IT 10 047
Examensarbete 30 hp
September 2010

Predictive Dialing

Jonatan Lindén

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Predictive Dialing

Jonatan Lindén

An algorithm for predictive dialing in a call center context is examined by simulation, both in an outbound environment, and a blend inbound/outbound environment. Busy factor and abandonment rate is related to different choices of parameters. Two modifications of the algorithm are proposed and compared, one that takes into account enqueueing of calling customers for a inbound/outbound environment, and one that makes the call rate dependent upon the agent occupancy level.

Handledare: Ulf Sahlin
Ämnesgranskare: Lars-Henrik Eriksson
Examinator: Anders Jansson
IT 10047
Sponsor: S2 Communications AB

Tryckt av: Reprocentralen ITC

Sammanfattning

Ett callcenter är en avdelning eller ett företag specialiserat på att hantera stora mängder av telefonsamtal, men även i allt större utsträckning mer generella sorters meddelanden eller samtal via andra informationskanaler som till exempel e-post och chatter.

Man skiljer ofta mellan olika sorters callcenter beroende på vilken typ av samtal som utförs: callcenters med ingående trafik som hanterar kunder som ringer för att exempelvis få support eller för att boka en resa, medan utgående callcenter ringer upp eventuella kunder för att till exempel få dem att byta telefonileverantör eller genomföra någon form av marknadsundersökning. Det finns även mixade callcenter som hanterar både inkommande och utgående samtal.

Moderna callcenter är starkt beroende av datorstöd. De tekniker som används och dess integration med callcentret går allmänt under namnet *computer-telephony integration* (CTI), och omfattar allt från programstöd för att underlätta försäljarnas arbete genom att hantera information som samlas in, till att ringa upp kunder, eller svara ringande kunder.

I CTI räknar man även in tekniker för bland annat: routing av inkommande samtal till rätt agent med rätt kunskap, styrning av hur utgående samtal är anslutna till det allmänna telefonnätet, att avgöra om kunderna verkligen har svarat eller inte (så att det inte rör sig om en telefonsvarare eller dylikt), att låta den uppringande kunder specificera mer exakt vad de vill göra genom någon form av knapp- eller röststyrning (*interactive voice response*, IVR) och därmed minska den totala mängden arbete som måste utföras manuellt. Ett annat relevant exempel, men som kanske är lite mer perifert, är hur datorstöd används för att generera arbetsscheman, det vill säga avgör hur många agenter som bör vara närvarande vid callcentret baserat på historiska data och annan tillgänglig information.

Beslutet om hur och när en kund ska ringas upp kan antingen tas av den enskilde agenten, det vill säga han eller hon kan välja om och när de vill ringa någon genom att klicka på en knapp (eller slå numret) eller så kan beslutet överlåtas till en dator. I det senare fallet kommer systemet att bestämma hur många samtal som behöver genomföras, företrädesvis baserat på hur många anställda som finns tillgängliga, bestämma vilka nummer som ska ringas, och till sist ringa själva samtalen, eventuellt efter en bekräftelse från agenten.

Ett av problemen med utgående call centers är att det under dagtid är mycket svårt att genomföra kampanjer med privatpersoner som målgrupp, eftersom mycket få personer verkligen hemma. Mer allmänt är callcentrets effektivitet (mätt i antal genomförda samtal) proportionell mot andelen svarande kunder, då kunder som inte svarar genererar extra väntetid.

Ett sätt att lösa detta problemet är att använda en metod för uppringning som försöker förutspå hur många kunder som kommer att svara baserat på tidigare samtalsdata, så kallad predictive dialing.

Användandet av predictive dialing innebär bland annat att det kommer i allmänhet finnas fler aktiva utgående samtal (det vill säga både samtal där samtal pågår men även samtal som väntar på svar) i systemet än det finns säljare, och att samtalen inte kommer att delas ut till säljarna förrän kunden har svarat. Om ingen säljare svarar inom en viss tid efter det att en kund har svart, kommer samtalet att kopplas ner, det vill säga ett missat samtal har genererats.

Predictive dialing kan möjliggöra för en mycket snabbare bearbetning av ringdata än med traditionell uppringning, eftersom den största delen av agentens väntetid elimineras. Det för emellertid med sig ett antal egenskaper som inte är lika eftertraktade, som till exempel de just nämnda missade samtalen, men även det faktum att säljarens förberedelsetid (det vill säga den tiden då det normalt sett är möjligt att läsa in sig på vad den uppringda personen heter och vad de har för sysselsättning, inte paustid) inför varje samtal blir så gott som obefintlig.

Den här uppsatsen utgår från en algorithm för predictive dialing, tidigare presenterad av Korolev et al. [15], och undersöker en implementation av den med hjälp av simulering. Vidare jämförs resultaten med två modifikationer av orginalalgoritmen, simulerade under samma villkor. För simuleringen implementerades även en callcenter simulator.

Resultaten av simuleringen av den ursprungliga algoritmen överensstämmer i sina huvuddrag med de som ges i Korolev et al. [15], men då fler parametrar har tillförts via simulatorn skiljer de sig till viss del. Simuleringarna av de modifierade algoritmerna visar att det finns möjligheter att förbättra algoritmen, och då i synnerhet med avseende på hur den hanterar inkommande telefonsamtal, samt genom att göra den beroende i större utsträckning av vissa faktorer i callcentret, istället för att enbart bero på avslutade samtal.

Contents

1 Introduction

A *call center* is a department or enterprise specialized in handling large amounts of telephone calls, or in its present-day equivalent, the *contact center*, large amounts of messages, where a message could be anything ranging from a chat message to a phone call.

One often makes a difference between call centers depending on what kind of calling is done; inbound call centers handle *customers*, calling to get support or make a booking, for example, while outbound call centers call the customers to, for example, sell something to them. There are also blend centers which handle both inbound and outbound calls.

Call centers of today are highly dependent on technology. The computerized support and its integration with the call center goes under the name of *computer-telephony integration* (CTI), and encompasses the whole spectra of technologies used to facilitate the work of individual salespersons, called *agents*, who call customers, or answer calling customers, as well as administrators.

In CTI one includes techniques for, among others, routing inbound calls to the right agent with the right knowledge, controlling how the outbound calls are connected to the public telephone network, deciding whether customers really have answered or not, letting the calling customers specify more exactly what they want to do through some kind of interactive menu (IVR) and thereby relieving the agent of this work, etc. A bit apart, but still pertinent as an example, is how technology is used for efficient rostering, deciding how many agents should be present at the call center based on historical data and other available information.

The decision of whom to call and when to do it can either be controlled by the individual agent, i.e., he or she may choose if and when they want to call somebody by clicking on a button or even dial the number, or this decision could be left to the computerized system. In the latter case, the system will decide how many calls to make, preferably based in some way on how many agents that are available, fetch the numbers from some database, and call the given numbers (possibly after an acknowledgement from the agent), and at some point in time hand out the calls to the right agents, who can then talk with the customer. We will henceforth denote these two approaches as agent-driven and server-driven, respectively. There are several server-driven calling strategies, and we refer to these as dialer methods.

The general problem that the two approaches address is finding a balance between efficiency and service quality, in the way that the agents have enough time to concentrate and prepare themselves for each call, so that the quality of the call is good enough, but at the same time avoiding inactivity. In some cases it might be better to have a low efficiency to achieve a higher quality, if for example the customers are important, which often is the case when calling to businesses. In some other cases this tends to be less important; it is seldom negligible.

One of the problems with outbound call centers is that during daytime it is very difficult to do customer bound campaigns, since very few people actually are at home. More generally, the effectiveness of the call center is proportional to the percentage of answering customers.

One approach solving this problem is to use a server-driven calling method that tries to predict the number of answering customers based on historical

and current call data, so-called predictive dialing. Predictive dialing makes it possible for call centers to operate at hours that would otherwise be impossible, for example during daytime as written above, or in other situations when the customers in question are less inclined to answer. Using some other method in these situations would give rise to far too high rates of unanswered calls, which equal wasted time.

Using a predictive method means in particular that there will in general be more outbound calls than there are available agents, and that the calls will not be dispatched to the agents until the customers answer. If no agent takes the call within a certain time limit after the moment that the customer has answered, that call will be dropped, a so-called *mismatch call* or *abandon calls*.

Nevertheless, this allows for a much greater call speed than with traditional call-center dialing, since most of the agent's waiting time is eliminated. It brings with it a number of properties that are less desired however, such as the just mentioned mismatch calls and leaving no time for agent to prepare prior to each call.

Predictive dialing will be the subject of this document. The main focus will be on implementations of a particular algorithm of the predictive dialing method presented by Korolev et al. [15], and two modifications of that algorithm, that will be studied in relation to the original algorithm. A call center simulator has been implemented to make it possible to evaluate and compare the different algorithms.

2 Background

There is a number of different widespread dialing methods/pacing algorithms. A dialing method could be seen as the logic or algorithm that decides who should be called next and when, and at what rate new calls will be made, possibly with some decisions left to the agent. This stands in contrast to an agent-initiated method, where the agent at least decides when to call.

Some vocabulary that might need an introduction is the *hit rate* of the outbound dialer, that is the percentage of customers that answers the calls made by the dialer, the *talk time* is the effective time of speaking and the *after call work* is the time after a successful call when some extra work might need to be done by the agent, like finishing an order or the like. We call the sum of the last two the *service time*. We define the *abandonment rate* to be the percentage of abandon calls in relation to the total number of calls answered. The busy factor is the average service time per agent in relation to the total work time.

A short introduction of some well-known dialing methods will follow, to get a general feeling of some basic problems that are encountered in an outbound call-center and how they are solved. The focus will then be returned to the predictive dialing method. It should be noted that this paper does not state that a certain method is better than another, a question that would be very difficult to answer, and that is more complex than finding out how much time of the agents' working time is talking time.

2.1 Preview

The preview dialing method is one of the agent-driven methods. The idea is to partly let the agent decide whom to call, but forcing him or her to accept a certain speed. The agent is presented with information about the customer, and is given a choice to make the call or to go on and view the next customer. If no choice has been done within a certain time limit, defined in advance, the customer is called. If the agent choose to not make the call, the next customer in line is presented, and the procedure is repeated. After a successful call is made, a certain amount of time is used for after call work (in common with all dialing methods).

There are some drawbacks with this method. For each call that is not answered by the customer, an overhead of the length of the time limit mentioned is generated, plus the waiting time of the call. Thus, if the hit rate is very low, this method becomes inefficient, and should either be used during times when the current group of customers could be expected to answer and with calling material that has been filtered in advance, or in some other scenario where one could expect a reasonable number of calls being answered. The good side is that it gives the agents some time for preparation, however still limiting it. It sets a minimum level of efficiency. What could be seen as an acceptable time of preparation varies from case to case, and can be adjusted for each case.

