

INSTITUTE OF
ACTUARIAL SCIENCE
&
ECONOMETRICS

REPORT AE 5/2004

To Pool or Not To Pool
in
Call Centers

N.M. van Dijk
E. van der Sluis

TO POOL OR NOT TO POOL IN CALL CENTERS

NICO M. VAN DIJK AND ERIK VAN DER SLUIS

Abstract

Should we pool or not?

This is a question of general interest for call center management. The general perception seems to exist that pooling is always beneficial either in terms of performance (mean waiting times and service levels) or agent capacity savings.

This paper aims to show that also at practical call center level a more detailed answer is in place, which may even show the opposite. To this end, it will provide

- 1) insights and approximate formulae,
- 2) numerical support and
- 3) general conclusions.

The orientation of the paper is threefold: *instructive*, *practical* and *research-oriented*. An instructive approach will be followed based upon:

- standard queueing results and insights,
- instructive examples and
- numerical results.

For practical call center purposes extensive numerical results and figures are given for the generic situation of pooling two agent groups of realistic orders. These include figures for practical usage.

It is also argued and numerically illustrated that an improvement of both the unpooled and pooled scenario can be achieved by overflow pooling. This in turn will lead to a variety of options for an optimization search, which is still highly open for scientific research.

As such, the results are aimed at both call center practitioners and management scientists.

1. Introduction

1.1 Motivation

Should different agents groups for one and the same skill be merged together into one single group is a natural question that often arises in call centers. Clearly, one large agent group seems more efficient than separate ones for given service targets. Of a similar nature is the question whether or not or to which extent separately located call centers should be virtualized with the rationale of load balancing.

These questions become more complicated when multiple skills become involved. Which skills and with which priorities should be combined. Should we allow overflow and to which extent. Here, as always a trade-off is to be made between capacity (number of agents) and performance (service levels).

A question thus of general concern within call centers. A question which involves a number of aspects and considerations of both quantitative and non-quantitative nature. On the quantitative side, there are the natural issues of capacities and service levels. On the non-quantitative, there are aspects of agent availability, training capacities, manageable team sizes, human resource aspects, planning flexibilities and so on.

A question thus of general interest for call center managers which addresses issues at different levels, like:

- architectural (dimensioning)
- strategic (single number, single-multi skilled), and
- operational (agent allocation and scheduling, overflow, planning),

from the level of individual agents (queues) for identical or different tasks, of separate agent groups within one call center up to the virtualization and centralized call routing between call centers.

Questions thus for call center managers far more complicated than just the question of whether we should pool or not in the simple situation of just two parallel queues. A situation for which simple answers can be expected to be at hand. And yet, practical call center questions for which lessons can still be learned from by even this simple situation and by standard queueing results. Lessons that can be useful for call center management decision-making that call center managers might benefit from.

This paper therefore will merely focus on the quantitative consequences of pooling, in terms of performance and capacities, as based upon queueing results.

1.2 Objective and approach

The objective of this paper therefore is to provide general insights as well as practical results based upon queueing theory directly (or indirectly) related to the question of pooling for call centers. As such, the paper is aimed at both:

- Call center practitioners (managers, engineers and performance analysts) with little if no prior knowledge of queueing to provide insights and practical results.
- OR/MS-specialists to illustrate the potential and necessity of queueing and simulation for call center managerial decision making as well as to motivate further (also scientific) research.

To this end, an illustrative approach will be followed based upon

- insights from queueing theory
- exact and approximate formulae
- instructive examples and extensive numerical illustration.

From a scientific point of view most of the results are thus not new. In contrast, some basic results from queueing theory and some more special ones from its literature will be advocated and used strongly. The contribution of this paper is thus meant to show just the opposite. That is, that well known results from queueing (can still) lead to practical awareness of results that do not seem to be in line with general perceptions in call center environments related to questions of pooling.

The presentation will therefore be kept rather descriptive and be supported on an easy to follow mathematical basis as well as by numerical examples and figures based on extensive computations. To this end, also some 'new' approximate formulae will be derived. In this instructive and illustrative form directed merely to the question of pooling and at practical numerically supported call center level, the results do not seem to have been reported before. Nevertheless, some most closely related literature results will be discussed in section 1.4 below.

1.3 Results

As will be discussed more detailed below, the contribution of this paper can thus be summarized by

1. insights and formulae
2. numerical and practical tables
3. awareness and general conclusions for call center management.

(1) Insights and formulae

By just an instructive two server example the first insights will be provided, as based upon standard $M/M/1$ and $M/M/2$ -results as well as Pollaczek-Khintchine's (PK) formula, that pooling

- can be and will generally be advantageous
- that it can also work disadvantageous if different call types are involved
- that in this case there are no exact analytic forms so that simulation may be required
- that also sensitivity aspects are to be considered
- that a trade-off may have to be made between which performance measures are considered as most relevant (mean waiting time allover, for one call type, service levels, capacity savings) (The observation that pooling is not always beneficial has been made before in the literature but not in its explicit instructive form as by this example).

Next, by standard ($M/M/s$) results from queueing, approximate formulae will be developed for pooling two agent groups. These formulae will indicate the order of the pooling effect that one may expect as a function of the number of servers, the traffic loads and the mix variability when different call types are involved. These formulae rely upon no more than P.K.-formula as an approximation, which will be shown to be fairly accurate, and a (so-called pooling) factor for doubling two standard exponential single server or multi-server queues. Though rather straightforward, these 'approximate' formulae seem unreported.

In essence, these formulae show that the effect on mean waiting times of pooling two server groups factorizes in two factors:

- this pooling factor (for pooling exponential queues)
- a mix factor (for mixing different services)

The pooling factor can approximately be fitted quite well by a simple analytic function in the traffic load and number of agents (see (3.1)). Without any mixing effect this factor would indeed lead to a substantial improvement of the mean waiting time by at least 50% and more the larger the number of agents.

The mix factor, however, may bring an effect in opposite direction and can be stronger. Furthermore, the formulae also show that the effect of pooling is virtually independent (insensitive) for the actual call distributions when similar call distributions are mixed (see (2.22)).

(2) Numerical support and practical tables

For practical call center purposes itself as well as to illustrate that the insights provided (as based on 'theoretical' queueing results and a simple example) also apply at realistic call center sizes, extensive numerical support is provided for pooling two agent groups (with agent numbers in the orders of 5, 10, 20 or 50).

Results are shown on:

- mean waiting times, service levels and capacities leading to different forms of conclusions
- sensitivity effects such as due to call volume and call duration aspects
- possible performance gains by a more specialized form of pooling

As two agent groups with possible different call characteristics can be seen as a generic situation for pooling, the results and its realistic orders, are believed to be sufficiently representative for results that one may expect and experience in real life call center situations.

Particularly, in combination with the approximate formulae, various figures provided (e.g. Figures 5, 7, 8 and 9), can directly be used as practical tables that can be used to predict (and compute) the effect

of pooling in a practical situation.

Despite the accuracy of some of the approximate formulae provided, for exactness, all numerical results in sections 3, 4 and 5, except those that could be computed directly as based on expressions for $M/M/s$ -queues, (in call center language that is, Erlang-C) are obtained by long run simulations.

In section 5 also for operational call center computation and optimization the combined usage of both queueing and simulation is discussed.

(3) Awareness and general conclusions for call center management

First of all, the results indicate that call center managers should be aware of different aspects when considering pooling:

- awareness of the role that different call types may play
- awareness of sensitivity consequences
- awareness of different objectives
- awareness of the gains, and
- awareness of queueing.

Furthermore, from the numerical results a number of general conclusions can be drawn as of interest for call center managers to decide upon pooling either within one call center or between call centers; at least that is from a quantitative point of view focusing at capacities and performance. Some general conclusions are:

- Pooling can indeed be highly beneficial in terms of mean waiting times (with gains over 50%). The larger the call center the more profitable (in %).
- Pooling seems less advantageous for (standard) service levels and for capacity savings. The larger the call center the less profitable (again in %).
- There is also a price to be paid: less robustness.
- When different call types are mixed by pooling, pooling might not even be profitable at all. In contrast, in different respects, the unpooled case can still be superior. In fact, in this case, an overflow scenario might lead to an improvement over both the pooled and unpooled scenario. In the latter case, optimization might be recommended such as by simulation.

Throughout, at the end of each (sub)section the main conclusions from that (sub)section will be stated explicitly.

1.4 Literature

The question of pooling parallel queues has already been touched upon in early references such as Stidham (1970) by merging servers into a single faster server, as well as in early introductory OR-textbooks such as Hillier and Lieberman (1974), Wagner (1975), Kleinrock (1976) and Bierman et al.(1986) for lumping queues into one single queue with multiple servers. In such textbooks, it is sometimes left as an exercise for the exponential case to reveal that the delay reduction can be in the order of pooling (that is by a factor s if s queues are pooled) as based upon the standard $M/M/1$ -delay formula for a single server.

An important ‘application’ of this (simple) result can be found in Kleinrock (1976) for comparing ‘frequency division’ and ‘statistical multiplexing’ techniques in telecommunication.

A first proof for the more general multi-server ($M/M/s$) case which shows that pooling always leads to a mean delay reduction seems to be given in Smith and Whitt (1981). In Wolff (1989) also a proof for the reduction factor s can be found. A survey and discussion on these results can also be found in Rothkopf and Lech (1987). Pooling is also elegantly addressed from a psychological point of view in Larson (1987).

In fact, both Smith & Whitt and Wolff also provide more general delay comparison results for the non-exponential case as based upon sample path comparison. Along this line both references, most notably Smith and Whitt, also contain early (counter) examples which show that pooling may enlarge rather than reduce a mean delay as due to more variability that might become involved. This observation, however, despite these early intriguing though theoretical examples, seems to have remained rather unlightened and unexploited in the literature for more practical consequences.

A first revealing exception in the context of manufacturing and production line systems can be found in Buzacott et al. (1994) and Buzacott (1996). Also in these references variabilities of task durations are highly taken up as of importance for configuring a production line. However, due to the serial rather than parallel structures in such applications, the conclusions drawn in these references seem to indicate just the opposite. The more variability the more advantage can be achieved by pooling. In a more general setting of reengineering principles, Loch (1998) reflects that managing variability can be advantageous.

As a second more recent elegant exception, Mandelbaum and Reiman (1998) consider the consequences of pooling and its trade-off between efficiency and variability in a general setting of exponential queueing networks. Both parallel and serial structures as well as combinations are hereby covered. More precisely, based upon Pollaczek-Khintchine's formula (as will also be used strongly in the present paper), in this reference the trade-off between efficiency on the one hand and variability on the other is made explicit and characterized by a utilization (E_u) and a variability (E_v) index.

As a mixture of both aspects: pooling of different services and regarding multiple services as in combination, and by expressions closely related to the PK-formula but at the level of multiple services rather than a single service, general results are concluded both for light and heavy traffic situations and a variety of configurations. However, a simple example (as our example from section 2.2) of two parallel queues with job-type depending (and arbitrarily distributed) services (such as possibly deterministic) is not precisely included (as in this reference the services are server rather than job-type dependent while in addition the service at each server itself is assumed to be exponential). Nevertheless, the main message and the general flavor of the results from this reference are completely in line with those from the present paper; stating that the effect of pooling heavily depends on the actual variability or mixture of services.

However, the focus of this reference is rather different from the current paper, and more dedicated to stability, optimal (control of) capacity allocation and design. In fact, the structure as of main interest in the present paper (that is of parallel groups of servers) is briefly dealt with by this reference in section 5.2 with the situation of mixed services (heterogeneous servers) referred to in section 5.3 as without explicit expression or numerical support or conclusions.

Most recently, in Wallace and Whitt (2004), in the setting of configuring skill based routing (SBR) staffing requirements for call centers, resource pooling is used to limit the staffing capacities. Here it is shown that rather than universal agents one should keep the multi-skill functionalities as limited as possible ("*a little flexibility may go a long way*"), say to only two skills. To this end, an efficient algorithm is developed as based upon the square-root relationship for capacity computations (cf. Halfin and Whitt, 1981). Though with different underlying reason, such as infeasibility and training costs for multi-skill functionalities, this general conclusion also corresponds to a conclusion in this paper: that is, to keep the mix variability to a 'minimum' by restricting pooling. (As shown by examples in section 5.3 with only 5% overflow substantial improvements might be achieved).

In fact, by the square-root relationship just mentioned, as first developed in Halfin and Whitt (1981) and discussed and extended in a number of references (Grassmann 1988, Jennings et al. 1996, Puhalskii 2000), also results on capacity pooling can directly be computed. Most notably, in Borst et al. (2004) the square root staffing principle is exploited and extended in a number of directions for dimensioning large call centers. Capacity savings by pooling (such as of two agents groups as in this paper) for exponential queues with one type of call are hereby easily concluded.

Other references somewhat related to pooling, as will be argued and shown in section 5, would be references on skill based routing and references on overflow. As for references on SBR, however, so far these are rather limited (e.g. Stanford and Grassmann 1993, Garnett and Mandelbaum 2000, Chevalier and Tabordon 2003, Koole and Talim 2000 and Borst and Seri 2000, Wallace and Whitt 2004). These references, however, do not explicitly contain results directed to pooling.

References on overflow, in contrast, are abundant but also rather limited for specific call center application with some exceptions such as Borst et al. (1999) and Chevalier and Tabordon (2003). (Also some other references related to these directions for further research on overflow and SBR will be referenced in section 5 as they come along).

Finally, stimulated by the technical and societal development of call centers, there has been an explosion of queuing references over the last decade that focus their attention on call centers. For an extensive recent survey, see Gans et al. (2003) and the long list of references therein. These references also include course materials by queuing experts, which transfer results from queuing theory to the application area of call centers.

As pooling aspects can intrinsically be involved in a variety of call center settings, this list may possibly contain references with results or observations of a similar nature as in this paper, from which conclusions with respect to pooling can directly or indirectly be drawn. Nevertheless, other than the queuing references mentioned no other references are known to us that explicitly reveal pooling results for call centers.

