servers, as advocated in QED systems, is the theme of Momcilovic et al. (2015). We demonstrate it in Figure 1[1], which is based on call center data. It depicts the topology of "who-can-be-served-by-whom" (customer-queues in the ellipsoids) or, equivalently, "who-can-serve-whom" (server-queues in the rectangles). In the present paper, we focus on the servers' perspective, which is illustrated by the three marked sub-networks in Figure 1:

- I-topology: A single pool of servers is dedicated to a single pool of customers. This is the most prevalent topology of queueing models, for example Erlang A,B,C (Gans et al., 2003).
- N-topology: The right server-pool caters to two types of customers, and the left customer type can be served by the two server pools.
- The left half of Figure 1 is a General complex connected component (G-topology), within which we marked an I-like sub-network. The latter is the focus of the present paper: a pool of servers catering to a primary queue, who occasionally become unavailable to serve customers in that queue due to a multitude of reasons. The rest of the call center, beyond that primary queue, will be modeled by a state ("black-box") which agents can visit and then return from. (This is similar to Erlang-R, in Yom-Tov and Mandelbaum (2014), where the focus is on doctors or nurses in an emergency department; here the "black-box" state represents the rest of the emergency department, where patients spend time between successive visits to providers.)

The I-,N- and G- topologies are of increasing complexity. We shall analyze two data sets corresponding to I and G. The I-like sub-network within the G-topology is of special importance since,

[1]Figures 1, 2 and 3 were created by SEEGraph, a structure-mining tool at the Technion SEELab, http://ie.technion.ac.il/Labs/Serveng/. Their underlying data can in fact be animated for further insight: see https://www.youtube.com/watch?v=1A6-jzS_scI, where Figures 1-3 appear after 1:42 minutes. (Animation of the overall flow of customers in a call center starts in minute 0:41.) In addition, each figure has its own animation, accessible via the link below the figure. (Note that it takes 5 to 30 seconds for the quality of the animations to improve and stabilize.) Some of this data is open to access and analysis in http://seeserver.iem.technion.ac.il/databases. Appendix E provides an explanation of the dynamics that animates the data.

when properly used, it has the potential to serve as a model of any I-like construct, within a general queueing network.

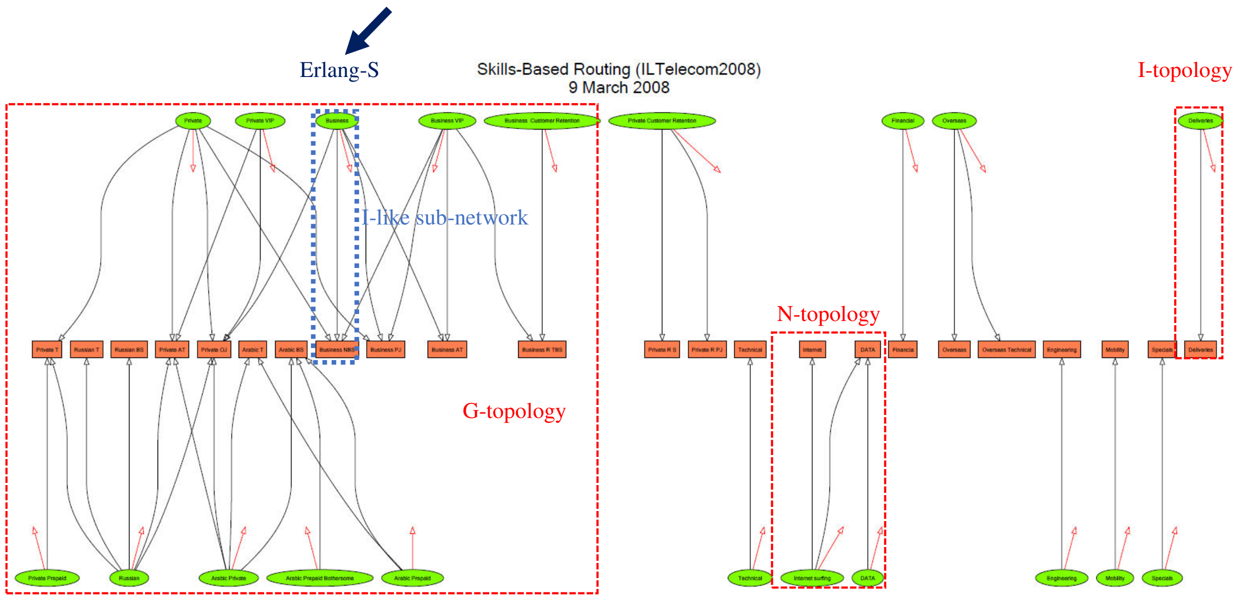


Figure 1: Topology of a call center. Server-queues are in the rectangles and customer-queues are in the ovals. (An animation of the data, underlying the figure, is accessible at http://youtu.be/_eyAXVXZU7o.)

Figure 2 is a daily-activity summary of a server pool, taken from Senderovich (2014). It acknowledges the states of agents who were present on that day: *ready* to serve (idle); *online* (busy) - serving various types of customers; and *unavailable* for service - due to various activities or engagements (wrap-up, breaks, meetings, . . .). Figure 3 is created from Figure 2 via aggregating the type-specific cycles of "online–wrap-up" into a single such cycle. Now, one clearly observes that servers alternate among three states: ready, online and unavailable (the latter aggregates wrap-up, breaks, meetings and back-office work). The model that we develop here is at this resolution of Figure 3. We refer to it as Erlang-S, with "S" standing for *Servers*.

1.1 Erlang-S: a simple server network

As discussed in Section 2 below, our motivation comes from data. Data tells us that servers change their availability status, rather than always stay available as call center models typically assume. The creation of the model, that is the way servers become available or unavailable, is again based on data. We now elaborate on this dynamics.

According to Erlang-S, the number of servers that are *present* in the system is fixed over time.

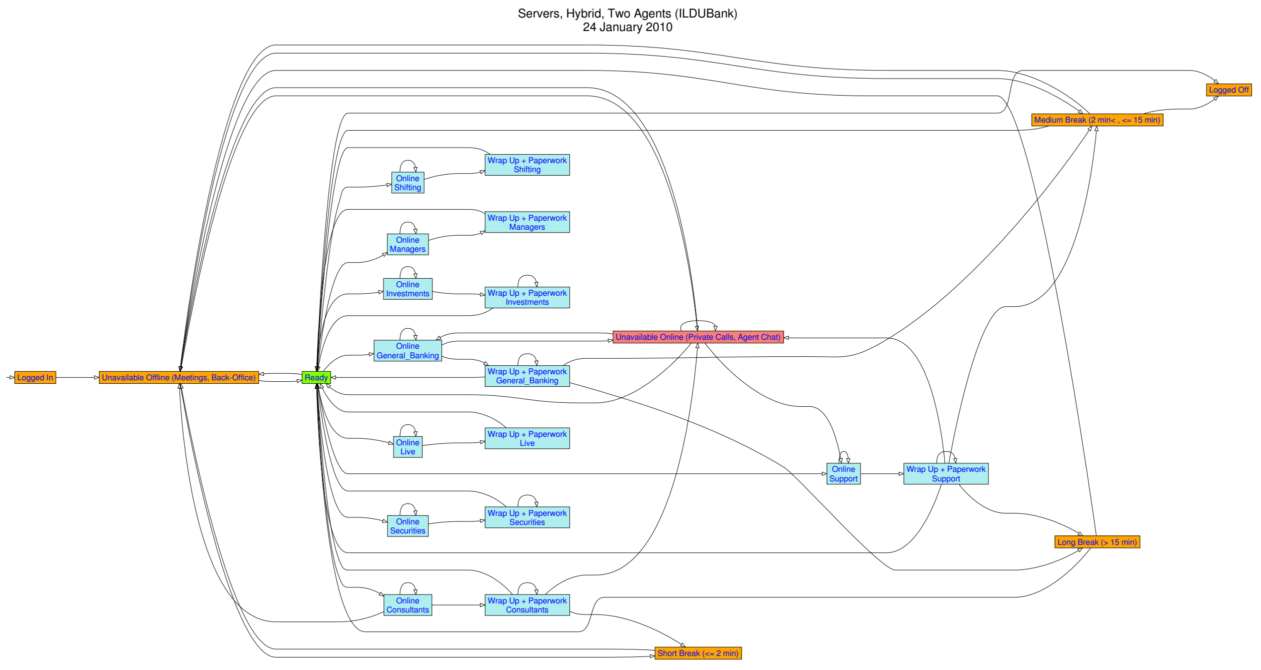

Servers, Hybrid, Two Agents (ILDUBank)
24 January 2010
Logged In
Unavailable Offline (Meetings, Back-Office)
Ready
Online Shifting
Wrap Up + Paperwork Shifting
Online Managers
Wrap Up + Paperwork Managers
Online Investments
Wrap Up + Paperwork Investments
Online General_Banking
Wrap Up + Paperwork General_Banking
Unavailable Online (Private Calls, Agent Chat)
Online Live
Wrap Up + Paperwork Live
Online Securities
Wrap Up + Paperwork Securities
Online Consultants
Wrap Up + Paperwork Consultants
Online Support
Wrap Up + Paperwork Support
Logged Off
Medium Break (2 min< , <= 15 min)
Long Break (> 15 min)
Short Break (<= 2 min)

Figure 2: Paths of agents in a call center, focusing on ready → online → wrap-up, separately for each customer-type. The light-blue rectangles indicate online service and wrap-up. The orange ones correspond to breaks. (The underlying data is animated in http://youtu.be/DHvObpKjYrQ.)

However, a server can either be *available* or *unavailable*, which does vary over time. Available servers serve customers from the queue, or they are ready to serve an arrival if the queue is empty (they cannot become idle if there are queuing customers); unavailable servers do not serve customers. Servers change their availability status in the following three ways:

1. Upon end of service, the (available) server can remain available or become unavailable.

2. Upon arrival of a customer, one of the unavailable servers can become available (and hence busy).

3. At a certain rate, an unavailable server can spontaneously become available.

The above three options were all observed in our data, and each can be given realistic interpretations. As already noted, a server that ends a service can become unavailable (Option 1) for its queue due to turning to serve a customer from another queue (Figure 1), or doing wrap-up work, going on a break or to a meeting (Figure 3). An arrival can change an unavailable server to being available (Option 2) if, for example, it is a VIP arrival. Finally, a server could spontaneously become available (Option 3) after returning from a break.

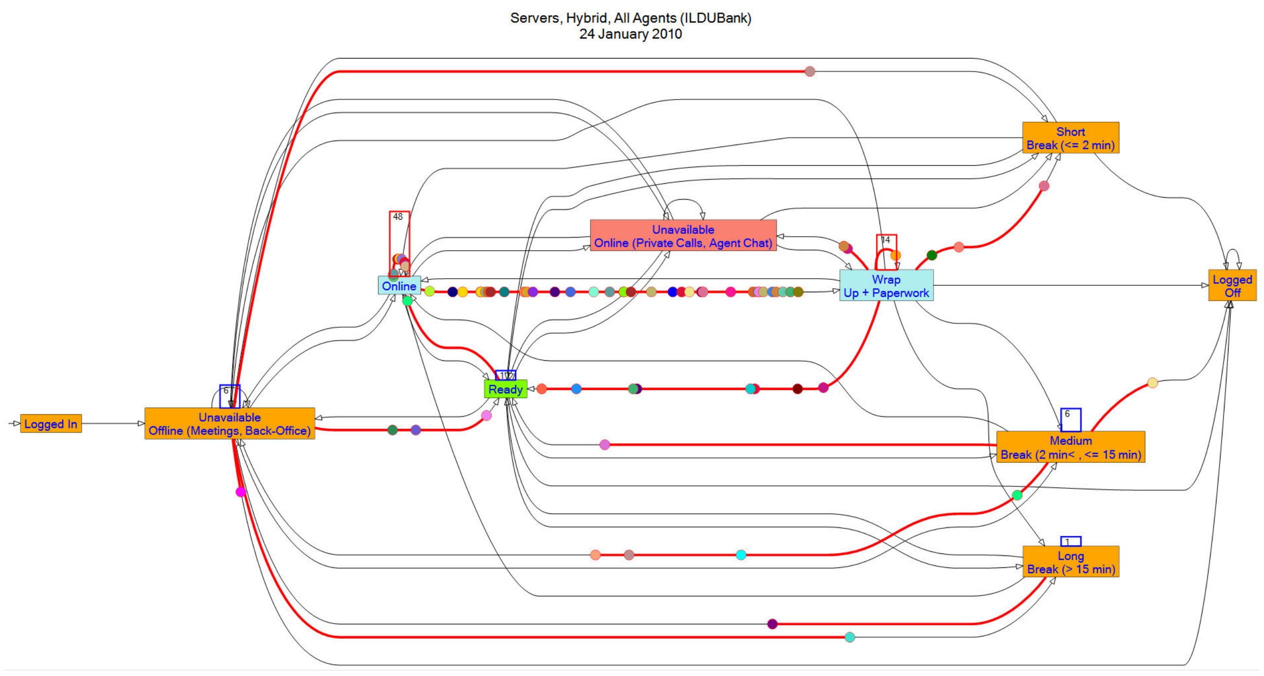

Servers, Hybrid, All Agents (ILDUBank)
24 January 2010
Short Break (<= 2 min)
48
Unavailable Online (Private Calls, Agent Chat)
14
Online
Wrap Up + Paperwork
Logged Off
6
Logged In
Unavailable Offline (Meetings, Back-Office)
12
Ready
6
Medium Break (2 min< , <= 15 min)
1
Long Break (> 15 min)

Figure 3: Snapshot of paths of agents in a call center, where states are aggregated into: ready, online, wrap-up, on break, and other miscellaneous unavailable states (in meetings, on private calls, ...). Each colored circle traces the activities of a specific agent. For technical reasons, the number of colors that is available for display is bounded by 50; hence, some customers in the graph may have the same colors. (The underlying data is animated in http://youtu.be/H-wMFSl95KU.)

Another phenomenon that we observed in our data is that servers tend to become more available as the system becomes increasingly loaded. Such phenomena are captured by the parameters of Erlang-S, that measure the "availability" of servers. However, estimating these parameters is challenging. Indeed, call center data typically has a time resolution of seconds and therefore already for a moderate-size call center, multiple events can occur simultaneously (during a single second [2]). It follows that the underlying process is not observed fully and a methodology that deals with this is required. Here we use the EM algorithm of Bladt and Sørensen (2005) for inference in the framework of discretely observed Markov jump processes (see Section 3.1). A formal description of the Erlang-S model is given in Section 2.2.

[2] A vivid demonstration appears in the following video http://youtu.be/lU3ZKy4Dnlg, which records the evolution of a US Bank call center in 1-second steps. The data from this call center is publicly available in http://seeserver.iem.technion.ac.il/databases

1.2 Literature review

In recent decades, call centers have become an important part of service systems; the number of call centers in the United States alone was estimated in 2008 to be over 47,000, employing 2.7 million agents (Aksin et al., 2007). Call centers are generally complex systems that have attracted ample research; see e.g., the review papers of Gans et al. (2003) and Aksin et al. (2007). Brown et al. (2005) describe an empirical analysis of a call center which gives rise to statistical analysis of its arrivals, services, and patience of customers.

