European Journal of Operational Research 197 (2009) 415–421

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Short Communication

Pooling is not the answer

Nico van Dijk, Erik van der Sluis *

University of Amsterdam, Faculty of Economics and Business, Roetersstraat 11, 1018 WB Amsterdam, Netherlands

ARTICLE INFO

Article history:
Received 20 July 2006
Accepted 16 June 2008
Available online 25 June 2008

Keywords:
Pooling
Queueing
Overflow
Priority rule
Threshold rule

ABSTRACT

This note studies practical and theoretical scenarios to improve a completely pooled or unpooled scenario for two server groups (e.g. call center groups) with short and long jobs (e.g. calls). First, simple overflow (reported earlier) scenarios and priority rules are compared. Next, threshold rules are investigated for further practical improvement. Finally, a threshold rule is sought for a strict improvement over pooling for both short and long jobs. The practical value is illustrated numerically and appears to be consistent also for larger server numbers. The results show:

(i) An overall average improvement by prioritizing short jobs.
(ii) A slight but strict improvement over pooling by prioritizing long jobs.

The first result is of practical interest; the second one is more theoretical. A practical scheme is provided for an ordering of scenarios, up to realistically large numbers of servers as in call centers.

1. Introduction

In service environments, such as call centers, the general perception seems to exist that it is advantageous to pool service capacities, at least from a performance and capacity point of view.

For a single type of service, this perception is supported by simple M/M/s queueing results as can already be found in early introductory OR-textbooks. When different services are involved, the general validity of this perception remains to be questioned. In this case, 'counter-intuitive' examples can be found in Smith and Whitt (1981) and Wolff (1989) for two single servers. (Also see INFORMS Transactions on Education by Cattani and Schmidt, 2005.) These single server examples illustrate an opposite effect of pooling for the average waiting time.

More recently, this phenomenon of pooling (or splitting) service capacity has been elaborated upon, both analytically and numerical by Whitt (1999) and purely numerical in Van Dijk and Van der Sluis (2008). Extensive lists of references related to pooling call centers capacities can be found in these references. Most notably, in most of these references much attention has been paid to the computation of agent numbers as based upon the square root principle, see Borst et al. (2004), Gans et al. (2003) and Wallace and Whitt (2005). For another recent related reference on pooling see Remark 2.

The numerical results in Van Dijk and Van der Sluis (2008) also include simple overflow scenarios which indicate that the question of pooling is not as simple as it may seem and of interest for improvement. More precisely, questions of both practical and theoretical interest remained open as:

- Does there exist other simple and practical rules, which lead to further improvement?
- To which extent can these simple rules be regarded as 'optimal'?
- Can we strictly improve the pooled scenario?

This note aims to respond to these questions for average waiting times. First, to be self-contained and for some insights an instructive example and numerical results in line with Van Dijk and Van der Sluis (2008) are briefly reviewed. In Section 3, the first question is addressed by the comparison of simple scenarios, which also includes two prioritizing scenarios (a preemptive and non-preemptive one). Section 4 addresses the second question by a comparison with optimizing threshold rules. It roughly turns out that only a single threshold will be required and only for small number of servers. For larger server numbers a simple non-preemptive priority rule will be nearly optimal. In Section 5, a threshold rule is presented that strictly improves the pooled scenario. A schematic ordering of scenarios and some global conclusions completes the paper.

* Corresponding author. Tel.: +31 20 5254318.
E-mail address: h.j.vandersluis@uva.nl (E. van der Sluis).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.06.014

2. Simple scenarios

2.1. Basic case and notation

This note exclusively considers the situation of two arrival streams of service (e.g. call) requests, referred to as of types 1 and 2, with arrival rates λ_1 and λ_2, and mean service times τ_1 and τ_2, with equal workloads $\boldsymbol{\rho} = \lambda_1\tau_1 = \lambda_2\tau_2$, and two groups of servers each with s (identical) servers (hence $2s$ servers in total) which can handle either type of service. Let

$\boldsymbol{k} = \lambda_1/\lambda_2 = \tau_2/\tau_1$,
$\tau = [\lambda_1/(\lambda_1 + \lambda_2)]\tau_1 + [\lambda_2/(\lambda_1 + \lambda_2)]\tau_2$, and
$\boldsymbol{W}_A$: Average waiting time for all jobs,
$\boldsymbol{W}_1$: Average waiting time for type 1 jobs,
$\boldsymbol{W}_2$: Average waiting time for type 2 jobs.