Particularly not, references that provide general results or numerical orders from which it can be concluded rather directly whether or not and for which objectives pooling should be employed for typical call center situations of two (or a small number of) agents groups with agent numbers of realistic order and loads.

1.5 Outline

In the next section, first we briefly discuss the psychological aspect related to pooling as of little if no impact for call centers. Next, some basic results from queuing are presented to argue that pooling will generally be but not necessarily always beneficial. This will be illustrated by an instructive example of two parallel servers.

In section 2.3, steps are undertaken for the generalization of this example for pooling two server groups of arbitrary size from which similar conclusions can be drawn. To this end, a number of (approximate) formulae will be derived. Sections 3 and 4 will provide extensive numerical support for agent groups of realistic call center size as based upon these formulae as well as upon simulation.

In both these sections 3 and 4, we only compare the situation of a strictly unpooled versus a fully pooled situation. Section 3 purely focuses on possible performance improvements (mean waiting times and service levels); section 4 on a possible capacity gain. In both sections, furthermore, an important distinction is made in whether the agent groups that are pooled have equal call characteristics (sections 3.1 and 4.1) or whether different call types are mixed (sections 3.2 and 4.2).

Finally, in section 5, first it is argued and instructively illustrated (section 5.1) that neither of these two situations (unpooled or pooled) is optimal and that further improvements can still be achieved by allowing a more restricted form of pooling by overflow. To this end, also the importance of simulation and its combination with queuing is briefly argued to search for further improvement (section 5.2). Next, numerical support is provided for another variant, which also illustrates an improvement at realistic call center size (section 5.3). In addition, another possible reason for a more specialized form of overflow (as based on a real life case) is briefly discussed and illustrated (section 5.4). Finally, three directions for further research are briefly discussed (section 5.5). A short evaluation concludes the paper.

2. Queueing Insights and Approximate formulae for pooling

2.1 Should we pool or not?

- A question as simple as whether queues (or line ups) in front of service desks should be combined (pooled) into a single queue (line up) or not.
- A question that also concerns a variety of other daily-life situations such as at banks, information desks, ticket offices, up to manufacturing with parts and tools lined up for parallel machines.
- A question that seems too simple to be asked as the answer seems so obvious.

2.1.1 Psychological aspects

“The only thing worse than waiting in line is waiting in the wrong line.”

Clearly, with queues pooled into one single line this cannot occur. In practical situations therefore, such as at banks, ticket desks and so on, with visible and physical queues and one type of client (service) already from a psychological point of view one important reason for pooling is obvious:

Pooling is as fair as fair can be.

In a single line, everyone is served in order of arrival. With separate queues, in contrast, few people will be able to keep their good humour when they notice that a line next to them moves faster – to say nothing of the increasing adrenaline level when you shift to the faster queue to observe that the line you just left is suddenly moving faster. That is not fair. You must have lost good luck. By pooling these line ups into one single line at least these frustrations of ‘unfairness’ are avoided.

As also indicated in literature, this fairness aspect is by far the most dominating factor for the perception (psychology) of having to wait (cf. Larson 1987, Maister and Rech 1985, Rothkopf and Rech 1987).

However, in call center environments this psychological argument hardly plays any role as the ‘calls’ are not visible for one another. In fact, here the psychology may work the other way around as opposed to visible queues. Upon entering a call center line the client is generally positively inclined as he is not frustrated by observing a long line. But the longer he waits, the more frustrated he can get, while in physical queues waiting customers may eventually become more pleased when they observe the server coming within reach.

A factor for the perception of having to wait that has a particular role for call centers is animation, such as by music tunes, tips and so on. Such animations, however, may turn out to generate a negative effect (also see Maister and Rech 1985).

Conclusion *As opposed to physical queues such as at check-in, at banks or information desks, for call center queueing the psychology of waiting callers is of little if no direct influence for the question of whether or not queues should be pooled (clearly it will be as far as the waiting times themselves may hereby be influenced).*

2.1.2 Mean waiting times: a first insight and formula for pooling

But more importantly, also from a quantitative point of view, pooling parallel queues (and servers) seems most natural and preferable as:

Pooling is as efficient as efficient can be.

When queues for more servers are pooled into a single line none of the servers can ever be idle as long as there are customers waiting. In other words, the service capacity is always used in its maximal possible way. The capacity cannot thus be used more efficiently.

By Intuition

Let us illustrate this somewhat more precisely and argue purely intuitively.

Consider a situation with two parallel servers that are strictly separated, each with a same arrival and service rate. By pooling these servers into one single 2-times faster server, both the arrival and service rate are doubled. Hence, it can handle 2 times more customers but it also receives 2 times more customers.

Consequently, and possibly somewhat counter-intuitively, the queue length (number of customers present) can be expected to be the same for each of the two single individual servers or this doubled server. (This will also be verified below by queueing formulae).

But since the doubled server works 2-times as faster also the queue moves forward 2-times faster, so that the mean waiting time or delay can be expected to be reduced by (roughly) a factor 2.

By Queuing formulae

Indeed, this intuition seems supported by the most standard queueing expression (e.g. Cooper, 1981):

$$D = \frac{1}{\mu - \lambda} \quad (2.1)$$

with

- μ : the service rate (or capacity) of the server (per unit of time)
- λ : the arrival rate (per unit of time) and
- D : the average (or mean) delay

Here the implicit assumption is made of *exponential* service times. This assumption though can be regarded as formal justification for speaking in terms of a service rate. Now, when s of such servers, each with arrival rate λ and service rate μ would be pooled as if they were merged into a single s -times faster server, the mean delay would thus be reduced to by a factor s as according to:

$$D = \frac{1}{s\mu - s\lambda} \quad (2.2)$$

This first reasoning for pooling as by single server merging can also be found in various introductory OR-textbooks as mentioned in section 1.4 and seems first made explicit by Stidham (1970).

Remark 2.1 (*WFM software and Erlang-C for call centers*) The exponentiality assumption also corresponds to the standard assumption within call center environments for the use *workflow management packages* or, relatedly, *Erlang-C*.

Remark 2.2 (*Call centers*) The efficiency gain will also be numerically illustrated later on (in section 3.1) for a wide range of realistic agent numbers within call centers.

Remark 2.3 (*Queue length*) As intuitively argued earlier, one may note that the queue length for the s -times faster server is indeed equal to the queue length for each individual server since (by Little's law) in either case:

$$L = (\text{arrival rate}) \times D = \frac{s\lambda}{s\mu - s\lambda} = \frac{\lambda}{\mu - \lambda} \quad (2.3)$$

This delay reduction seems to result from the efficiency gained by pooling individual queues. The inefficiencies in the non-pooled case are avoided, as one server can no longer be idle while there are still customers waiting at another. The intuitive reasoning above thus seems to be supported by queueing theoretic results and in line with the general perception that pooling is beneficial. Most notably, for call centers it may enable one to balance (or spread) the workload in the 'best' possible way to minimize waiting times.

Conclusion 2.1 *Pooling seems more efficient and can lead to a (mean) delay reduction (roughly) proportional to the order of the pooling. (However, as will be illustrated later on, this conclusion is not generally valid).*

2.1.3 Another factor: Variability

However, there is also another factor that has not yet been considered. This is the factor of variability, more precisely, of the different forms or types of services in addition to just the mean service times. For example, for the most simple situation of a single server system in queueing theory there is Pollaczek-Khintchine's famous formula (e.g. Cooper, 1981).

$$W_G = (1 + c^2)W_D = \frac{(1 + c^2)}{2} W_E \quad (2.4)$$

where

W_G , W_D and W_E are the expected (average or mean) waiting times for the situation of a general service distribution (G) with coefficient of variation c^2 , respectively for a deterministic (or fixed) service time D (hence with coefficient of variation $c^2=0$) and for an exponential distribution E (for which $c^2=1$), with

- c^2 : coefficient of variation = σ^2 / τ^2 with
- σ^2 : the variance of the service time;
- τ : the mean service time.

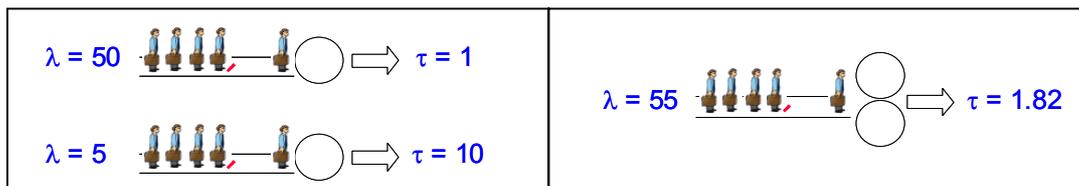
In words, this formula tells us that also the variation (as expressed by σ^2) around (relatively to) the mean τ of the service times plays an essential factor for the average delay (as compared to the situation of fixed service times). As such, it can also be of influence in relation to pooling.

2.2 An instructive two server example

2.2.1 A numerical example

As a simple example, consider a service system with two parallel servers and two types of customers:

- { Type 1 customers with an arrival rate 50 per hour and a fixed service time of 1 minute;
- { Type 2 customers with an arrival rate 5 per hour and a fixed service time of 10 minutes.



With strict separation of the two servers, one devoted to type 1 customers and one to type 2 customers and without any jockeying between them, by virtue of the waiting time expression for the deterministic case, the expected (mean) waiting times will be:

$$\begin{aligned}
 W_1 &= 2.5 \text{ minutes for type 1} \\
 W_2 &= 25 \text{ minutes for type 2 and} \\
 W_A &= \frac{10}{11} \times 2.5 + \frac{1}{11} \times 25 = 4.55 \text{ minutes allover.}
 \end{aligned}$$

Pooled case

However, if the two customer types are combined into a single line which feeds the two servers simultaneously, a *variation in service times* will be introduced as an arbitrary job can be of either type 1 or type 2, according to the arrival ratios. In this case the variation of service time would become:

$$\sigma^2 = \frac{10}{11}[1-1.82]^2 + \frac{1}{11}[10-1.82]^2 = 6.69 \text{ (minutes)}^2$$

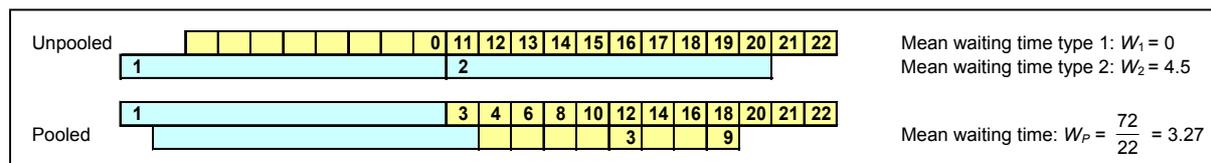
and would this variance not be introduced by pooling, one could have expected that pooling would lead to a reduction factor in the order of 50% for the average waiting time. However, as it turns out (by simulation), due to this mix variability we now obtain:

$$W_P = 6.15 \text{ minutes}$$

for both types of customers. We thus observe a substantial increase of the waiting time for the majority of customers (the type 1 customers) as opposed to a reduction for just a small percentage of customers. To provide some more insight the following (counter-intuitive) example illustrates the effect of different service characteristics at sample path basis.

Sample path example

This example has 22 customers arriving at interarrival times of 1 minute. The customers 1 (arriving at $t = 0$) and 2 are type 2 customers with 10 minutes of service time, while all the other are type 1 customers with 1 minute of service time.



Conclusion 2.2 *Pooling is not necessarily beneficial (allover) when different service characteristics are mixed.*

2.2.2 Approximate expression for pooling two exponential and deterministic servers

The mean waiting time result in the pooled example above was computed by simulation (as there is no exact general expression for the general case of $M/G/2$ -queues). Nevertheless, we could provide a simple but fairly accurate approximation. To this end, let

$W_E(s, \rho, \tau)$: the mean waiting time for an exponential server group with s servers, (that is an $M/M/s$ -queue as computed by Erlang-C) with traffic load $\rho = \lambda / s\mu$ per server where λ is the arrival rate, μ the service rate and $\tau = 1/\mu$ the mean service time.

Then, by straightforward calculation from standard $M/M/s$ -expressions, we find

$$\frac{W_E(2, \rho, \tau)}{W_E(1, \rho, \tau)} = \frac{\tau \rho^2 / (1 - \rho^2)}{\tau \rho / (1 - \rho)} = \frac{\rho}{1 + \rho} \quad (2.5)$$

This expression does in fact represent the effect on the mean waiting time for pooling two exponential servers with identical services. More importantly, it has two pooling consequences. In the first place it shows that the effect of pooling two parallel servers depends on the traffic load ρ and will even lead to a reduction of at least 50%, as argued before based upon regarding the pooled server as a single server with double service rate.

As a second consequence, by assuming, as will be shown in the next section, that the PK-formula is still quite accurate for multi-server models, and by combining (2.5) with the PK-formula we can derive an expression for pooling two servers (each with equal traffic load ρ and for instructive purposes let us first restrict to deterministic services say with mean τ_1 and τ_2).

For the mean waiting time W_i ($i = 1, 2$) at each server in the unpooled situation for type 1 and 2 and by the PK-formula it holds: $W_i = \frac{1}{2} W_E(1, \rho, \tau_i)$. Let W_A be the average of the unpooled system. For the pooled system, the mean service time becomes $\bar{\tau} = p_1\tau_1 + p_2\tau_2$ with $p_i = \lambda_i / (\lambda_1 + \lambda_2)$.

Furthermore, the mean waiting time for the pooled servers can still be approximated reasonably well, as will be shown in section 2.3, by the PK-formula, that is by: $W_{P=(1)} \frac{1}{2} (1+c^2) W_E(2, \rho, \bar{\tau})$.

