INSTITUTE OF
ACTUARIAL SCIENCE
&
ECONOMETRICS

REPORT AE 5/2005

Pooling is not Always the Answer

N.M. van Dijk
E. van der Sluis

University of Amsterdam

POOLING IS NOT ALWAYS THE ANSWER
(A Technical Note)

NICO M. VAN DIJK AND ERIK VAN DER SLUIS

University of Amsterdam

Abstract

This technical note studies whether pooling two (e.g. call center) server (e.g. agent) groups with different service (e.g. short and long call) types is always 'optimal'.

First some recent results are reviewed in an instructive manner to argue and illustrate a substantial improvement by simple overflow scenarios, as based upon a standard queueing result for the effect of variability. Next it shows that further improvements are possible in two directions:

(i) by prioritizing short jobs for an average improvement;

(ii) by optimal thresholds, which prioritize long jobs,

for a slight but strict improvement over pooling.

Also the dependence on the number of servers (agents) and the ratio of short and long jobs is investigated. It shows that the improvement patterns are robust up to some trade-off number of agents for the overflow scenarios and more or less constant allover for the prioritizing scenarios. The scenario can thus be regarded as practical.

The results are of both practical and theoretical interest: practical for possible implementation and awareness for extensions to more complex situations; theoretical for the 'counter- intuitive' results and further research.

1. Introduction

In call centers (as well as other service environments) the general perception appears to exist that it is always advantageous, at least from a performance and capacity point of view, to pool service capacities; for call centers that is: agent groups.

This perception seems supported by simple queueing results as has already been taught in early introductory OR-textbooks (e.g. [5], [6]) Nevertheless, the general validity of this perception remains to be questioned as shown in a recent paper by the authors [10]. (For the simple single server case it has also simultaneously been addressed in INFORMS Transactions on Education [2]).

More precisely, both [9] and [12] do not only prove a reduction factor in the case of identical services but also present a counterintuitive example for the simple case of a single server. These single server examples illustrate an opposite effect of pooling for the average waiting time when different services are involved. And also non-quantitative aspects, such as of psychological nature, that may have to be taken into account in relation to pooling are reported (see, [7] and [8]).

A more extensive list of and discussion on other and recent references related to pooling for call centers can be found in [10]. Most notably, in most of these references much attention has been paid to the computation of agent numbers as based on the square root principle introduced in [4]. For elegant surveys see [1], [3] and [11].

In [10], in contrast, it was investigated whether or not it was advantageous to pool two call center groups with different call types while keeping the capacities unchanged. Both approximate analytic expressions and insights were provided, as based on classical queueing results, to show and argue that pooling is not necessarily beneficial. These general results were supported by extensive numerical results for agent groups of reasonably realistic size.

Furthermore, it was also shown and numerically supported that considerable improvements could still be achieved by simple overflow scenarios (two-way or one-way), with just a small % of overflow. The numerical results indicated that the question of pooling is not only of theoretical but particularly also of practical interest. Nevertheless, two questions of both practical and theoretical interest remained open:

- At practical level: Does there exist a simple scenario which is practical to implement and which leads to further improvement also for realistic call center sizes?
- At theoretical level: Can we strictly improve the pooled scenario in all respects; that is for any type of call or customer rather than just on average?

This technical note aims to respond to these questions as well as to provide further insights and practical awareness that call centers managers can benefit from also for more complicated situations. To this end the following outline will be followed.

First, in section 2, for self-containedness and essential insights an instructive example and some numerical results from [10] are reviewed. Next, in section 3, new results are presented. First, in section 3.1 we address the first question by showing a substantial improvement for two prioritizing scenarios. In section 3.2 we address the second question by optimizing threshold scenarios. In section 3.3 we investigate to which extent the results hold true for arbitrary agent numbers. Section 3.4 gives a schematic overview and section 3.5 contains some general conclusions.

2. To pool or not?

2.1 Basic queueing results

In service operations pooling separate servers is generally perceived to be beneficial. Indeed, when queues for more servers are pooled into a single line none of the servers can ever be idle as long as there are calls / customers waiting. In other words, the service capacity cannot be used more efficiently.

This is also supported by the standard delay formula for a single (exponential) server (e.g. [6]) with arrival rate λ and service rate μ: $\boldsymbol{D} = 1 / (\mu - \lambda)$ as by pooling two of these servers, each with arrival rate λ and service rate μ, as if they are merged into one single server which is twice as fast, the mean delay would thus be reduced to by a factor 2 as by $\boldsymbol{D} = 1 / (2\mu - 2\lambda)$. In other words, at first glance pooling two servers thus seems to lead to a mean delay reduction of roughly 50%.