We now describe several studies that focus on servers. Gans et al. (2010) study call centers with multiple types of agents, each with its own service time distributions. The heterogeneity of agents is caused in part by the agent learning-curves; that is, experienced agents behave differently from beginners, which has operational consequences if ignored. Ward and Armony (2013) introduce a routing policy that assigns customers to servers so as to minimize customer holding costs subject to fairness (fair allocation of idleness among servers); along the way, they establish asymptotically ASTA and Little's law from the view-point of servers.

Kc and Terwiesch (2009) investigate health care systems and demonstrate empirically that service rate is accelerated as load increases. This phenomenon is modeled in Delasay et al. (2013), who enrich Erlang-C with the feature that service rate is state-dependent, such that it increases with system overload; their model differs from ours in that servers are always available and abandonment is not acknowledged. Adaptation of service-rate is consistent with our findings in call centers, which show that agents tend to become more available as system load increases; see Section 3.2.

Sun et al. (2007) study wireless networks where the capacity changes stochastically with time and system load. They demonstrate that this stochastic variation can have a major impact on different performance measures such as call blocking probability and queueing delay.

It will help to put Erlang-S in perspective, relative to some existing Erlang models. Erlang-C (M/M/n) is a simple queueing model that ignores caller abandonment and thus it is typically inappropriate for modeling call centers. On the other hand, the Erlang-A model (M/M/n +M), which can be traced back to Palm (1957), accounts for customers' impatience while waiting. Brown et al. (2005) argue that "using Erlang-A for capacity-planning purposes could and should improve operational performance. Indeed, the model is already beyond typical current practice (which is Erlang-C dominated), and one aim of this article is to help change this state of affairs". A survey of Erlang-A and its application to call centers is given in Mandelbaum and Zeltyn (2003).

The Erlang-R model was introduced by Yom-Tov and Mandelbaum (2014), with "R" standing for Reentrant customers. It was motivated by healthcare systems, in which patients often go through a repetitive service process, alternating between being available and unavailable for service. The Erlang-S model, presented here, can be thought of as Erlang-A in which servers enjoy Erlang-R features: they leave the system (become unavailable) and then return.

Our model is, in fact, a vacation model of servers (Tian and Zhang, 2005), but it differs from these models in that it describes how servers go on vacation and come back, based on data. Also, previous similar models have lacked impatient customers, and their parameter estimation has not been of concern. The closest model to ours is in Takagi and Taguchi (2014), where servers have after-call work, thus becoming unavailable after each service completion. The latter model is a special case of ours, in that Erlang-S is richer in options for when servers become available or unavailable.

1.3 Contributions and contents

Data tells us that agents change their tendency to be available according to system needs. Consequently, considering only the average number of available agents, as done for example when fitting Erlang-A to data, leads to overestimation of the queue length and the abandonment rate. Other versions of Erlang-A, which model servers' unavailability by prolonging or adding an extra service time, could work well for simple sub-networks (I-topology) but they do not for more complicated ones (e.g., I-sub-network within G-topology). Overall, when comparing predictions of the two models against real data, Erlang-S predicts reality more closely and usefully.

Moreover, queueing models typically require, as an input parameter, the number of agents N that are present. However, data on N originates in server-logs, which are often separated from customer-logs, and typically the latter are the focus of attention. It follows that the value of N must often be estimated, and here Erlang-S provides a useful resource. Specifically, in Section 4 we successfully estimate N, via an EM algorithm that is based on the Erlang-S model.

To sum-up, our main contributions are as follows:

- We propose and use a new service model, Erlang-S: it is developed from data, in order to capture the behavior of servers who change their availability status stochastically.

- Model parameters are estimated (at various levels of granularity) via EM algorithms, and the model is fitted using call center data.

- The model can fit a complex queueing system, specifically (sub) networks of parallel multiclass customers and multi-skill servers, with a general topology of class-skill matching. This is done by modeling the rest of the network system, beyond the queue of primary interest, as a node which agents visit and from which they return.

- In general, our model is much more accurate and versatile than Erlang-A, which has become the most prevalent model for queues with impatient customers. (We also have anecdotal evidence that Erlang-A is replacing Erlang-C as the (call center) industry standard.)
- Nevertheless, Erlang-A is still found useful for isolated I-topology (as apposed to I- within a G-topology) if it parameters are appropriately tuned.
- Estimated parameters reveal interesting phenomena. For example, the rate at which agents become spontaneously available increases with queue length.
- Several parametric forms of the availability functions are compared. We find that, for the I-topology, a low-dimensional form works well; but for the more complex case of G-topology, a higher dimensional form is truly needed.
- We use Erlang-S to estimate the number of agents present in the system, a quantity that is often needed for modeling but is beyond readily available data.

The rest of the paper is organized as follows: Section 2 starts with a brief description of the Erlang-A model. Through call center data, we demonstrate that the assumption on the number of available agents being constant, is violated even over short time periods (seconds). This leads one to formally introduce the Erlang-S model and its parameters. Section 3 discusses estimation of parameters and the use of Erlang-S for prediction of the abandonment fraction. We then show, via real data, that Erlang-S predicts abandonments better than Erlang-A. Section 4 addresses the problem of estimating the number of agents present, when it is unobservable. Call center topology can be complex, as in Figure 1; this motivates the application of Erlang-S, in Section 5, to a general queue structure. We conclude with remarks and comments on future research in Section 7. Finally, Appendices A-E include material that is not directly required for reading continuity.

2 Erlang-S (vs. Erlang-A)

In this section we introduce the Erlang-S model. It is compared with Erlang-A, which is the prevalent call center model (and is a special case of Erlang-S).

2.1 Erlang-A

Figure 4 shows a schematic representation of Erlang-A, or $M/M/n+M$ in Kendall notation, the "language" of birth-and-death queues (Baccelli et al., 1984). Arrivals are assumed to follow a Poisson process with rate λ; they immediately enter the queue or get served, depending on whether all n servers are busy or there exists an idle agent, respectively. Each customer abandons the queue at rate θ and the individual service rate is μ. Let $x(t)$ denote the number of customers at time t in the system, either in service or in queue. According to Erlang-A, $x(\cdot)$ is a birth-and-death process with death rates

$$\mu_A(x) := \begin{cases} \mu x & x \le n \\ \mu n + (x-n)\theta & x > n \end{cases},$$

and constant birth rates λ.

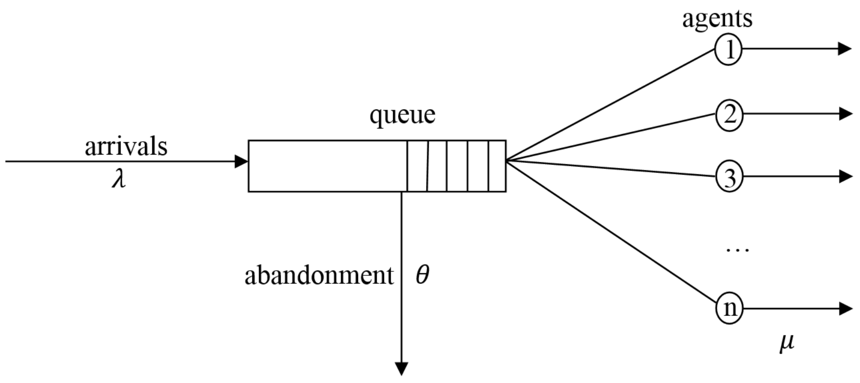


Figure 4: Schematic representation of the Erlang-A model.

Let $q(t) = \max\{x(t)-n, 0\}$ denote the number of customers in the queue at time t. Note that, for every t, if $q(t) > 0$ then $x(t) - q(t) = n$; this implies what is referred to as "work conservation"; no idle servers if there are customers awaiting service. Thus, Erlang-A assumes that:

1. Work conservation: All the n servers present are available in the sense that they either serve customers or are ready to serve (the latter implies that there are no customers in the queue).

2. The number of available agents is constant in time.

These assumptions are frequently violated in the call center data sets that we analyzed. Specifically, while changes in the number of available agents over moderate (hours) to long (shifts) periods are expected, and could be accommodated by Erlang-A, here we find that such changes occur over short periods (seconds) of time: these changes, therefore, must be captured stochastically, as we do in

Erlang-S. We present an example where Assumption 2 is violated; this implies that Assumption 1 is also violated, even when the number of agents present is constant.

The example is shown in Figure 5, displaying a specific half-hour in the telesales part of a U.S. bank. This part is isolated from the rest of the call center and therefore can be treated separately (I-topology). At times $t =$10:36:54, 10:43:07, 10:47:23, we have $x(t) - q(t) = (44, 58, 61)$ respectively (and $q(t) > 0$). That is, triggered by arrivals of many customers, and within approximately 11 minutes, 17 agents became available and then immediately busy serving customers. During this time frame, demand and capacity are almost balanced: q is usually 0 and increases up to 1 or 2 over short periods. From 10:47 on, the number of agents does not grow any more and customers who arrive to the system join the queue. Consequently, the average waiting-time and abandonment fraction increase dramatically at this time, as clearly seen in (b) and (c) of Figure 5. Figure 5 (d) focuses on one minute, during which $q > 0$ (except for the first few seconds). Over this minute, while q increased from 1 to 10, x increased by more – from 58 to 70. This implies that 3 agents became available during that minute.

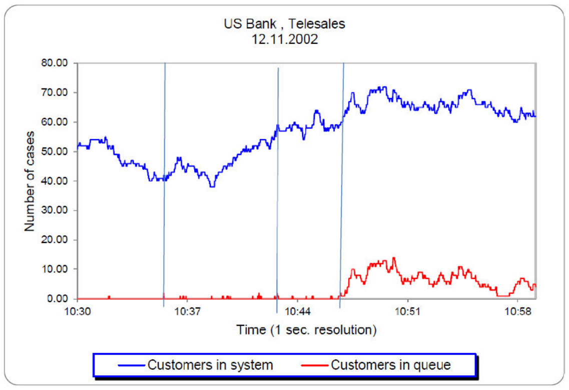
A second example of violating the constant-agents assumption is given in Appendix A. This example is taken from the data set that is extensively analyzed in Section 3.2 below. In that data set, the percentage of minutes where the constant-agents assumption is violated is 67.7%. When considering 5 and 10 minute intervals, the percentage is 98.0% and 99.9%, respectively. We conclude that the constant-agents assumption is frequently violated in the data sets we analyzed, even over short time periods.

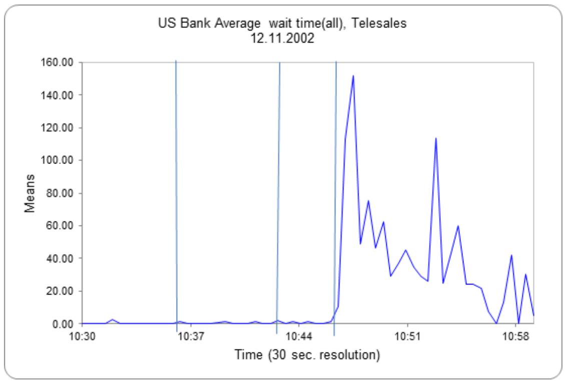
2.2 Erlang-S

The violation of the two assumptions, namely constant number of available agents and work conservation, calls for modifying Erlang-A so that the number of available agents may vary in time. This gives rise to Erlang-S.

Figure 6 shows a schematic representation of Erlang-S. As in Erlang-A, arrivals are assumed Poisson with rate λ and they enter the queue or get served; each customer abandons the queue at rate θ and service rate is μ. In Erlang-S, the agents present may be available or unavailable: the latter are present in the system but they are not serving customers from the queue. The number of agents present, denoted by N, is assumed constant, but whether agents are available or not is time varying.

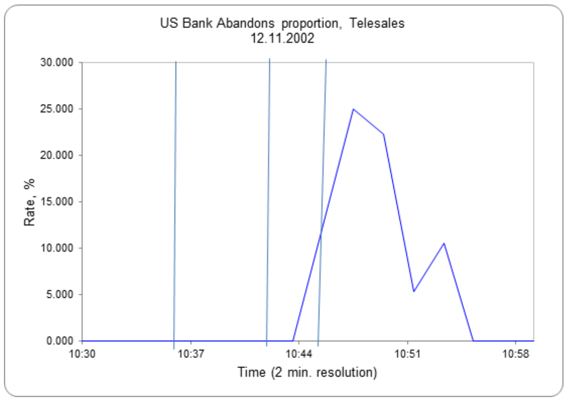
Agents change their availability status according to the following dynamics. Upon service com-

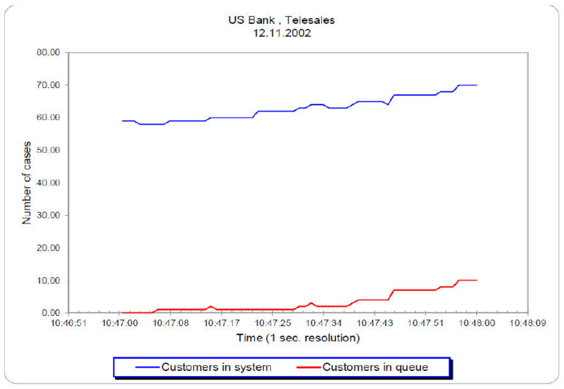




(a)

(b)





(c)

(d)

Figure 5: (a) Illustration of $x(t), q(t)$ (blue, red) in data from a U.S. Bank, the telesales part. At times $t=$10:36:54, 10:43:07, 10:47:23 , $x(t)-q(t)=(44,58,61)$ respectively (and $q(t)>0$). (b) Waiting time, averaged over 30-second intervals. (c) abandonment fraction, accumulated over 2-minute intervals. (d) A specific one minute (10:47) from the data of Figure (a).

pletion, the customer leaves the system and the agent stays available with probability p_2 or becomes unavailable with probability $1-p_2$. An unavailable agent can become available in one of two ways: either at rate ξ (spontaneous availability), or when a customer arrives at the queue and there is no available idle agent; in the latter case, the server immediately becomes available and serves the new customer (even if there are other customers in the queue), which occurs with probability p_1. The parameters p_1, p_2 and ξ are all state-dependent.

Let $n(t)$ denote the number of available agents at time t. Recall that $x(t)$ and $q(t)$ are the overall number of customers and the queue-length at time t, respectively. The following relations

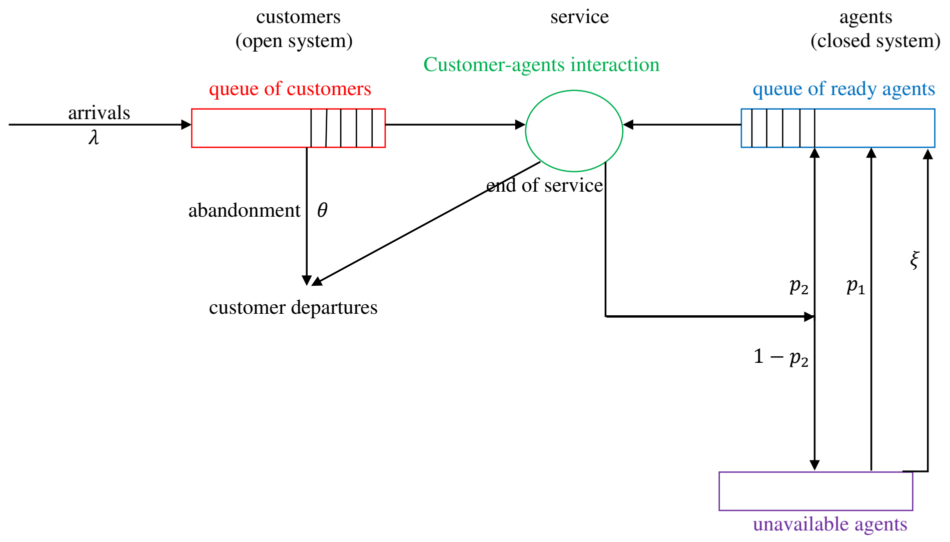


Figure 6: Schematic representation of the Erlang-S model.

hold both in Erlang-S and in Erlang-A (but in Erlang-A, $n(t) \equiv n$ is constant):

$$\begin{aligned}
&\text{Number of customers being served} = \text{number of busy agents} = \min\{x(t), n(t)\} = x(t) - q(t);\\
&\text{Number of agents that are ready to serve} = \max\{n(t) - x(t), 0\};\\
&\text{Number of customers in the queue} = q(t) = \max\{x(t) - n(t), 0\}. \qquad (1)
\end{aligned}$$

For Erlang-S we also have:

$$\text{when } q > 0, \text{ number of unavailable agents} = N - (x - q).$$

Erlang-S could be defined as a two-dimensional model, where a state is the pair (x, n). However, we found, for reasons we now discuss, that the pair (x, q) is in fact more convenient to work with.