2.2 Progressive

The progressive dialing method is straightforward: in every moment we compute the number of agents that are free and can receive calls in this moment. The corresponding number of calls are made.

As defined by Korolev et al., the number of calls to be made, when using the progressive method, are given by

$$N = N_r - N_d,$$

where N_r is the number of free agents, and N_d the number of already dialed calls. The abandon rate is theoretically zero, but the method equally keep the busy factor at a low level when the hit rate is low. It may be suitable for a warming up period of a call center, i.e., as a data initializer for another algorithm, as mentioned by Korolev et al. In fact it does perform very well if the hit rate is high. As the hit rate approaches 1, this algorithm will perform as well or better than a predictive algorithm, since it will not generate mismatch calls.

As specified by Korolev et al. [15], the busy factor is given by

$$u = \frac{T_s}{T_s + (1-p)\tau_1/p + \tau_a}, \tag{1}$$

where T_s is the average service time, p the hit rate, τ_1 and τ_a the average time an unsuccessful and a successful call, respectively, stay in the dialer, i.e., is not hung up automatically or dispatched to an agent. This could be understood by seeing that $(1-p)\tau_1/p$ is the average time the dialer is occupied waiting for unsuccessful calls to finish, *per* successful call.

This method is more efficient than the preview method, since the progressive method avoids repetitive extra waits after each failed call. In the preview case,

if the call is not answered, the agent will be presented with a new choice, and a certain decision time follows. In the progressive case a new call will be made directly after this is recognized (with a delay of a few seconds). We can also see that it will become more efficient if we shorten the time limit of a call to be seen as unsuccessful. However, a too short time period will prevent customers from being able to answer, and apart from being questionable from a nuisance point of view, this will also lead to a decreased hit rate, and will as such lower the busy factor.

On the downside, we have the same situation as with the predictive dialer (which will be seen shortly): the agent has no possibility to review to whom he or she is going to talk until the conversation already has begun, and the customer information has to be reviewed during the talking. However, if the call data is of bad quality, i.e., if it contains stale information, and other factors that decreases the hit rate, this method is definitely of interest in comparison to preview dialing.

2.3 Predictive

As the name suggests, a predictive pacing algorithm tries to predict how many customers to call. Roughly put, the number of placed calls is based on how many agents the algorithm expects to be free in a short moment, as well as how many customers it expects to answer, a description that should be appropriate for all such algorithms. It has the same basic pros and cons as the progressive method, i.e., it avoids unwanted waiting times for the agents, but does not give the agents a chance to review the customer information before the call. On top of that, the predictive method can be more efficient, with respect to avoiding agent waiting times, than the progressive method, if the predictions are accurate enough. This is also the crux of the method, since inaccurate predictions directly incur either mismatch calls or a lower agent activity. A mismatch call, or abandon call, is a call made to a customer, without there being any free agents left in the call center at the moment the customer answers, and thus has to be dropped. In certain countries there is legislation regarding how many such calls are allowed, and if being exceeded, heavily fined. Whether or not such legislation exists, mismatch calls are unwanted, and this is one of the main objectives of the algorithms that will be studied.

3 Statement of Problem

We will study an algorithm for predictive dialing, i.e., an algorithm that continuously calculates the number of calls to be made from the call center. The algorithm that will be examined has been presented by Korolev et al. [15]. They identify several optimization problems that define the possible goals of predictive dialing, and also give a basic solution to each of the optimization goals.

As per Korolev et al., the problem can be stated as to define a pace for call generation such that:

- All agents should be sufficiently occupied, in other words, receive enough calls so that the busy factor of the call center is satisfactory.

- Mismatched calls should be bounded from above by some level, such that the abandonment rate of the call center does not become too high.

Further, this should be treated as two separate optimization problems, since they might not be satisfied simultaneously, as stated by Korolev et al. [15]. Hence:

1. The abandonment rate should be maintained at an admissible level $A_{\max}$, while maximizing the agents' busy factor.

2. The agents' busy factor is maintained at an admissible level $u_{\min}$, while the abandonment ratio is as low as possible.

Korolev et al. gives two basic algorithms, one for each optimization problem, and examines them by means of analysis.

The two basic algorithms they present are based on the Erlang loss formula, see for example Cooper [6] for a short introduction (an even shorter introduction will follow in the next section).

In this paper, the first algorithm presented in Korolev's paper, i.e., the one related to the first optimization problem, and which undoubtedly must be seen as the most useful one, is implemented and analyzed by simulation. The scope of the algorithm has been somewhat extended, and includes estimating some of the parameters of the algorithm that are taken for given by Korolev et al. Questions of interest are how well the algorithm performs with respect to abandon rate and busy factor in a simulated environment. During the course of writing, some possible refinements of the algorithm were reflected upon, and two alternative algorithms were implemented as well. These are compared to the original algorithm.

It should be noted that Korolev et al. also proposes another algorithm for predictive dialing that fits especially well for the case 5 - 25 agents, baptized hyper-progressive. It performs better than the progressive as well as the predictive algorithms in that particular case, which perform very bad for that few agents.

4 Tools

In this section follows a short introduction to some various subjects that are relevant to understand the theoretical foundation of the algorithms.

4.1 Poisson processes

A Poisson process is a continuous time Markov process, a counting process with independent increments, such that $N(0) = 0$ and $N(\tau + t) - N(\tau) \sim \text{Poi}(\lambda t)$, $\forall \tau, t \geq 0$, where λ is the rate, or intensity, of the process and Poi is the Poisson distribution.

The Poisson process has some very nice properties. If we add two independent processes with rate λ_1 respective λ_2, we will get a new process with rate $\lambda_1 + \lambda_2$, i.e., if one type of event occurs with a certain rate and another type of event occurs with another rate, then any of the two types of events occur with the sum of the rates. The number of events in any interval $[a, b]$

depend only on the length on that interval, and has a Poisson distribution with parameter $\lambda_{a,b}$. On the other hand, there are non-homogeneous Poisson processes. Non-homogeneous (the one described above being homogeneous) or equally, non-stationary, Poisson processes, are Poisson processes where the parameter λ, the *arrival rate* or *intensity*, varies over time, i.e., is a function of the time, $\lambda(t)$. In that case, the number of events in any interval $[a, b]$ also depends on the intensity function on that interval. This adds of course some levels of complexity, and there are a lot of papers studying how to approximate these processes.

There exists also various generalizations of Poisson processes that can model more complex situations with, for example, dependencies. To be mentioned here is renewal theory and Markov arrival processes and its relatives. This will not be touched upon in this document, but is somewhat related to complex queueing models.

A good starting point for an introduction to Poisson processes can be found in the book by Lefebvre [17].

4.2 Queueing theory

Queueing theory is at the core of this problem. It defines and formalizes the general notion of a queue, be it in a high-speed network, your local grocery shop or a call center.

A queueing system is defined, in its most basic form, as a number of servers that handles arriving customers. A server represents a clerk/agent/channel, a customer represents a customer(!)/incoming phone call/incoming IP-packet. The queue has a capacity ranging from 0 to ∞. With the queueing system is associated a stochastic process, having certain properties that may or may not be known, representing the flow of arriving and departing customers. This process has a certain arrival rate. When a customer is handled by a server, such an event is connected with a thereupon following service time, that may or may not vary between each customer. With each queueing system is associated a queueing discipline, First Come First Served (FCFS), Last Come First Served (LCFS), etc. We will only be interested in the FCFS case, and this will not be explicitly mentioned in the rest of the text.

The general way to denote a queueing system is by $A/S/s/r$ in Kendall's notation (first introduced in Kendall [13]), where A is the distribution of the time between two arrivals, S is the distribution of the service time, s the number of servers, r the capacity of the queue. Occasionally in some papers, r denotes the total capacity of the system, and in that case the queue size would be $r - s$; this is a pitfall and has to be understood on a per paper basis. (Someone else might state that, occasionally r denotes the capacity of the queue) Thus, in this document, r will always denote the queue length and will always be explicitly written. More general notations exists, but this one suffices for our needs.

Further, some of the most common distributions occurring in queueing theory have been baptized by capital letters as well: M - exponential, G (or GI) - general (or general independent), D - deterministic (constant). G is almost always assumed to be independent as well, GI is just a more explicit way to show it. A queue with exponential inter-arrival times (M) has thus a homogeneous Poisson arrival process. We might be interested in a queue with a

non-homogeneous Poisson arrival process, this is succinctly denoted M_t, showing that inter-arrival times depend on the time t.

The models that we mainly will study are $M/GI/s/0$ (i.e., no queue) and the $M/M/s/\infty$ (hence with the capacity of a very long queue, often just written $M/M/s$).

To get a more thorough introduction I direct the interested reader to for example Lefebvre [17] or Bhat [1].

Erlang formulas

The danish mathematician Agner Erlang was one of the first to study queueing systems (see for example Erlang [9]), and came up with some of the first formulas to calculate the blocking probabilities of phone calls in telephone networks, that nowadays also are used in the context of, for example, call centers and in queueing systems in general.

The Erlang B formula, or Erlang's loss formula, calculates the blocking probability of a $M/G/s/0$ queueing system, that is, the probability that a call is blocked caused by all servers being occupied and thereby dropped since there is no queue.

$$E_B(m, \rho) = \frac{\rho^m/m!}{\sum_{i=0}^{m} \rho^i/i!} \tag{2}$$

The blocking probability depends on the number of servers/agents m and the current traffic offer ρ, where $\rho = T_s\lambda$ (the traffic offer is sometimes given a unit called Erlang, which adds nothing to the result of the just given relationship, more than maybe clarifying of what one is speaking), the product of the current service time T_s and the rate λ of the Poisson process. This rate might of course vary continually over time, $\lambda(t)$.

This should be compared with Little's theorem, one of the fundamental results of queueing theory, that states that the asymptotic average number L of customers in the system can be related to the arrival rate and the service time, $L = T_s\lambda$, where T_s and λ in this case represents the asymptotic average service time respective asymptotic average arrival rate.

What is worth to mention is how the formula in the stationary case is independent of the distribution of the service time beyond the mean. This is remarkable, since it means that no matter how the service times are distributed, be it normal, exponential, bimodal: the blocking probability is independent of the arrival rate and will hence be the same as long as the mean is the same. The proof was first given by Sevastyanov [24].

It should be noted that the result is only valid for the blocking probability, i.e., the asymptotic variance and workload factor are sensitive to the service time distribution. Furthermore, this result is no longer valid if the Poisson arrival process is non-stationary (non-homogeneous): if so the result is dependent on the service-time distribution beyond the mean.