2.2. Pool-or-not-or-overflow (earlier results)

In service operations, pooling service capacities is generally perceived to be advantageous. Indeed, when one type of service is involved this is shown directly by standard M/M/s expressions. (In Van Dijk and Van der Sluis, 2008, an approximate expression is provided for the effect of pooling two identical M/M/s systems, which shows a reduction factor of always at least $\rho/(1+\rho)$ for the mean waiting time.) However, when services are pooled with different service means, the effect is less beneficial as due to the mix variability then introduced. The observation is also in line with the famous Pollaczek–Khintchine (PK) formula, though only exact for a single server, which expresses the effect of service variability. An extensive and elegant analytic treatment of the PK-formula in relation to the question of pooling can be found in Whitt (1999).

As a consequence, both a pooled or unpooled scenario might be preferable depending on the mix ratio $\boldsymbol{k}$ and server number s. Also other scenarios can now be thought of to combine the advantage of both scenarios, i.e.

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

An overflow system therefore might lead to further improvement for the overall mean waiting time. As illustrated in Fig. 1, this turns out to be the case for the instructive example of $s = 1$ (two parallel servers) and $\boldsymbol{k} = 10$ (hence with 10 times more short jobs which are 10 times shorter) ($\lambda_1 = 50$, $\tau_1 = 1$; $\lambda_2 = 5$, $\tau_2 = 10$) by comparing the four basic scenarios of the pooled (P), the unpooled (U), a two-way overflow (2WO) and a one-way (1WO-1) scenario as specified by:

	Scenario	Specification
2WO	Two-way overflow	A separate queue for each type. An idle server, when there are no jobs of its own type waiting, will take a job waiting of the other type, if any.
1WO-1	One-way overflow type 1	A separate queue for each type. Only an idle server of type 2 and if there are no jobs waiting of type 2 will take a job from the other queue, if any.

Remark 1 (*Service time distribution*). Throughout deterministic service times are used to illustrate the effect of a mix ratio most distinctively. As shown in Van Dijk and Van der Sluis (2008) by both (approximate) analytic expressions (for the pooled and unpooled scenarios) and by simulation, similar results can be obtained for arbitrary service (call) distributions.

2.3. Larger server numbers

The results for the two-server example might be thought of as of merely theoretical interest. But similar results also apply to larger server numbers such as with up to over 100 (for the pooled case) servers, as of realistic call center size. This is illustrated by Table 1 and Fig. 2. The results are only shown for the one-way overflow scenario for type 1 calls (the short calls) to server 2.

Table 1
Results for the three scenarios ($k = 10$; $\rho = 0.9$)

s	Pooled	Unpooled		One-way overflow (type 1)		
	$\boldsymbol{W}_P$	$\boldsymbol{W}_1$	$\boldsymbol{W}_A$	W_1	W_A	% overflow
1	11.53	4.49	8.18	3.40	7.20	5.1
5	1.76	0.78	1.41	0.53	1.20	4.9
20	0.26	0.15	0.26	0.08	0.21	4.0
30	0.13	0.08	0.15	0.04	0.12	3.6
50	0.05	0.04	0.07	0.02	0.05	2.9
60	0.03	0.03	0.05	0.01	0.04	2.6

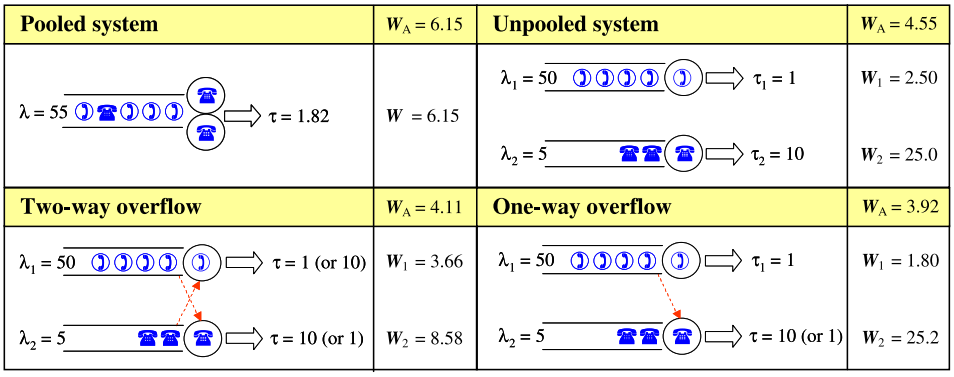


Fig. 1. Scenario comparison ($\boldsymbol{k} = 10$; $\rho = 0.83$).