We first observe that, when $q > 0$, then $n = x - q$ and hence, in this case, (x, q) and (x, n) are equivalent. However, when $q = 0$, the pair (x, q) does not determine n, as implied by (1): the number of busy agents is equal to x, but the number of ready agents is undetermined, and can take any value $0, 1, \ldots, N - x$.

One reason that a model with (x, q) is preferred is that the former is easier to implement. Data on n is often ambiguous, as is the case in our data sets. The main difficulty is to determine if an agent is ready to serve a newly arrived customer when $q = 0$. According to our terminology, a ready agent is one that will serve a customer immediately upon arrival; however, such a status generally does not exist in call center data. The "ready" agents in call centers serve arriving customers according to their importance, but they do not necessarily serve any customer upon arrival as we assume. It follows that, when $q = 0$, we actually have no data on the number of ready agents. Another related reason to prefer (x, q) is that it is simpler. Such a model does not require knowledge about ready agents. Thus, it need not account for agents that change their status from ready to unavailable when $q = 0$, while the full model on (x, n) does.

A state of Erlang-S is thus the pair (x, q). Figure 7 illustrates the feasible transitions and the parameters of the model: at state (x, q), for $q > 0$, five transitions are possible, as detailed in Figure 7. Given that a customer arrives, which occurs at rate λ, a customer (either the one that arrived or a customer from the queue) is served with probability $p_1(x, q; N)$, otherwise the queue increases by one. When an end of service occurs, at rate $\mu \cdot (x - q)$, the next customer is served with probability $p_2(x, q; N)$, or otherwise stays in the queue. An abandonment occurs at rate $q\theta$ and a "spontaneous service" occurs at rate $\xi(x, q; N)$.

Summing up, the transition rates from (x, q) to (x', q') are

$$\text{if } q > 0 \begin{cases} \lambda[1 - p_1(x, q; N)] & (x', q') = (x + 1, q + 1) \quad \text{(arrival, enter queue)} \\ \lambda p_1(x, q; N) & (x', q') = (x + 1, q) \quad \text{(arrival, enter service)} \\ \mu(x - q)p_2(x, q; N) + \theta q & (x', q') = (x - 1, q - 1) \quad \text{(end of service, service for the next customer)} \\ \mu(x - q)[1 - p_2(x, q; N)] & (x', q') = (x - 1, q) \quad \text{(end of service, no service for the next customer)} \\ \xi(x, q; N) & (x', q') = (x, q - 1) \quad \text{(spontaneous service)} \end{cases}, \tag{2}$$

$$\text{if } q = 0 \begin{cases} \lambda[1 - p_1(x, 0, N)] & (x', q') = (x + 1, 1) \quad \text{(arrival, enter queue)} \\ \lambda p_1(x, 0, N) & (x', q') = (x + 1, 0) \quad \text{(arrival, enter service)} \\ \mu x & (x', q') = (x - 1, 0) \quad \text{(end of service)} \end{cases}. \tag{3}$$

Since the number of customers being served cannot exceed N, if $x - q \geq N$ then $p_1(x, q; N) = \xi(x, q; N) = 0$. Table 1 conveniently summarizes the parameters and relevant quantities of Erlang-S.

2.3 Properties of Erlang-S

The Erlang-S model can be considered as a generalization of Erlang-A to a two-dimensional state space: it has two transitions that represent an arrival, and the sum of their rates is λ, the arrival rate in Erlang-A; similarly, at state (x, q), there are two types of "deaths" and the sum of their

Table 1: Parameters, states, and status of agents for Erlang-S.

λ	arrival rate	x	overall number of customers
μ	service rate	q	number of customers in queue
θ	abandonment rate (individual)	available agents =	$\begin{cases} x-q & \text{if } q>0 \\ \text{any of } x, x+1, \dots, N & \text{if } q=0 \end{cases}$
N	Number of agents present	unavailable agents =	$\begin{cases} N-(x-q) & \text{if } q>0 \\ \text{any of } 0, 1, \dots, N-x & \text{if } q=0 \end{cases}$
		ready agents =	$\begin{cases} 0 & \text{if } q>0 \\ \text{any of } 0, 1, \dots, N-x & \text{if } q=0 \end{cases}$
		busy agents =	$x-q$

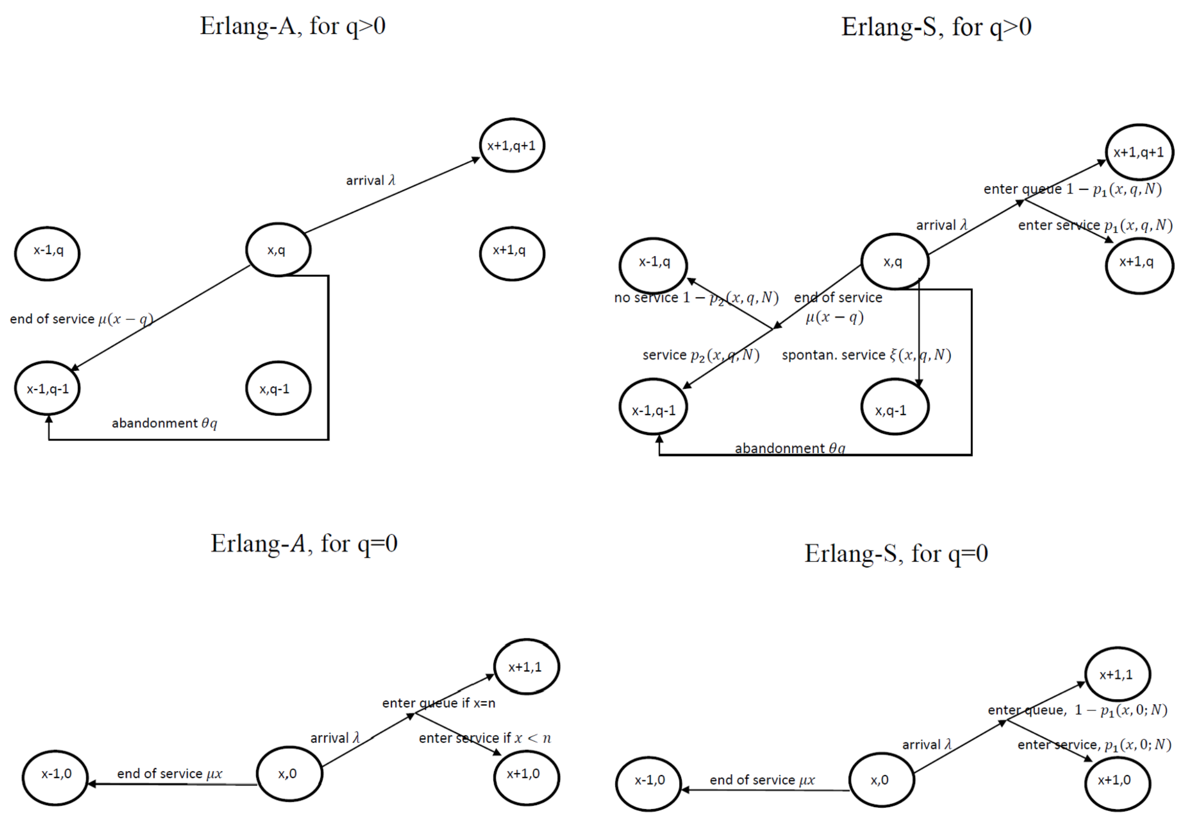


Figure 7: Possible transitions and rates at state (x, q), for Erlang-S (right) and Erlang-A (left). The upper (lower) figure shows the case $q > 0$ ($q = 0$).

rates is $\mu \cdot (x - q) + \theta q$, as in Erlang-A. Due to the two-dimensional state space, there is another possible transition $(x, q) \to (x, q - 1)$, which is not possible in Erlang-A. (Notice also that Erlang-S reduces to Erlang-A when $\xi(x, q; N) = 0$, $p_2(x, q; N) = 1$, and $p_1(x, q; N) = \begin{cases} 1 & x < n, q = 0 \\ 0 & \text{otherwise} \end{cases}$, for $n \leq N$.)

Erlang-S always has a stationary distribution, as stated in the following proposition and proved in Appendix B.

Proposition 1. *If $\theta > 0$, Erlang-S is ergodic for all parameter values.*

Generally, the projection of Erlang-S on x is a Markovian process only with respect to the full filtration of Erlang-S. In the special case where $\mu = \theta$, the projection is a Markov process in itself, as stated in the following proposition. The proof is again given in Appendix B.

Proposition 2. *If $\theta = \mu$ then the projection of Erlang-S on x is a Markov process, specifically the $M/M/\infty$ queue.*

3 Using Erlang-S to predict abandonment fraction

In this section, we introduce the estimation procedure for the availability functions of Erlang-S. Specifically, we describe how to estimate (p_1, p_2, ξ), from call center data. To this end, we consider two parametric forms of these functions. The first form is a "high-dimensional" function, which depends on relatively many parameters, while the second form is simpler. The main difficulty in estimating the parameters is that the call center is observed at discrete times, specifically seconds. Since several events can occur during a second, the process is not observed fully (recall the footnote on Page 5.) To overcome this problem, we use the EM algorithm of Bladt and Sørensen (2005) to estimate the parameters of Erlang-S.

3.1 The EM algorithm

Let $\{x(t)\}_{0 \leq t \leq T}$ be a Markov process on the state space $\{1, \ldots, s\}$, and let Q be its $s \times s$ infinitesimal generator matrix. If the process was observed continuously from 0 to T, then the maximum likelihood estimate (hereafter MLE) of Q_{ij} would be $\widehat{Q_{ij}} := \frac{N_{ij}}{R_i}$, where N_{ij} is the number of transitions $i \to j$, and R_i is the sojourn time in state i. However, when the process is observed only in discrete times, say $\{1, 2, \ldots, T\}$, then N_{ij} and R_i are not observable. To overcome this problem, Bladt and Sørensen (2005) propose the following EM algorithm:

1. Set $\tilde{Q}$ to be any generating matrix.

2. **E step:** Compute

$$\tilde{N}_{ij} := E_{\tilde{Q}}\left[\text{number of transitions } i \to j | \{x(t)\}_{t=1}^{T}\right], \tag{4}$$

$$\tilde{R}_i := E_{\tilde{Q}}\left[\text{sojourn time in } i | \{x(t)\}_{t=1}^{T}\right]. \tag{5}$$

3. **M step:** Set $\tilde{Q}_{ij} = \frac{\tilde{N}_{ij}}{\tilde{R}_i}$, check for some stopping criterion to apply, stop if it applies or return to Step 2 if it does not.

Specific stopping criteria will be formally described for each subsequent application of the EM algorithm. The heart of the algorithm is the computation of the conditional expectation in (4) and (5). Bladt and Sørensen (2005) provide explicit formulas for these expectations, which are given in Appendix D.

The EM algorithm is useful when the model depends on unobserved latent variables. Here, the latent variables are N_{ij}, R_i, which are functions of the unobserved continuous process. Its discrete observation are used to estimate its unobserved states and the parameters of the model.

The authors also prove that the algorithm converges to a stationary point of the likelihood function. Below we implement this algorithm for Q's of a certain structure, namely, our parameter space is $\mathcal{Q}_{\mathcal{I}} := \{Q | Q_{ij} = 0 \text{ for } i, j \in \mathcal{I}\}$, where $\mathcal{I}$ is some set of indices. This differs from the parameter space in Bladt and Sørensen (2005), where $Q_{ij} \neq 0$, for every $i \neq j$. However, their theoretical result continues to hold in our reduced parameter space, since we assume that the real Q belongs to the interior of $\mathcal{Q}_{\mathcal{I}}$, and therefore no boundary problems arise. To maximize the likelihood over $\mathcal{Q}_{\mathcal{I}}$, note that if we start with $\tilde{Q}_{ij} = 0$ for certain i, j, then $\tilde{Q}_{ij}$ remains 0 in all iterations of the algorithm. Furthermore, in order to apply the algorithm to our data set, we truncated the state space so that the part removed is negligible (time spent in it is about 0.5% of the total time, in the training data). We use variations of this algorithm to estimate the parameters of interest: $p_1(x, q; N), p_2(x, q; N), \xi(x, q; N)$.

3.2 Estimation in an Israeli call center

We now consider a specific foreign language service. It has an I-topology (see Figure 1) and thus could be analyzed in isolation. At each second t, we observe $(x(t), q(t))$, which is the number of customers in the system and in the queue, respectively. We analyze the data from weekdays in 2004-2005, from 10:00 till 11:00; the data from 2004 (232 days) is training data while from 2005

(226 days) serves as test data. This hour was chosen since the arrival rate does not change too much during this hour, as apposed to earlier in the morning.

3.2.1 High-dimensional form

In this section, we assume a certain high-dimensional parametric form for $p_1(x,q;N), p_2(x,q;N), \xi(x,q;N)$ and estimate the parameters. The process by which we obtained this specific parametric form is described in Appendix C. To summarize it, we start with a nonparametric estimate that yields a general form. Subsequently, we parametrize the estimate via logistic regression (p_1, p_2) and linear regression (ξ). (Logistic regression is commonly used to estimate probabilities.) We show below that our resulting estimates have good prediction power in the data sets we studied.

The parametric form that we worked with is

$$\begin{aligned} p_1(x,q;N) &= \frac{I(x-q<N)}{1+\exp\{-(c_1+\alpha_1 x+\beta_1 q+\gamma_1 N)\}} \\ p_2(x,q;N) &= \frac{1}{1+\exp\{-(c_2+\alpha_2 x+\beta_2 q+\gamma_2 N)\}} \\ \xi(x,q;N) &= (c_3+\alpha_3/x+\beta_3 q+\gamma_3 N)I(x-q<N). \end{aligned} \tag{6}$$

In order to calculate the MLE of $(c_1, \alpha_1, \ldots, \beta_3, \gamma_3)$ we used a variation of the EM algorithm of Bladt and Sørensen (2005), which is introduced formally in Appendix D. We obtained the following estimates for (6):

$$\begin{aligned} p_1(x,q;N) &= \frac{I(x-q<N)}{1+\exp\{-(-0.084-0.265x+0.039q+0.023N)\}}, \\ p_2(x,q;N) &= \frac{1}{1+\exp\{-(-8.010+0.206x+0.069q+0.166N)\}}, \\ \xi(x,q;N) &= (-0.116+0.720/x+0.002q+0.005N)I(x-q<N). \end{aligned} \tag{7}$$

The parameter p_1 is the probability of being served immediately upon arrival. It decreases in x since when x gets larger (with q fixed), there are fewer unavailable agents $N-n=N-x+q$ (which holds for $q>0$) and it increases in N due to having more agents present. The parameter p_2 is the probability to be served immediately upon service completion. It is more or less constant in x, q, N since serving a customer from the queue, after a service is completed, does not require that an additional agent becomes available. The parameter ξ decreases in x and increases in N for the same reasons as p_1. It increases in q since agents tend to become more available when the queue is longer. Notice the special form of $\frac{1}{x}$ in ξ, which is discussed in Appendix C.