The blocking probability can fortunately be calculated in a more efficient way than applying the formula (2) directly, to avoid the computations of the very large factorials, that become increasingly cumbersome with an increasing number of agents. Particularly, 171!, which would occur when working with call centers with more than 170 agents, cannot be represented with the most widespread floating point standard in use today, as stated by Qiao and Qiao [22].

The algorithm used has been presented in Qiao and Qiao [22] and is based on a rewriting of the Erlang-B formula:

$$E_B(m,\rho) = \frac{1}{1+\sum_{i=1}^{m}\left(\frac{m}{\rho}\right)\left(\frac{m-1}{\rho}\right)\cdots\left(\frac{m-i+1}{\rho}\right)} = \\ = \frac{1}{1+\frac{m}{\rho}\left(\cdots\left(1+\frac{2}{\rho}\left(1+\frac{1}{\rho}\right)\right)\cdots\right)}$$

The Erlang-C formula, or equivalently, the Erlang delay formula, calculates the probability that a customer has to wait to get served in a queueing system $M/M/s/\infty$, when incoming calls are put in a queue.

$$E_c(m,\rho) = \frac{\frac{\rho^m}{m!}\frac{m}{m-\rho}}{\sum_{i=0}^{m-1} m^i/i! + \frac{\rho^m}{m!}\frac{m}{m-\rho}}$$

One basic property of the Erlang-C formula is that $E_C(m,\rho) \geq E_B(m,\rho)$ for all m,ρ, which could be explained by the fact that in the $M/M/s/\infty$ model, if all servers are busy then the customers will be placed in a queue, and hence increasing the risk that later customers get blocked. This formula does not extend to the more general $M/G/s/\infty$ model, i.e., it is dependent on the service-time distribution beyond the mean.

The Erlang C formula can be calculated directly from the Erlang B formula, see for example Cooper [6], by

$$E_c(m,\rho) = \frac{mE_B(m,\rho)}{m-\rho(1-E_B(m,\rho))}.$$

The $M/M/s/\infty$ model for which the Erlang-C formula is defined, is considered as deficient or at least sub-optimal for many cases. This stems at least partly from the fact that it only handles exponentially distributed service times, often not the case in reality, as shown by Brown et al. [5]. Further, it ignores abandonment from the waiting queue (customer impatience) and retrials (new calls from the customers that abandoned the queue), both important factors when estimating the waiting time of the queue. The model $M/M/s/r+M$, often called Erlang-A, addresses the first of these issues, and is the subject of many papers regarding pure inbound call centers, see for example Garnett et al. [11], Mandelbaum and Zeltyn [18] and Whitt [27]. The $+M$ represents that each customer is associated with a patience parameter, in this case exponentially distributed. It was first introduced by a swede named Palm in the forties. We will only work with the Erlang-C formula here due to its relative simplicity, but any of the Erlang-A or its derivatives, as the $M/GI/N+GI$, should probably be used when a better model is necessary.

4.3 Discrete-Event Simulation

A general introduction to simulations can be found in for example Kleijnen [14] and Law & Kelton [16], the latter focusing more on so-called discrete-event simulation, which is our case. Further papers that are of particular interest

when working with queueing simulations are for example Srikant and Whitt [25], who treat so called bootstrapping of simulations, deciding which data extracted from the simulations are interesting to study and not under the influence of some initial instability of the system.

5 Related work

As is conveyed by Gans et al. as well as Korolev et al., the literature regarding predictive dialing is as good as nonexistent:

> In fact, we are aware of almost no academic work devoted to pure outbound operations, the exception being Samuelson. [1] [10]

> It should be noted that literature devoted to outbound dialing is very rare. We can only mention a paper [Samuelson][2] that presents only the problem statement but lacks a description of a solution. The reason might be that the subject was considered to be a no-no in the operation research society. [15]

The paper mentioned above, Samuelson [23] should rather be seen as an account of personal achievements with some additional problem statements, i.e., a story of how a problem was solved and led to a patent, together with some theoretical questions.

Korolev et al. seem to be unaware of one paper presented by Filho et al. [7] during WSC 2007, i.e., one and a half year before the publishing of their paper, where a simulation based algorithm is proposed, albeit in a very general manner, without going into details. The paper investigates the usage of a simulation model that runs in parallel with the call generator, and where the calculated optimal call rate from the simulation model continuously will be used as a decision basis for the call generator, while the data from the call center continuously will be reincorporated in the simulation model.

Most of the current academic research regarding call centers is related to inbound call centers. There are some papers that study blend call centers, but the outbound part is often neglected, at most mentioned in the introduction and in some auxiliary comments, and in particular not being pertinent to predictive dialing. If there is a lack of research that explicitly treats applications of theories to outbound call centers and predictive dialing, then there is (of course) a wealth of related research that could be applied when studying outbound call centers.

Gans et al. [10] gives a general and thorough introduction to contact centers, presenting general problems.

In addition to the papers already mentioned, Mehrotra et al. [21] presents some introductory details regarding call centers.

Mazzuchi et al. [20] have some pointers regarding how to implement the simulation per se.

See for example Whitt [26] and Srikant and Whitt [25] for an extensive treatment of simulation of queueing models and how to estimate the needed running length of the simulations.

[1] See Samuelson [23]
[2] Ibid.

Brown et al. [5] have found some interesting results about the distribution of service times, and criticise some of the most frequently used models of inbound calls, using as basis collected call data from a call center of a banking company.

The blocking probability of the Erlang loss model $M/G/s/0$ is known to be insensitive to the service distribution beyond the mean, see Sevastyanov [24]. It has been shown that in the case of a non-stationary (non-homogeneous) Poisson arrival process, this does not hold, and the distribution may have a strong influence on the blocking probability.

In the case of inbound call centers, the research is gearing towards quite complex models where customer impatience and retrials are taken into account, which are mostly variations of the Erlang-A model, $M/M/s/r + M$. Brandt and Brandt [3] study a $M(n)/M(m)/s/k + GI$ queue model, Garnett et al. [11] analyze a model $M/M/s/r + M$. Mandelbaum and Zeltyn [18] extended this analysis to the case $M/M/s/r + GI$, taking into account some of the findings of Brown et al. [4], later published as [5]. Other models have later been analyzed, probably influenced by the results of Brown [5]. Whitt [27] notes that $M/GI/s/r + GI$ is a good model but computationally intensive, and presents some approximations of the latter, analyzing a $M/GI/s/r + M(n)$ model as an approximation of the $M/GI/s/r + GI$ model. A good summary of different earlier models taking impatience into account is found in Brandt and Brandt [3].

There is a lot of reading regarding routing of calls (Skill Based Routing, SBR, value-based routing, etc.) and call center rostering, which theoretically has nothing do with this subject, but is generally interesting when designing an *Automatic Call Distributor* (ACD), responsible of distributing inbound and outbound messages and calls to available agents. See Gans [10] for pointers to such research.

6 Method and the algorithms

This section briefly discusses why simulation has been chosen to evaluate the algorithms, introduces various properties of the simulator and lastly presents the three algorithms that will be evaluated and compared.

6.1 Method

A choice has been made to evaluate the different algorithms by simulation of their respective implementations. Korolev et al. compute the values of the predictive algorithm from the formula they have described in their paper, but they evaluate their hyper-progressive algorithm by simulation, not knowing how to compute it analytically. The same could be said for the variations of the base algorithms here: they depend on some extra parameters, and are consequently more difficult to understand analytically. Furthermore, one more dimension of complexity has been added, having the algorithms take into account the aggregation of historical data and also the estimation of some of the input parameters to the algorithm, which influences the final outcome. Finally, by doing simulations for all the algorithms they work on equal terms and the results will be more interesting to compare.

In general, the simulation outcome is affected by the design of the simulator, so it is of course of great interest that the simulation comes as close as possible

to the reality, even though that might be a vain attempt.

Properties of the simulator

The simulator plays the role of the customers answering the calls made by the dialer (outbound), and the customers calling the call-center (inbound). The customers answers the calls made by the dialer with a certain probability, given as a parameter to the simulation. This parameter corresponds to the hit rate. Answering customers answer randomly within a 15 second interval, any other calls would in this case be ignored, since the dialer is configured to hang up any calls not answered within that time frame.

The service-time distribution used is exponentially distributed with mean 100. As already mentioned several times, Brown et al. [5] have shown that at least in the case of an inbound call center it would be better described by a lognormal distribution. For the outbound case, no extensive analyses of the phone call length have been found, but Samuelson [23] mentions observing a bimodal distribution in some cases, which probably could be explained by two groups of customers, those being interested and those not.

No attention has been given to the number of available phone lines, which effectively will put an upper bound on the number of outgoing phone calls, and hence on the busy factor. But from an algorithmical point of view this is an acceptable simplification (in this case), the algorithm will do its best and might be capped from above if reality offers limiting conditions.

The simulation of inbound calls is very simple: Calls are made by the simulator at random intervals, with an exponential distribution, to the call center. They are put in a waiting queue for a certain time if they cannot be handled directly. The calling customers will never hangup and will hence stay in the queue forever, if necessary, waiting for the dialer distributing it to a free agent. As such, it is a bit simplified compared to reality, where some of the customers would hangup while waiting. This goes by the name of customer impatience in the literature. Nor are retrials from customers that have already abandoned the queue once taken into account. However it might of course be possible (and of interest!) to improve the algorithm by taking into account these dropout values as well.

Customer impatience is treated by Garnett et al. [11], and many others, according to whom a model with hazard rates (called Erlang-A and denoted $M/M/s/\infty + M$) gives a notable improvement over other classical models, even "far superior" as stated by Garnett et al. In that way it would also be superfluous to experiment with those more advanced models when simulating a blend center, since the simulator would be less complex than the algorithm.

Other notes on the simulation

As stated by Srikant and Whitt in for example [25], when simulating a queueing model, we can expect an initial period of the simulation where the process approaches its steady-state and hence should be excluded from the result, followed by a period where the process has attained its steady-state. This method is called bootstrapping. In the measure that one can talk about a steady-state in a process where the parameters change continuously, some consideration has

been given to bootstrapping, though not with the same rigor as described by Srikant and Whitt.

As a side note, an example of an implementation detail of the simulator that was found to make a substantial difference (to the worse from the perspective of the algorithm) to the performance of the algorithm during the simulations, was whether the so-called phone calls were answered immediately or rather were answered randomly with a uniform distribution during the 15 seconds that followed the instant the call was made. To make the simulations somewhat more realistic, another probability distribution should probably have been chosen than a uniform one, but it has its points anyway, and the choice is not obvious.