This reasoning however relies upon the implicit assumption of two identical servers or rather identical service characteristics, most notably, identical means. When different services are involved in contrast, the advantage of pooling remains questionable, as already addressed in the literature for the simple two-server case by references as [11] and [12]. More precisely, here a second basic result from queueing theory is to be realized: a result that seems hardly realized in practical call center environments. This result is known as Pollaczek-Khintchine (PK) formula. This formula, which is exact for the single server case, expresses the effect of service variability, as by

$\boldsymbol{W}_G = ½(1 + c^2)\, \boldsymbol{W}_E$ with $c^2 = \sigma^2 / \tau^2$ and

$\boldsymbol{W}_G$: the expected mean under a general service distribution G
(and E standing for the exponential case) with mean τ and standard deviation σ.

2.2 Instructive two-server example

As a consequence, if we would pool two separate servers, say for services of type 1 and type 2, as if it becomes one twice as faster server, we should also realize the variability due to mixing. By mixing different services (call types) extra service variability is brought in as each consecutive service can be of either type 1 or 2. As by the PK-formula this may lead to an increase of the mean waiting time (and delay).

This is illustrated in Figure 1 for the situation of two job (call) types 1 and 2 with mean service (call) durations $\tau_1 = 1$ and $\tau_2 = 10$ minutes but arrival rates $\lambda_1 = 10\ \lambda_2$. (In what follows we will refer to this situation as by a mix ratio $\boldsymbol{k} = 10$). The traffic load $\boldsymbol{\rho} = \lambda_1\tau_1 = \lambda_2\tau_2$ is set at 83%, so that both call types bring in an equal workload.

The results show that the unpooled case is still superior, at least for the average waiting time and particularly also for the mean waiting time of type 1 calls, the vast majority (91%) of all calls. Due to the variability, it thus seems preferable to keep the servers separate despite its inefficiency as a server might be idle when there is still a call waiting (at the other server). But also another scenario can now be thought of that may combine the advantage of both scenarios:

- No (or minimum) idleness as for the pooled case
- No (or minimum) service variability as for the unpooled case

To this end, the servers should still be kept devoted for type 1 and type 2 jobs (calls). But in addition, server 2 can also take a type 1 job (call) if there is no type 2 job (call) waiting and vice versa. The results for this two-way overflow system, as shown in Figure 1, indicate an improvement over both the pooled and the unpooled case. (Note however that the unpooled case is still by far superior for type 1 jobs).

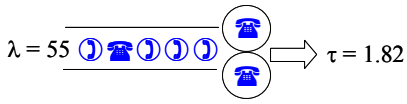
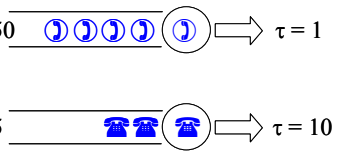
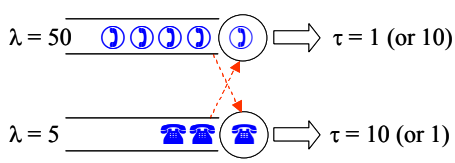
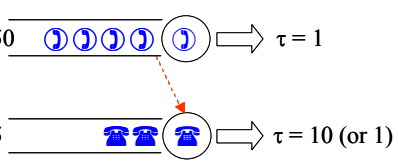
Pooled system	$W_A = 6.15$	**Unpooled system**	$W_A = 4.55$
	$W = 6.15$		$W_1 = 2.50$ $W_2 = 25.0$
Two-way overflow	$W_A = 4.11$	**One-way overflow**	$W_A = 3.92$
	$W_1 = 3.66$ $W_2 = 8.58$		$W_1 = 1.80$ $W_2 = 25.2$

Figure 1. Single server simulation comparison ($k = 10$; $\rho = 0.83$)

In this two-way overflow scenario there is no idleness at all (none of the servers can ever be idle while there is still a call waiting). However, the overflow of type 2 calls may imply long disturbances (and thus a large variability) at server 1. It might also be advantageous to allow some idleness at server 1. As a further improvement, a one-way overflow is therefore suggested in which type 1 calls may but type 2 calls may not overflow. This scenario not only shows a further reduction of the waiting time allover but also improves the unpooled scenario for type 1 (91%) calls. (The price to pay here is a waiting time increase for the small % of type 2 calls). In addition, the overflow % of calls is less than 10%.