Notation (=₍₁₎) Here, as well as throughout, we use the notation =₍₁₎ to indicate an exact expression up to the PK-approximation as by (2.12). The accuracy of this approximation depends on the number of servers s as illustrated in Figure 2.

Deterministic case

Let $P(2, D_1, D_2)$ denote the effect of pooling two servers with deterministic services D_1 and D_2 , but equal traffic load $\rho = \rho_1 = \lambda_1\tau_1$ and $\rho = \rho_2 = \lambda_2\tau_2$, defined as

$$P(2, D_1, D_2) = \frac{W(\text{pooled model})}{W(\text{two single servers})} = \frac{W_P}{W_A}$$

Then with $p_i = \lambda_i / (\lambda_1 + \lambda_2)$ and

$$c_{mix}^2 = \frac{p_1(\tau_1 - \bar{\tau})^2 + p_2(\tau_2 - \bar{\tau})^2}{\bar{\tau}^2} \quad (2.6)$$

and using $\rho_1 = \rho_2$ we have

$$\begin{aligned} W_A &= p_1W_1 + p_2W_2 = p_1\frac{1}{2}\tau_1\rho_1/(1-\rho_1) + p_2\frac{1}{2}\tau_2\rho_2/(1-\rho_2) \\ &= \frac{1}{2}\bar{\tau}\rho/(1-\rho) = \frac{1}{2}W_E(1, \rho, \bar{\tau}) \end{aligned}$$

$$W_{P=(1)} = \frac{1}{2} (1 + c_{mix}^2) W_E(2, \rho, \bar{\tau})$$

The effect of pooling now leads to a factor

$$P(2, D_1, D_2) = \frac{\frac{1}{2}(1+c_{mix}^2)W_E(2, \rho, \bar{\tau})}{\frac{1}{2}W_E(1, \rho, \bar{\tau})} = (1+c_{mix}^2) \left[\frac{\rho}{1+\rho} \right] = P(2, E_1, E_2) \quad (2.7)$$

Exponential case

Here the last equality in (2.7), where $P(2, E_1, E_2)$ represents the pooling factor for two exponential services E_1 and E_2 , is easily verified along the same lines, as will also be verified more generally later on in section 2.3.4.

For the instructive example in section 2.2, the pooling factor is $(1+2.03) 0.83 / 1.83 = 1.375$, which corresponds with the 35% increase in mean waiting time as shown by simulation.

Mix ratio

In the example there were $k = 10$ times more short jobs than longer jobs with exactly a 10 times longer service time. In fact, in this special case with a mix ratio of k times shorter jobs but equal traffic loads, the following general expressions can be derived

$$\bar{\tau} = \frac{\lambda_1}{\lambda_1 + \lambda_2} \tau_1 + \frac{\lambda_2}{\lambda_1 + \lambda_2} \tau_2 = \frac{2k}{(k+1)} \tau_1 \quad \text{with } \lambda_1 = k\lambda_2 \text{ and } \tau_2 = k\tau_1 \quad (2.8)$$

and

$$\begin{aligned} c_{mix}^2 &= \frac{p_1(\tau_1 - \bar{\tau})^2 + p_2(\tau_2 - \bar{\tau})^2}{\bar{\tau}^2} = p_1 \left(\frac{\tau_1}{\bar{\tau}} \right)^2 + p_2 \left(\frac{\tau_2}{\bar{\tau}} \right)^2 - 1 \\ &= \frac{k}{k+1} \left(\frac{k+1}{2k} \right)^2 + \frac{1}{k+1} \left(\frac{k+1}{2} \right)^2 - 1 = \frac{(k-1)^2}{4k} \end{aligned} \quad (2.9)$$

As a consequence, since $\rho/(1+\rho)$ is less than 50%, the pooling factor will already be less than 1, which means that pooling will be advantageous for the mean waiting time, for $k \leq 5$. However, situations with $k > 5$ do seem realistic (specifically in call center environments).

Conclusion 2.3 *Pooling is average beneficial for a mix ratio $k \leq 5$, but not necessarily for $k > 5$.*

Increase for the majority of customers

Furthermore, as in the example, note that the majority (in the example more than 90%) of the jobs (the type 1 jobs) will substantially increase their mean waiting time (in the example more than doubled) by pooling, despite the fact that the service capacity made available to them (and with the traffic loads kept the same), is doubled.

More precisely, from the expressions above we easily obtain

$$\frac{W_p}{W_1} = \left(1 + c_{mix}^2\right) \frac{2k}{k+1} \left[\frac{\rho}{1+\rho} \right] = \frac{1}{2}(k+1) \left[\frac{\rho}{1+\rho} \right] \quad (2.10)$$

Conclusion 2.4 *Pooling is necessarily beneficial for all types only for $k \leq 3$. Whereas, there can be a possible increase for $k/(k+1)$ 100% of the customers for $k > 3$. Whether this is acceptable, despite an improvement of the mean waiting time allover, may depend on the specific situation under consideration.*

2.3 Formulae for pooling two server groups.

In this section, we aim to argue the global effect of pooling of two server groups with different service characteristics in line with the simple two-server example from section 2.2.1 and 2.2.2. As a first step therefore, it is shown that the PK-formula still holds reasonably well as an approximation for larger server groups so that it can be used for each group separately as well as their combination.

Next, as there are no simple expressions for the comparison of these larger groups, as a second step, we like to qualitatively regard each group as an $s\mu$ -server. As will be shown below in section 2.3.2 this seems reasonably justified. Finally, by combining these steps, an expression like (2.5) can then be argued as a rough approximation for the order of pooling.

2.3.1 PK-formula for larger groups

For the simple example of two parallel servers, the pooled case was evaluated by still using the Pollaczek-Khintchine formula. This however is still to be justified. In fact, as a two server example is not representative for more realistic call center group sizes, like $s = 5, 10, 20, 50$ or even 100 agents, one might question whether or to which extent this aspect of variability and of variability introduced by mixing different services when pooling, is still relevant also for larger server numbers.

In contrast, in realistic call center practices it is generally perceived that the larger the call center, which can possibly be resulting from pooling, the less the actual service distribution G or its coefficient of variation c^2 will play a role. An often heard argument here is the law of large numbers (or the central limit theorem) by which the variations should average out the more calls are involved. The application of Erlang-C and relatedly of WFM (Workforce Management) tools, as based upon the standard exponentiality assumptions, seems hereby generally justified.

The opposite of this perception and argument appears to be true. The size of the call center, at least for normal traffic loads such as $\rho = 0.7, 0.8$ or 0.9 , hardly seems to be of influence on the effect of the coefficient of variation as shown for the single server model by PK-formula. This becomes apparent, more precisely, by the following approximate formula for the mean waiting time for $M/G/c$ -queues. This formula as based upon earlier results by Cosmetatos (1976), as well as related variants that have been developed in the literature, is not exact but has been shown to be fairly accurate for a wide spectrum of natural traffic situations (e.g. see Tijms, 1994, p. 297-300).

$$W_G = \left[(1 - c^2) F + c^2 \right] W_E$$

with

(2.11)

$$F = \frac{1}{2} + \frac{(s-1)(1-\rho)(\sqrt{4+5s}-2)}{s\rho \cdot 32}$$

In Figure 1 below one can observe that the reduction factor of 50% for a deterministic service as opposed to an exponential service (Erlang-C) as 100% exact for a single server case, still holds reasonably well for $s = 30$ servers (agents). Particularly for realistic (high) traffic loads, such as 85%, the F -value seems to remain within roughly [0.5 - 0.6].

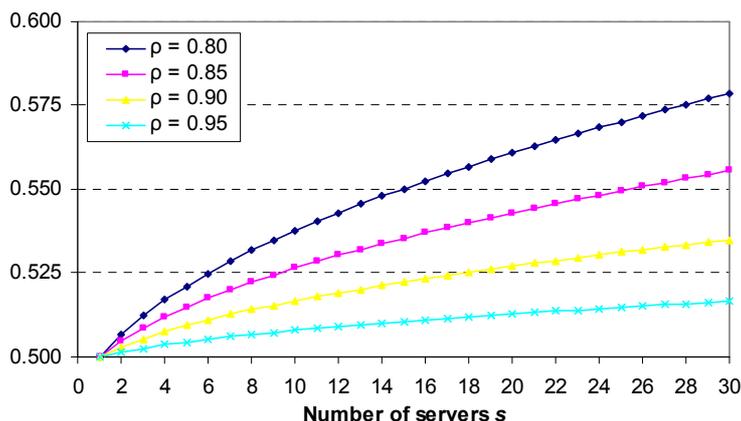


Figure 1: The F -factor for the generalized PK-formula (2.11) for different number of servers and ρ .

In Figure 2 it is shown how this (approximate) formula only slightly deviates from the linear extrapolation $\frac{1}{2}(1+c^2)$, as exact for the single server case by PK-formula, as function of c^2 for larger numbers of servers s , say $s = 5, 10, 20$. For the purpose of arguing the effect of pooling, as already

used in section 2.2.2 for $s = 2$, and as will be used in section 2.3.3, we could thus roughly state that also for larger numbers of s

$$W_G =_{(1)} \frac{1}{2}(1 + c^2)W_E \quad (2.12)$$

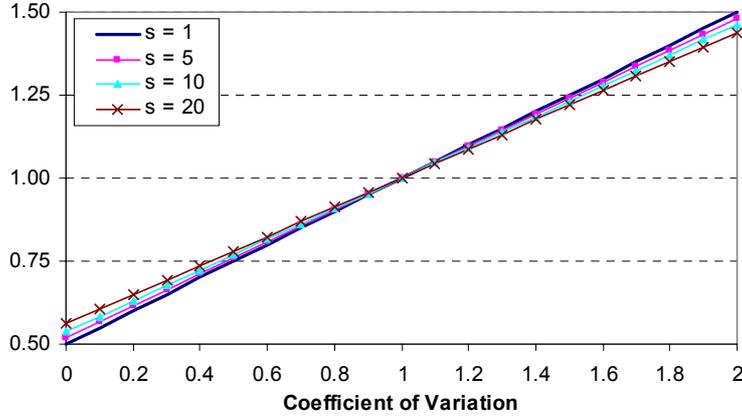


Figure 2: Generalization of PK-formula for multi server systems for $\rho = 0.8$.

2.3.2 Fast server approximation

Unfortunately, expressions for the mean waiting time W_E of multi server exponential queues (for more than $s = 2$) do not lead to a simple form such as (2.5). Therefore as a second step, in line with the simple two server example from section 2.2, let us argue the order of the effect for pooling multi server groups by regarding each group, say with s servers, as a single server with a service rate $s\mu$. To this end, let us first provide some support.

At first intuition, one may think that the (exponential) $s\mu$ -server is superior to s parallel (exponential) servers as the service rate is always at least as large (and larger when the number of customers is less than s). However, one has to be careful. Also here a sample path (counter) example can easily be constructed to violate this intuition. More precisely, the following inequalities can be shown (see Wolff 1989). With

- D : the expected total delay
- W : the expected waiting time

of an $M/G/s$ -queue and correspondingly \bar{D} and \bar{W} for an s -times faster $M/G/1$ (s -times faster) queue, and arbitrary service distribution G with coefficient of variation c^2 we have:

$$\begin{aligned} \bar{W} &\leq W + (s-1)(1+c^2)/2s\mu \\ \bar{D} &\leq D - (s-1)(1+c^2)/2s\mu \end{aligned} \quad (2.13)$$

In fact, in general $\bar{W} > W$ appears to be true, as shown below. The $s\mu$ -queue will have s times faster services but its waiting time before getting into service, as clearly also its probability of having to wait in the first place, is generally larger.

Nevertheless, as illustrated in the figure above the qualitative behaviour of a parallel system with s servers and of an $s\mu$ -server with respect to s appears to be rather comparable. This will be used in section 2.3.3 to obtain a zero and first order approximation for the effect of pooling also for larger numbers.

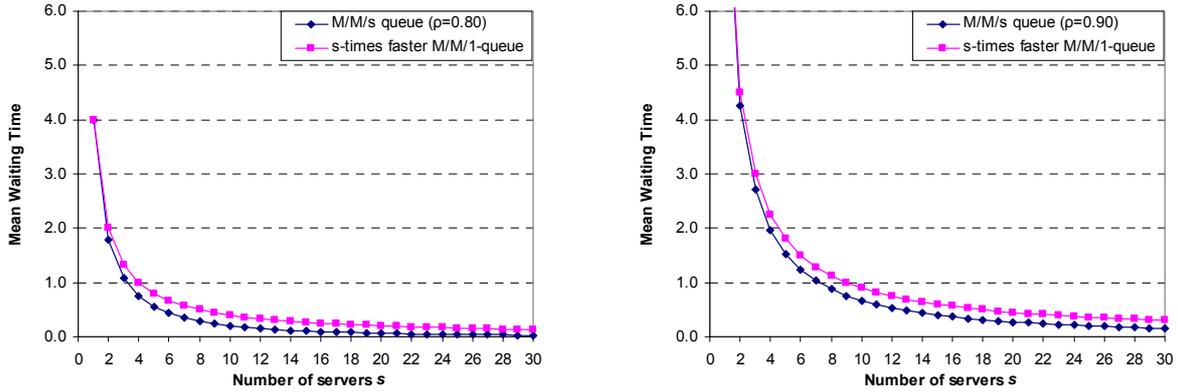


Figure 3: Comparison of multi-server and s -times faster single server.

2.3.3 The effect of pooling two server groups

Now as most representative situation as of interest for pooling, let us consider the situation of two server groups $i = 1, 2$ with an equal number of servers s , and two customer types 1 and 2 at server group 1 and 2 respectively and with different service distribution G_i for customer type group i with characteristics (τ_i, c_i^2) with c_i^2 the coefficient of variation, arrival rate λ_i and traffic load $\rho_i = \lambda_i \tau_i / s$ where we assume that $\rho_1 = \rho_2 = \rho$. Furthermore, as we aim to study the option of pooling, we implicitly assume that any server can serve any type of customer.