6.2 The base algorithm

Korolev et al. [15] proposes an analytic solution to find the call rate based on events in a relatively small time frame in history. It is recalculated each time calls are about to be generated.

The general idea presented by Korolev et al. is as follows: we regard the outbound call center's answering customers as the arrivals of calls to an inbound call center with zero length queue, i.e., a call center where a call is immediately disconnected if there is no free agent to answer. This is a reasonable way of seeing things: if we assume that we produce outgoing calls at a constant rate (explanatory example), once in a while a customer will answer, and some agent will be connected to the call, as if the customer was calling him/her. We can expect different number of answers during different times of the day, and if a lot of customers answer at the same time, this will generate a lot of simultaneous connections to the waiting agents. If all agents are busy, an answering customer will generate a connection without an agent, and will directly be disconnected. This is also referred to as a *multi-server loss system*, the agents being the servers. *Multi-server*, since there are multiple agents, *loss system* since any excess of calls will be lost. It is generally accepted that the arrival rate of an inbound call center is best modeled as a non-homogeneous Poisson process, which could maybe be guessed from the example above. In this case the answered calls by the customers constitute the events which occur in the Poisson process, whose rate is influenced by the customers possibility to answer the calls, as well as by the outbound call flow. This makes it possible to alter the Poisson arrival process rate, the key of the algorithm. This rate should be adjusted so that it matches the rate of customers leaving the system, i.e., hanging up after having talked to an agent. Hence we try to adjust the call rate such that the calls finally being answered by the customers, and thus have to be taken by the agents, represent a homogeneous Poisson process, and since we know how many agents there are, we know the optimal arrival rate of the homogeneous arrival process.

In an m-agent call center, this is described as a $M/G/m/0$ system, and since the queue is always empty, the Erlang-B formula can be used to find the probability of all agents being busy, a standard formula used in queueing theory.

So, more or less directly from Korolev et al. [15]: Let λ be the rate of the Poisson process representing the answered outbound calls, T_s the average service time, then $\rho = \lambda T_s$ is the current traffic offer. The blocking probability in a m-agent call center is then given by the Erlang-B formula, and it should be less than $A_{\max}$, the upper admissible level of the abandon rate (since the blocking probability will translate directly to the abandon rate). Thus,

$$B(m, \rho_m) \leq A_{\max}, \tag{3}$$

where we just want to maximize ρ_m, the maximal traffic offer that can be handled by an m-agent call center such that the abandon rate is less than $A_{\max}$, since the number of (free) agents m are fixed. If we let λ_i be the rate of the Poisson process representing the inbound calls, and p the hit rate, we get

$$\rho_m = (p\lambda_m + \lambda_i)T_s, \tag{4}$$

So to find the new call rate, we estimate p, λ_i and T_s from recent data, and we get ρ_m as the maximum satisfying (3), and we solve (4) with respect to λ_m, the new call rate adapted for a call center with m free agents. For more details, see Korolev et al.

In reality it would be more correct to see the arriving calls as the events of a non-homogeneous Poisson process, described by a non-stationary queueing model $M_t/G/s/0$, where the arrival rate depends on the hit rate, and, with a slight time lag, on the number of outgoing calls made. Since we can't exactly know how many customers who will answer a call, we will see changes in the arrival rate all the time, even though we try to mitigate those variations. They depend on incorrect calculations of the parameters of the algorithm, estimated from the aggregated data, as well as incomplete knowledge of all the incoming calls. The changes in the arrival rate will in its turn generate mismatch calls when increasing, and, correspondingly, generate free servers when decreasing.

The algorithm was pseudo-implemented as follows:

Algorithm 1: Algorithm as adapted from Korolev et al.

Data:
p - the hit rate
μ - parameter of service distribution
δ - the dial frequency
λ_i - inbound call flow
A_{max} - The maximum abandon rate
m - the number of free agents
repeat
  if *is time to make calls* **then**
    calculate p, μ, λ_i from aggregated data
    find greatest ρ_m such that $E_B(m, \rho_m) < A_{max}$
    calculate new outbound callflow $\lambda_m = (\rho_m \mu - \lambda_i)/p$
    generate new Poisson distributed variate $N_c = f(\delta, \lambda_m)$
    dial N_c new numbers
until *end of time*

There has been research regarding the distribution of service times in inbound systems by Brown et al. [5]. For inbound traffic Brown et al. find that the service time has a lognormal distribution, however it wouldn't be too surprising if this was substantially different in the case of outbound traffic. When such information is available a more suitable distribution should be used.

The algorithm could maybe be improved upon if the information about the distribution of the outbound service times was used (it could be hard in the case of an exponential distribution), i.e., knowing that after a short presentation in the phone call a customer will either hang up or talk for quite a long time is, just as an example, something that clearly could be exploited.

6.3 Modifications of the algorithm

Two modifications of the algorithm will be studied.

- Blend center: An alternative model that takes into account the fact that incoming calls do not hangup immediately if all lines are taken, but rather may wait.
- State dependency: Inhibit outgoing calls directly if for example the level of occupation of the agents (this is what actually will be used in the simulations), or some other parameter, becomes too high (one could also take the same approach in the case of a lower limit). Hence we add a dependency between the call generation pace and the number of occupied servers (or some other parameter).

Blend inbound/outbound center algorithm

The algorithm with modifications done to handle the enqueueing of the inbound calls:

Algorithm 2: New algo based on Erlang-C

Data:
p - the hit rate
μ - parameter of service distribution
λ_i - inbound call flow
A_{max} - The maximum abandon rate
S_{min} - service level
δ - the dial frequency
m - the number of free agents
ρ_i - the inbound traffic offer
repeat
  if *is time to make calls* **then**
    calculate λ_i, p, μ, ρ_i from aggregated data
    find smallest n, the number of agents needed to answer the inbound flow, such that $E_C(n, \rho_i) > S_{min}$
    find greatest ρ_{m-n} such that $E_B(m-n, \rho_{m-n}) < A_{max}$
    calculate new outbound call flow $\lambda_{m-n} = (\rho_{m-n}\mu - \lambda_i)/p$
    generate new Poisson distributed variate $N_c = f(\delta, \lambda_{m-n})$
    dial N_c new numbers
until *end of time*

The new algorithm, algorithm (2) proposed is very simple: it uses the Erlang-C formula to estimate the number of agents that have to take care of the inbound calls, with respect to the current service quality or chosen acceptable enqueueing probability. Those agents are then excluded from the calculation of the new outbound call flow. Since the model proposed by Korolev et al. supposes that inbound calls either are answered or refused instantly, we should expect this to perform better, in the way that the new model is closer to reality, where someone making a call does not hangup if not answered immediately. It should be noted that all agents may receive inbound or outbound calls, i.e., no agents are in fact locked to exclusively handle inbound calls, but the call rate is adjusted as to allow for agents being busy with inbound calls. It should also be noted that outbound calls to some extent have priority over the inbound calls. This means that after the current outbound calls have been handled, if there are free agents left, they will be given the inbound calls. In the next "round" this has a negative impact on the abandon rate, since more agents will be occupied with an active call.

State dependent algorithm

Since the algorithm is completely dissociated from the current server state, the chance of mismatching calls increases, i.e., whatever that happens in the call center at the moment of calling is unknown to the algorithm and is not available until later, in an aggregated form, like the average service handling time or the average hit rate. Certain information is never used by the algorithm and doesn't fit to be used in some aggregated form, but could be useful anyway. There could be a gain in not only react to changes in recently things that has happened but also react to the current state of some information.

And further, as of now, the arrival rate does not depend on some factors internal to the queuing system (number of customers being served, number of customers in queue) but rather depends on an external factor, being the hit rate and its changes.

What we look for is a model where the arrival rate is dependent on some factor in the system, and of this model we should normally see a blend of two different states, one which is the usual Poisson process, and one which is a modified state, having a close to zero arrival rate.

Several solutions are probably possible, the first thing that springs to mind would be to decrease the call rate when the server occupation is very high, i.e., when it is probable that any action taken by the dialer will generate mismatch calls. This could be done in a positive direction as well: if there is a low number of active agents, increase the call rate with a certain percentage.

Several successful analytical approaches of state-dependent queueing systems exist, see for example Brandt and Brandt [2], which focuses on queues where the state depends on the number of customers in the system, either being served or waiting in the queue. Their research is however done in another setting, they study the interdependency between customer impatience and the arrival rate, but the general idea is relevant to this case as well.

It is customary to use the notation $M(n)$ for a queueing model to imply dependence on some parameter, and the queueing model would in this case be $M(n)/M/s/0$. Here we will focus on the case where the arrival rate is dependent on the number of customers in the system, or equally, the number of busy servers. It is of interest what happens to the blocking probability when we decrease the call rate if too many of the agents are busy, using a chosen limit c that defines the admissible level. So in our situation all the states, if we consider a state as being defined by the current number of calls in the system, are equal except when all or almost all the servers are taken, i.e., if m is the number of servers, then we have that $M(0) = M(1) = \cdots = M(m-c-1)$, and only $M(m-c), M(m-c+1), \ldots, M(m)$ differ, being lower. We can't expect $M(m) = 0$, since calls already have been made, and could be answered by a customer during a certain time all servers are taken, however the probability is lower. (Because of the earlier mentioned rule that we imposed upon ourselves that a call already made is not disconnected before a certain time has passed)

The algorithm has been implemented as follows.

Algorithm 3: New algo based on Erlang-B, dependent on the service load

Data:
p - the hit rate
μ - parameter of service distribution
λ_i - inbound call flow
τ - the number of cycles to inhibit calls
δ - the dial frequency
m - the number of free agents
repeat
  if *is time to make calls and calls are not inhibited* **then**
    if *#free agents < free agents minimum* **then**
      inhibit calls for τ cycles
      continue
    calculate λ_i, p, μ from aggregated data
    calculate the number n of agents needed to answer the inbound flow
    find greatest ρ_{m-n} such that $E_B(m-n, \rho_{m-n}) < A_{max}$
    calculate new outbound call flow $\lambda_m = (\rho_{m-n}\mu - \lambda_i)/p$
    generate new Poisson distributed variate $N_c = f(\delta, \lambda_m)$
    dial N_c new numbers
until *end of time*

6.4 Data aggregation

The parameters of the equation presented by Korolev et al. have to be estimated from current and historical data, as already mentioned. This is a sufficiently important part of the dialing to be seen as a part of the algorithm. Korolev et al. [15] leave out the estimation/approximation of the variables upon which the formula will depend, in particular what interval of time that has been used in the approximation.