2.3 Larger server numbers

The results as shown above for the two-server example might be thought of as of merely theoretical interest. However, as recently shown in [10], similar results also apply for larger server numbers as of more practical call center order such as with up to over 100 (for the pooled case) agents. This is illustrated in Table 1. The results are shown for the one-sided overflow scenario with overflow for type 1 calls (the short calls) to server 2. Also note that this overflow variant improves the performance for type 1 calls (more than 90%). In addition, one may observe the small percentage of overflow. The one-way overflow scenario (called variant 3) appears to improve both the unpooled (variant 1) and the pooled (variant 2) scenario for small and medium sized agent groups. This scenario keeps the call types separate and only leads to or requires a small percentage of overflow from type 1 calls (the short calls) to server 2.

	Unpooled (variant 1)			Pooled (variant 2)	Overflow (variant 3)			
s	W1	W2	Wav	W	W1	W2	Wav	% overflow
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 1. ***Simulation results of the three variants for pooling two unequal groups*** **($k = 10$; $\rho = 0.9$)**

Remark (Service distribution) As mentioned and shown in [10] by both (approximate) analytic expressions (for the pooled and unpooled scenarios) and by simulation, similar results can be obtained for arbitrary service (call) distributions. The choice for deterministic service times is merely based on illustrating the mix effect most distinctively.

3. Can we do better?

As illustrated in section 2 the basic queueing insights did not only lead to intriguing (and possibly counterintuitive) results for the simple example of two servers but also to results that seem to contradict the general perception that pooling is (always) beneficial for larger server numbers, such as for more realistic call center agent groups.

As is turned out, the one-way overflow scenario appeared to be superior up to some reasonably large number of servers. Only beyond this number the pooled scenario, seems to win. Nevertheless, the results and insights provided did not guarantee at all that this is the end. As also indicated in the introduction, questions of both practical and theoretical interest remain:

1. (*Practical*) Can we still do better? And if so, is this highly complex or still practical to implement?
2. (*Theoretical*) Can there exist a scenario that improves the pooled scenario for both type 1 (short) and type 2 (long) jobs?

This section will respond to these questions.

3.1 Average improvements

To this end, in addition to the four earlier scenarios presented, we introduce three more scenarios as specified and listed in Table 2 below. Here we note in advance that in all scenarios a job might

No.	Scenario	Specification
1	Pooled	One queue for type 1 and 2.
2	Unpooled	A separate queue for each type. No overflow.
3	Two-way	A separate queues for each type. Servers of type 2 can take a type 1 job if there is no type 2 job waiting and vice versa.
4	One-way	A separate queue for each type. Only servers of type 2 can take a type 1 job if there is no type 2 job waiting. No overflow to servers of type 1.
5	Thr(Opt)	See section 3.2.
6	Prio(1,N)	A separate queue for each type. Type 1 jobs always have priority over type 2 jobs. Type 2 jobs are served only if there is no type 1 job waiting.
7	Prio(1,P)	As scenario 6, but in addition: When a type 1 job arrives, a type 2 job is pre-empted. When no more type 1 jobs are waiting, type 2 jobs are resumed.

eventually be served by a server from either group 1 or 2, except for scenario 2 (unpooled) and scenario 4 (one-way) for type 2 customers. Scenario 5, which is somewhat more complicated, will be discussed in section 3.2, but it is included to give the complete overview.

Before presenting numerical results, let us briefly elaborate on these scenarios. In addition to the 'variability' aspect, the underlying reasoning for these scenarios is threefold:

- *Handling idleness*. The scenarios primarily differ in what happens when a server becomes idle (scenarios 3, 4, 5 and 6) or when an arrival takes place (scenario 7). This is in line with the general insight, as argued and illustrated in section 2, that 'pooling' is not so much about *'pooling capacity'* but about *'pooling idleness'*.
- *Majority of short jobs*. As the average waiting time treats each customer as equally important, the scenarios should be tailored to the largest customer group, that is type 1 (scenarios 4, 6 and 7).
- *Practical*. With exception of scenario 5, all scenarios can be regarded as simple and non-dynamic (in that they do not depend on the queue length). Accordingly, they can be seen as practical for possible implementation. Only scenario 7, in which services can be interrupted, seems less practical.