Let $p_i = \lambda_i / (\lambda_1 + \lambda_2)$, $i = 1, 2$. Further, then as a first step, as argued in section 2.3.1, for both the unpooled and pooled case we can use PK-formula (as approximate) to conclude

$$W_{A=(1)} = p_1^{1/2}(1 + c_1^2)W_E(s, \rho_1, \tau_1) + p_2^{1/2}(1 + c_2^2)W_E(s, \rho_2, \tau_2) \quad (2.14)$$

and

$$\begin{aligned} W_P &=_{(1)} \frac{1}{2}(1 + c_{pooled}^2)W_E(2s, \rho, \bar{\tau}) \\ &= \frac{1}{2} \left[1 + p_1 \left(\frac{\tau_1}{\bar{\tau}} \right)^2 c_1^2 + p_2 \left(\frac{\tau_2}{\bar{\tau}} \right)^2 c_2^2 + c_{mix}^2 \right] W_E(2s, \rho, \bar{\tau}) \end{aligned} \quad (2.15)$$

Here, c_{pooled}^2 in the latter expression for the pooled case has been obtained by straightforward computation from standard relations for conditional expectations and its variance where c_{mix}^2 reflects the variability due to mixing (pooling) the different services, as expressed by (2.9).

Now note that by scaling τ_i to $\bar{\tau}$ while assuming that the traffic loads are kept the same (as by adjusting the arrival rate) we can conclude

$$W_E(s, \rho, \tau_i) = \frac{\tau_i}{\bar{\tau}} W_E(s, \rho, \bar{\tau}) \quad (i = 1, 2) \quad (2.16)$$

by which we can rewrite (2.14) for the unpooled case as

$$\begin{aligned} W_{A=(1)} &= p_1^{1/2}(1 + c_1^2) \frac{\tau_1}{\bar{\tau}} W_E(s, \rho, \bar{\tau}) + p_2^{1/2}(1 + c_2^2) \frac{\tau_2}{\bar{\tau}} W_E(s, \rho, \bar{\tau}) \\ &= \frac{1}{2} \left[1 + p_1 \frac{\tau_1}{\bar{\tau}} c_1^2 + p_2 \frac{\tau_2}{\bar{\tau}} c_2^2 \right] W_E(s, \rho, \bar{\tau}) \\ &= \frac{1}{2} \left[1 + \frac{1}{2} c_1^2 + \frac{1}{2} c_2^2 \right] W_E(s, \rho, \bar{\tau}) \end{aligned} \quad (2.17)$$

Hence, by (2.15) and (2.17) we obtain

$$P(2s, G_1, G_2) = \frac{W_P}{W_A} \stackrel{(1)}{=} \left[\frac{W_E(2s, \rho, \bar{\tau})}{W_E(s, \rho, \bar{\tau})} \right] \frac{\left[1 + p_1 \left(\frac{\tau_1}{\bar{\tau}} \right)^2 c_1^2 + p_2 \left(\frac{\tau_2}{\bar{\tau}} \right)^2 c_2^2 + c_{mix}^2 \right]}{\left[1 + \frac{1}{2} c_1^2 + \frac{1}{2} c_2^2 \right]} \quad (2.18)$$

Pooling Factor (First order approximation)

Let the pooling factor $P(2s, \rho)$ be defined as

$$P(2s, \rho) \equiv \frac{W_E(2s, \rho, \tau)}{W_E(s, \rho, \tau)} \quad (2.19)$$

Clearly, for computational purposes standard analytical forms are available for $P(2s, \rho)$. However, these do not provide a simple analytical expression. As a second step therefore a fast server comparison can be suggested, as outlined in section 2.3.2. As for the simple two-server example of section 2.2, a zero order approximation would simply be obtained by

$$P(2s, \rho) \approx \frac{W_E(1, \rho, \bar{\tau}/2s)}{W_E(1, \rho, \bar{\tau}/s)} = \frac{1}{2} \quad (2.20)$$

by which a pooling reduction of 50% is argued. To also include some effect of the traffic load, a first order approximation (which from hereon will be denoted by $\approx_{(2)}$) is provided by

$$P(2s, \rho) \approx_{(2)} \frac{W_E(2, \rho, \bar{\tau}/s)}{W_E(1, \rho, \bar{\tau}/s)} = \frac{\rho}{(1 + \rho)} \quad (2.21)$$

which shows that the reduction is indeed at least 50%. In fact, in the next section a more accurate approximation will be provided depending on both ρ and s . In line with expression (2.5), as rough first order approximation we may thus conclude the role of both the mix variability and of the traffic load ρ by combining (2.18) and (2.21).

Notation: Rather than the notation $\stackrel{(1)}{=}$ which is regarded as a relatively minor approximation, here we use $\approx_{(2)}$ as well as $\approx_{(3)}$ in section 2.3.5 to indicate a rather rough approximation.

Remark 2.4 Clearly, the situation above with equal number of servers and equal traffic load for both server groups to be pooled (or not), may not strictly fit an actual realistic situation. Nevertheless, the results are meant as indicative for the order of results that can be expected also in somewhat related situations. Roughly speaking, the assumption of more or less equal traffic loads (per server) can be regarded as essential.

Conclusion 2.5 *Pooling two equal server groups leads to a mean waiting time reduction of at least 50% as given by $\rho/(1+\rho)$.*

2.3.4 Special case: Equal distributions

First note that if $G_1 = G_2 = G$ and thus also $\tau_1 = \tau_2 = \tau$ and $c_1^2 = c_2^2 = c_{12}^2$ then (2.18) becomes

$$P(2s, G, G) = P(2s, E, E) = P(2s, \rho) \quad (2.22)$$

Hence, the pooling factor becomes independent of the service distribution.

When just the service distributions are assumed to be the same (such as most notably for call center applications: exponential), but allowing $\tau_1 \neq \tau_2$, or more precisely under just the assumption that $c_1^2 = c_2^2 \equiv c_{12}^2$, we can simplify the expressions (2.17) and (2.15) to

$$\begin{aligned}
W_A &\approx_{(1)} \frac{1}{2}(1+c_{12}^2)W_E(s, \rho, \bar{\tau}) \\
W_P &\approx_{(1)} \frac{1}{2}\left[1+c_{12}^2(1+c_{mix}^2)+c_{mix}^2\right]W_E(2s, \rho, \bar{\tau}) = \frac{1}{2}(1+c_{12}^2)(1+c_{mix}^2)W_E(2s, \rho, \bar{\tau})
\end{aligned} \tag{2.23}$$

As a consequence, also in this case the pooling factor appears to be independent of the actual service distribution as expressed by:

$$P(2s, G_1, G_2) \approx_{(1)} (1+c_{mix}^2)P(2s, \rho) \quad \text{For any type of distribution } G_1, G_2 \tag{2.24}$$

Conclusion 2.6 *(Insensitivity result) The effect of pooling becomes independent of the (type of) call distributions when these (or rather their coefficients of variation but not necessarily their means) are equal.*

2.3.5 Unequal loads and number of servers (Approximate formula with $\rho_1 \neq \rho_2$ and $s_1 \neq s_2$)

Clearly, also for the more general case with significantly different traffic loads ($\rho_1 \neq \rho_2$) and server numbers ($s_1 \neq s_2$) the expressions (2.14) and (2.15) can be extended to

$$W_A \approx_{(1)} p_1 \frac{1}{2}(1+c_1^2)W_E(s_1, \rho_1, \tau_1) + p_2 \frac{1}{2}(1+c_2^2)W_E(s_2, \rho_2, \tau_2) \tag{2.25}$$

$$\begin{aligned}
W_P &=_{(1)} \frac{1}{2}(1+c_{pooled}^2)W_E(s_1+s_2, \bar{\rho}, \bar{\tau}) \\
&= \frac{1}{2}\left[1+p_1\left(\frac{\tau_1}{\bar{\tau}}\right)^2 c_1^2 + p_2\left(\frac{\tau_2}{\bar{\tau}}\right)^2 c_2^2 + c_{mix}^2\right]W_E(s_1+s_2, \bar{\rho}, \bar{\tau})
\end{aligned} \tag{2.26}$$

with $\bar{\tau}$ and c_{mix}^2 as before and

$$\bar{\rho} = \frac{s_1 \rho_1 + s_2 \rho_2}{s_1 + s_2} \tag{2.27}$$

With the analytic expressions for $M/M/s$ -queues, computational results can thus be obtained directly. But also a rough first order approximation (which will be indicated by $\approx_{(3)}$) could still be concluded by setting

$$W_E(s_i, \rho_i, \tau_i) \approx_{(3)} W_E(1, \rho_i, \tau_i / s_i) = \frac{1}{s_i} \frac{\rho_i}{(1-\rho_i)} \frac{(1-\bar{\rho})}{\bar{\rho}} \frac{\tau_i}{\bar{\tau}} W_E(1, \bar{\rho}, \bar{\tau}) \tag{2.28}$$

$$W_E(s_1+s_2, \bar{\rho}, \bar{\tau}) \approx_{(3)} W_E(2, \bar{\rho}, 2\bar{\tau} / (s_1+s_2)) = \frac{2}{s_1+s_2} W_E(2, \bar{\rho}, \bar{\tau}) \tag{2.29}$$

leading to an (approximate) pooling factor:

$$\begin{aligned}
P(s_1+s_2, G_1, G_2) &=_{(1)} \frac{(1+c_{pooled}^2)W_E(s_1+s_2, \bar{\rho}, \bar{\tau})}{p_1(1+c_1^2)W_E(s_1, \rho_1, \tau_1) + p_2(1+c_2^2)W_E(s_2, \rho_2, \tau_2)} \\
&\approx_{(3)} \left[\frac{\bar{\rho}}{1+\bar{\rho}} \right] \frac{(1+c_{pooled}^2)}{\frac{(1-\bar{\rho})}{2\bar{\rho}^2} \left[(1+c_1^2) \frac{\rho_1^2}{(1-\rho_1)} + (1+c_2^2) \frac{\rho_2^2}{(1-\rho_2)} \right]}
\end{aligned} \tag{2.30}$$

3. Call Center Pooling: Performance Effects

In this section, we aim to apply the results and insights from section 2 in order to provide a number of representative results and conclusions as of practical call center interest related to the question of pooling.

Clearly, by the general formulae and insights a variety of different situations and aspects can be investigated such as with

- different numbers of agent groups
- equal and unequal loads at these groups
- a variety of mixtures of different job types
- different call types (like exponential, deterministic, and other coefficients of variation in between)
- different performance measures (average waiting time, waiting times for specific types, service levels, workloads and efficiencies) and
- different forms of pooling (e.g. by overflow, only after so many seconds, skill based routing).

Clearly, this would be impossible. In contrast though and in order to keep the presentation restricted while still aiming to provide sufficiently instructive and representative situations to draw some general conclusions, in what follows in section 3, 4 and 5 we aim to include a representative situation for most of them, without being far from exhaustive. To this end, from hereon we will limit the presentation to the representative and generic case of:

- just two agent groups each with one type of call
- exponential and deterministic calls and
- equal workloads for both

3.1 Equal call types

In this sub section, let us first provide support for the general perception that pooling is beneficial. A first rough distinction here is to be made in whether the groups, that are considered to be pooled, have more or less the same type of calls (in terms of means and variation).

3.1.1 Reducing mean waiting times

As most natural situation, pooling will be thought of to be beneficial when groups are pooled for stochastically identical calls, or at least for calls from which the basic characteristics as the mean and variance are more or less equal and which can be dealt with by both agent groups.

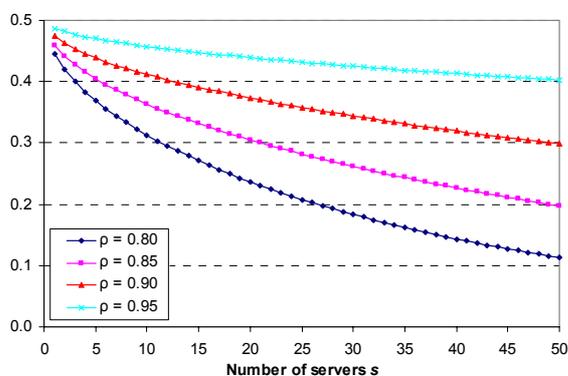


Figure 5: Pooling factor $P(2s, \rho)$.

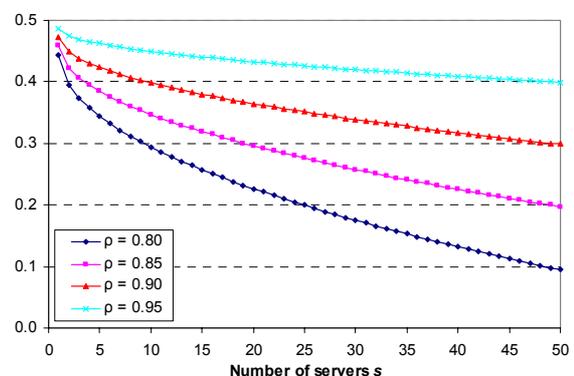


Figure 4: Approximate formula (3.1) for $P(2s, \rho)$.

In section 3 this was already argued for two identical server groups, each with s servers, by approximately regarding these server groups as by two single but s -times faster servers, so as to illustrate a mean waiting time reduction factor of at least 50% and its dependence on the traffic loads.

In the figures above the pooling factor $P(2s, \rho)$ is studied more precisely. First in Figure 4, the pooling factor is computed by exact computation for $M/M/s$ -queues (as by Erlang-C). In all situations the pooling leads to a reduction of at least 50% and still features the phenomenon $\rho/(1+\rho)$, as already roughly argued in section 2 as a rough first order approximation. However, the results also show that pooling becomes more advantageous for mean waiting times the larger the call centers.