The service talk time may differ depending on the time of day, just as well as the hit rate. It may also depend on what weekday it is, in the way that Friday evenings recurrently see lower hit rates, and special holidays and vacations may see other patterns. There could be a great benefit of being able to use those patterns. We will only interest us of how to calculate the input parameters based on the most recent historical data.

In all cases we need some way of estimating the parameters of the outbound flow and the service time. During all operation time, call data will be aggregated over specific time intervals, from which the new call flow later on will be estimated. This is a common approach, and has been shown to give a good approximation if the intervals are correctly chosen in proportion to the amount of data (calls etc.). Various problem might however arise: overfitting, when data is given to much importance due to the chosen interval being too short, and smoothing, when it is given little importance. See for example Henderson [12] and Massey et al. [19] for discussions.

The interval between aggregations can be chosen to be fix-sized or varying. In the case of fix-sized intervals the duration might affect the general algorithm through smoothing or overfitting. Smoothing occurs when the fix sized time interval is too long, and changes in the data are ignored because of the amount of data that is included in the calculation of, for example, the mean. (This will not occur when the result only depends on one value, like the maximum or similarly) However a too short time interval will lead to overfitting, i.e., local spikes and variations are given to much importance, and the rate might even be estimated to zero, since no new events have happened in the short interval during which data has been aggregated. A zero rate would effectively prevent a single event (call) from occurring.

The span in which these effects occur would be interesting to investigate, as well as how the period of the call cycle and the choice of moments to dispatch calls from the dialer affect the outcome.

Some assumptions regarding the algorithms

- Zero length queue, since no customer will wait when getting a silent call. In reality the admissible queueing time will be around 1 second, the time it takes for a customer to lift the phone to the ear, but this will in general only have a very small impact on abandon rate and busy factor (slightly positive for the abandon rate).

- Talk-time and after-call work time will not be separated, both will be included in the service time. This will have some consequences in reality, since trunk lines will be freed when the agent does not talk any longer, and there is nothing like an infinite amount of trunk lines except in theoretical models. This could probably be used for some optimizations in a production environment, but it will not be taken into consideration here.

- Usually, predictive dialing is limited by the number of available phone lines, which normally has to be significantly greater than the number of agents. This will not be touched upon, but should be kept in mind when designing a system for predictive dialing (since phone lines generally cost money). This is not possible to overcome through some sort of optimization without starting to fiddle with the waiting time of the call.

Implementation details and problems

Some variations were tried with the output from the algorithm. The amount of numbers to dial each round, that is, each time after the algorithm had been run, was either taken as is, as well as taken to be the mean of a Poisson random variable. The second approach has been used throughout all the simulations. There was trouble using the first approach; taken as is it generates too few calls, and when starting from a fully occupied state the activity will decrease to a lower level. If statically adding a small number of calls dependent upon the total number of agents each time, the occupancy was higher, but the relation abandon rate/busy factor was much worse than when choosing the number of calls randomly. So in some way (it could be speculated), the good results depend on the fact that more calls than expected are generated sometimes, but not very often, thereby increasing the overall occupancy level.

Most of the algorithms used to generate variates with different distributions that have been used (and that had to be implemented) in the simulations come from Devroye's book [8]. The implementations have undergone basic verification of correctness, mean and variance as well as Kolmogorov-Smirnov tests.

7 Simulations

In this section we present the different simulations that will be conducted with each of the three algorithms. The parameters to test in relation to each other have been selected in a more or less intuitive manner, and would have been done differently, by some method described by for example Kleijnen [14] to get a better and more structured approach.

Here follows a list of all the simulations that will be carried out with the different algorithms.

Outbound traffic only

In the purely outbound case, ($\lambda_i = 0$ in algorithm 1 and 2), the following simulations have been done with the Korolev algorithm:

- The demanded abandon rate and the number of agents are put in relation to the busy factor, just as a comparison to the paper by Korolev et al. How does an actual implementation of the algorithm compare to the expected behavior?
- The demanded abandon rate and the number of agents are put in comparison to the actual abandon rate - how closely do the calls generated by the algorithm follow the expected value? It is the same case here as above, we can compare this with the results presented by Korolev et al.
- Time span of collected data in relation to the actual abandon rate - what happens as the time frame increases respective decreases? Does smoothing occur? Overfitting? To what extent?

Inbound/outbound

Simulations that will be done with the blend inbound/outbound algorithm 1 presented by Korolev et al., where all responses are instantaneous or the call is supposed to be discarded. The calling customers of the simulator are as already stated put in a waiting queue, thus the simulation might not be really fair.

- Demanded abandon rate and inbound flow in relation to the actual abandon rate. How does the algorithm perform with increasing levels of inbound calls? Here a fixed number of agents will be used.

Simulations that will be done with the modified algorithm 2 that takes into account the inbound traffic that is queued up:

- The demanded abandon rate and the enqueueing probability/service quality in relation with the actual abandon rate. How great of an impact does an increasing service quality have on the abandon rate generated by the

outbound calls made? Does the algorithm give a good estimate on how many agents that are needed to answer inbound calls with the current service level?

- The demanded abandon rate and the enqueueing probability is put in relation to the busy factor. Same questions as above. Here we should see a clear decrease if to many agents are expected to be needed to answer inbound calls, since then the algorithm will lower the pace of the outbound calls.

State-dependent algorithm

These simulations will be done with the algorithm 3, in a pure outbound scenario:

- The demanded abandon rate and the level of the state dependency are put in relation to the actual abandon rate. Does coupling the algorithm to the current state (in the way that has been proposed by algorithm 3) have some benefices?

- The demanded abandon rate and the level of the state dependency is put in relation to the busy factor. How big is the negative impact on the busy factor?

8 Results and discussion

In this section the results of each simulation are presented, followed by a comparison between the results of the different algorithms. Finally a short summary of performance results presented in other papers are compared to those found here.

8.1 Base algorithm, pure outbound

The range presented here is 20 - 100 agents, and demanded abandon rates between 0.2 % - 2 % (since it's doubtful that anyone would be interested in allowing an abandon rate of, say, 30%). As can be seen in figure 1, the actual abandon rate is consistently higher than the one asked for, no matter the number of agents. This shows that for example the estimations of the service time and the hit rate bring with them some further negative effects in addition to the ones that follows from the equations defined by Korolev et al. In the case of the busy factor presented in figure 2 there is nothing new, we can see that the algorithm adjusts the call speed to maintain the abandon rate, to the detriment of the busy factor.

We can thus conclude that those basic results conform roughly to what has already been presented by Korolev et al., with some modifications due the simulation environment and the actual implementation of the algorithm, i.e., the fact that the parameters to the algorithm really have to be estimated.

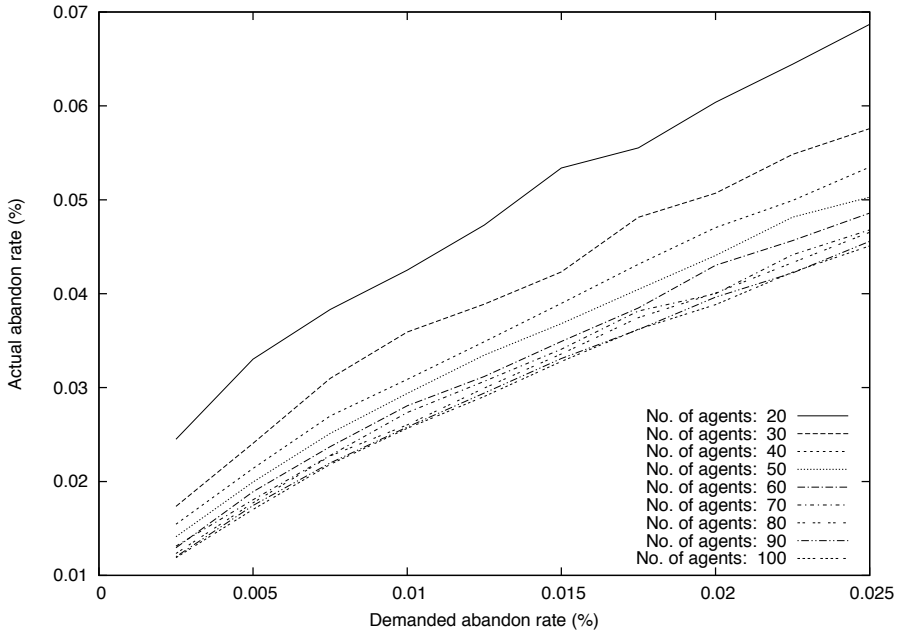

Fig. 1: The actual abandon rate of the base algorithm. Cf. Korolev et al. [15]

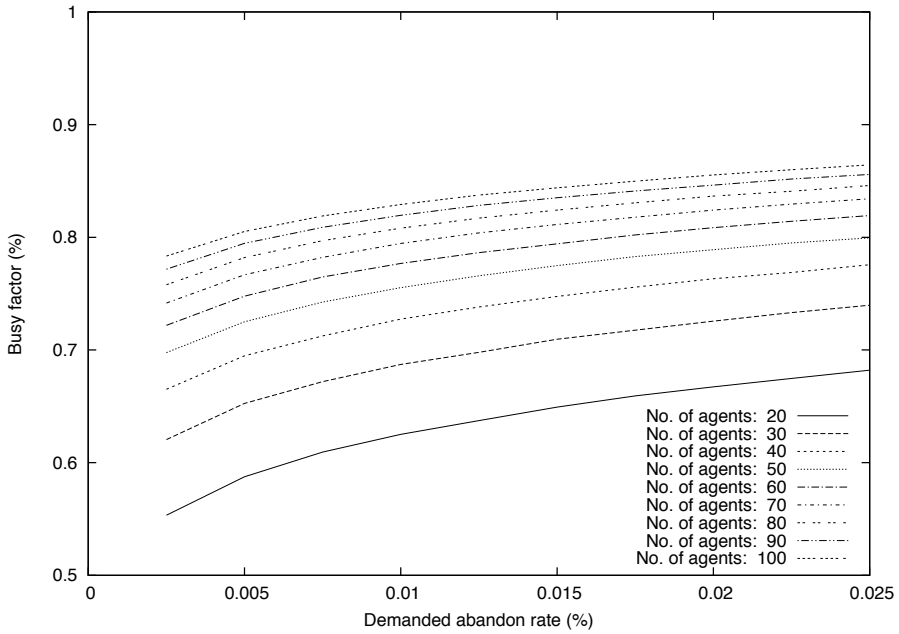

Fig. 2: The busy factor of the base algorithm. Cf. Korolev et al. [15]