In line with these underlying reasons, both scenarios 6 and 7 provide full priority to type 1 jobs when a server becomes available, without and with pre-emption respectively. As a consequence, the results for type 1 jobs and, as these form the majority, the overall average can be expected to substantially improve either (one or two-way) overflow scenario. This is illustrated in Figure 2 for the same case as in Table 1 with $s = 10$ at a traffic load of $\rho = 90\%$. (Similar patterns and with the same ranking order are obtained for $\rho = 70\%$ or 80% and $s = 5$ or 20.)

Rank	No.	Scenario	W_A
7	1	Pooled	0.71
6	5	Thr(Opt)	0.68
5	2	Unpooled	0.63
4	3	Two-way	0.58
3	4	One-way	0.52
2	6	Prio(1,NP)	0.38
1	7	Prio(1,P)	0.20

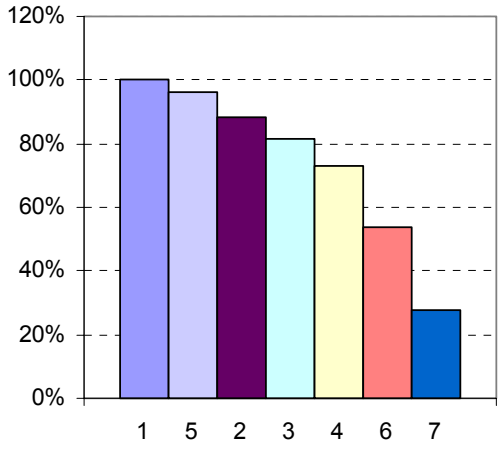


Figure 2. Average waiting times for different scenarios.

Overall average From this figure we draw the general conclusions:

1) *Not only the pooled but also the one-way scenario can still be improved substantially, at least for a reasonably small server group.*

2) *'Prioritization' of type 1 jobs is therefore required when a server idles.*

3) *'Prioritization' of type 1 jobs by pre-emption upon arrivals leads to further improvement.*

As the improvements are mainly due to the improvements for type 1 jobs with a price to be paid by type 2 jobs, Figure 3 shows the individual results for type 1 and type 2 jobs separately.

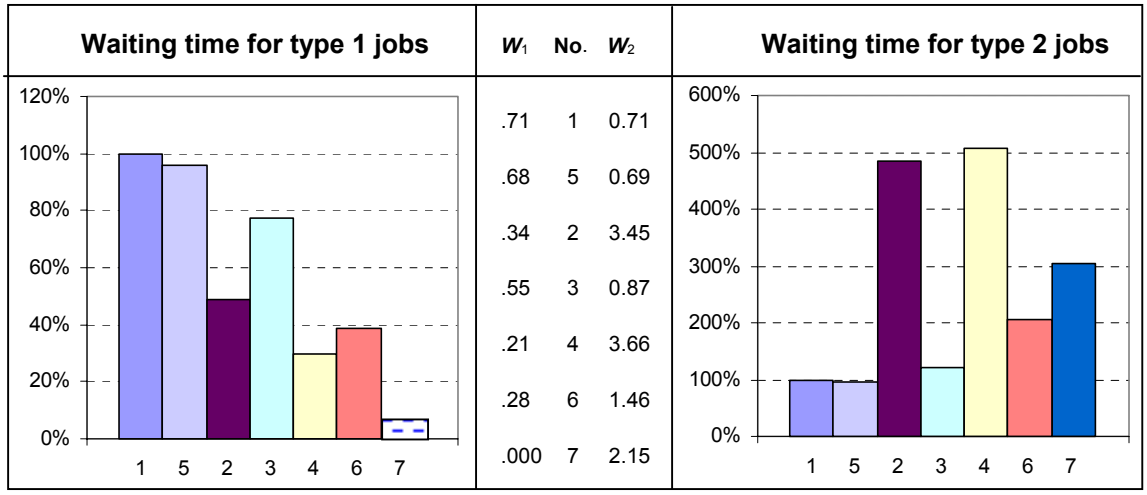


Figure 3. Average waiting time for type 1 and type 2 respectively for different scenarios.