To get some rough idea about the dynamics of the system, one can look at the value of the parameters in different states. For example, consider the case of eight customers in the system, two of whom are in the queue, and there are 12 agents in the system, i.e., $x = 8, q = 2, N = 12$. In this state, out of the 12 agents present in the system, only half are serving customers. The values of the parameters are

$$p_1(8, 2, 12) = 0.136,\ p_2(8, 2, 12) = 0.014,\ \xi(8, 2, 12) = 0.038; \tag{8}$$

the average arrival rate λ is approximately 0.02. That is, in this state the spontaneous service rate is about twice the arrival rate and therefore an unavailable agent is likely to become available before another customer arrives. The probability that an agent will stay available after an end of service, p_2, is low. Similarly, the probability of a newly arrived agent to start service, p_1, is low. From the point of view of the agent, what typically happens in state $(8, 2, 12)$ is that the agent becomes unavailable after the end of service, and then relatively shortly after, will become available again.

When the queue gets relatively large, e.g., $q = 5$, we have

$$p_1(8, 5, 12) = 0.150,\ p_2(8, 5, 12) = 0.018,\ \xi(8, 5, 12) = 0.045,$$

which represent an increase of about 10% for p_1, p_2 and about 20% for ξ, relative to the values in (8). This means that a newly arrived customer, when $q = 5$, is more likely to be served upon arrival, and, more importantly, an unavailable agent will faster turn available. Thus, as queue length increases, agents tend to become more available, which prevents the queue from becoming even longer.

3.2.2 First-principle form

We now consider a simplified version of Erlang-S, where each one of the functions $p_1(x, q, N)$, $p_2(x, q, N)$, $\xi(x, q, N)$ depends only on a single parameter.

The function $p_1(x, q, N)$ is the probability that a customer will be served immediately upon arrival. For simplicity, we assume now that $p_1 = 0$ when $q > 0$; thus, a customer can enter service immediately only when there is no one in the queue. When $q = 0$ and an arrival occurs, we assume that each of the $N - x$ agents that are not serving flips a coin with probability p_1^0 to serve the new customer, and the customer is served if at least one of those agents is willing to serve. Thus, we assume that $p_1(x, q, N) = I(q = 0; N > x)\{1 - (1 - p_1^0)^{N-x}\}$.

For $p_2(x, q, N)$, we assume that it is equal to a constant p_2^0, i.e., an agent that completes service moves on to the next customer with probability p_2^0 or becomes unavailable with probability $1 - p_2^0$. For the spontaneous service rate, we assume that, when $q > 0$, each of the $N - (x - q)$

agents that are unavailable has rate of ξ^0 to become available and to serve a customer from the queue. Agents become available independently of each other. Thus, we assume that $\xi(x, q, N) = \{N - (x - q)\}\xi^0 I(q > 0)$.

Summing up, the low-dimensional version of Erlang-S has the following structure:

$$p_1(x, q, N) = I(q = 0; N > x)\{1-(1-p_1^0)^{N-x}\},\ p_2(x, q, N) = p_2^0,\ \xi(x, q, N) = \{N-(x-q)\}\xi^0 I(q > 0) \tag{9}$$

where p_1^0 and p_2^0 are, respectively, the probability for a server to serve a customer that arrives when the queue is empty and upon service completion; ξ^0 is the spontaneous service rate of each agent.

The MLE of the parameters p_1^0, p_2^0, ξ^0 was calculated using an EM algorithm, similar to the one in Appendix D. We obtain

$$p_1^0 = 0.0515\ , p_2^0 = 0.0115,\ \xi^0 = 0.0111. \tag{10}$$

According to this parametrization, we calculate that

$$p_1(8, 2, 12) = 0.0,\ p_2(8, 2, 12) = 0.011,\ \xi(8, 2, 12) = 0.055.$$

Comparing these values to (8), we note that ξ is larger while p_1 is smaller ($= 0$ since $q > 0$) and p_2 is almost the same. Hence, in this state, for the low-dimensional form, an unavailable agent is more likely to become available, and a newly arrived customer will not be served upon arrival.

3.3 Predicting abandonment fraction

We compare abandonment predictions of Erlang-S models against both real test data and Erlang-A predictions. We denote the high-dimensional Erlang-S of Section 3.2.1 by Erlang-S$_H$, and the simple model of Section 3.2.2 by Erlang-S$_L$.

Recall that we obtained estimates for p_1, p_2, ξ (Sections 3.2.1, 3.2.2); for μ and θ we use the mean service time and the average of $\frac{\text{mean abandonment proportion}}{\text{mean waiting time}}$, over all half-hours in the training data, respectively. The additional two parameters λ, N are assumed known for each half-hour of the test data. The assumption that N is known in advance is reasonable since a system manager knows how many agents are present in the system. The arrival rate, λ, is not known in advance but there are efficient ways to estimate it. One can in fact improve the estimates by updating them periodically; see e.g., Goldberg et al. (2014). Consequently, for each half-hour, the parameters of the model are either estimated or assumed known and the steady-state equations can be numerically solved to compute $E[q]$, the average queue length. The prediction of the abandonment fraction is computed by

$$P(\text{abandonment}) = \frac{\text{abandonment rate}}{\text{arrival rate}} = \frac{E[q]\theta}{\lambda}. \tag{11}$$

In this work we focus on abandonment fraction. Similar analysis can be carried out for average queue length due to the relationship in (11), which would yield similar conclusions.

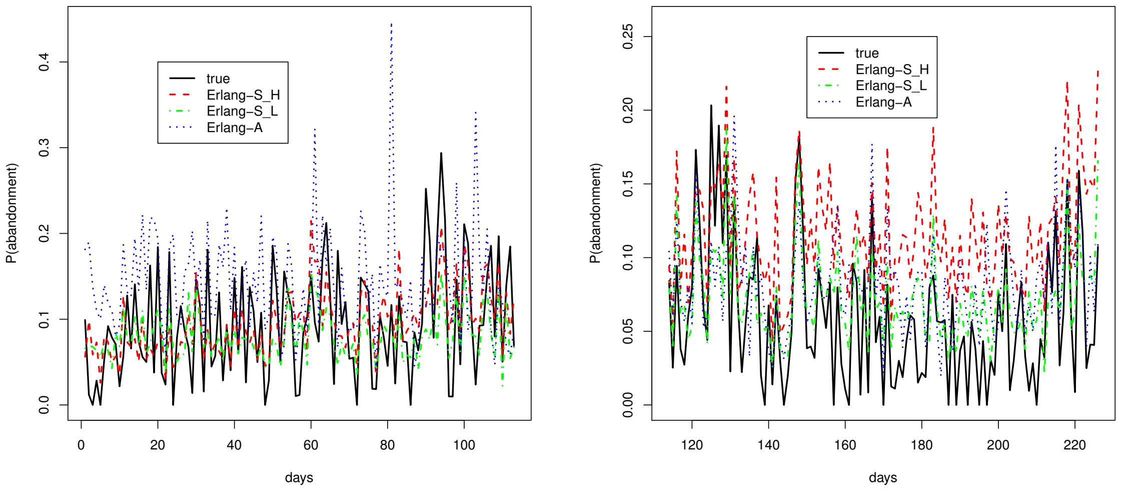
For Erlang-A, we follow a similar procedure: μ, θ are the same estimates as above and λ, n are assumed to vary between half-hours. Here n is the averaged number of available agents, as defined in (12) below. In the next subsection we consider other choices of n. Note that the average n is much less likely to be known in advance than N.

We compared the abandonment fraction between the real data and the models. The results are shown in Figure 8 and in Table 2. These indicate that Erlang-A predicts longer queues than in reality and hence the abandonment fractions are also overestimated. Both Erlang-S models generally provide good predictions of the abandonment fraction in the first half of the test data. The reason for the overestimation of queue length in Erlang-A is that agents tend to become available when they are needed and, therefore, considering only the average number of available agents leads to upward biased predictions.

Table 2: Abandonment fractions - comparing predictions against real data. The Mean (std) is presented.

	First half (2/1/2005 - 27/6/2005)	Second half (28/6/2005 - 22/12/2005)
Erlang-A	0.142 (0.063)	0.083 (0.034)
Erlang-S$_L$	0.080 (0.031)	0.073 (0.031)
Erlang-S$_H$	0.096 (0.039)	0.115 (0.038)
Data	0.094 (0.064)	0.060 (0.048)

Comparing Erlang-S$_H$ and Erlang-S$_L$ in the first half of the test data, we find that the predictions are similar but Erlang-S$_L$ provides slightly biased predictions. After about 150 days, the approximations of the Erlang-S models deteriorate. Erlang-S$_L$ has less parameters and therefore is more robust and provides better predictions than Erlang-S$_H$ in the second half. In fact, there are several changes between the two halves of the test data. The mean service times in the first and second half are 254 and 267 seconds, respectively. Also, the averages of N differ: 11.1 and 13.4 respectively. Figure 9 plots the average number (over the two half-hours: 10:00-10:30, 10:30-11:00) of N, for all days in the test data. It is clearly seen that, in the second half, there are generally



(a) First half (2/1/2005 - 27/6/2005). (b) Second half (28/6/2005 - 22/12/2005).

Figure 8: Abandonment fractions according to Erlang-A, Erlang-S$_H$, Erlang-S$_L$ and the test data, divided into the first (2/1/2005 - 27/6/2005) and second (28/6/2005 - 22/12/2005) half of the test data.

more agents present. All of this indicates that the agents of the first and second halves have different characteristics and, hence, the availability functions p_1, p_2, ξ that are estimated, based on the training data, are no longer valid in the second half. In practice, deterioration of prediction accuracy would suggest that parameters should be re-estimated periodically, where the time between successive estimators may vary depending on the dimensionality of the parametric form (the higher the dimension the longer is the time).

In summary, both Erlang-S models provide good useful predictions of the abandonment fraction, while Erlang-A overestimates it.

3.4 Versions of Erlang-A

In this section, we consider versions of Erlang-A, which differ by their number of servers and estimated service time. We use each version to predict abandonment fraction and study the performance of our predictors.

Erlang-S is a modification of Erlang-A where the number of available agents, n, is time dependent and varies with system load. As we demonstrated in Section 2, in call center data, n changes even during short time periods. In Section 3.3 we set n to be the mean and we now study alternative choices. We consider the Erlang-A model where n is the median and denote it by Erlang-A$_{median}$.

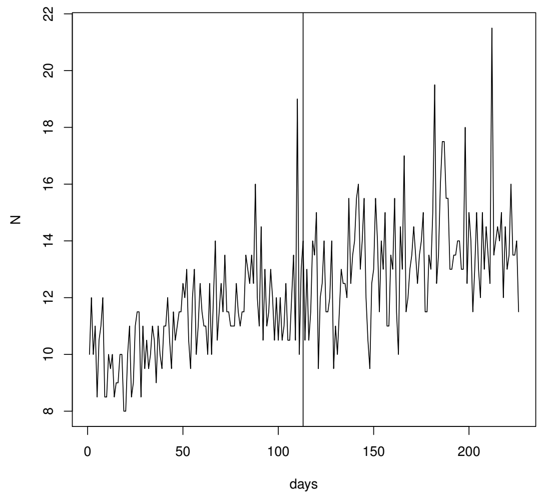


Figure 9: The average number of N in the test data (2/1/2005 - 22/12/2005) for each day; the average is over the two half hours: 10:00-10:30, 10:30-11:00. The vertical line separates the first and second half.

We also study the performance of Erlang-A when n is larger than the mean by a fixed number k; we denote it by Erlang-A$_k$. The latter models are not justified by certain features of the call center but rather serve as optional models that potentially could have good prediction power.

Furthermore, the analysis in Section 3.2.1 implies that agents usually become unavailable after service and become available again after some time. This implies that service is in fact composed of two parts, one is the observed service time and and the second part, in which agents remain unavailable to serve other customers, is unobserved. We should note here that, in the above data set, there is very little after-call work; for 98.2% of the calls, the wrap-up time is 0 and for 1.32% it is one second. Therefore, the second part of the service is not simple after-call work but rather some unobservable time during which the agent is unavailable. We call the sum of the two parts *virtual service time* and we denote the corresponding Erlang-A model by Erlang-A$_{virtual}$.

The four parameters of Erlang-A$_{virtual}$ are λ, θ, which are the same as in the regular Erlang-A; n equals to N, the number of agents present; and the service rate μ is the unobserved virtual service rate. For each half-hour h, we estimate the latter as follows: let $\lambda_h, \theta_h, N_h, p(Ab)_h$ be the arrival rate, abandonment rate, number of agents present and abandonment fraction at h, respectively. Also, let $F_A(\lambda, \mu, \theta, n)$ be the function that maps the vector of parameters $(\lambda, \mu, \theta, n)$ to the corresponding abandonment fraction according to Erlang-A (by solving the steady state equations and using (11)). The estimated virtual service time at half hour h is the quantity $\hat{\mu}_h$ that solves the equation

$F_A(\lambda_h, \hat{\mu}_h, \theta_h, N_h) = p(Ab)_h$. The mean virtual service time in the training data is 437 seconds, while the mean standard service time is 241.

We concentrate on the first half of the test data due to the reasons discussed above. We then compared the predictions of the different versions of Erlang-A, Erlang-S$_L$ and Erlang-S$_H$ as done in the previous subsection. Each of the models we considered predicts $\widehat{p(Ab)}_h$ for each h, a half hour in the first half of the test data, and it is compared to the true value $p(Ab)_h$ in terms of the following metrics:

- **Mean**: the mean $\frac{1}{H}\sum_h \widehat{p(Ab)}_h$, where H is the total number of half hours (compared to true mean $\frac{1}{H}\sum_h p(Ab)_h$).

- **p-value**: The p-value of the two-sample Wilcoxon test that identifies stochastic dominance between $\{\widehat{p(Ab)}_h\}$ and $\{p(Ab)_h\}$ (we used the function wilcox.test from the R software package).

- **RMSE**: the root mean square error $\left\{\frac{1}{H}\sum_h (\widehat{p(Ab)}_h - p(Ab)_h)^2\right\}^{1/2}$.

- **Error**: the mean $\mathcal{L}_1$ error $\frac{1}{H}\sum_h |\widehat{p(Ab)}_h - p(Ab)_h|$.

- **Relative error**: the mean relative error $\frac{1}{H}\sum_{h:p(Ab)_h>0.02} |\widehat{p(Ab)}_h - p(Ab)_h|/p(Ab)_h$, over the half-hours where the abandonment fraction is larger than 0.02. (If we consider also half hours with smaller abandonment fraction, then they are over-weighted.)

- **% over/under**: The percentage of over- or under-estimation, i.e., $100 \times |\frac{1}{H}\sum_h I(\widehat{p(Ab)}_h > p(Ab)_h) - 0.5|$, where $I(\cdot)$ denotes the indicator function.