8.2 Comparison progressive vs. predictive

We can see from figure 3 how it compares with the progressive dialer. Note that the x-axis in the progressive case (left hand side) represents increasing levels of

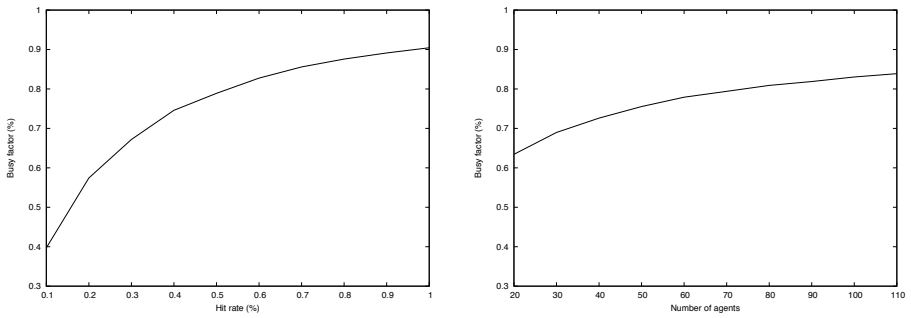

Fig. 3: a) Busy factor when using the progressive algorithm with increasing hit rate. b) Busy factor when using the predictive algorithm with increasing number of agents.

hit rate, while in the predictive case (right hand side) it represents increasing number of agents. As we already know, the progressive dialer is independent of the number of agents, and the predictive dialer is independent of the hit rate (it is sensitive to changes of the hit rate, but the results shows in figure 3 are the result of runs with constant hit rates). We can see that at a certain lower limit of the number of agents, and at a certain upper limit of the hit rate, the progressive method outperforms the predictive, and since the predictive method always brings with it a small but not negligible amount of mismatch calls, the progressive method is to prefer in these cases. This conforms to what we already know, that the predictive method does not work well with few agents, and that the progressive method improves with an increasing hit rate while the predictive does not. The result of the progressive method is similar to what equation (1) gives, with a mean service-time of 100 t.u. and average service times of 107 t.u. and 15 t.u. in the successful and unsuccessful case, respectively.

Worst-case scenario

The algorithm has been run both with a fixed and a variable answer rate (hit rate), and it adapts the changes of the hit rate quite well. The worst case occurs when the rate has been constant for a duration that equals the time length of the bins used for data aggregation followed by a sudden increase. This can partly be counterbalanced by adding state dependency to the algorithm, as will be treated later, but the base problem remains, the algorithm is not capable of handling quick changes in the call rate, due to smoothing.

Figure 4 shows the behavior of the algorithm vis-à-vis the abandon rate, with a hit rate that has been modeled by a sine curve whose period ranges from 100 - 1200 (time units). It is possible to see that there is some kind of correlation between the length of the time bins and period of the sinusoidal hit rate.

This could maybe be mitigated by having an adaptive aggregation frequency, dependent on the derivative of the perceived hit rate. i.e., when the hit rate is changing, the length of the buckets should decrease so that the data used by the dialer is updated more often and thus it can react faster to changes. Adaptive aggregation of data is, as already mentioned, also useful for avoiding zero values when activity is low. But then again, this would make the algorithm more sensitive to local hit rate and service time changes, a problem that also

Samuelson [23] has encountered.

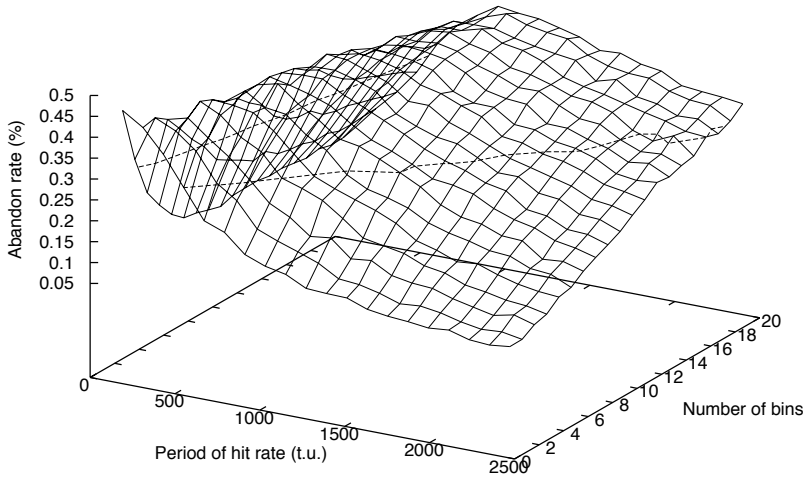

Fig. 4: Worst case scenario with an increasing period of a sinusoidal hit rate function. A 30 % marker line is shown on the surface.

8.3 Blend center

When the outbound dialer is a part of a blend center, where customers are calling to the call center as well as being called to, the conditions are a bit different. The outbound algorithm constantly has to take measures for the inbound calls.

Old blend center algorithm

The algorithm presented by Korolev et al. performs well in the blend center case under the assumption that inbound calls will be answered instantly or rejected if no agent is free, i.e., that there would be a certain amount of inbound mismatch call. This is seldom what is sought after, inbound calls will rather be queued up in a waiting queue if there is no free agent available. This ability has been included in the new algorithm. Since the simulator queues up incoming calls, and since the Korolev algorithm was not really created to handle that case, the results were quite uninteresting and have been excluded.

New blend center algorithm

The new algorithm, algorithm 2, performs better in the situation where inbound calls are queued up if they are not answered directly. This method was tested both with an exponential and lognormal service time distribution.

The new model performs quite well for its simplicity, both with respect to the abandon rate and busy factor. The outbound call flow is decreased, taking

into account the inbound call flow, and if the inbound flow is high enough, the outbound call flow might be stopped more or less completely. This should optimally be taken care of by the algorithm, since the predictive algorithm performs very badly with a small number of agents. However, what is interesting is of course that the algorithm balances the outbound/inbound flows in such a way that the inbound flow is handled correctly, the abandon rate stays at its normal level and the busy factor as well, even though in this case the agents might be busy answering inbound *or* outbound calls. This can be seen in figure 5 and 6.

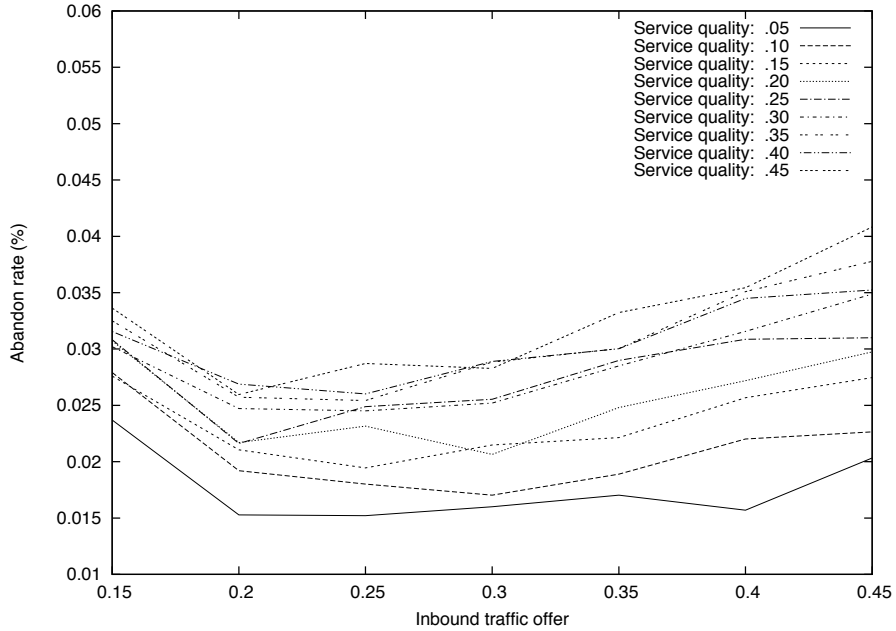

Fig. 5: The abandon rate of the new blend center model.

The leftmost values should be ignored, as they are caused by the old algorithm being used when the inbound traffic is very low, because of the calculation of the Erlang-C formula in my implementation becoming increasingly cumbersome for very low values of inbound traffic. Apart from that, what can be seen is that the Erlang-C formula slightly overestimates the number of needed agents to take care of the inbound calls when the number of agents is large, thus resulting in a decrease of the abandon rate. Since more agents are supposed to take care of inbound calls than necessary, fewer agents are supposed to be available for outbound calls, and thus fewer outbound calls are made. Thus the overall flow of calls, inbound as well as outbound, is lower. It would be better if the abandon rate stayed the same, since this would mean that the algorithm 2 has made a good partition of the number of agents handling inbound calls vs. the number of agents handling outbound calls. It is of course satisfactory if the abandon rate is low, but this rate should be decided by the outbound part of the algorithm, and not as a side effect of an overestimation of the number of inbound calls.

So the new algorithm works well when the inbound calls have an exponential

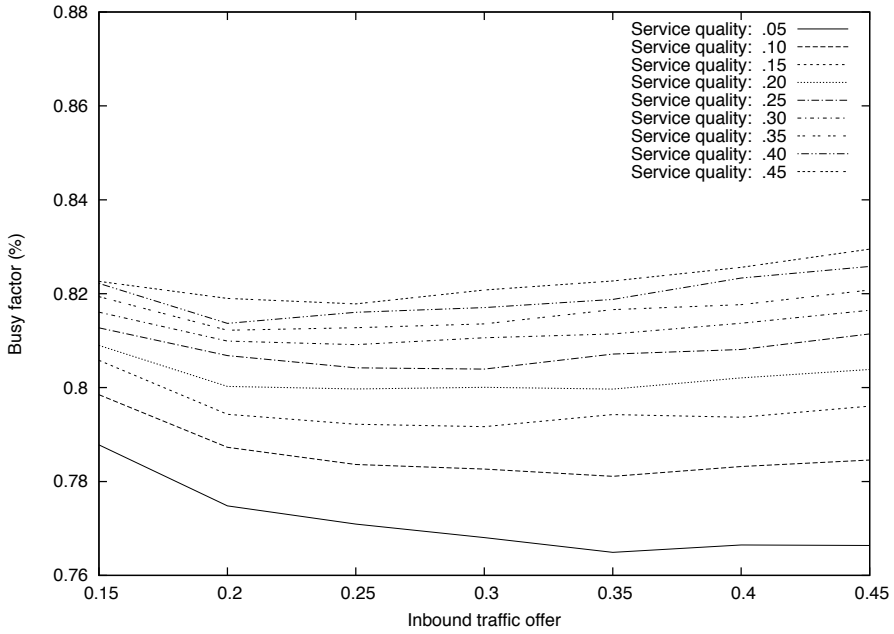

Fig. 6: The busy factor of the new blend center model.

distribution, however we should expect them to be lognormal, as stated in Brown et al. [5].