Type 1 Clearly, for type 1 jobs the improvements are even more distinctive. Particularly:

1) *Under the pre-emptive scenario (7) waiting times seem to have completely vanished.*

2) *The simple one-way scenario (4), with no priority for type 1 jobs but also no interference by type 2 jobs, still performs slightly better than the Prio(1,N) scenario (6) with full priority for type 1 jobs.*

3) *Either the one-way scenario (4) or Prio(1,N) scenario (6) leads to a substantial reduction for type 1 jobs*

Type 2 Also some observations for type 2 jobs (in same scenario order) are of interest in themselves:

1) *Both the Prio(1,N) and Prio(1,P) scenarios (6 and 7), that provide substantial attention and accordingly improvement for type 1 jobs, also improve the completely unpooled (2) or one-way scenario (4) for type 2 jobs.*

2) *As opposed to the pooled reference scenario (1) the relatively practical Prio(1,N) scenario (6), with an allover improvement of roughly 50% or more, (for the average and type 1 jobs), only leads to a deterioration of roughly 50% for type 2 jobs.*

3.2 Thr(Opt) scenario

Now let us return to the second intriguing question whether a scenario can be found that improves pooling for both type 1 and type 2. This is achieved by scenario 5. This scenario is dynamic in that the queue lengths play a role. Here, a search has to be executed to find optimal threshold numbers. To this end, let the scenario Thr(θ_1, θ_2) be specified by:

Thr(θ_1, θ_2): when a server of server group j (j = 1, 2) becomes available it will give priority to a job of type i where,

$$i = \begin{cases} 1 & \text{if } m_1 \geq \theta_1 \wedge m_2 < \theta_2 \\ 2 & \text{if } m_1 < \theta_1 \wedge m_2 \geq \theta_2 \\ j & \text{if } m_1 \geq \theta_1 \wedge m_2 \geq \theta_2 \text{ or } m_1 < \theta_1 \wedge m_2 < \theta_2 \end{cases}$$

with m_i: the number of type i jobs waiting, i = 1, 2.

Note that

Thr(1, ∞) = Prio(1,N) and

Thr(1, 1) = Thr(∞, ∞) = Two-way.

Among these dynamic (or queue length dependent) scenarios Thr(θ_1, θ_2) a scenario is sought which strictly improves the pooled scenario. A scenario Thr(Opt) is determined that takes into account the waiting times of both job types by:

Step 1: $\min_{\theta_1,\theta_2} \max\{\boldsymbol{W}_1[\text{Thr}(\theta_1,\theta_2)], \boldsymbol{W}_2[\text{Thr}(\theta_1,\theta_2)]\}$ (2.1)

This leads to an optimal threshold combination $(\theta_1, \theta_2)^*$
and overall average waiting time under $(\theta_1, \theta_2)^*$: $\boldsymbol{W}_A[\text{Thr}(\theta_1, \theta_2)^*]$.

Step 2: If $\boldsymbol{W}_A[\text{Thr}(\theta_1, \theta_2)^*] < \boldsymbol{W}_{\text{Pooled}}$ compute:

$$\begin{aligned} \boldsymbol{W}_{\text{A}}[\text{Thr(Opt)}] &= \min_{\theta_1,\theta_2} \boldsymbol{W}_A[\text{Thr}(\theta_1,\theta_2)] \\ &\text{s.t. } \max\{\boldsymbol{W}_1[\text{Thr}(\theta_1,\theta_2)], \boldsymbol{W}_2[\text{Thr}(\theta_1,\theta_2)]\} < \boldsymbol{W}_{\text{Pooled}} \end{aligned} \quad (2.2)$$

This leads to an optimal threshold combination $(\theta_1, \theta_2)^{**}$.

Step 3: If $(\theta_1, \theta_2)^{**}$ exists then $\boldsymbol{W}_A[\text{Thr(Opt)}] = \boldsymbol{W}_A[\text{Thr}(\theta_1, \theta_2)^{**}]$,
otherwise $\boldsymbol{W}_A[\text{Thr(Opt)}] = \boldsymbol{W}_A[\text{Thr}(\theta_1, \theta_2)^*]$.