- **% win**: The percentage of hours, h, where $\widehat{p(Ab)}_h$ is closer to the true value than the prediction of Erlang-S$_H$.

The results are presented in Table 3.

Erlang-A$_{virtual}$ provides unbiased predictions to the abandonment fraction. This is not surprising since the service time was estimated so that it would yield unbiased estimate of the abandonment fraction. The error under Erlang-A$_{virtual}$ is slightly larger than Erlang-S$_H$, Erlang-S$_L$: the RMSE and $\mathcal{L}_1$ error is about 5-10% higher. Overall, Erlang-A$_{virtual}$ is comparable to the Erlang-S models in this data set. Note, however, that we are fitting here a simple I-topology. The situation changes dramatically with a G-topology; then, the advantage of Erlang-S clearly emerges (Section 5.4) and errors increase by an order of magnitude.

We find that the mean and the median of n are close and hence Erlang-A and Erlang-A$_{median}$ perform similarly. From the different versions of Erlang-A$_k$, we found that Erlang-A$_1$ outperforms the other versions. That is, if we consider n to be bigger than the mean by 1, we obtain unbiased estimates of the abandonment fraction (the p-value of the Wilcoxon test is not small). The average n in this period is 5.7 and hence Erlang-A$_1$ represents an increase of about 20% over the mean. When comparing Erlang-A$_1$ to the Erlang-S models, we find that the error of the latter models are about 40% smaller than Erlang-A$_1$. Furthermore, Erlang-A$_1$ yields unbiased estimates, but this comes at the cost of high percentage of underestimation, unlike the Erlang-S models.

In short, certain versions of Erlang-A provide unbiased predictions, but Erlang-S$_L$ and Erlang-S$_H$ yield better predictions using the metrics we employed here. The prediction error under the best Erlang-A model is about 5-10% larger than for the Erlang-S models; and we repeat the above observation, that prediction errors become dramatically different in favor of Erlang-S when considering the G-topology data set (Section 5.4).

Table 3: Abandonment fractions - comparing predictions against real data in the first half of the test data (2/1/2005 - 27/6/2005).

	mean (std)	p-value	RMSE (std)	error (std)	rel. error (std)	% over/under	% win
Data	0.094 (0.064)						
Erlang-S$_H$	0.096 (0.039)	0.373	0.052 (0.057)	0.042 (0.03)	0.476 (0.552)	8.4%	
Erlang-S$_L$	0.08 (0.031)	0.281	0.054 (0.063)	0.043 (0.033)	0.425 (0.412)	5.8 %	50.4%
Erlang-A$_{virtual}$	0.093 (0.061)	0.922	0.057 (0.067)	0.046 (0.034)	0.541 (0.528)	2.2 %	46.9%
Erlang-A	0.142 (0.063)	< 0.001	0.092 (0.126)	0.072 (0.058)	1.052 (1.696)	28.8%	31.9%
Erlang-A$_1$	0.082 (0.045)	0.233	0.068 (0.089)	0.05 (0.046)	0.571 (0.942)	11.1 %	47.8%
Erlang-A$_2$	0.044 (0.031)	< 0.001	0.079 (0.095)	0.063 (0.048)	0.628 (0.465)	29.6 %	31.0%
Erlang-A$_{median}$	0.135 (0.062)	< 0.001	0.041 (0.114)	0.066 (0.056)	0.956 (1.519)	23.5%	34.5%

4 Estimating the number of agents present N

The number of agents that are present in the system, namely the parameter N, is often unknown. Erlang-S can then be used to estimate it, which we do in two ways: calculate the N that maximizes the likelihood of the model (MLE), or find the N that best fits a specific performance measure, e.g., abandonment fraction. The two approaches will be now formalized, implemented and compared.

4.1 Two approaches

Suppose that the functions p_1, p_2, ξ are given. For a specific time period h (e.g., 10:00-11:00 a.m. on a certain day), the parameters λ, μ, θ were estimated and the goal now is to infer N.

The first approach is to compute the MLE of N using the EM algorithm in Appendix D. The second seeks N that best predicts some performance measures. Formally, for a specific performance measure M, the Erlang-S model induces a function $M(\lambda, \mu, \theta, p_1, p_2, \xi; N)$ that is computed using the given parameters. As before, given p_1, p_2, ξ and a specific time period, the parameters λ, μ, θ are assumed estimated for that time period. Then one chooses an N for which $M(\lambda, \mu, \theta, p_1, p_2, \xi; N)$ matches best the observed performance.

4.2 Implementation and comparison

We now compute several estimates of N, by using the data set of the foreign language service from Section 3.2. These estimates are compared to the true values, in each hour of the first part of the test data, for which information on N is available. (We found that a half-hour is too short a period for obtaining reasonable estimates of N and therefore we considered hours.) The estimated N was restricted to be one of $7, 8, \ldots, 15$, which are the feasible values for N in the training data.

Several estimates were considered, which are based on Erlang-S. For the MLE, we considered the low and high-dimensional forms for p_1, p_2, ξ, denoted by MLE_L, MLE_H and introduced in Sections 3.2.2 and 3.2.1, respectively. In addition, we estimated N via two performance measures: abandonment fraction and average queue length, denoted by M-a and M-q, respectively. For each performance measure, we considered the low and high-dimensional forms, denoted by subscript L and H; for example M-a_L, denotes the estimate based on abandonment fraction under the low-dimensional form.

We also computed estimates via the Erlang-A model. They are based on the same performance measures: abandonment fraction and average queue length; they were computed as before, with the function M induced by the Erlang-A model. They are denoted by M-A-a and M-A-q for the abandonment fraction and average queue length, respectively. Note that N cannot be estimated through Erlang-$\text{A}_{virtual}$ in Section 3.4, since the virtual service time is not observed and N is used in order to estimate it. Thus, if N is missing than the virtual service time cannot be estimated.

As a benchmark, we compared the estimates to a naive estimate, which we took to be $\max_t\{x(t) - q(t)\}$, namely the maximum number of busy agents during the time period.

The estimates were compared via the mean error (absolute value) between the estimated N and the true one, for each hour in the first half of the test data (113 days). (For the second half, the models do not fit the data well, as discussed in Section 3.3, and therefore the second half is not considered here.) We also computed the fraction of cases where the error was less than or equal to 1. The results are displayed in Figure 10 and Table 4. The naive estimate provides downward biased estimates since it is rarely the case that all agents present are busy. The estimates based on Erlang-A are also biased, because the parameter n in Erlang-A measures the number of available agents, which is below the number of agents present. The estimates based on Erlang-S outperform the ones based on Erlang-A. Among them, MLE$_L$ provides the best results: 51% of the hours it is accurate in the sense that the error is at most one; also its average error, 1.74 agents, is the smallest.

If the errors in each hour of the test data are i.i.d, then the standard deviation of the sample mean is about $1.4/\sqrt{113} = 0.13$. It follows that the difference of 0.28 between MLE$_L$ and MLE$_H$, for example, is statistically significant.

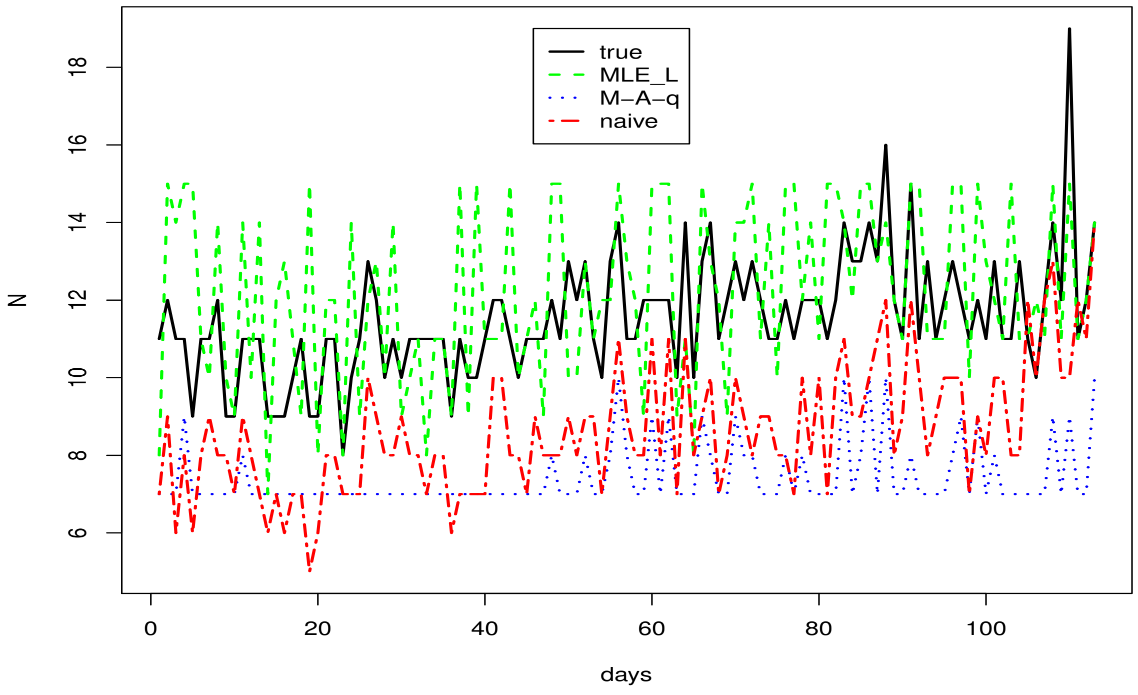


Figure 10: Comparison of estimates of N and the true N (2/1/2005 - 27/6/2005).

Table 4: Comparison of various estimates of N.

	Mean error (std)	% of error ≤ 1
MLE$_L$	1.74 (1.4)	51.3%
MLE$_H$	2.02 (1.4)	39.8%
M-q$_L$	1.99 (1.4)	41.6%
M-a$_L$	2.04 (1.5)	38.1%
M-q$_H$	2.33 (1.3)	26.6%
M-a$_H$	2.35 (1.4)	30.0%
M-A-q	4.12 (1.3)	1.0%
M-A-a	4.12 (1.3)	1.0%
naive	2.93 (1.3)	12.4 %

5 Erlang-S for a general queue.

So far, we have discussed the application of the Erlang-S model to an I-topology (recall Figure 1). Here customers of a certain type are served by agents from a single pool, which serves only that type of customer. However, as seen in Figure 1, this is uncommon in call centers. In this section, we discuss queues with a general structure (G-topology).

5.1 General topology

The general case is depicted in Figure 11. There are two types of customers, A and B, and two skills of agents, namely those whose primary service is A, and those whose primary service is B. There are N_A and N_B agents of the two skills present in the system. Both skills of agents serve both types of customers, but agents of certain skill tend to serve costumers of their primary type. This gives rise to a rather general model since one can focus on part A, while B is regarded as "the rest of the call center". In terms of Figure 1, A is business, while B is an aggregation of private, private VIP, business VIP, etc.

5.2 Estimation in an Israeli call center

We consider a specific part of an Israeli call center, analyzing data of weekdays in 2004-2005 from 10:00 till 11:00. The data from 2004 (232 days) is training data and from 2005 (189 days) is test

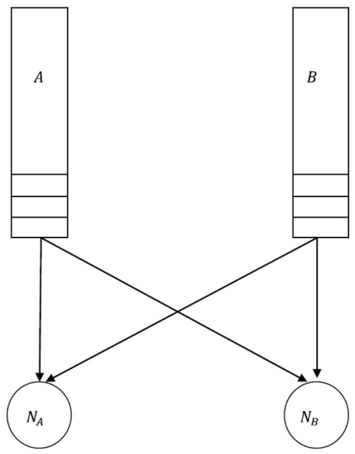


Figure 11: Aggregation of G-topology into an X-topology.

data. In this system, Types A and B are (almost) isolated from the rest of the call center: about 99% of the calls from customers A or B were handled by agents counted in N_A and N_B, and about 95% of the customers that these agents served were either A or B. Figure 12 plots N_A versus N_B, demonstrating that there is a negative correlation between N_A and N_B. The reason is that there is a roughly constant number of agents in this part of the call center, namely A and B, and when N_B increases, N_A tends to decrease.

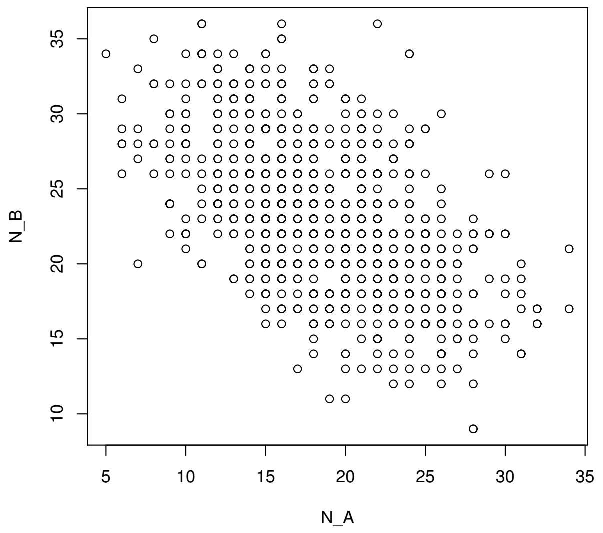


Figure 12: Plot of $\{N_A(h), N_B(h)\}$ for each half-hour h in the training data.

5.2.1 High-dimensional form

To apply Erlang-S to Figure 11 in its high-dimensional form, the parameters p_1, p_2, ξ are considered functions of N_A, N_B (as well as of x, q). Similarly to the I-topology case, $p_1(x, q; N_A, N_B) = \xi(x, q; N_A, N_B) = 0$ for $x - q \geq N_A + N_B$. To estimate the parameters, the same EM algorithm as in Section 3.2 can be used, namely the one in Appendix D; however, now the terms in (6) are functions of N_A and N_B, not just N. The parametric form implies that the estimates of p_1, p_2, ξ now depend on 15 parameters.

The MLE's are:

$$p_1(x, q; N_A, N_B) = \frac{I(x - q < N_A + N_B)}{1 + \exp\{-(0.0974 - 0.0605x - 3.5865q + 0.0461N_A - 0.0083N_B)\}},$$

$$p_2(x, q; N_A, N_B) = \frac{1}{1 + \exp\{-(-5.7315 + 0.1267x - 0.0753q + 0.0365N_A + 0.0195N_B)\}},$$

$$\xi(x, q; N_A, N_B) = (0.0140 + 0.0001x + 0.0027q + 0.0008N_A + 0.0004N_B)I(x - q < N_A + N_B).$$

There are several characteristics that are noteworthy. The probability p_1 decreases quite rapidly in q, hence the probability of being served immediately upon arrival is relatively high for $q = 0$, and much lower for $q > 0$. This is to be expected since customers in the queue are more likely to be served than a newly arrived customer. The coefficient of N_B in p_1 is negative due to the negative correlation between N_A and N_B discussed above. In this case, ξ does not have the special form of $1/x$ as before, but it is more or less constant in x; it increases in q, so that when the queue is starting to build up, agents tend to become available so as to accommodate the increasing workload.