No greater difference was observed on the abandon rate and busy factor between when the simulator produces lognormal service times versus exponential service times, in the case when the parameters of the lognormal distribution were chosen such that it fits quite close to an exponential distribution. i.e., meaning that it only differs at the left end of the distribution, representing the fact that very few calls tend to be excessively short. In another context however, Brown et al. also showed that the Erlang-A model ($M/M/s+M$) was sufficiently good even though the sampled call data was not exponential. The same conclusion could maybe be valid here.

It might be that the queue length varies when switching between the two service time distributions. Since we treat performance of the outbound dialing in this paper, no time has been put into analyzing queue lengths and optimization of the flow of the inbound calls.

However, if the parameters to the lognormal distribution are chosen in such a way that the shape of the probability density function not even closely resembles the probability density function of the exponential distribution, then there is a big difference, which is expected. As Korolev et al. state, the service time parameter does have a big influence on the call flow. Hence the performance of the algorithm 2 depends on the properties of the inbound call data and how closely it resembles the exponential distribution. The solution is to use any of the more advanced inbound models instead of Erlang-C when calculating the number of agents necessary to handle the inbound calls, and is as such not a problem.

8.4 State dependent call rate

The aggregated data from the simulation runs shows that abandon calls has a somewhat burst-like arrival manner. I.e., once the calculation is bad, the algorithm subsequently does bad decisions and continue to call in the same rate during a certain time. A short inspection of the aggregated data from the simulation data shows that the lengths (measured in bins of aggregated data) of the bursts are somewhat exponentially distributed, so about 40 % has a length longer than 1, depending on the rate parameter. This is demonstrated in figure 7, where the three different boxes represent the number of mismatch calls in the case of not cutting the outgoing call flow at all, and cutting it when the number of free agents reach zero and one, respectively. One can see that the tail gets shorter when one starts to cut the call flow.

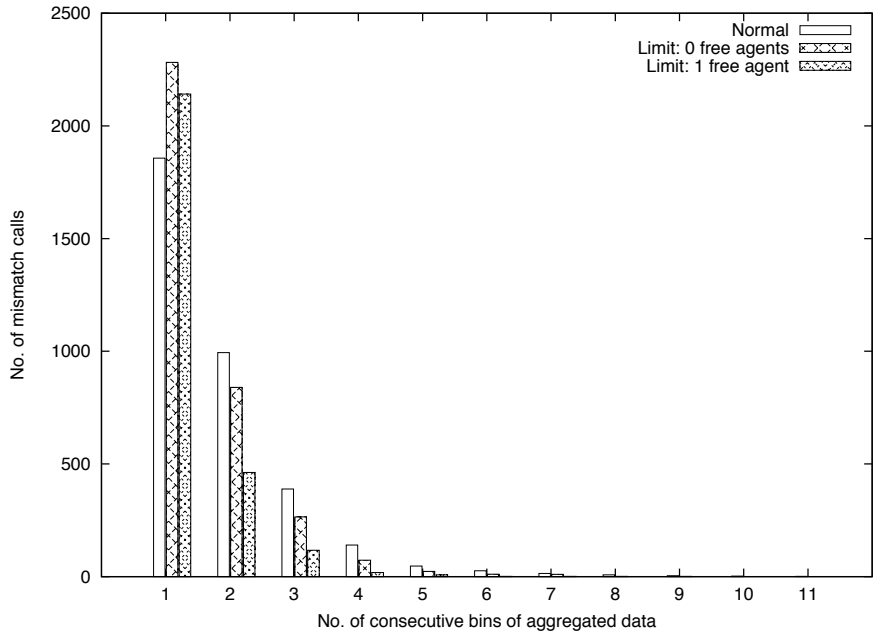

Fig. 7: The number of consecutive bins with aggregated data that contains mismatch calls.

This problem is addressed by algorithm 3. The fact that the base algorithm continues to make calls even when there are no agents shows very clearly that it doesn't depend on the state of the system. As we can see in figure 8 and 9, the state dependent algorithm has a positive effect on the abandon rate, decreasing it by 50 % or more, and has only a slight negative effect on the busy factor (about 1 % when the limit is 0 agents), which further implies the burstiness of the abandoned calls.

It is of course highly desirable that the algorithm is not too tightly coupled to the current state of the servers, the advantage of the algorithm lies in how it can react in advance, and not after, server events. However, this shows indeed that much better performance can be achieved by making the model slightly dependent on the current server state. Some other possible modifications spring to mind, for example applying exactly the same approach but in a positive

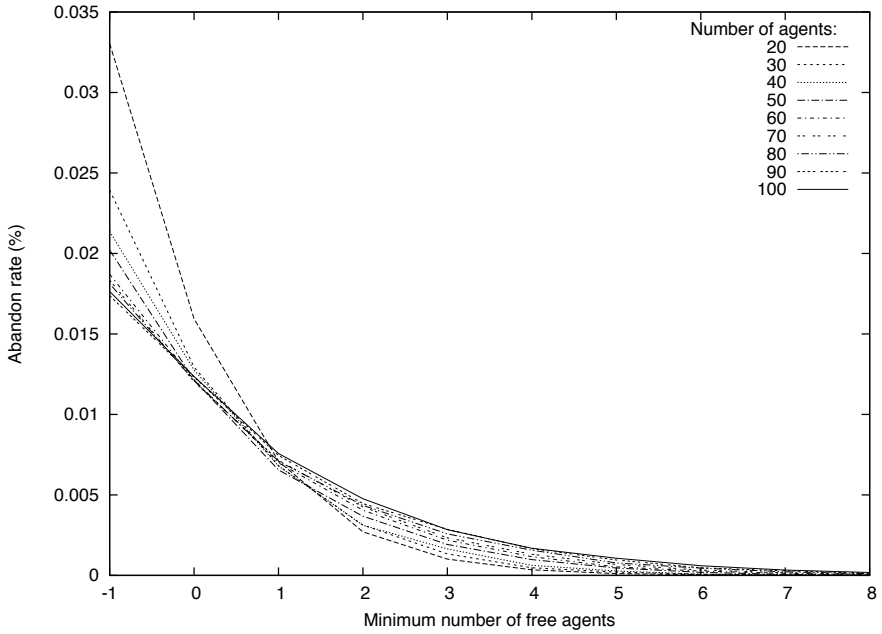

Fig. 8: The abandon rate with state dependency. -1 denotes that no limit was set.

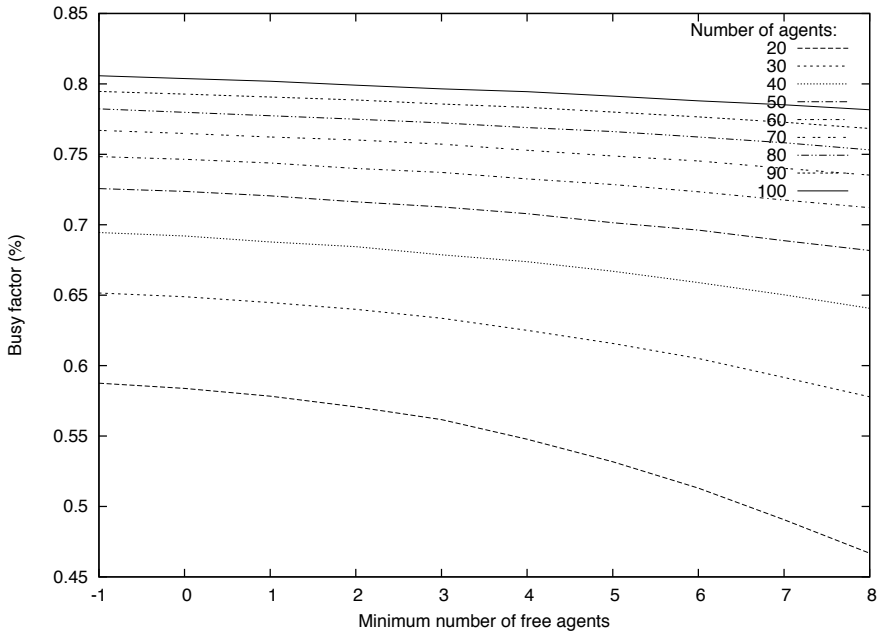

Fig. 9: The busy factor with state dependency. -1 denotes that no limit was set.

manner, i.e., increasing the call rate by a certain percentage when the number of agents is very low.

It is also a fact that when all agents already are taken, calls made during a

period starting 15 seconds before that state have already been done, and might have been answered by the customer. So it is interesting to see how the busy factor is deteriorating while the abandon rate is getting better, for more values, i.e., stopping the outbound calls when one or more agents are free.

8.5 Further discussion and comparison

With the additions to the algorithm mentioned above it could be interesting to see how it compares to the different results presented in other papers, already mentioned here; by Korolev et al., Filho et al., and Samuelson. The comparison will in all cases be done with algorithm 3, since everything that has been presented in other papers only covers the outbound case.

It should be noted that only Samuelson and Filho presents results from a real call center, unfortunately without specifying the number of agents in it. Thus, their results should be given some extra weight, and the comparison should only be seen as giving a hint of how the algorithm performs.

Samuelson mentions a busy factor between 83 - 90 %, with an abandon rate below 3 %, which he says agrees to what he had expected. In comparison, the implementation used here, with results only valid in simulations, attains a busy factor of 79 % and 3 % abandon rate for 30 agents, 81 % busy factor and 3 % abandon rate for 40 agents, and 88 % busy factor and 3 % abandon rate for 100 agents. It is quite probable that this would be worse if run under real conditions.

In contrast, Korolev et al. have only calculated the busy factor as a function of the abandonment rate and the number of agents by solving the given equations. A 3 % abandon rate then corresponds to busy factors 69 % and 83 % in the cases of 30 and 100 agents, respectively. As written above, 30 agents with 3 % effective abandon rate gives a busy factor of 79 % with the new algorithm, and 86 % with 100 agents and 2 % abandon rate.

Filho et al., just as well as Samuelson, leave out information regarding the number of supposed agents in the call center (even though this surely has an impact on the given information), and they present an abandon rate of 3 - 5 % giving a busy factor of 85 - 92 %. In our case this compares to 79 - 84 % for 30 agents and the same abandon rates, and 88 - 91 % for 100 agents. They have, on the positive side, run their simulations on data collected from real call centers, just as Samuelson.

The results of the simulations thus seem to be consistent with what Filho et al. and Samuelson have found.