Table 2 lists optimal threshold combination $(\theta_1, \theta_2)^{**}$ (if it exists), for which the pooled scenario is improved allover, and optimal threshold combinations $(\theta_1, \theta_2)^{*}$ otherwise for different values of s, mix ratios k and $\rho = 0.9$. It shows that $(\theta_1, \theta_2)^{**}$ does not always exists. For example, for $k = 10$ and s=2, at least one of the two job types will always be worse than for the pooled case. However, for most s, k-values $(\theta_1, \theta_2)^{**}$ appears to exist.

k \ s	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	(4,2)	(4,2)	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	**(3,2)**	**(3,2)**	**(3,2)**	**(3,2)**	(3,2)
3	(4,1)	(4,1)	**(3,1)**	**(3,1)**	**(3,1)**	(5,2)	(5,2)	(5,2)	(5,2)	(5,2)	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**
4	**(5,1)**	(7,2)	**(4,1)**	**(4,1)**	**(6,2)**	**(6,2)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	(3,1)	(3,1)	**(5,2)**	**(5,2)**	**(2,1)**	**(2,1)**	**(2,1)**	**(2,1)**
5	**(9,2)**	(6,1)	**(5,1)**	**(5,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(7,2)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(6,2)**	**(6,2)**	**(6,2)**
6	**(8,1)**	**(7,1)**	**(6,1)**	**(5,1)**	**(5,1)**	**(5,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**
7	(13,2)	**(8,1)**	**(7,1)**	**(6,1)**	**(10,2)**	**(5,1)**	**(5,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**	**(3,1)**
8	**(14,2)**	**(9,1)**	**(8,1)**	**(7,1)**	**(6,1)**	**(11,2)**	**(5,1)**	**(5,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(9,2)**	**(3,1)**	**(3,1)**	**(9,2)**	**(9,2)**	**(9,2)**	(9,2)
9	**(12,1)**	(15,2)	**(9,1)**	**(8,1)**	**(7,1)**	**(6,1)**	**(6,1)**	**(5,1)**	**(5,1)**	**(5,1)**	**(5,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(3,1)**	**(3,1)**	**(3,1)**
10	(18,2)	(12,1)	**(15,2)**	**(8,1)**	**(7,1)**	**(7,1)**	**(6,1)**	**(6,1)**	**(5,1)**	**(5,1)**	**(5,1)**	**(5,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(4,1)**	**(3,1)**	**(3,1)**

(θ1, θ2)** solution (θ1, θ2)* solution (θ1, θ2)* solution, pooling average better

Table 2. Optimal threshold combinations $(\theta_1, \theta_2)^{**}$ or $(\theta_1, \theta_2)^{*}$ for different mix ratios k and server group sizes s.

Some noteworthy observations from Table 2 are:

1) *To optimize the pooled scenario for both type 1 and type 2 jobs some more "preference" or "prioritization" appears to be required as indicated by the "optimal" levels (θ_1, θ_2).*

2) *The optimal thresholds seem to prioritize type 2 (long) jobs. ($\theta_2 = 1$ or 2)*

3) *A strict improvement over the pooled scenario for both job types does not only apply to large mix ratios ($k = 10$) but also to small mix ratios ($k = 2, 3$).*

4) *The optimal strategy appears to be rather robust for mix ratio k and number of agents s.*

The improvements are only in the order of a few % but consistently outside 95% confidence intervals with a range of ½%. Figure 4 shows the relative improvements (waiting time reduction) for $k = 10$ and $s = 1$ to 20.

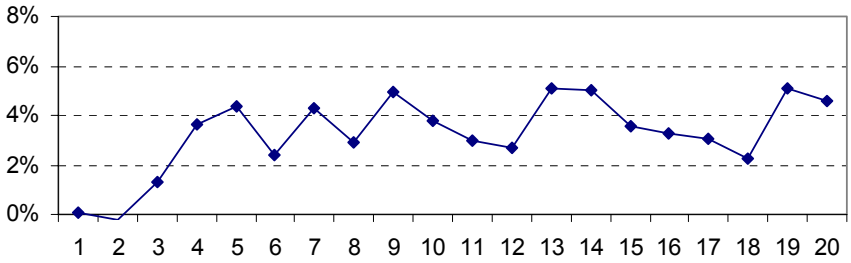


Figure 4. Relative improvement in average waiting time.

3.3 Large number of servers

One may question to which extent the observations and conclusions as made in section 3.2 can be regarded as representative, particularly for other numbers of servers (agents). (As already mentioned the same pattern also applies for s = 10 or 30 and ρ = .70 or .80). Figure 5 also illustrates the pattern for s = 1, 5, 10, 20 and 50 for ρ = 0.9 and k = 10. This leads to the following observations:

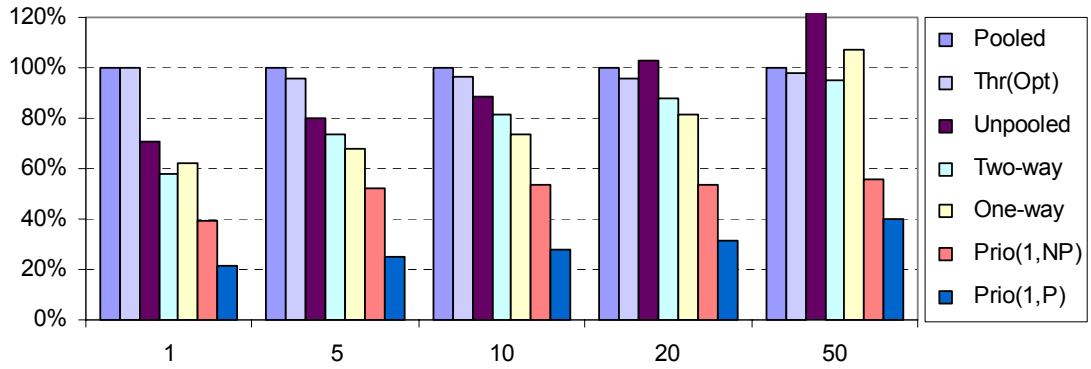


Figure 5. Average waiting times for different scenarios and server group sizes from 1 to 50.

- *The improvement patterns do indeed seem rather robust for the size of the agent group except that the unpooled and one-way scenario (2 and 3) are eventually outperformed by pooling for sufficiently large numbers of servers (agents).*
- *The type 1 priority scenarios (either 6 or 7), in contrast, appear to provide a constant improvement regardless of the server (agent) number.*
- *Furthermore, one may note the double interchange of rankings for the one-way and two-way overflow scenarios (3 and 4) for s = 1 and s = 50.*

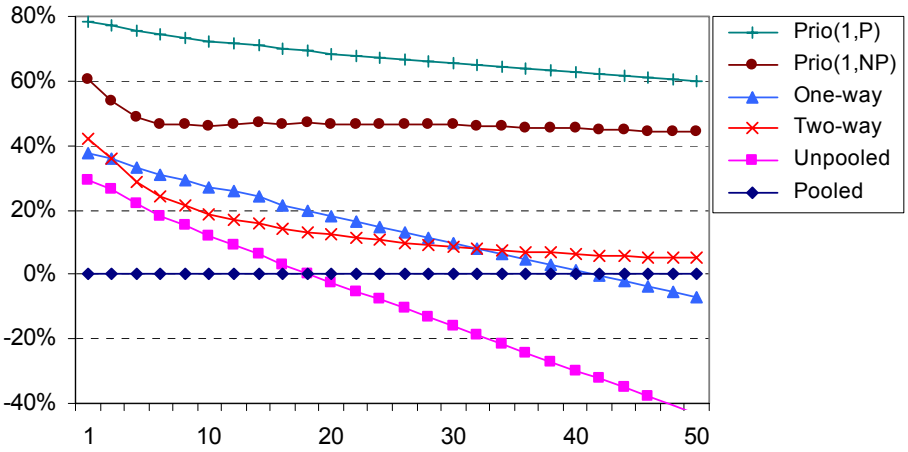


Figure 6. Reduction in average waiting time compared to pooling for different scenarios and server group sizes from 1 to 50.

These observations are also supported by Figure 6 which presents the *s*-dependence more graphically. From Figure 6 it also becomes clear that

- *For scenarios 6 and 7 the improvement appears to be become (asymptotically) constant. For the scenarios 2 and 4 the improvement vanishes for sufficiently large s.*
- *The improvements are monotonically decreasing in s.*

3.4 A schematic overview

So far, we have restricted the results presented to the representative case of $\rho = .9$ and $k = 10$. This case can be regarded as rather illustrative. But while $\rho = .7$, .8 or .9 seems quite realistic, $k = 10$ might be considered as rather extreme. For small k one may expect less distinctive if not opposite results. In Table 3 we therefore provide a rough schematic overview for different agent numbers (the actual intervals slightly differ for different scenarios as well as values for k) and mix ratios k. Herein, the inequality symbol < indicates a smaller average waiting time. Table 3 illustrates that pooling is indeed always 'optimal' but only in the situation of identical services ($k = 1$).