5.2.2 First-principle form

The MLE's of p_1^0, p_2^0, ξ_0, as defined in (9), were computed similarly to Section 3.2.2, using $N = N_A + N_B$. That is, our low-dimensional model ignores the different skills of agents and considers only one pool of $N = N_A + N_B$ agents. Here, unlike the I-topology case, agents that are not available (to A-type customers) may be serving a B-type customer (Figure 11). We obtain the following MLE's:

$$p_1^0 = 0.026, \; p_2^0 = 0.040, \; \xi^0 = 0.001.$$

To compare both forms of Erlang-S, consider the situation where $x = 12, q = 2$ and there are $N_A = N_B = 20$ agents. The values of p_1, p_2, ξ are $0, 0.040, 0.039$ for the low-dimensional form, and $0.001, 0.037, 0.044$ for the high-dimensional form. Thus, p_1, p_2 are similar in both forms but ξ is slightly higher in the high-dimensional form; the means (1/rate) are 26 and 23 seconds for the low and high-dimensional forms, respectively. Hence, an unavailable agent is expected to become available earlier in the high-dimensional form.

5.3 Predicting abandonment fraction

We use the same setting of Section 3.3 to compare the predictions of Erlang-A, Erlang-S$_L$ (low-dimensional) and Erlang-S$_H$ (high-dimensional) to real data. Specifically, all parameters, excluding λ and N_A, N_B (or n for Erlang-A), are estimated from the training data and assumed constant throughout the test data. The arrival rate λ, as well as N_A, N_B and n, are assumed to be known in advance, over half-hours. For each half-hour in the test data, we compared the predictions of Erlang-A and Erlang-S against real data. The results are summarized in Figure 13 and Table 5. Due to the relationship in (11), the results for the average queue length are similar.

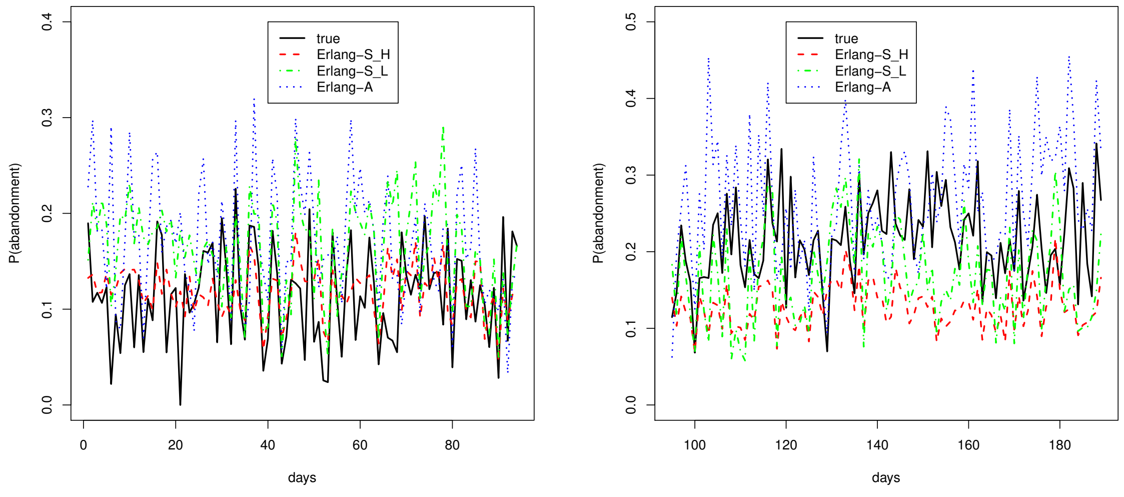
As before, Erlang-A overestimates the queue length and the abandonment fraction; Erlang-S$_H$ works well for the first half of the test data but not for the second. Unlike before, Erlang-S$_L$ does not provide good predictions and overestimates the queue length. This result indicates that Erlang-S$_L$ may be over-simplified for a general topology. Nevertheless, our conclusion is that Erlang-S, at least in its high-dimensional form, is still useful for the general topology.

Table 5: Daily mean (std) of abandonment fractions - comparing predictions against real data.

	First half (2/1/2005 - 23/05/2005)	Second half (24/5/2005 - 31/10/2005)
Erlang-A	0.171 (0.064)	0.266 (0.083)
Erlang-S$_L$	0.162 (0.053)	0.168 (0.063)
Erlang-S$_H$	0.118 (0.026)	0.127 (0.029)
Data	0.114 (0.051)	0.215 (0.058)

5.4 Versions of Erlang-A

As in Section 3.4, we consider here different versions of Erlang-A and compare them to Erlang-S$_H$ and Erlang-S$_L$. We use the same models and the same metrics of comparison as in Section 3.4. For Erlang-A$_{virtual}$, we study here two versions, one where the estimate of the virtual service time is based on $n = N_A + N_B$, and one where it is based on $n = N_A$; the latter is denoted by Erlang-A$_{virtual-A}$.



(a) First half (2/1/2005 - 23/05/2005). (b) Second half (24/5/2005 - 31/10/2005).

Figure 13: Abandonment fractions according to Erlang-A (blue, dotted), Erlang-S$_H$ (red, dashed), Erlang-S$_L$ (green, dotdashed), and in the test data (solid), separately for the first and second half of the test data.

We focus on the first half of the test data and the results are presented in Table 6. Here, unlike the I-topology data set, Erlang-A$_{virtual}$ is biased and highly variable. Indeed, the improvement over Erlang-A$_{virtual-A}$ in terms of RMSE and $\mathcal{L}_1$ error is 250-260% (whereas in the I-topology data set, the improvement was 5-10%). The rest of the results are not very different from that of Section 3.4. Erlang-A and Erlang-A$_{median}$ are similar. Erlang-A$_2$ (respectively, Erlang-A$_1$) provides the smallest mean square error (respectively, bias) among the Erlang-A$_k$ models but Erlang-S$_H$ outperforms all the Erlang-A models in all the metrics of comparison that we considered.

6 Erlang-S versus Erlang-A: a numerical study

So far we have studied the implementation of Erlang-S to call center data, with our main argument being that Erlang-S fits the data better than Erlang-A. In this section we perform a simulation study where the underlying process that generates the data is known. In this way, one can investigate the performance of Erlang-A when the real process is Erlang-S and vice versa. As importantly, the estimation procedure can be assessed since the true parameters are known.

Table 6: Abandonment fractions - comparing predictions against real data in the first half of the test data (2/1/2005 - 23/05/2005).

	mean (std)	p-value	RMSE (std)	error (std)	rel. error (std)	% over/under	% win
Data	0.114 (0.051)						
Erlang-S$_H$	0.118 (0.026)	0.526	0.049 (0.054)	0.04 (0.029)	0.482 (0.606)	5.3%	
Erlang-S$_L$	0.162 (0.053)	< 0.001	0.076 (0.092)	0.06 (0.047)	0.735 (0.83)	30.9%	39.4%
Erlang-A$_{virtual}$	0.313 (0.12)	< 0.001	0.227 (0.209)	0.201 (0.106)	2.172 (1.832)	44.7%	7.4 %
Erlang-A$_{virtual-A}$	0.241 (0.126)	< 0.001	0.178 (0.205)	0.14 (0.11)	1.597 (1.957)	33%	19.1%
Erlang-A	0.171 (0.065)	< 0.001	0.087 (0.106)	0.068 (0.054)	0.912 (1.531)	31.9%	37.2%
Erlang-A$_1$	0.124 (0.055)	0.304	0.059 (0.076)	0.044 (0.039)	0.561 (1.052)	7.4%	47.9%
Erlang-A$_2$	0.088 (0.045)	< 0.001	0.058 (0.066)	0.048 (0.033)	0.487 (0.672)	24.5%	41.5 %
Erlang-A$_{median}$	0.171 (0.068)	< 0.001	0.057 (0.104)	0.07 (0.056)	0.934 (1.459)	27.7%	37.2%

6.1 Real process is Erlang-S

In what follows, we assume that the real process is Erlang-S and calculate relevant performance measures via numerically solving its steady-state equations. Since our model uses states (x, q) rather than (x, n), we define and compute $n(t)$ for every t in a particular way and study the stationary distribution of $n(t)$. We then compare the performance measures of Erlang-S to that of Erlang-A whose n parameter is the average of $n(t)$.

Suppose that $\theta = \mu = 1/240$ (i.e., both mean service time and mean patience are four minutes), $\lambda = 0.07$ (per second) and there are $N = 30$ agents. Assume also the low dimensional form (9) with the values given by (10), which are the estimates from the data set described in Section 3.2.

The choice $\mu = \theta$ was made so that the process $x(\cdot)$ is Erlang-A, as the model is then tractable, being distributed as $M/M/\infty$; see Proposition 2. In order to compare the performance of Erlang-S to Erlang-A, one must define the corresponding n. In Erlang-S, the number of available agents n varies with time. For each t, we now define $n(t)$ as follows: let $[A_1, B_1], [A_2, B_2], \ldots,$ be the time intervals during which the queue is empty, that is $q(t) = 0$; then

$$n(t) := \begin{cases} x(t) - q(t), & \text{if } q(t) > 0 \text{ (equivalently } t \notin \bigcup_i [A_i, B_i]) \\ \max_{A_i \le s \le t} x(s), & \text{if } t \in [A_i, B_i] \end{cases} . \tag{12}$$

The idea behind this definition is that when $q = 0$, there is the possibility that some unavailable agent becomes available. If this occurs, then a customer that arrives is immediately served, and both

x and n increase by 1, but q stays 0. If an agent becomes unavailable, then an arriving customer will enter the queue and n will decrease by 1. Therefore, when the queue is empty, the number of available agents, n, is updated when an arrival occurs.

Recall that when $q(t) = 0$, the parameter $n(t)$ is undetermined (see Table 1). The steady-state distribution of $n(\cdot)$ cannot be directly deduced from that of (x, q). It is possible to compute this distribution by adding n to the state-space, following the dynamics in (12), and then resolve the new steady-state equations. Alternatively, one can compute it via simulation, which is a simpler procedure and hence adopted here. We simulated the process for 5×10^6 seconds and recorded $n(t)$ for each t. The empirical distribution is displayed in Figure 14. It is seen that $n(t)$ is usually around 17 but it varies. In order to compare the performance to Erlang-A, we use the average $n(t)$ in the latter, which is ≈ 17. The distribution is close to being symmetric and the median is also 17. Our estimate for $P\{n(t) = 17\}$ is 0.11.

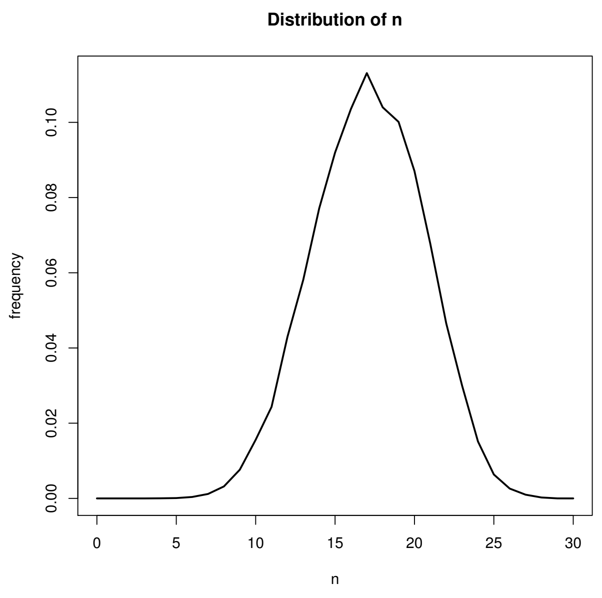
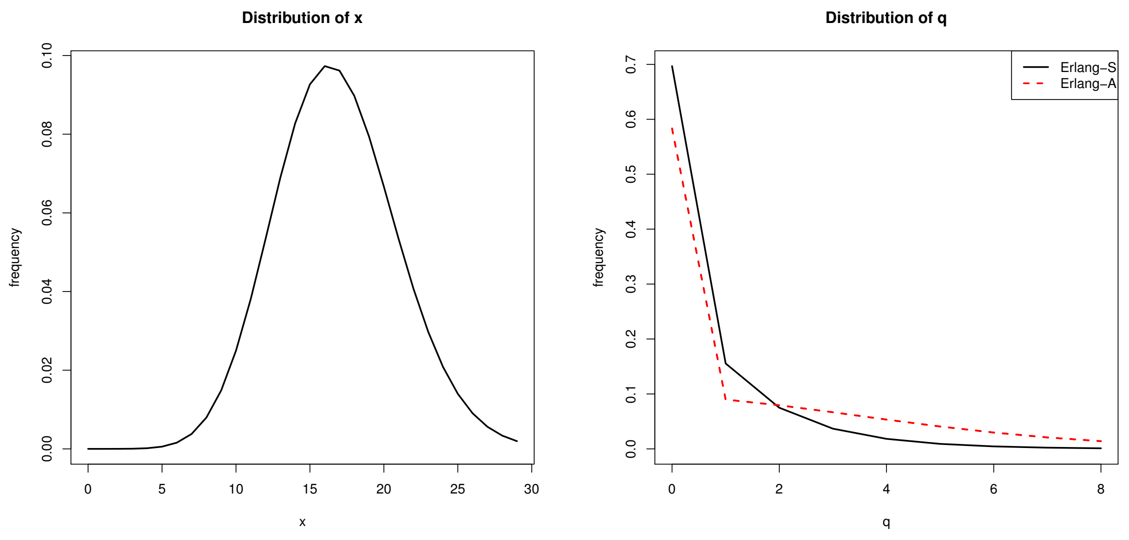


Figure 14: Distribution of $n(t)$.

Figure 15 shows the marginal stationary distribution of x and q for Erlang-A and Erlang-S; that of x is Poisson$(\lambda/\mu = 16.8)$ in both models. Under Erlang-A, the stationary distribution of q is $\sim \max(Y - 17, 0)$, where $Y \sim$ Poisson(16.8). For Erlang-S, the marginal stationary distribution of q does not seem to have a closed form expression and it was hence computed by numerically solving the steady-state equations. For the numerical calculation the state space was truncated so that $x < 39$ and $q < 15$ (when enlarging the state space so that $x < 50$ and $q < 5$, the additional probability is of order of 10^{-6}).

It is seen that, while for x the distributions are the same in both models, for q the distributions differ. Under Erlang-A there is a non-negligible probability that the queue is very large; while in Erlang-S the queue very rarely exceeds 6, say. Overall, the average queue-length in Erlang-A is 1.53 while in Erlang-S it is shorter, 0.59. In both models, the abandonment fraction is given by (11). Thus, the abandonment fraction in Erlang-A is higher than in Erlang-S: 9.1% vs. 3.5% respectively.



(a) Distribution of x.

(b) Distribution of q.

Figure 15: Histogram of x and q for Erlang-A and Erlang-S.

6.2 Estimation

We now study the estimation process. We simulated the above Erlang-S process $\{x(t), q(t)\}_{t=1}^{T}$, for $T = 3600 \times 200$, i.e., 200 hours. We observed the process only in discrete times (seconds) as in the call center data. We then computed the MLE in the same way as in section 3.2.2. The resulting estimates are

$$\hat{p}_1^0 = 0.0449,\ \hat{p}_2^0 = 7 \times 10^-5,\ \hat{\xi}^0 = 0.0115.$$

and the true values are

$$p_1^0 = 0.0515,\ p_2^0 = 0.0115,\ \xi^0 = 0.0111.$$

Thus, the estimates of p_1^0 and ξ^0 are close to the true values unlike the estimate of p_2^0. The reason might be that since p_2^0 is close to zero then the problem becomes one of rare-event estimation, which is a difficult problem. The estimated process yields an abandonment fraction of 3.5%, which is equal to the true value (up to the third decimal digit). Hence, although the estimated p_2^0 differs from the true value, it does not impact much the estimation of the abandonment fraction, which is very accurate.