More precisely, as illustrated in Figure 5, the following analytic approximation has been found which appears to fit reasonably well. This approximation shows the role of both the traffic load ρ and the number of agents $2s$, when pooling two call centers of size s .

$$P(2s, \rho) = \frac{W_E(2s, \rho, \tau)}{W_E(s, \rho, \tau)} \approx \left[\frac{\rho}{(1+\rho)} - \frac{1}{4}(1-\rho)\sqrt{s-1} \right]^+ \quad (3.1)$$

Conclusion 3.1 (i) *The pooling effect on mean waiting times for two agent groups with one and the same call type (and with capacities kept) is a reduction of at least 50%. It is increasing in both the number of servers s and the traffic load ρ .*

3.1.2 Increasing service levels

Clearly, as another performance measure of interest also service levels can be considered. Similar results can be provided as illustrated in Figure 6 below, which show the increase in service level (the percentage of calls with a waiting time less than some threshold level t) by pooling two agent groups of size s and equal call characteristics.

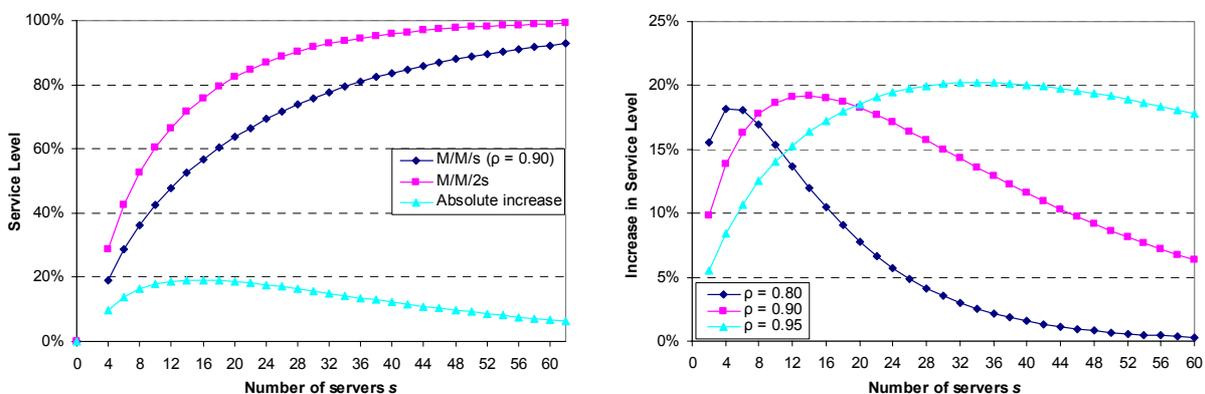


Figure 6: *Increase in service level by pooling two identical groups with Exponential calls and threshold level $t = \tau/4$.*

Clearly, the performance improvements for service levels of practical interest such as for 80% or 90% are generally somewhat less than for mean waiting times. Furthermore, as the service levels are bounded from above, the (also relative) improvement in service level appears to reduce the larger the call center size s .

Conclusion 3.1 (ii) *The pooling effect on service levels in the same situation will also be an improvement all over. But in contrast it is decreasing in the number of agents (call center size) and (for sufficiently large call centers also in) traffic load ρ .*

3.2 Different call types

As shown in section 3.1, simply pooling two call center groups while keeping the agent capacities thus always turns out to be beneficial (regardless of the call center sizes) provided identical call characteristics are involved (regardless of whether exponential or not). However, as illustrated and argued in section 2, even with unchanged capacities the effect of pooling is no longer necessarily clear and depends on the mix variability introduced by pooling as made explicit by expression (2.18). Herein, the coefficient of mix variation c_{mix}^2 and the pooling factor $P(2s, \rho)$ can be substituted as according to (2.9) and (2.19) or (3.1).

Furthermore, the parameter k will represent the mix ratio of k times more short calls (as by ((2.8)). The results will be obtained by simulation. (One may note that Erlang-C is no longer sufficient and approximate computations could be executed as by (2.18) and (3.1) for mean waiting times and as referred to in section 3.2.4 for service levels.)

3.2.1 Mean waiting times

Consider the situation in which different services are mixed as outlined in section 2.3.3. That is, with two agent groups of equal size s and traffic loads ρ , but each for a different call type with characteristics (τ_i, c_i^2) and c_i^2 at group $i, i=1, 2$ and mix ratio k as in (2.8). For studying the effect of pooling we also implicitly assume that each agent could handle a call of either type.

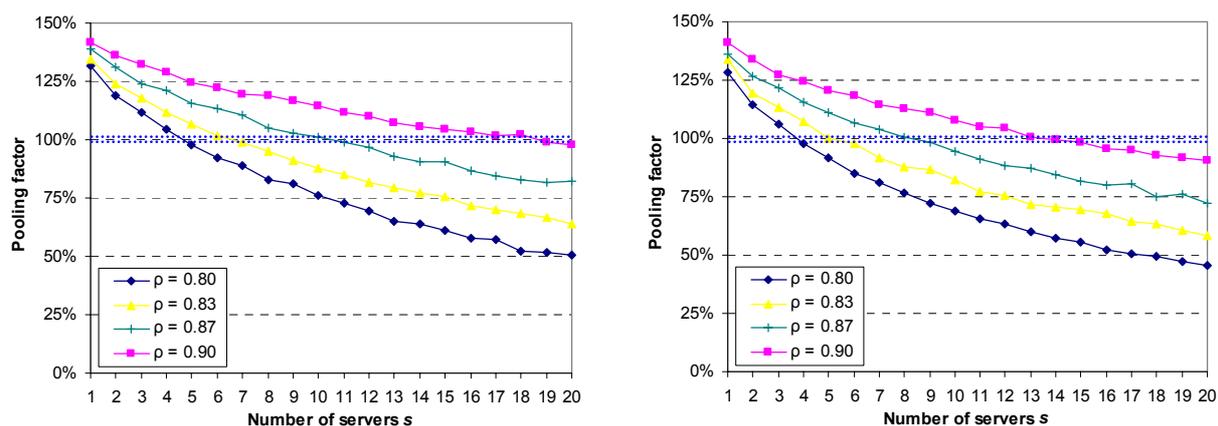


Figure 8: Pooling factor in case of Deterministic (left) and Exponential (right) call durations for different levels of traffic load ($k=10$).

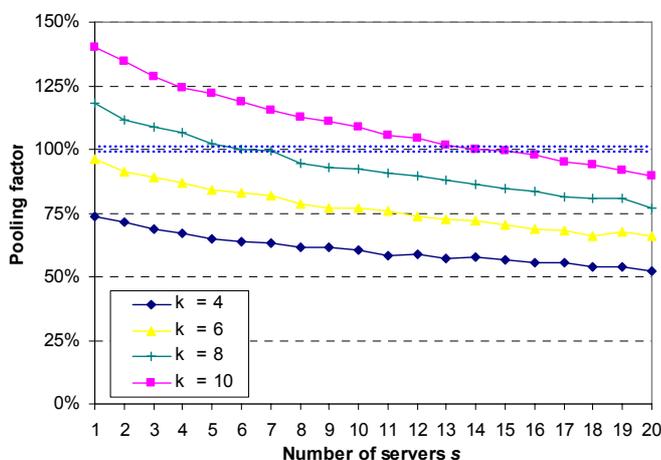


Figure 7: Pooling factor for different mixing ratios k and Exponential call durations with $\rho = 0.9$.

Let the *pooling point* \mathbf{P} denote the number s at which pooling becomes beneficial (for the mean waiting time) while keeping the number of servers unchanged.

In Figure 7 we study the situation of both the extreme case of deterministic call durations, as rather unrealistic for call durations and of exponential calls. The results show that the pooling point might not have or just have been reached in realistic situations, such as with ρ in the order of 80% up to 90% and s between 5 up to 20. In Figure 8 we also study the effect for different values of the mix ratio k .

Conclusion 3.2 (i) *The pooling effect for two agent groups with unequal call types is not beneficial for mean waiting times for larger mix ratios ($k > 5$) and small agent numbers $s < \mathbf{P}$, where \mathbf{P} is the pooling point. This pooling point will be:*

- *larger for larger mix ratios*
- *larger for larger traffic loads*
- *larger for smaller coefficients of variation (e.g. deterministically as extreme), of the call durations themselves.*

3.2.2 Service levels

Also in this case, rather than mean waiting times, service levels can be considered. In Figure 9 below we provide some illustration as based upon simulation. (For an approximate computation, see 3.2.4.) Let t be the threshold level for which the service level is computed.

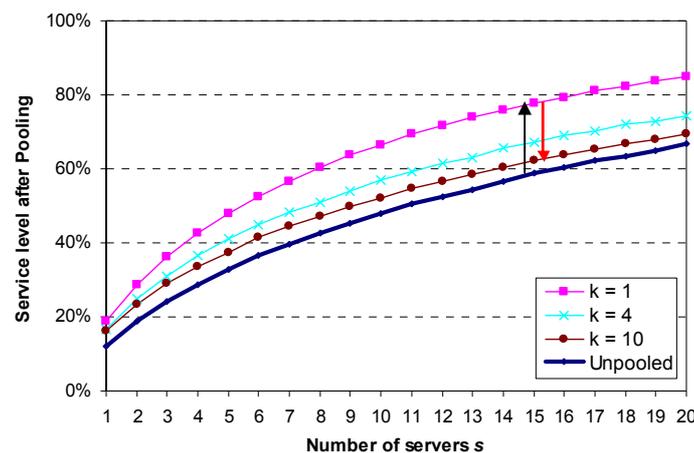


Figure 9: *Service levels before and after pooling groups with different call types ($t = \tau/4$).*

For rather standard service levels that are relatively small, say smaller than the mean call time like $t = \tau/4$ as shown in Figure 9, pooling will still appear to remain to be advantageous if the call types were equal. However, the mix variability already leads to a substantial quality reduction as illustrated in Figure 9. For example, for the situation of $s = 15$, it illustrates that the service level of 60% for separate agents groups would have been increased by pooling to almost 80% when the call types would have had identical characteristics. However, it is reduced to nearly 60% again (in case of $k = 10$) as a consequence of mixing their different characteristics. For threshold levels substantially larger, in contrast, for example $t = 4\tau$, indeed pooling may turn out even negatively as shown in Figure 10. For example, for the situation of $s = 5$, instead of an increase from 89.7% to 98.8% for call types with identical characteristics, the service level decreases (in case of $k = 10$) to 71.5%.

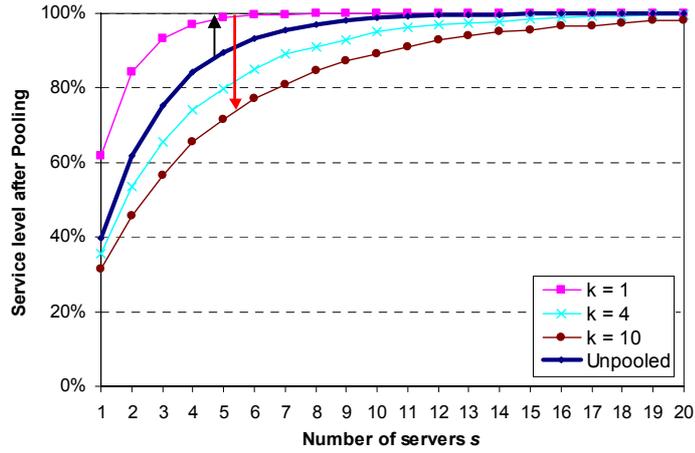


Figure 10: Service levels before and after pooling groups with different ($x = 4\tau$).

Conclusion 3.2 (ii) *In the situation of unequal call types also for service levels the pooling effect can be substantially less than thought for (e.g. as based on Erlang-C). The effect still seems generally beneficial for standard threshold levels though, but can even become negative for large threshold levels.*

3.2.3 Type 1 calls

As in section 2.2 in Conclusion 2.4 for the two server example, also more generally, even though the overall effect of pooling is positive, it remains to be realized that it may still lead to a (substantial) reduction for the (majority) group of customers with the shortest call duration, when the mix factor k is large, say more than 3.

s	k = 4		k = 6		k = 8		k = 10	
	$\rho = 0.8$	$\rho = 0.9$						
1	1.11	1.18	1.56	1.66	2.00	2.13	2.44	2.61
2	0.99	1.12	1.38	1.57	1.78	2.02	2.17	2.47
3	0.93	1.10	1.31	1.53	1.68	1.97	2.06	2.41
4	0.89	1.08	1.25	1.51	1.61	1.94	1.97	2.37
5	0.86	1.06	1.21	1.48	1.55	1.91	1.89	2.33
10	0.74	1.00	1.03	1.40	1.33	1.79	1.62	2.19
15	0.64	0.95	0.90	1.33	1.16	1.71	1.42	2.09
20	0.57	0.91	0.79	1.28	1.02	1.64	1.25	2.01
25	0.50	0.88	0.70	1.23	0.90	1.58	1.10	1.93
30	0.44	0.85	0.61	1.19	0.79	1.53	0.96	1.86

Table 1: Pooling factor for type 1 calls for different mix ratios k and traffic loads ρ .

More precisely, similar to expression (2.10), also in this more general case, the following expression for W_p/W_1 can easily be derive. For example, with equal call distributions, hence with $c_1^2 = c_2^2 = c_{12}^2$, along the lines as in 2.2.3 to 2.2.5 and by substituting (3.1), we find

$$\frac{W_p}{W_1} =_{(1)} (1 + c_{mix}^2) \frac{2k}{k+1} \mathbf{P}(2s, \rho) = \frac{1}{2}(k+1) \mathbf{P}(2s, \rho) \approx \frac{1}{2}(k+1) \left[\frac{\rho}{(1+\rho)} - \frac{1}{4}(1-\rho)\sqrt{s-1} \right]^+ \quad (3.2)$$

Conclusion 3.2 (iii) *In the case of unequal call types the pooling effect for type 1 calls, which represent the majority, $k/(k+1)100\%$ of all calls, will be (substantially) less than in the situation of one call type. For small number of servers, high traffic load and high mix ratio it can also be non-beneficial despite an overall improvement.*

3.2.4 Approximate service level computation

Rather than by simulation also an approximate computation of service level can be suggested. To this end, let $\eta_G(s, \rho, \alpha)$ the percentile t such that the probability for waiting time to exceed threshold t for an $M/G/s$ -queue at traffic load ρ is equal to $1-\alpha$.