s	**1** **2** **3**	**4** **5** **....** **10** **....** **15** **....** **20** **....** **30** **....** **40** **....** **50** **....**
***k* = 10** ***k* = 9**	Prio(1,P) < Prio(1,N) < 2-way < 1-way < unpooled < **pooled**	Prio(1,P) < Prio(1,N) < 1-way < 2-way < unpooled < **pooled** … … Prio(1,P) < Prio(1,N) < 2-way < **pooled** < 1-way < unpooled
***k* = 8** ***k* = 7**	Prio(1,P) < Prio(1,N) < 2-way < 1-way < unpooled < **pooled**	Prio(1,P) < Prio(1,N) < 2-way < 1-way < **pooled** < unpooled Prio(1,P) < Prio(1,N) < 2-way < **pooled** < 1-way < unpooled
***k* = 6**	Prio(1,P) < Prio(1,N) < 2-way < 1-way < **pooled** < unpooled	Prio(1,P) < Prio(1,N) < 2-way < **pooled** < 1-way < unpooled
2 ≤ *k* ≤ 5	Prio(1,P) < Prio(1,N) < 2-way < **pooled** < 1-way < unpooled	
***k* = 1**	**pooled** = 2-way =Prio(1,N) ≤ Prio(1,P) << 1-way < unpooled	

Table 3. Ranking of scenarios for different mix ratios and server group sizes.

Remark 3.1 (General service distribution) As in section 2, throughout this section the service times are assumed to be deterministic, but similar patterns and results can be obtained for arbitrary service times (for example, but not necessarily, exponentially distributed).

4. Global conclusions

To summarize the various observations and as an evaluation the following global conclusions are drawn:

1. Pooling can be substantially improved by a number of simple 'practical' scenarios as:
   - A strictly separated scenario (for sufficiently small number of agents),
   - A one-way or two-way overflow scenario with improvements up to some sufficiently large number of agents, and by
   - A priority scenario (pre-emptive or non-preemptive) for short (type 1) jobs for arbitrary number of agents and with a more or less constant improvement order.
2. Pooling can even be improved for both short and long jobs by optimal threshold scenarios. These optimal scenarios generally tend to prioritize long (type 2) jobs.
3. Pooling is thus far from optimal, at least from an average waiting time perspective. This could already have been concluded from basic queueing insights from.
4. Also the weights and the proportions for different job classes may have to be taken into account (say other than proportional) and may effect which scenario is optimal.
5. Simulation was necessarily required to evaluate the scenarios.
6. Particularly for more complex pooling questions, with multiple skills and preferences, a combination of queueing (for insights) and of simulation (for evaluation and optimization) is therefore suggested if not required.

References

[1] BORST, S.C., A. MANDELBAUM AND M.I. REIMAN (2004). Dimensioning large call centers. *Operations Research* **52**, 17-34.

[2] CATTANI, K. AND G.M. SCHMIDT (2005). The pooling Principle. *INFORMS Transactions on Education* **5**. (http://ite.pubs.informs.org/Vol4No2/CattaniSchmidt/)

[3] GANS, N., G. KOOLE AND A. MANDELBAUM (2003). Telephone call centers: Tutorial, Review and Research prospects. *Manufacturing and Service Operations Management* **5**, 79-141.

[4] HALFIN, S. AND W. WHITT (1981). Heavy-traffic limits for queues with many exponential servers. *Operations Research* **29**, 567-588.

[5] HILLIER, F.S. AND G.J. LIEBERMAN (1974). *Introduction to Operations Research*, 2nd edition, Holden-Day, San Francisco.

[6] KLEINROCK, L. (1976). *Queueing Systems, Vol. II: Computer Applications*, Wiley, New York.

[7] LARSON, R. C. (1987). Perspectives on Queues: Social Justice and The Psychology of Queueing, *Operations Research* **35**, 895-905.

[8] ROTHKOPF, M.H. AND P. RECH (1987). Perspectives on Queues: Combining Queues is not Always Beneficial, *Operations Research* **35**, 906-909.

[9] SMITH, D.R. AND W. WHITT (1981). Resource Sharing for Efficiency in Traffic Systems *Bell System Tech. J.* **60**, 39-55.

[10] VAN DIJK, N.M. AND VAN DER SLUIS (2005). To Pool or not to Pool in Call Centers. Submitted to Production and Operations Management. (http://www.fee.uva.nl/ke/sluis/)

[11] 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. (http://www.columbia.edu/~ww2040/)

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

AUTHORS ADDRESS:

UNIVERSITY OF AMSTERDAM
FACULTY OF ECONOMICS AND ECONOMETRICS
ROETERSSTRAAT 11, 1018 WB AMSTERDAM, THE NETHERLANDS