6.3 Real process is Erlang-A

As mentioned in Section 2.3, Erlang-A is the same process as Erlang-S when $\xi(x, q; N) = 0$, $p_2(x, q; N) = 1$, and $p_1(x, q; N) = \begin{cases} 1 & x < n, q = 0 \\ 0 & \text{otherwise} \end{cases}$, for certain $n < N$. It coincides with the low dimensional form as described above, when $p_1^0 = 1$, $p_2^0 = 1$, $\xi^0 = 0$ and when $N = n$ (every agent present is available for service). Indeed, when we simulated the Erlang-A process that was described in Section 6.1, over a time period of 100 hours as in the previous section, we obtained the following estimates

$$\hat{p}_1^0 = 0.9999,\ \hat{p}_2^0 = 0.9999,\ \hat{\xi}^0 = 8 \times 10^{-5},$$

which almost coincides with the Erlang-A structure.

7 Discussion and further research

In what follows, we provide a short summary of our contributions and mention worthy future research directions. We divide the discussion into four short subsections: modeling, statistics, operations, analysis.

7.1 Modeling

Our main conclusion is that one must account for agents' operational behavior and availability in order to develop useful models that can accurately predict abandonment. We developed a model, Erlang-S, where servers can change their availability status. An important feature of our model (when fitted to real data) is that it reveals that agents tend to become available when the system is loaded and they are needed. Erlang-A, which cannot allow this feature, overestimates the queue length and abandonment fraction in our data sets. Other versions of Erlang-A, that add an extra after-service time, may work well for I-topology queues, but they are not expressive enough for G-topology systems.

We studied parametric forms of the availability functions at different levels of complexity and dimensionality. Within our data, we found that the simple low-dimensional form fits well the simple I-topology queue, while a high-dimensional form is required for the G-topology. These conclusions suggest that further research, with further data sets, is required to determine the complexity needed to model various types of queues, in particular when a simple modification of Erlang-A would suffice, and when and why it would not.

7.2 Statistics

On the statistical side, the main challenge was to estimate parameters of the continuous time process, while observations are made in discrete times. To overcome this difficulty, we used variations of the EM algorithm of Bladt and Sørensen (2005). It is worth mentioning that we also calculated the MLE's in the data set of Section 3.2, ignoring time discreteness; that is, maximizing the likelihood as given in Stage 5 of the EM algorithm in Appendix D, where $M_h[(x,q),(x',q')]$ and $R_h[(x,q)]$ are the observed number of transitions from (x,q) to (x',q'), and the observed sojourn time in state (x,q), respectively. The resulting prediction error (squared distance) of the abandonment proportion is about 7% higher than the prediction error of Erlang-S$_H$ with the estimated parameters (14). This finding points to higher prediction error when ignoring discreteness.

In the present work, we have not developed confidence intervals for the predicted estimates. These can be computed by calculating the information matrix, which is based on the quantities, $M[\cdot,\cdot], R[\cdot,\cdot]$, estimated in the EM algorithm. It is also desirable to be able to construct confidence intervals for the predictions of abandonment fractions. One possible approach would be a parametric bootstrap that simulates the process based on the estimates, and then predicts abandonment fractions according to the simulated process. This approach would be computationally intensive and it is left for future research.

7.3 Operations

In Erlang-S, not all of the agents present in the system are available for service. Thus, performance can be improved even without recruiting new agents. This insight gives rise to many questions and possibilities such as: can management measure and control the availability of each agent, perhaps reward agents accordingly in some way? We postulated three parameters of availability, p_1, p_2, ξ; is one of them more important to improve than the others?

To obtain some idea about improvements that can be achieved, consider a specific half-hour from the data set discussed in Section 3.2. In this half-hour, $\lambda = 0.01667, \mu = 0.00409, \theta = 0.00424$. The Erlang-S$_L$ model predicts an abandonment fraction of 0.065 (and the real fraction is 0.0645). If the spontaneous service rate, ξ^0, was doubled, then the abandonment fraction would be 0.032. If p_2^0 was doubled, then the abandonment fraction would stay almost the same. If p_1^0 was doubled, then the abandonment fraction would be 0.052. In summary, increasing p_1 or ξ, e.g., via some form of control, can improve significantly the abandonment fraction, and doing so without adding more agents to the system. On the other hand, increasing the availability parameters may come at the cost of other necessary agent activities. Further study is required to understand this trade-off.

7.4 Analysis

We compared the predictions of abandonment fractions of Erlang-S and Erlang-A, against real data. Erlang-A uses a constant number of available agents and, therefore, does not account for changes in availability status. This leads to overestimation of the queue length and abandonment fraction. Erlang-S, on the other hand, provides more accurate predictions.

In this study, we were able to solve the steady-state equations only numerically. We are relegating the explicit solutions, at least for some important special cases, to future studies (Takagi and Taguchi (2014) or Delasay et al. (2013) can serve as a starting point); however it seems that, ultimately, the now-prevalent asymptotics of many-server queues will be called upon to provide insight and numerical support. All this would lead to a better understanding of how the values of the parameters affect performance.

References

Aksin, Z., Armony, M., Mehrotra, V. (2007). The modern call-center: a multi-disciplinary perspective on operations management research. *Production and Operations Management*, **16**, 665–668.

Baccelli, F., Boyer, F. P., Hebuterne, G. (1984). Single-server queues with impatient customers. *Advances in Applied Probability*, **16**, 887–905.

Bhaskaran, B. G. (1986). Almost sure comparison of birth and death processes with application to M/M/s queueing systems. *Queueing Systems*. **1**, 103–127.

Bickel, P. J., Doksum, K. A. (2001). *Mathematical Statistics: Basic Ideas and Selected Topics*. Upper Saddle River, N.J. : Prentice-Hall.

Bladt, M., Sørensen, M. (2005). Statistical inference for discretely observed Markov jump processes. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, **67**, 395–410.

Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Shen, H., Zeltyn, S., Zhao, L. (2005). Statistical Analysis of a telephone call center: a queueing-science perspective. *Journal American Statistical Association*. **100**, 36–50.

Delasay, M., Ingolfsson, A.,Kolfal, B. 2013. Modeling load and overwork effects in queueing systems with adaptive servers. Submitted. Available at http://professor.business.ualberta.ca/armanningolfsson/Research/~/media/business/FacultyAndStaff/AOIS/ArmannIngolfsson/Documents/Modeling_Load_and_Overwork.pdf.

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

Gans, H., Liu, N., Mandelbaum, A., Shen H., Ye, H. (2010). Service times in call centers: Agent heterogeneity and learning with some operational consequences. In *Borrowing Strength: Theory Powering Applications–A Festschrift for Lawrence D. Brown, Institute of Mathematical Statistics*, 99–123.

Goldberg, Y., Ritov, Y., Mandelbaum, A. (2014). Predicting the continuation of a function with applications to call center data. *Journal of Statistical Planning and Inference*, in press. Available at http://ie.technion.ac.il/serveng/References/predicting_continuation.pdf.

Kc, D. S., Terwiesch, C. (2009). Impact of workload on service time and patient safety: An econometric analysis of hospital operations. *Management Science*, **55**, 1486–1498.

Mandelbaum, A., Zeltyn, S. (2007). Service engineering in action: the Palm/Erlang-A queue, with applications to call centers, *Advances in Services Innovations, Editors D. Spath and K.P. Fahnrich*. Springer-Verlag, 17–48.

Momcilovic, P., Armony, M., Mandelbaum, A. (2015). Data-based models of resource-driven activity networks. In preparation.

Niebel B.W., Lohmann M.R., Mee J.F. (2012). Motion And Time Study: An Introduction To Methods, Time Study And Wage Payment. Homewood, R.D. Irwin, Palm, C. (1957). Research on telephone traffic carried by full availability groups. Tele, vol.1, 107 pp. (English translation of results first published in 1946 in Swedish in the same journal, which was then entitled *Tekniska Meddelanden fran Kungl. Telegrafstyrelsen.*)

Senderovich, A. (2014). Service analysis and simulation in process mining. Ph.D. Research Proposal. Available at: http://ie.technion.ac.il/serveng/References/Research_Proposal_Final_12_2_14_Arik.pdf

Takagi, H., Taguchi, Y. (2014). Analysis of a queueing model for a call center with impatient customers and after-call work. *International Journal of Pure and Applied Mathematics*, **90**, 205–237.

Tian, N., Zhang, Z.G. (2005). *Vacations Queueing Models: Theory and Applications*, New York: Springer.

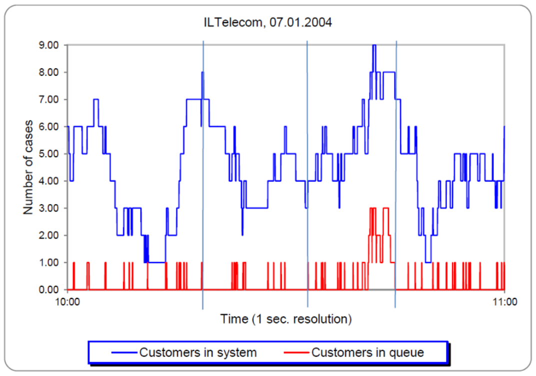
Sun, H., Dharmaraja, S., Williamson, C., Jindal, V. (2007). An Analytical Model for Wireless Networks with Stochastic Capacity. *Proceedings of SCS SPECTS, San Diego.*

Ward, A.R., Armony, M. (2013). Blind Fair Routing in Large-Scale Service Systems with Heterogeneous Customers and Servers. *Operations Research*, **61**, 228–243.

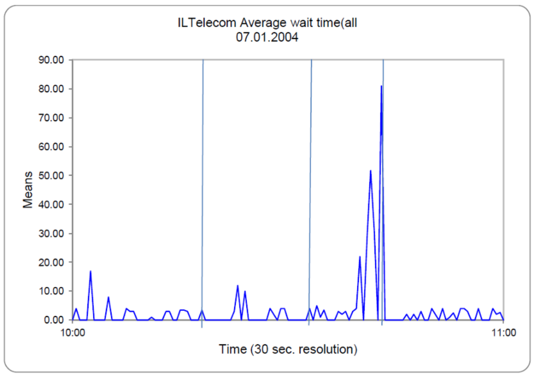
Yom-Tov, G., Mandelbaum, A. (2014). Erlang-R: a time-varying queue with reentrant customers, in support of healthcare staffing. *Manufacturing and Service Operations Management*, in press. Available at http://ie.technion.ac.il/serveng/References/Erlang_R_Nov_2013.pdf.

Appendix A: A second real example of violating the constant-agents assumption

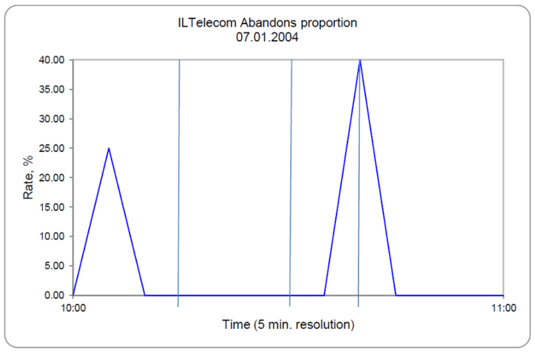
In this appendix, we present another example of violating the constant-agents assumption. It is taken from the data set that is analyzed in detail in Section 3.2. Here, we focus on data from a specific hour, illustrated in Figure 16. During this hour, relatively many customers arrived by 10:15 and they were served immediately, more or less (the blue line increases but not the red); these customers were served by around 7 available agents in the system. At approximately 10:30, agents became unavailable and only 3 are still serving customers. As a result, the service level deteriorates: the average waiting time and abandonment rate increase, as illustrated in (b) and (c) of Figure 16. Then, at around 10:45, 7 agents became available again and service level improved.



(a)



(b)



(c)

Figure 16: (a) Illustration of $x(t), q(t)$ (blue, red) in data from an an Israeli Telecom call center. At time $t =$10:18:30, 10:33:01, 10:44:21, $x(t) - q(t) = (7, 3, 7)$ respectively (and $q(t) > 0$). (b) Waiting time, averaged over 30-seconds intervals. (c) abandonment fraction, accumulated over 5-minutes intervals.

Appendix B: Proofs

Proof of Proposition 1

By definition, $q(t) \leq x(t)$ almost surely, for every $t \geq 0$. The process $\{x(t)\}_{t\geq 0}$ is a birth-and-death process with respect to the filtration of Erlang-S. It has a constant birth rate λ, and death rate $\mu \cdot (x - q) + \theta q$, which is larger than $x \min(\mu, \theta)$. (We allow here some harmless abuse of notation, in which x denotes both a process and its state.) Thus, if $\theta > 0$, it is stochastically smaller than an $x' \sim M/M/\infty$ process with arrival rate λ and service rate $\min(\mu, \theta)$, and the latter is stable. Therefore (Bhaskaran, 1986), there exists a probability space such that $x(t) \leq x'(t)$ and, of course, $q(t) \leq x(t)$, where both inequalities hold almost surely for all $t \geq 0$, and $(x, q) \sim$ Erlang-S, $x' \sim M/M/\infty$. Hence, the return time to $(0, 0)$ in Erlang-S is bounded above by the return time to 0 in $M/M/\infty$. Also, it can be easily deduced from (2) and (3) that every state can be reached from any other state in a finite number of steps with positive probability. Hence, since every state x (and in particular $x = 0$) in $M/M/\infty$ is positive recurrent, then Erlang-S is ergodic. □

Proof of Proposition 2

Let $\mathcal{F}_t := \sigma\{x(s), q(s)\}_{s \le t}$ denote the history of the Erlang-S process at time t. The Markovian property implies that, for every $t, s \ge 0$ and $A \subseteq \mathbb{N}^2$,

$$P\left[\{x(t+s), q(t+s)\} \in A|\mathcal{F}_t\right] = P\left[\{x(t+s), q(t+s)\} \in A|\{x(t), q(t)\}\right];$$

and in particular, for any $A \subseteq \mathbb{N}$,

$$P\left\{x(t+s) \in A|\mathcal{F}_t\right\} = P\left[x(t+s) \in A|\{x(t), q(t)\}\right].$$

Since the birth and death rates do not depend on $q(t)$, we conclude that,

$$P\left\{x(t+s) \in A|\mathcal{F}_t\right\} = P\left\{x(t+s) \in A|x(t)\right\}. \tag{13}$$

Let $\mathcal{F}_t^{(x)} := \sigma\{x(s)\}_{s \le t}$ denote the history of x. The law of iterated expectations implies

$$P\left\{x(t+s) \in A|\mathcal{F}_t^{(x)}\right\} = E\left(E\left[I\{x(t+s) \in A\}|\mathcal{F}_t\right]\mathcal{F}_t^{(x)}\right).$$

By (13), the inner expectation is equal to $P\left[x(t+s) \in A|x(t)\right]$ which is measurable with respect to $\mathcal{F}_t^{(x)}$. So,

$$P\left\{x(t+s) \in A|\mathcal{F}_t^{(x)}\right\} = P\left[x(t+s) \in A|x(t)\right]$$

and the Markov property follows. The birth rate is λ and the death rate at state x is μx, as in the $M/M/\infty$ queue. □

Appendix C: Choosing the parametric form

The estimation of the parameters in this form is carried out in two steps. The first step is to estimate the rates with no parametric assumption; that is, for each (x, q) there are five possible transitions for $q > 0$, and three for $q = 0$, and each rate is estimated separately. The second step is to assume a certain parametric structure and to estimate the parameters. The reason that the estimation was done in such an ad-hoc fashion is that we first sought to obtain a general insight of how the rates change with x, q, N and only then to fit a parametric curve.