9 Conclusions

Three algorithms based on a predictive dialing algorithm for a inbound/outbound call center presented by Korolev et al. have been analyzed by simulation in this paper.

The results of the simulations of algorithm 1, based directly on the original algorithm, without any inbound calls, came very close to the analytic results presented by Korolev et al., and as such did not surprise. The results from the simulations were slightly worse, but this could possibly be attributed to the

circumstances of the simulator and how the algorithm actually computes some parameters from call data.

When simulating the same algorithm with inbound calls, it performed much worse. This bad performance could probably be attributed to the fact that the original algorithm uses the Erlang-B formula for the inbound calls as well, in combination with the simulator implementation that have been used, that queues up the inbound calls not answered instantly.

Algorithm 2 in this paper is identical to the first algorithm, except that the Erlang-C formula is used instead of the Erlang-B to decide how to handle the inbound calls, given a chosen quality of service. In the simulations, this algorithm performed better with respect to the abandon rate and the busy factor when handling inbound calls. However, the abandon rate increases moderately when the service quality of the inbound queue increases, indicating that the balancing of the handling of the inbound and outbound calls could be improved upon.

Algorithm 3 is similar to the original algorithm with the difference that it depends partly on the current service load of the call center. It will stop making new outbound calls if there is only a certain number of free agents in the call center left. This has a positive impact on the abandon rate and does not lower the busy factor beyond an acceptable level.

The busy factor and the abandon rate of algorithm (3) roughly agrees with results presented in two papers apart from Korolev's, where other algorithms have been employed.

10 Further Research

The most interesting path to go would of course be to try the algorithm on some data collected from a call center, and later on use it in a production environment.

It would be fruitful to compare these results with for example a simulation-based algorithm as proposed by Filho et al. [7], to see if, as he states,

> The circumstances in which predictive dialers operate are very dynamic, non-deterministic and with a high number of stochastic variables to deal with. In such a world the use of mathematical (analytical) models to define the best dialing rate will not work well. [7]

i.e., the problem is too complex to be handled well by an analytical algorithm. It has been shown that this algorithm can keep its promises regarding the abandon rate in a simulated environment, so a comparison between this algorithm as an example of an analytical one and another, that uses simulation to control the call flow, on the same set of collected or simulated data would be very interesting.

Brown et al. [5] have presented some interesting conclusions about the distribution of the service times of inbound calls. A similar research about the service times of outbound calls would be interesting. Samuelson [23] mentions that in some cases the service time had a bimodal distribution when he experimented with his algorithm, this kind of general results would be profitable. It could maybe be exploited by a dialer algorithm, adjusting the call rate when the probability that one or more calls soon will end.

The state dependency approach was successful, however not very formalized (nor rigorous). It would be interesting to formalize the corresponding model to a greater extent.

It would be interesting to improve upon the inbound part of the model to handle the inbound calls, taking into account customer impatience (hazard rates) as well as retrials. More specifically, using an Erlang-A model ($M/M/s/r+M$) as for example outlined by Garnett et al. [11], or some even more advanced model like approximation models of the $M/GI/M+M(n)$ model, as presented by Whitt [27], that supposedly is better than the Erlang-A model. That is also a very interesting subject, since there is a lot of research efforts in that area, and thus a lot of things happening.

Extending the data aggregation to take historical data into consideration by means of some weighting system, i.e., data from the same weekday from earlier weeks, seasonal data and special occasions, etc.

11 Changes to the thesis and general ideas

During the course of the writing of this master thesis the weight has shifted more and more from some of the original questions to a more pure investigation of a few algorithms for predictive dialing.

One of the first question that I added to the problem statement was whether it was feasible or not to expose the dialer as a web service. I've realised during the writing that this question had more to do with response times than with the actual dialer. Since then, I've focused more and more on the algorithm (or the implementations thereof) in itself, and how different parameters influence the outcome, more in-depth than what the original authors did.

If I would have started writing this master thesis now (that is, *after* having written everything else), I would probably have used some more organized approach to the simulations (I thought I was organized when I started), in the choice of parameters of which to evaluate the algorithms sensitivity, and applied several of the methods presented by for example Kleijnen [14], and Law and Kelton [16]. But first a foremost, I would not have implemented my own simulator, but used some kind of existing simulator framework, saving both time and pain. Now the simulations have been done in an intuitive manner and some combinations of values of the parameters that drastically change the performance of the algorithm might have been overseen.

Further, when working with the simulations, I have asked myself which parts of the models that could have been analytically understood and described, since a lot of the papers regarding inbound call center models (by Ward Whitt et al.) are trying to find models that are analytically understandable. Two of the algorithms presented by Korolev et al. are analyzed in their paper (even though the hyper-progressive algorithm is examined by simulation), but in this paper they have been extended somewhat.

12 Buzz words

talk time The time an agent actively talks with a customer

after call time A limited amount of time reserved to some book-keeping activity after an successful call has been completed

busy factor The average time per agent spent talking with the customers.

abandon call A call that is made by the call generator and successively disconnected when the customer answers due to a lack of free agents in the call center.

mismatch call See abandon call.

abandonment rate The amount of abandon calls in relation to the number of calls answered by customers.

hit rate The amount of answered calls in relation to the total amount of calls made by the call generator.

smoothing Loss of information due to the method used to extract information from a sample.

overfitting Exaggeration of minor variations in the information due to the method used to extract it.

service time The total amount of time it takes an agent to treat a customer, including talk time and after call time.

service load The current percentage of occupied servers.

holding time See service time.

ACD Automatic Call Distributor

SBR Skill Based Routing

CPD Call progress detector

RPC Right Party Connects

CTI Computer Telephony Integration

IVR Interactive Voice Response

References

[1] U. Bhat, Narayan. *An Introduction to Queueing Theory*. Birkhäuser, 2008.

[2] Andreas Brandt and Manfred Brandt. On a two-queue priority system with impatience and its application to a call center. *Methodology and Computing in Applied Probability*, 1(2):191–210, Sept. 1999.

[3] Andreas Brandt and Manfred Brandt. On the M(n)/M(n)/s queue with impatient calls. *Performance Evaluation*, 35(1-2):1–18, Mar. 1999.

[4] Lawrence Brown, Noah Gans, Avishai Mandelbaum, Anat Sakov, Haipeng Shen, Sergey Zeltyn, and Linda Zhao. Statistical analysis of a telephone call center: A queueing-science perspective. The same as Brown2005, but it was available before being published by JASA. Available at http://iew3.technion.ac.il/serveng/References/references.html, 2002.

[5] Lawrence Brown, Noah Gans, Avishai Mandelbaum, Anat Sakov, Haipeng Shen, Sergey Zeltyn, and Linda Zhao. Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association*, 100(469):36–50, 2005.

[6] Robert B. Cooper. Queueing theory. In *Encyclopedia of Computer Science*, pages 1496–1498. John Wiley and Sons Ltd., Chichester, UK, 2003.

[7] Paulo J. de Freitas Filho, Geovani Ferreira da Cruz, Rui Seara, and Guilherme Steinmann. Using simulation to predict market behavior for outbound call centers. In *WSC '07: Proceedings of the 39th conference on Winter simulation*, pages 2247–2251, Piscataway, NJ, USA, 2007. IEEE Press.

[8] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986.

[9] Agner K. Erlang. The theory of probabilities and telephone conversations. *Nyt Tidsskrift for Matematik B*, 20:33–39, 1909.

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

[11] O. Garnett, A. Mandelbaum, and M. Reiman. Designing a call center with impatience. *Manufacturing & Service Operations Management*, 4(3):208–227, 2002.

[12] S. G. Henderson. Estimation of nonhomogeneous poisson processes from aggregated data. *Operations Research Letters*, 31:375–382, 2003.

[13] D. G. Kendall. Stochastic processes occuring in the theory of queues and their analysis by the method of the imbedded markov chain. *Annals of Mathematical Statistics*, 24:338–354, 1953.

[14] Jack P.C. Kleijnen. *Design and Analysis of Simulation Experiments*. Springer, 2008.

[15] Nikolay Korolev, Herbert Ristock, and Nikolay Anisimov. Modeling and simulation of a pacing engine for proactive campaigns in contact center environment. In *SpringSim '08: Proceedings of the 2008 Spring simulation multiconference*, pages 249–255, San Diego, CA, USA, 2008. The Society for Computer Simulation, International.

[16] Averill M Law and W. David Kelton. *Simulation Modeling and Analysis*. McGraw-Hill Series in Industrial Engineering and Management Science. McGraw-Hill, fourth edition, 2007.

[17] Mario Lefebvre. *Applied Stochastic Processes*. Springer, 2007.

[18] A. Mandelbaum and S. Zeltyn. The impact of customers' patience on delay and abandonment: Some empirically-driven experiments with the M/M/N + G queue. *OR Spectrum*, 26(3):377–411, 2004.

[19] William A. Massey, Geraldine A. Parker, and Ward Whitt. Estimating the parameters of a nonhomogeneous poisson process with linear rate. *Telecommunication Systems*, 5(2):361–388, 1996.

[20] Thomas A. Mazzuchi and Rodney B. Wallace. Analyzing skill-based routing call centers using discrete-event simulation and design experiment. In *WSC '04: Proceedings of the 36th conference on Winter simulation*, pages 1812–1820. Winter Simulation Conference, 2004.

[21] Vijay Mehrotra and Jason Fama. Call center simulations: call center simulation modeling: methods, challenges, and opportunities. In *WSC '03: Proceedings of the 35th conference on Winter simulation*, pages 135–143. Winter Simulation Conference, 2003.

[22] Sanzheng Qiao and Liyuan Qiao. A robust and efficient algorithm for evaluating erlang B formula. Technical Report CAS98-03, Department of Computing and Software, McMaster University, Canada, 1998.

[23] Douglas A. Samuelson. Predictive dialing for outbound telephone call centers. *Interfaces*, 29(5):66–81, Sep. - Oct. 1999.

[24] B.A. Sevastyanov. An ergodic theorem for markov processes and its applications to telephone systems with refusals. *Teor. Verojatnost. i Primenen. (Theoretical Probability Applications)*, 2:104–112, 1957.

[25] Rayadurgam Srikant and Ward Whitt. Simulation run length planning for stochastic loss models. In *WSC '95: Proceedings of the 27th conference on Winter simulation*, pages 1384–1391, Washington, DC, USA, 1995. IEEE Computer Society.

[26] Ward Whitt. Planning queueing simulations. *Management Science*, 35(11):1341–1366, 1989.

[27] Ward Whitt. Engineering solution of a basic call-center model. *Management Science*, 51(2):221–235, 2005.