Then in line with (2.10) an approximate relation is provided by (see Tijms, 1994, p. 299-300)

$$\eta_G(s, \rho, \alpha) = (1 - c_s^2) \eta_D(s, \rho, \alpha) + c_s^2 \eta_E(s, \rho, \alpha) \quad (3.3)$$

However, instead of mean waiting times as in (2.11) here there is no simple approximation as by a factor F that relates percentiles for the deterministic and exponential case. For computational purposes, however, an exact and numerical stable expression for $M/D/s$ -queues, as has recently been developed in an elegant paper by Franx (2001) is strongly recommended.

4. Capacity savings

In section 3 it was assumed allover, that the capacities (the number of agents) were kept unchanged upon pooling with the objective of a performance improvement (the mean waiting time or service level).

In call center practices, however, it is rather standard to determine the capacities (the required number of agents) to meet a specific service performance (usually an $\alpha\%$ service level within a pre-specified threshold t of so many seconds or minutes). Consequently, the situation of section 3.1, though illustrative for the substantial performance gains that could be achieved by pooling, will often not be applicable. Instead pooling might be aimed at capacity gains.

4.1 Equal call types: capacity saved

Below the possible capacity savings (in total number of agents) are computed for the simple case of two identical exponential agent groups with equal traffic loads and determining (by standard $M/M/s$ or Erlang-C calculations) the number of agents for both the separate groups and the pooled group to meet a given service target: in Table 2 for the mean waiting time and in Table 3 for a given service level.

Load	Separate Group			Pooled Groups		Saved
	s	SL	W	s	SL	
2	3	94.5%	2.73	5	97.2%	1
5	5	93.0%	3.13	8	92.1%	2
10	8	92.1%	3.35	14	93.6%	2
20	14	93.6%	2.61	25	94.0%	3
40	25	94.0%	2.51	45	90.2%	5
100	56	93.1%	3.09	106	90.1%	6

Table 2: Capacity savings for pooling two identical groups with service level target $P(W \leq 15 \text{ sec}) > 90\%$.

Load	Separate Group			Pooled Groups		Saved
	s	SL	W	s	W	
2	3	94.5%	2.73	4	5.22	2
5	5	93.0%	3.13	7	9.72	3
10	7	80.3%	9.72	13	5.71	1
20	13	86.5%	5.71	23	8.31	3
40	23	80.4%	8.31	44	6.48	2
80	44	84.1%	6.48	84	8.36	4

Table 3. Capacity savings for pooling two identical groups with mean waiting time target $W \leq 10$.

These savings are by far not as large as the performance reductions in section 3.1 (due to the strong nonlinear or exponential growth of waiting times as a function of the traffic load) and might not be as large as generally perceived (say in the order of just a few percent).

Furthermore as larger systems operate more efficiently, that is, can work at a higher traffic load to meet the same performance target, the larger the call centers the less the saving by pooling as can also be seen in Figure 11.

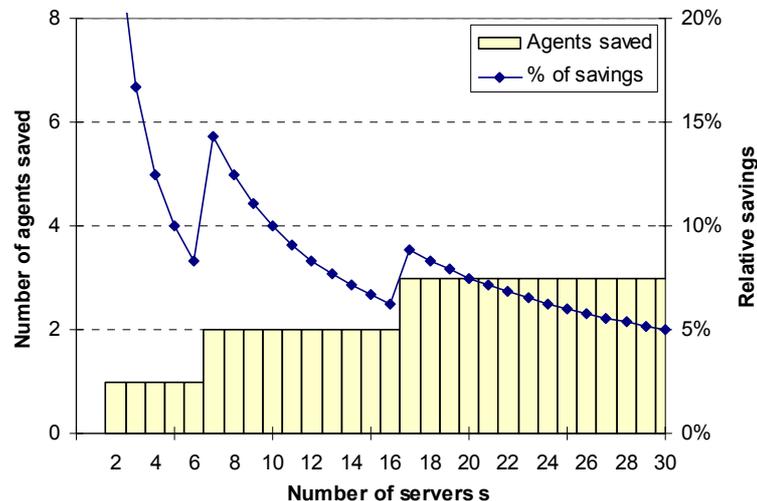


Figure 11: Savings in number of agents when two identical groups are pooled with service level target: $P(W \leq 15 \text{ sec}) > 80\%$.

Conclusion: In other words, in terms of savings pooling becomes less beneficial the larger the call centers, as opposed to the statements for mean waiting times in section 3.1 As illustrated below similar type of results can also be obtained for service levels.

Conclusion 4.1 *Instead of a performance improvement pooling can also be used to achieve capacity savings (but not both at the same time). This capacity saving will generally,*

- *become smaller the larger the call center*
- *be in the order of a few percent.*

4.2 Sensitivity effects (equal call types)

So far, pooling has been aimed at to be and it has been shown that pooling can be profitable in either of two ways: a performance or a capacity gain. However, in the case of a capacity gain there is also a price to be paid. This price is higher the larger the call center. This is the price of robustness. More precisely, larger systems as can be regarded as due to pooling, are more sensitive as the traffic loads will generally be higher (for given service targets).

4.2.1 Call volume forecasts

Most notably, the dimensioning of call centers needs to rely upon forecasts for call volumes as by historical statistics. But what happens if the actual call volumes turn out to be somewhat higher than anticipated for.

Example (call volume) As a simple example consider some 10 medium sized call centers, say with traffic volumes of 200 calls each and its pooled version of 2000 calls with exponential call durations with means of two minutes. Now suppose that the actual call volumes turn out to be 10% higher. The service level of the medium sized group will hardly be affected. For a larger agent group, however, it will drop much stronger (see Figure 12).

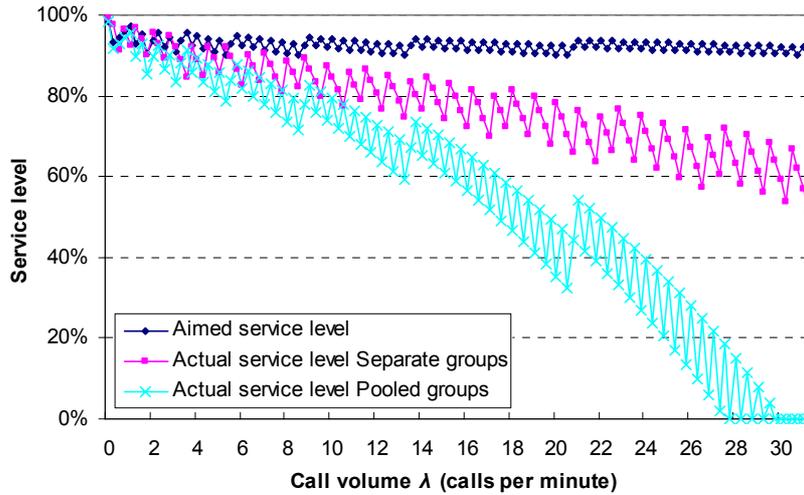


Figure 12: Service levels when call volume is 10% higher with service level target: $P(W \leq 15 \text{ sec}) > 90\%$.

For example, for an agent group of size $s = 20$ a 10% increase in traffic will lead to a service level of around 75% (instead of around 90%). In the pooled case, ($s = 36$) the service level would even drop to 40%.

4.2.2 Longer call durations

Similarly, combining two groups may lead to somewhat longer call durations such as due to agent handling times or overhead. Suppose that two equal groups are pooled with the minimal number of agents computed as based on the original call duration. However, due to slightly enlarged call duration the actual service level will drop. This effect will be stronger the larger the call centers, like the situation in the previous section.

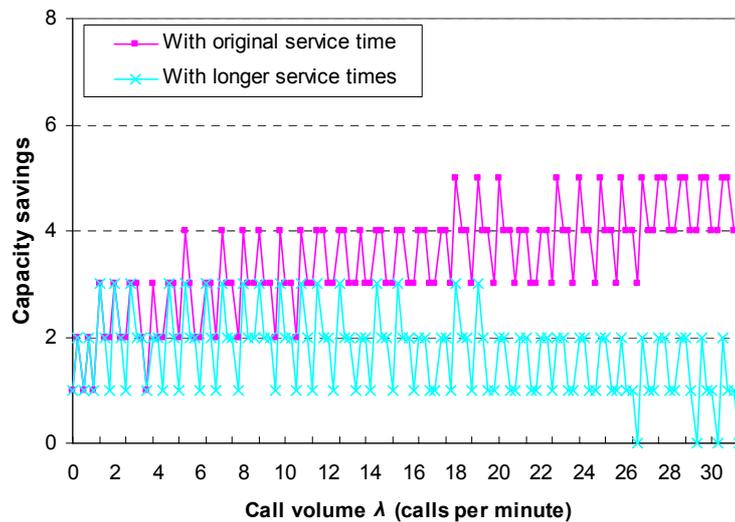


Figure 13: Savings in number of agents with 5% longer call durations and service level target: $P(W \leq 15 \text{ sec}) > 90\%$.

Alternatively, to compensate for this degradation, the minimal number of agents required can be computed as based on the longer call durations. In that case, the saving in the number of agents is less,(or might even vanish) to meet the service level (see Figure 13).

Conclusion 4.2 *When employing pooling for capacity savings (as based on a fixed service target) one should be aware of an increased sensitivity (as due to a higher efficiency level) such as for higher than predicted call volumes or enlarged call durations, which may have a drastic effect.*

4.3 Unequal call types: mixing effects

As shown in section 3.2, the positive effect of pooling will at least be reduced and might even vanish due to the mix variability introduced when different call characteristics are involved.

This also holds for capacity savings. Capacity calculations only based on average call durations and Erlang-C, the pooled case would lead to an optimistic reduction in the number of required agents. However, due to the mix variability the actual service levels can be considerably less then expected as shown in Figure 14.

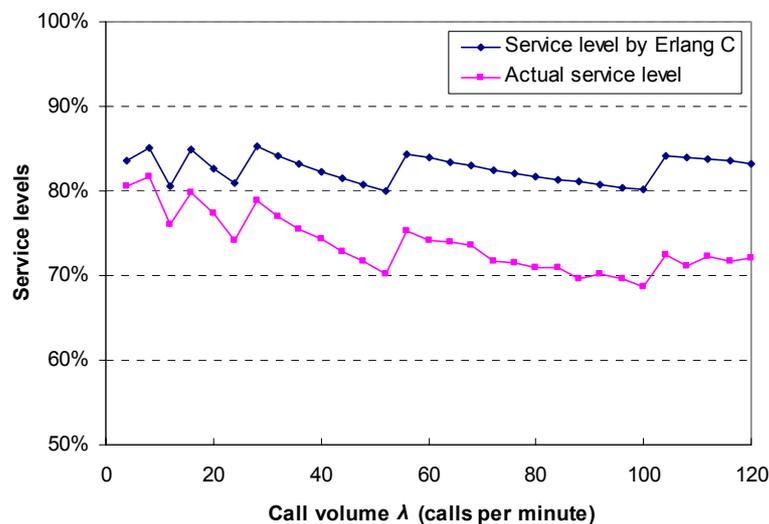


Figure 14: Service level number when two unequal groups ($k=10$) are pooled and the number of agents based on Erlang C.

As in section 4.2.2., to compensate for the mix variability, additional agents might be required in order to regain the pre-specified performance target. A similar figure as Figure 13 could than be obtained. Due to this factor the savings might not be as large as one would have expected (as based upon standard Erlang-C calculations).

Conclusion 4.3 *Capacity savings by pooling, as computed by Erlang-C when pooling groups with unequal call types, will reduce the actual service performance which may no longer meet the target. The actual capacity savings to meet this target will at best be equal but generally be less (and might even vanish).*

5. Overflow pooling

So far, in sections 3 and 4 we have only considered the options of a strict unpooled situation as opposed to a fully pooled situation in order to investigate a performance (section 3) or a capacity (section 4) gain.

In other words, with either criterion we have simply provided the insights and numerical support to conclude, in a given situation and with a given objective, which of these two variants performs best, when different services are involved.

However, this is not to say that in that case there is no variant (if not many more) that is (that are) superior to both. Particularly, that there is no a variant that is generally superior to the fully pooled case, despite the fact that capacity then seems to be used in the most ‘effective’ way, with minimal idleness.

In fact, here the disadvantage of either should be realized

- For the unpooled case: a form of inefficiency
- For the pooled case: extra mix variability

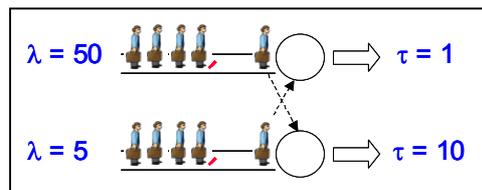
A more careful combination of the (dis)advantages of both might therefore lead to even better results. To this end, in the next section let us first consider the instructive example again.

5.1 The instructive example revisited

Reconsider the example from section 2.2 with just 2 parallel servers, each for one traffic type 1 or 2, with equal traffic load $\rho = 5/6$ but a traffic ratio of 10. More precisely, at

- Server 1: Arrival rate 50 and mean call duration 1 for type 1
- Server 2: Arrival rate 1 and mean call duration 10 for type 2

In order to avoid the disadvantage of both we would like to avoid the inefficiencies on the one hand while keeping the mix variability to a minimum.



To this end, as a simple overflow scenario, let the calls of type 1 line up at server 1 (as in the unpooled scenario) but also allow server 2 to take a type 1 call if there is no type 2 call waiting, and vice versa.