We now describe the first step of the estimation. Figure 17 presents the state-space on which the process is observed; there are 50 possible states and 188 possible rates. In fact, the real process visits more states than the 50 assumed. Transitions to or from these other states are not used in the estimation procedure, nor is the sojourn time in these states. Due to this truncation, parameter estimates on the boundary become slightly biased, but the effect on accuracy is negligible since the sojourn time near the boundary is small. Furthermore, this first step of estimation is used to just determine parametric structure, hence the specific values of the estimates are of little importance.

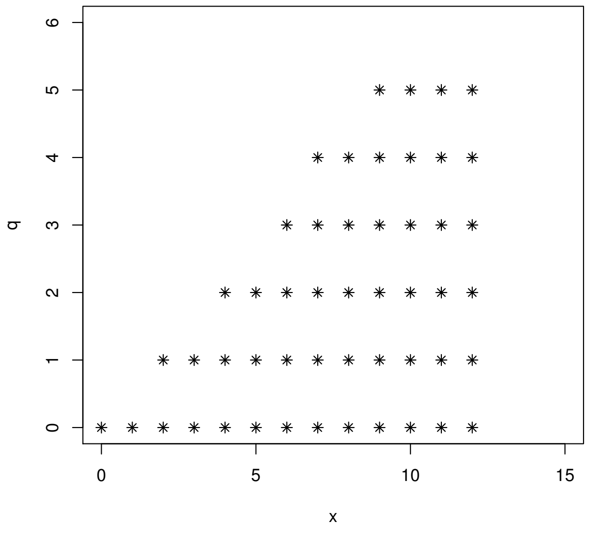


Figure 17: The state space.

The number of agents that were present in the system, N, is considered fixed over half-hours. Since agents arrive and leave, N can change some during half hours. However, we focus on 10:00-11:00, which is in a middle of a shift and changes are small. For simplicity, we worked with the number of agents present at the beginning of each half hour. The histogram of this N, for the 464 half-hours in the training data, is presented in Figure 18. We divide the half-hours data into 7 groups such that, in each group, the number of agents N is the same (see Figure 18; half-hours with 7 and 8 agents and with 14 and 15 agents were considered as one group each). For each group, we first estimated the transition rates from (x, q) to the five possible states (if $q > 0$), as illustrated in Figure 7, and doing so with no further parametric assumptions.

We used the EM algorithm of Bladt and Sørensen (2005), described in Section 3.1, with the following stopping rule: stop when the maximum difference (in absolute value) between the new

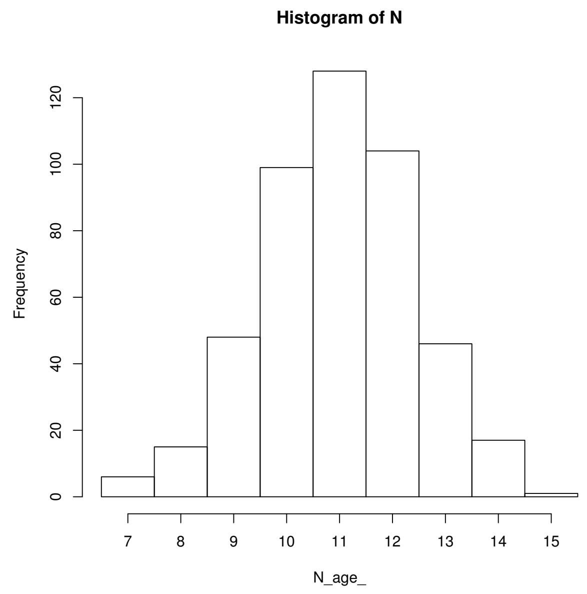


Figure 18: The histogram of N, covering the 464 half-hours in the training data. The mean is 10.97 agents, and the standard deviation 1.45.

and old estimates of Q_{ij} is smaller than 2×10^{-4}. We found that the following parametric form works well, in the sense that the parametric and non-parametric estimates are similar :

$$p_1(x, q; N) = \frac{1}{1 + \exp\{-(c_1 + \alpha_1 x + \beta_1 q + \gamma_1 N)\}}$$
$$p_2(x, q; N) = \frac{1}{1 + \exp\{-(c_2 + \alpha_2 x + \beta_2 q + \gamma_2 N)\}}$$
$$\xi(x, q; N) = c_3 + \alpha_3/x + \beta_3 q + \gamma_3 N.$$

For p_1 and p_2, the standard logistic form was assumed. The parametric form for the rate ξ is linear in q and N but not in x; for the latter, we found that the rate decreases like $1/x$ and not linearly (see Figure 19 below). A similar form was found in other data sets, which are not reported here. Notice that, if we make the following two assumptions:

1. Each of the $N - (x - q)$ present but unavailable agents has a constant rate of becoming available;
2. Agents become available independently of each other,

then the rate would be linear in x. This suggests that the aforementioned assumptions can not both hold in general. If only Assumption 2 prevails, then the convexity of $1/x$ suggests that agents have higher individual spontaneous return-to-service rate for small x than for large x (for fixed q).

We obtained the following estimates:

$$p_1(x,q;N) = \frac{I(x-q<N)}{1+\exp\{-(-0.084 - 0.265x + 0.039q + 0.023N)\}}, \tag{14}$$

$$p_2(x,q;N) = \frac{1}{1+\exp\{-(-8.010 + 0.206x + 0.069q + 0.166N)\}},$$

$$\xi(x,q;N) = (-0.116 + 0.720/x + 0.002q + 0.005N)I(x-q<N).$$

Figure 19 illustrates the rate ξ for certain choices of x, q, N, under the parametric form and according to the non-parametric estimate. This demonstrates that the rate decreases like $1/x$ and that the parametric and non-parametric estimates are indeed similar.

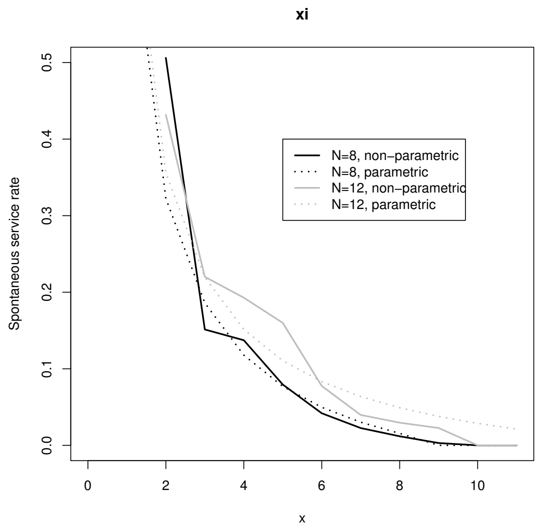


Figure 19: Estimates of $\xi(x, q = 1, N)$ for $N = 8$ and $N = 12$, for $x \in \{0, 1, \ldots, 11\}$.

Appendix D: EM algorithms

The heart of the EM algorithm is the computation of the conditional expectation in (4) and (5). Bladt and Sørensen (2005) provide explicit formulas for these expectations which are given now. Recall that

$$\tilde{N}_{ij} := E_{\tilde{Q}}\left[\text{number of transitions } i \to j | \{x(t)\}_{t=1}^{T}\right],$$

$$\tilde{R}_i := E_{\tilde{Q}}\left[\text{sojourn time in } i | \{x(t)\}_{t=1}^{T}\right],$$

then,

$$\tilde{N}_{ij} = \sum_{t=1}^{T-1} \tilde{f}^{ij}_{x(t),x(t+1)}, \;\; \tilde{R}_i = \sum_{t=1}^{T-1} \tilde{M}^i_{x(t),x(t+1)}, \tag{15}$$

for

$$\tilde{M}^i_{k,\ell} = M^i_{k,\ell}/\{\exp(\tilde{Q})\}_{k,\ell}, \;\; \tilde{f}^{ij}_{k,\ell} = f^{ij}_{k,\ell}/\{\exp(\tilde{Q})\}_{k,\ell}$$

where the notation $\{A\}_{k,\ell}$ denotes the k, ℓ entry of matrix A and

$$M^i_{k,\ell} = \exp(-\lambda)/\lambda \sum_{n=0}^{\infty} \frac{\lambda^{n+1}}{(n+1)!} \sum_{m=0}^{n} \{B^m(e_i e_i')B^{n-m}\}_{k,\ell},$$

$$f^{ij}_{k,\ell} = \tilde{Q}_{i,j} \exp(-\lambda)/\lambda \sum_{n=0}^{\infty} \frac{\lambda^{n+1}}{(n+1)!} \sum_{m=0}^{n} \{B^m(e_i e_j')B^{n-m}\}_{k,\ell}$$

for $\lambda \geq \max_{i=1,\ldots,s}(-\tilde{Q}_{i,i})$, e_i is the i-th unit vector, a' denotes the transpose of vector a, and finally $B = I + \tilde{Q}/\lambda$.

MLE of $(c_1, \alpha_1, \ldots, \beta_3, \gamma_3)$ (used in Section 3.2)

1. Start with some $(c_1, \alpha_1, \ldots, \beta_3, \gamma_3)$.

2. For each half-hour h in the training data, estimate $\mu(h), \theta(h), \lambda(h)$; here $\mu(h)$ is one over the mean service time, $\theta(h)$ is $\frac{\text{mean abandonment proportion}}{\text{mean waiting time}}$ (Mandelbaum and Zeltyn, 2003), and $\lambda(h)$ is the average number of arrivals per second (we know $N(h)$).

3. For each half-hour h in the training data, compute the generator matrix $Q_h[(x,q),(x',q')]$, based on $\mu(h), \theta(h), \lambda(h), N(h)$ and $(c_1, \alpha_1, \ldots, \beta_3, \gamma_3)$, according to (2), (3) and the parametric form (6).

4. Compute

$$M_h[(x,q),(x',q')] := E_{Q_h}\left[\text{number of transitions } (x,q) \to (x',q') \text{ in h}|\{x(t),q(t)\}_{t\in h}\right],$$

$$R_h[(x,q)] := E_{Q_h}[\text{sojourn time in state } (x,q) \text{ in h}|\{x(t),q(t)\}_{t\in h}],$$

and $M[(x,q),(x',q')] := \sum_h M_h[(x,q),(x',q')]$, $R[(x,q)] := \sum_h R_h[(x,q)]$.

5. **Estimating** $c_1, \alpha_1, \beta_1, \gamma_1$: using logistic regression (see, e.g., Bickel and Doksum, 2001, Section 6.4); at each state (x,q), there are $M[(x,q),(x+1,q)]$ "successes" and $M[(x,q),(x+1,q+1)]$ failures.

   Estimating $c_2, \alpha_2, \beta_2, \gamma_2$: maximizing the log likelihood

$$\sum_{x,q,h} \{M_h[(x,q),(x-1,q)]\log(q_1) - q_1 R_h[(x,q)] + M_h[(x,q),(x-1,q-1)]\log(q_2) - q_2 R_h[(x,q)]\},$$

where

$$q_1 = \mu(h)(x-q)[1-p_2(x,q;N(h))],\ \ q_2 = \mu(h)(x-q)p_2(x,q;N(h)) + \theta(h)q.$$

Estimating $c_3, \alpha_3, \beta_3, \gamma_3$: maximizing the log likelihood

$$\sum_{x,q,h} \{M_h[(x,q),(x,q-1)] \log\{\xi(x,q;N(h))\} - \xi(x,q;N(h))R_h[(x,q)]\}\,.$$

6. If the maximum distance (absolute value) between the new and previous estimates of the 12 parameters is bigger than 10^{-4}, go to step 3.

The conditional expectations of Step 3 were computed using (15). The maximization in Step 5 was performed numerically by using the "optim" function of R. The R code that implements the algorithm will be gladly provided by the first author.

MLE of N (used in Section 4)

1. Start with some N.

2. Compute the transition matrix $Q[(x,q),(x',q')]$, based on μ, θ, λ, the functions p_1, p_2, ξ and N.

3. Compute

$$M[(x,q),(x',q')] := E_Q\left[\text{number of transitions } (x,q) \to (x',q') \text{ during h}|\{x(t),q(t)\}_{t\in h}\right],$$
$$R[(x,q)] := E_Q[\text{sojourn time in state } (x,q) \text{ during h}|\{x(t),q(t)\}_{t\in h}].$$

4. Find an N that maximizes the log likelihood

$$\sum_{x,q} M[(x,q),(x+1,q)]\log(q_1^{(1)}) - q_1^{(1)}R[(x,q)] + M[(x,q),(x+1,q+1)]\log(q_2^{(1)}) - q_2^{(1)}R[(x,q)]$$
$$+ M[(x,q),(x-1,q)]\log(q_1^{(2)}) - q_1^{(2)}R[(x,q)] + M[(x,q),(x-1,q-1)]\log(q_2^{(2)}) - q_2^{(2)}R[(x,q)]$$
$$+ M[(x,q),(x,q-1)]\log(\xi(x,q;N)) - \xi(x,q;N)R[(x,q)],$$

where

$$q_1^{(1)} = \lambda p_1(x,q;N),\ \ q_2^{(1)} = \lambda\{1-p_1(x,q;N)\},\ \ q_1^{(2)} = \mu(x-q)[1-p_2(x,q;N)],$$
$$q_2^{(2)} = \mu(x-q)p_2(x,q;N) + \theta q.$$

5. If the new N is different from the previous N, go to Step 2.

Appendix E: Animating the data underlying Figures 1, 2, 3

This appendix serves as a brief documentation for the animations of Figures 1–3. Each colored circle traces the state of a specific customer or agent, and the arcs represent possible state transitions. A circle that moves within an arc from X to Y (say) stands for a customer or an agent who is still in X and will next switch to Y. The time to traverse that arc reflects the sojourn time in X; thus, slow and fast motion of circles corresponds to long and short sojourn times, respectively.

The real time is given in the lower left corner. The dynamic bars, which appear in some parts of the animations, represent the total number of customers or agents in the corresponding state. Also, in some of the animations, the thickness of the arc from X to Y corresponds to the number of customers that are in X and are about to switch to Y (and similarly for arcs from X to out of the network). We now elaborate on what is shown in each of the animations.

Figure 1, http://youtu.be/_eyAXVXZU7o: The first 19 seconds show the flow of customers from the queues in the ellipsoids to the servers in the rectangles. Next (till 0:53), the customers in the queue and those who are being served, are animated through bars that display the corresponding customer count. Red bars represent overloaded queues (waiting time exceeding 2 minutes). Next (till 1:20), the circles and bars are shown together. Next (till the end), the customer flow is represented by snapshots and thick/thin arcs: the arc width corresponds to the total number of customers that traversed that arc since the previous snapshot. Red arrows stand for abandonment.

The animation of **Figure 2**, http://youtu.be/DHvObpKjYrQ, shows both the circles and the bars. It focuses on two agents: the red circle stands for a "supervisor" and the green is a regular agent. While the regular agent spends most time in the "serving-cycle": ready → online → wrap-up, the supervisor often goes (after 16:00) to meetings and breaks.

The animation of **Figure 3**, http://youtu.be/H-wMFSl95KU, is similar to that of Figure 2, but here the different service types (investments, general banking, ...) of online and wrap-up are aggregated into one, and the animation covers all the agents present in the call center (not just two of them).

As a final comment, our data and its corresponding animations enable almost automatic time-studies, which traditionally have been performed through human observational studies (Niebel et al., 2012)