In the same line as in section 2.2.1 a sample path example can be constructed to illustrate an improvement of this overflow scenario over both the unpooled as the pooled scenario. However, at realization basis these examples can be constructed with results in any intuitive as well as counterintuitive direction. A long run average is therefore required as by simulation.

Simulation results:

The following results are obtained by simulation for the deterministic case.

Unpooled			Pooled	Overflow			% overflow	
W1	W2	Wav	W	W1	W2	Wav	1 => 2	2 => 1
2.50	24.87	4.54	6.12	3.66	8.58	4.11	20.9%	22.0%

In the first place, the results illustrate that indeed, as already proved in section 2.2 for mix ratios $k > 5$, the pooled case performs less than the unpooled case, not only for type 1 customers but all over. Secondly, it shows that this overflow scenario, which one may also regard as some form of pooling, improves both the pooled as well as unpooled scenario.

Conclusion 5.1 *In situations with sufficiently different call types an overflow scenario might perform superior to both the pooled and unpooled case.*

5.2 Optimal overflow pooling and simulation

As illustrated in section 5.1, when a service mix is involved, a fully pooled case is not necessarily optimal, despite the fact that it minimizes the idleness (the inefficiencies) of the agents. As full pooling will introduce a maximum of mix variability, variants might still be thought of that may reduce or limit this mix variability, possibly to the cost of some ‘extra’ idleness.

In the overflow scenario there is no ‘extra’ idleness at all (none of the agents is ever kept idle while there is still any call waiting). But, it can still be advantageous to allow some ‘idleness’ (inefficiency) of servers to keep the mix variability more restricted.

For example, only allow overflow of type 1 customers to server 2, but not type 2 customers to server 1 so that server 1 could sometimes be kept idle with type 2 calls waiting. Or, only allow overflow from type 1 customers to server 2 if the type 1 queue at server 1 exceeds some threshold. Many other variants with ‘overflow’ pooling might so be thought of which may improve the overflow scenario in the previous section. As an illustration, for the example from section 5.1 the first variant with only overflow for short calls would lead to the following results, which appear to improve all scenarios.

Unpooled			Pooled	Overflow (one-sided)			% overflow	
W1	W2	Wav	W	W1	W2	Wav	1 => 2	2 => 1
2.50	24.99	4.55	6.14	1.80	25.18	3.92	7.7%	0.0%

In other words, by allowing some specific form of overflow by which specific capacities (or rather traffic flows) are pooled in specific situations, some ‘optimal’ form of ‘overflow pooling’ can be searched for. Here, in practice a trade off is to be made between not only

- the best (queueing) performance and
- the (in)efficiencies of agents

but also

- whether or not or and to which extent the pooling scenarios with overflow are sufficiently practical for implementation;
- the multi-functional skills of agents and corresponding aspects (such as training and operational costs).

The necessity of simulation

As overflow systems are generally analytically unsolvable (e.g. Van Doorn, 1984), simulation seems necessarily required or at least is a natural remaining resource to evaluate the performance. Without simulation the overflow scenarios could not be evaluated easily and other scenarios as mentioned for further optimization become even more complicated.

In addition, as WFM-packages or Erlang-C calculations are based on $M/M/s$ -calculations, and thus with the implicit assumptions of modelling (also different type) calls as by one single exponential call duration, one either has to correct for some approximate PK-factor, as in line with the results from section 2, or simulation is to be executed.

Last but not least, as a third important reason for simulation, in realistic call center environments one cannot simply assume that waiting times can be computed as if consecutive time intervals (usually half hours) can be regarded as independent steady state situations with some constant arrival rate and some constant capacity.

The waiting times from one half hour will influence those in the beginning of the next, the capacities may not be constant such as due to short breaks or “on hold”, and the arrival rates may also be more irregular than by the underlying standard Poissonian assumption.

Most notably, one may even experience bursts of calls that violate any steady state computation at all. Clearly, simulation cannot resolve these aspects but at least it can handle them more realistically. For example by simulating an entire day rather than just separate time intervals.

Summarizing, in order to investigate the question of pooling simulation is preferred if not required for a number of reasons:

- to simulate configurations with different variants of overflow pooling
- to evaluate the effect of non-exponential services and most notably of mix variability
- to also deal with realistic phenomena of fluctuating call rates, variable capacities and burstiness.

This, however, is not to say that basic insights from and (approximate) computations by queueing theory as presented herein as well as numerous other references are not useful for analyzing call center performance questions of pooling. At least these insights and computations may strongly limit the (simulation) search for a ‘practical optimal’ solution.

Conclusion 5.2 *A variety of overflow scenarios might be thought of for further improvements over the unpooled and pooled scenario. A combined queueing and simulation approach is generally recommended to search for an ‘optimal overflow’ scenario.*

5.3 Larger groups

In this section, we aim to illustrate that the instructive example from section 5.1 also applies to larger numbers of agents of realistic call center order. Here we will use the overflow scenario only with overflow for type 1 calls (the short calls) to server 2, as illustrated in section 5.2.

s	Unpooled (variant 1)			Pooled (variant 2)	Overflow (variant 3)			% overflow
	W1	W2	Wav	W	W1	W2	Wav	
1	4.49	45.24	8.18	11.53	3.40	45.47	7.20	5.1%
2	2.14	21.40	3.89	5.27	1.55	21.62	3.37	5.2%
3	1.38	14.15	2.51	3.33	0.98	14.35	2.17	4.9%
4	1.01	10.24	1.83	2.34	0.70	10.43	1.57	4.9%
5	0.78	7.57	1.41	1.76	0.53	7.75	1.20	4.9%
10	0.34	3.52	0.63	0.71	0.21	3.66	0.52	4.5%
15	0.21	2.15	0.38	0.40	0.12	2.26	0.31	4.2%
20	0.15	1.44	0.26	0.26	0.08	1.54	0.21	4.0%
30	0.08	0.81	0.15	0.13	0.04	0.88	0.12	3.6%
40	0.06	0.54	0.10	0.08	0.02	0.59	0.08	3.2%
50	0.04	0.38	0.07	0.05	0.02	0.42	0.05	2.9%
60	0.03	0.29	0.05	0.03	0.01	0.33	0.04	2.6%

Table 4. *Simulation results of the three variants for pooling two unequal groups with deterministic call durations and mix ratio $k = 10$.*

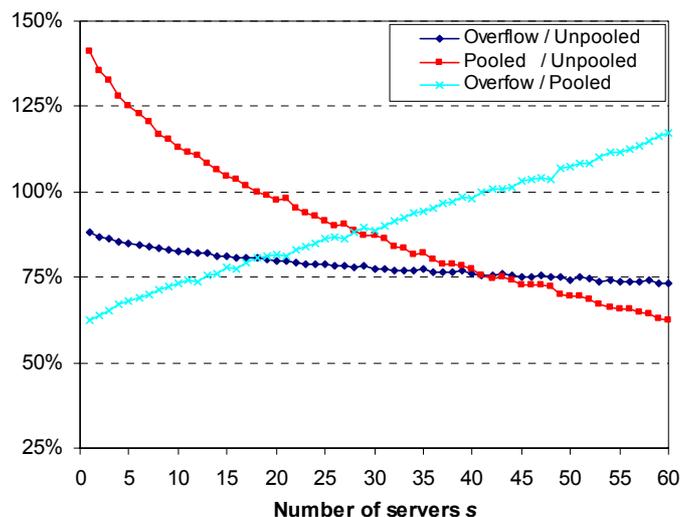


Figure 15: *Comparison of three pooling variants for two unequal groups with deterministic call durations and mix ratio $k=10$.*

Again, in the first place the results show that pooling is not necessarily superior to the unpooled case when different call types are involved. Again, this special scenario (called variant 3) is shown to improve both the unpooled (variant 1) and the pooled (variant 2) scenario for small and medium sized agent groups, at least in overall mean waiting time. This scenario keeps the call types separate and only leads to a small percentage of overflow from type 1 calls (the short calls) to server 2.

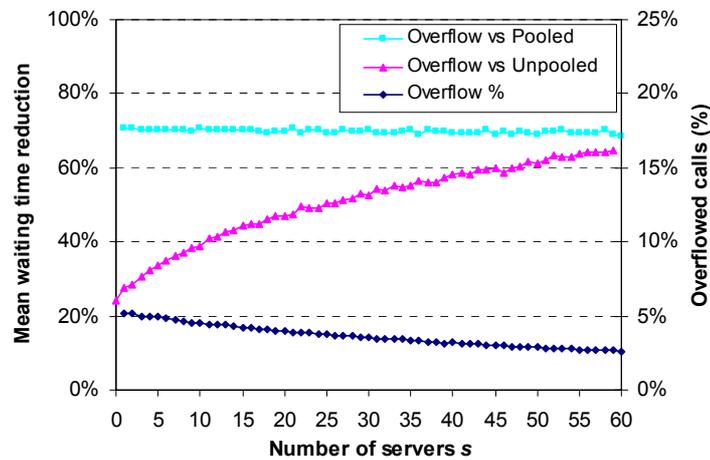


Figure 16: Reduction in waiting times for the type 1 calls under the one-sided overflow scenario compared to the pooled and unpooled case.

Also note that the third variant improves the performance for type 1 calls, which constitute the majority (more than 90%) of calls. This improvement is also shown in Figure 16 in relation to the small percentage of overflow.

This observation is in line with earlier results in the literature, which also indicate that just a relatively limited overflow flexibility may already lead to substantial improvement (e.g. see Turner 1998 and Wallace and Whitt 2004).

Conclusion 5.3 *Conclusions 5.1 and 5.2 also apply to realistic call center sizes showing substantial improvement over both the unpooled and pooled case by just a small % of overflow (pooling). Further steps for optimization such as by simulation can be recommended.*

5.4 Other overflow considerations

As another consideration, rather than for not for finding the best performance, restrict pooling and to only allow for overflow but in a restricted manner, one may wish to keep the overflow to a minimum. Most notably, multi-skilled agents may still have a preference (be best fitted) for some primary call type (e.g. server 2 for calls of type 2). Or, particularly in situations of separately located, say regional call centers, which are virtualized as one common call center, as overflow from call center (or call center group) to another may lead to extra costs.

	Unpooled	Pooled	Overflow after 28 sec
Service level	73%	98%	96%
Waiting Time	10 sec 60%	0 sec 80%	10 sec 60%
% Overflow	-	60%	20%

Table 5. Three different overflow scenarios for the Dutch AAA.

Case-example (Dutch AAA)

As a real-life example for the Dutch AAA (called ANWB), four regional call centers were considered to operate as one pooled center by letting calls, that were still arriving regionally, overflow directly if they had to wait. As shown above in Table 5 instead of a service level of around 73% within 30 seconds (with an average call duration of 2 minutes) when the call centers were kept separate, the overflow led to an improvement of a 96% service level. However, in this case 60% of all calls was overflowed. In addition to the disadvantage of additional telecommunication costs hereby involved, these overflowed calls were thus connected to agents from other regions with less knowledge of the specific region, and thus less “suited”. Alternatively, by letting calls overflow but only after 28 seconds (so that they could still be counted as successful within 30 seconds if a free agent in another region could be found), the service level was just slightly reduced to 96%. The overflow % however drastically dropped to 20%. Clearly, a variety of practical reasons can so be thought (costs, skills, recognition, handling, responsibility), that may have to be taken into account for considering and limiting overflow (pooling). Again, a combination of queueing insights and simulation may here appear a most valuable approach to find a practical balance between such aspects and an acceptable performance.

5.5 Overflow optimization methods

As argued and illustrated in sections 5.3 and 5.4, different and a large if not infinite number of overflow scenarios might be considered with different objectives and performance indicators that may have to be balanced or optimized. Unfortunately, for this purpose there is no general standard procedure or tool. In contrast, this aspect is still highly open for research.

Let us just briefly mention a few possible approaches.

Simulation

Clearly, a first and most common procedure and tool is to extensively execute simulation (e.g. Mehrotra and Fame, 2003 and Anton et al. 1999). As mentioned earlier, queueing insights and results can here be still most useful for a number of reasons (verification, validation, orders of magnitude, insights and scenario development) (see van Dijk and Van der Sluis, 2005, for a more detailed discussion on this combined approach). Also specialized simulation optimization packages are nowadays available that may speed up and automate an ‘optimization’ search (e.g. see Krug 2002).

Structural results

Nevertheless, specialized results and insights for the specific optimization of call center overflow problem of interest will generally still be required. In this respect, also results from a second approach can be highly useful. An approach of establishing structural results (e.g. Koole 1998, Bhulai and Koole 2003, Wallace and Whitt 2004). Recently, results in this direction are also developed as based on stochastic monotonicity and Markov decision theory (Koole and Pot, 2004).

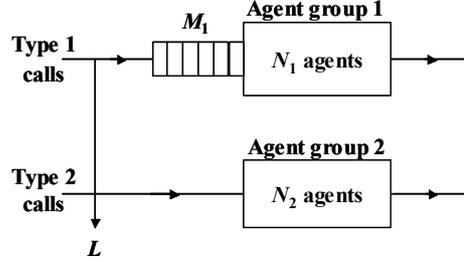
Bounding and error bound approach

Finally, as a third approach, rather than by accurate computation and optimization, a bounding approach might be followed. In this approach, the unsolvable system is modified into a solvable system which guarantees secure analytic performance bound. In addition, analytic error bounds might come along that quantify the (in)accuracy of these performance bounds.

As such, an ‘optimization’ of system parameters such as an agent number or finite queue size can be executed on computational or analytic basis.

Call packing example.

As a generic example, consider a simple agent group 1 exclusively for regular calls of type 1 with N_1 agents and a finite waiting facility for at most M_1 calls. In addition, there is a second group of N_2 agents, which is primarily meant for type 2 calls but which can also handle type 1 calls as described below. This second group, however, is assumed to have no waiting facility. If a type 1 call cannot be accommodated at the first agent group or its waiting capacity, it may attempt to find a free agent from



the second group. If the second group is fully occupied, a call request at the second group of either type 1 or type 2 is lost. Let type i calls arrive at a rate λ_i with exponential call durations with parameter μ_i , $i = 1, 2$.

Here one may typically think of the second group as higher qualified agents (possibly staff members) for less regular but more complicated type 2 calls, which can also be used as standby for regular type 1 calls in excess situations.

As a measure of interest, let us focus on the loss probability of type 1 calls denoted by L_1 .

Unfortunately, the present system has no closed product form solution for the joint steady state distribution of the first and second agent group, so that there is no analytic expression for L_1 . (Clearly, the first agent group can be treated as a standard $M/M/N_1/N_1+M_1$ -queue. However, as the overflow stream is hyperexponential (e.g. Van Doorn, 1984) the second group cannot be seen as if in isolation). A variety of approximations and special results for overflow systems have therefore been presented in the literature (e.g. Akimaru and Takahashi 1983, Geish and Kobayasi 1980, Berezner et al.1998, El-Taha and Heath 2000, Lee 2004, Franx et al. 2004). Of special interest here is the equivalent random method (e.g. see Borst et al.1999, Shortle 2004).

As another direction (which is related to Hordijk and Ridder 1987 and Van Dijk 1997) we could also modify the system into a product form system. To this end, assume the call packing regime under which a type a call at group 2 is “artificially” switched back to the first agent group if a group (type) 1 agent becomes available.

Let $\mathbf{n} = (m_1, o_1, n_2)$ denote the number m_1 of type 1 calls at group 1, o_1 of overflowed type 1 calls at group 2 and n_2 of type 2 calls at group 2 and let $\bar{\pi}(m_1, o_1, n_2)$ be its steady state distribution. Then under call packing and with c a normalizing constant at the state space \bar{S} , one easily verifies the product form solution for $(\mathbf{n} \in \bar{S})$:

$$\bar{\pi}(m_1, o_1, n_2) = c \frac{1}{o_1!} \frac{1}{n_2!} \prod_{k=1}^{m_1} \frac{1}{f(k)} \left(\frac{\lambda_1}{\mu_1} \right)^{m_1+o_1} \left(\frac{\lambda_2}{\mu_2} \right)^{n_2} \quad \text{with } f(k) = k\mu_1 I_{\{k \leq N_1\}} + N_1\mu_1 I_{\{k > N_1\}} \quad (5.1)$$

Let \bar{L}_1 be the corresponding loss (%) of type 1 calls as by

$$\bar{L}_1 = \bar{\pi}(m_1=N_1, o_1+n_2=N_2) \quad (5.2)$$

The following result has then been proven in [Van Dijk and Van der Sluis (2004)]. (The proof is rather technical and relies upon a Markov reward comparison approach as outlined in Van Dijk 1998.)

$$\bar{L}_1 - \delta \leq L_1 \leq \bar{L}_1 \quad \text{with} \quad (5.3)$$

$$\delta = \bar{\pi}(o_1 > 0) \frac{N_1 \mu_1}{\lambda_1} \left(\frac{\lambda_1 + \mu_2}{\mu_2} \right).$$

As it turns out by numerical verification, as shown in this reference, the secure upper bound \bar{L}_1 , as well as in combination with the lower bound $\bar{L}_1 - \delta$ appears to be rather accurate (comparable and often more accurate than a standard type approximation by using loss expressions as if the overflow stream can also be regarded as a separate arrival rate). As a consequence, an optimization such as for dimensioning N_1 , M_1 and N_2 can be carried out on analytic basis as based upon the product form (5.1). Further steps in this direction for more complicated overflow and Skill Based Routing-structures seem of interest for further research.

Evaluation

Whether we should pool or not agent groups within or between call centers is a question for which there is no simple answer. Despite first intuition and simple early results in the queueing literature which seems to support that pooling is indeed beneficial, also opposite (counterintuitive) results can be concluded (and have been provided in the literature) as also based on standard queueing theory.

Nevertheless, in practical call center environments the general perception seems to exist that pooling is beneficial, both in terms of performance and capacities. To a large extent this appears to be true but not by far as advantageous as might be thought of by standard Erlang-C computations. Also for realistic call center orders of number of agents, traffic loads and performance norms, the effect of pooling, say of two equally loaded agent groups, may even turn out to be negative.

An awareness of the underlying reason for causing these degradations: the mixing of different call characteristics, the order of its effect and tools to predict and evaluate them, should therefore be present in call center environments.

In fact, a more selected form of pooling by overflow may lead to superior results in more than one respect over both an unpooled and fully pooled situation. Just some limited overflow may already lead to substantial improvement. This also appears to be in line with recent observations in the literature, most notably on SBR. As such, the question of pooling, though less complicated, appears to be related to and of a similar challenging nature as questions on Skill Based Routing (SBR).

The search for an ‘optimal pooling’ or an ‘optimal skill base routing’ structure thus appears to boil down to the same underlying queueing principles and trade off between efficiencies, variabilities and dynamic or flexible use of capacities. For this trade off, however, there is no simple answer. Further research in this direction is therefore strongly suggested.

Different research approaches in this direction have been opened, such as by the square-root principle, by automated simulation, by Markov decision and stochastic monotonicity theory or by stochastic bounding techniques. But all of them, as well as other that can be thought of, are still highly open for continuing and future research as well as practical call center application.

As such we would like to invite the interested reader, either call center practitioner or researcher, to join and pool this research.

References

AKIMARU, H. AND H. TAKAHASHI (1983). An Approximate Formula for Individual Call Losses in Overflow Systems, *IEEE Trans. Comm* **31**, 808-811.

- ANTON, J., V. BAPAT AND B. HALL (1999). *Call Center Performance Enhancement Using Simulation and Modelling*. Purdue University Press.
- BEREZNER, S.A., A.E. KRZESINSKI AND P.G. TAYLOR (1998). A Product-Form “Loss Network” with a Form of Queueing, *J. Appl. Probab.* **34**, 1075–1078.
- BHULAI, S. AND G.M. KOOLE (2003). On the structure of value functions for threshold policies in queueing models, *Journal of Applied Probability* **40**, 613-622.
- BIERMAN, H. JR., C.P. BONINI AND W.H. HAUSMAN (1986). *Quantitative Analysis for Business Decisions*. Irwin, Homewood, Ill.
- BORST, S. AND P. SERI (2000). Robust Algorithms for Sharing Agents with Multiple Skills, Bell Labs Lucent Technologies. Unpublished report.
- BORST, S., BOUCHERIE, R.J. AND O.J. BOXMA, ERMR: A Generalised Equivalent Random Method for Overflow Systems with Repacking, in: P. Key, D. Smith (Eds.), ITC 16, Elsevier, Amsterdam, 1999, pp. 313-323.
- BORST, S.C., A. MANDELBAUM AND M.I. REIMAN (2004). Dimensioning large call centers. *Operations Research* **52**, 17-34.
- BUZACOTT, J.A. (1996). Commonalities in Reengineered Business Processes: Models and Issues, *Management Science* **42**, 768-782.
- BUZACOTT, J.A., SHANTHIKUMAR, J.G. AND D.D. YAO (1994). Jackson Network Models of Manufacturing Systems. *Stochastic Modeling and Analysis of Manufacturing Systems*, D.D. Yao (ed.), Springer-Verlag, Chapter 1, 1-46.
- CHEVALIER PH. AND N. TABORDON (2003). Overflow Analysis and Cross-trained Servers, *International Journal of Production Research* **85**, 47-60.
- COOPER, R.B. (1981). *Introduction to Queueing Theory*. North-Holland, Amsterdam.
- COSMETATOS, G.P. (1976). Some Approximate Equilibrium Results for the Multi-Server Queue M/G/r, *Operations Research Quarterly* **27**, 615-620.
- EL-TAHA, M. AND J.R. HEATH (2000). Traffic Overflow in Loss Systems with Selective Trunk Reservation, *Performance Eval.* **41**, 295-306.
- FRANX, G.J. (2001). A Simple Solution for the M/D/c Waiting Time Distribution, *Operations Research Letters* **29**, 221-229.
- FRANX, G.J., G.M. KOOLE AND A. POT (2004). Approximating multi-skill blocking systems by hyper-exponential decomposition, submitted for publication.
- GANS, N., G. KOOLE AND A. MANDELBAUM (2003). Telephone call centers: Tutorial, Review and Research prospects. *Manufacturing and Service Operations Management* **5**, 79-141.
- GEIHS, K. AND H. KOBAYASHI (1982). Bounds on Buffer Overflow Probabilities in Communication Systems, *Performance Eval.* **2**, 149-160.
- GRASSMAN, W.K. (1988). Finding the Right Number of servers in Real-world Queueing Systems, *Interfaces* **18**, 94-104.
- HALFIN, S. AND W. WHITT (1981). Heavy-traffic limits for queues with many exponential servers. *Operations Research* **29**, 567-588.
- HILLIER, F.S. AND G.J. LIEBERMAN (1974). *Introduction to Operations Research*, 2nd edition, Holden-Day, San Francisco.
- HORDIJK, A. AND A. RIDDER (1987). Stochastic Inequalities for an Overflow Model. *J. Appl. Prob.* **24**, 696-708.
- JENNINGS, O.B., A. MANDELBAUM, W.A. MASSEY AND W. WHITT (1996). Server Staffing to Meet Time-Varying Demand, *Management Science* **42**, 1383-1394.

- KLEINROCK, L. (1976). *Queueing Systems, Vol. II: Computer Applications*, Wiley, New York.
- KOOLE, G.M. (1998). Structural results for the control of queueing systems using event-based dynamic programming. *Queueing Systems* **30**, 323-339.
- KOOLE, G.M. AND A. POT (2004). Profit maximization and monotonicity results for inbound call centers, submitted for publication.
- KRUG, W. (2002). *Modelling, Simulation and Optimisation for Manufacturing, Organisational and Logistical Processes*, Erlangen, Gruner Druck GmbH.
- LARSON, R. C. (1987). Perspectives on Queues: Social Justice and The Psychology of Queueing, *Operations Research* **35**, 895-905.
- LEE, J. (2004). Asymptotics of Overflow Probabilities in Jackson Networks. *Oper. Res. Lett.* **32**, 265-272.
- LOCH, C. H. (1998). Operations Management and Reengineering, *European Management Journal* **16**, 306-317.
- MANDELBAUM, A. AND M.I. REIMAN (1998) On Pooling in Queueing Networks, *Management Science* **44**, 971-981.
- MAISTER, D.H. AND P. RECH (1985). The Psychology of Waiting Times, in *The Service Encounter*, eds. J.A. Czepiel, M.R. Solomon and C. Suprenant, Heath and Company, Lexington Books. <http://www.davidmaister.com/articlesby.asp>
- MEHROTRA, V. AND J. FAMA (2003). Call Center Simulation Modeling: methods, challenges, and opportunities, *Proceedings of the 2003 Winter Simulation Conference*, 135-143.
- MITZENMACHER, M. (1996). *The Power of Two Choices in Randomized Load Balancing* Ph.D. Thesis.
- PUHALSKII, A.A. AND M.I. REIMAN (2000). The Multiclass GI/PH/N Queue in the Halfin-Whitt Regime, *Adv. Appl. Prob.* **32**, 564-595.
- ROTHKOPF, M.H. AND P. RECH (1987). Perspectives on Queues: Combining Queues is not Always Beneficial, *Operations Research* **35**, 906-909.
- SHORTLE, J.F. (2004). The Equivalent Random Method with Hyper-Exponential Service, *Performance Evaluation* **57**, 409-422.
- SMITH, D.R. AND W. WHITT (1981). Resource Sharing for Efficiency in Traffic Systems *Bell System Tech. J.* **60**, 39-55.
- STANFORD, D. AND W. K. GRASSMANN (1993). The Bilingual Server Systems: a Queueing Model Featuring Fully and Partially Qualified servers, *Management Science* **31**, 221-277.
- STIDHAM, S.JR (1970). On the Optimality of Single Server Queueing Systems, *Operations Research* **18**, 708-732.
- TIJMS, H.C. (1994). *Stochastic Models: An Algorithmic Approach*. Wiley, Chichester.
- TURNER, S. (1998). The Effect of Increasing Routing Choice on Resource Pooling. *Probability in the Engineering and Informational Sciences* **12**, 109-124.
- VAN DIJK, N.M. (1997). Bounds and Error Bounds for Queueing Networks. *Ann. Oper. Res.* **79**, 295-319.
- VAN DIJK, N.M. AND E. VAN DER SLUIS (2004). Call Packing Bounds for Overflow Queues, submitted for publication.
- VAN DIJK, N.M. AND E. VAN DER SLUIS (2005). Check-In Computation and Optimization by Simulation and IP in Combination, *European Journal of Operations Research* (to appear).
- VAN DOORN, E.A. (1984). On the Overflow Process from a Finite Markovian Queue. *Performance Eval.* **4**, 233-240.
- WAGNER, H.M. (1975). *Principles of Operations Research*, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ.

WALLACE, R.B. AND W. WHITT (2004). Resource Pooling and Staffing in Call Centers with Skill-Based Routing, submitted to *Manufacturing and Service Operations Management*.

WHITT, W. (1992). Understanding the Efficiency of Multi-Server Service Systems, *Management Science* **38**, 708-723.

WOLFF, R.W. (1989). *Stochastic Modelling and the Theory of Queues*. Prentice-Hall, Englewood Cliffs, NJ.

UNIVERSITY OF AMSTERDAM
FACULTY OF ECONOMICS AND ECONOMETRICS
ROETERSSTRAAT 11
1018 WB AMSTERDAM
THE NETHERLANDS
E-MAIL: N.M.vanDijk@uva.nl
H.J.vanderSluis@uva.nl
URL: <http://www.uva.fee.nl/ke/sluis>