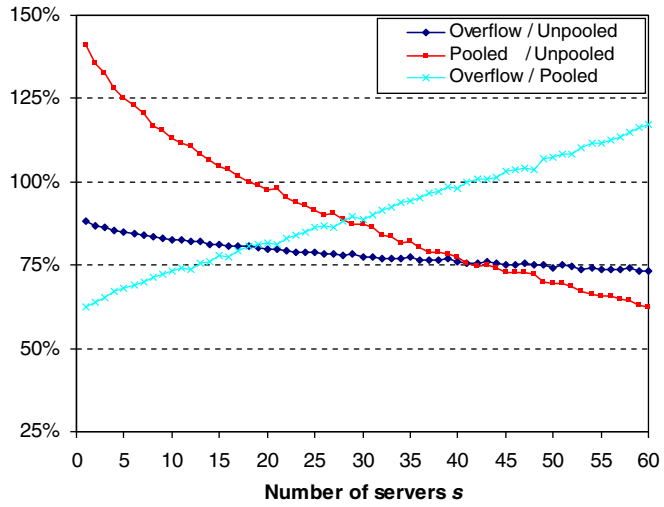


Fig. 2. Comparison of scenarios ($k = 10$).

(These results have not been reported explicitly in Van Dijk and Van der Sluis.)

Here, we can observe that the one-way overflow scenario is superior up to $s = 60$ (hence 120 servers in total) for the mean waiting time $\boldsymbol{W}_1$ of type 1 jobs, which in the example constitutes over 90% of all jobs. In addition, despite the significant positive effect of the type 1 overflow, the overflow percentage appears to remain rather small (say less than 5%). Though in a different setting of skilled based routing and with an objective of costs, a similar intriguing observation has been made in Wallace and Whitt (2005): "*a little flexibility may go a long way*".

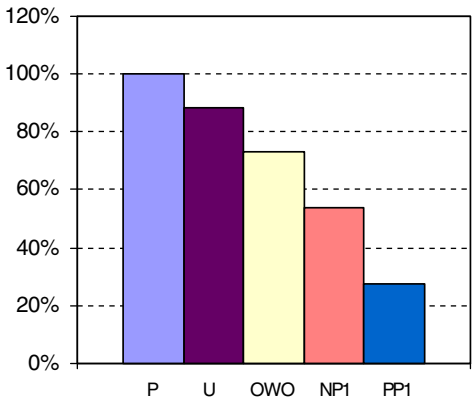
For the overall average waiting time the pooled scenario only appears to become preferable for a sufficient large number of servers, in the example of say more than 40 servers, as illustrated by Fig. 2.

3. Simple priority rules

As is turned out, among the four simple scenarios (P, U, 2WO, 1WO-1) the one-way overflow scenario for the short jobs appeared to be superior up to some reasonably large number of servers. Only beyond this number, the pooled scenario will win. Nevertheless, the results and insights do not guarantee at all that this is the end. Can we still do better? And if so, is it still practical to implement? Here two general queueing insights can be thought of

- *Handling idleness*. Waiting times are essentially dealt with by idleness (e.g. think of the relation $\boldsymbol{W} \sim 1/(1-\rho)$ for the M/M/1-queue). Pooling is not so much about *'pooling capacity'* but about*'pooling idleness'* when a server becomes available.
- *Majority of short jobs*. Waiting times are generally reduced all-over by handling shortest jobs first; in the present setting that is, type 1 jobs.

Consequently, two simple scenarios to improve the overall average waiting time are to prioritize type 1 jobs, either without preemption (service interruption) or with preemption of type 2 jobs when a type 1 arrives. In either way, service for a type 2 job only starts (or is resumed) when no more type 1 jobs are waiting.



Rank	Scenario	$\boldsymbol{W}_A$
5	Pooled	0.71
4	Unpooled	0.63
3	One-way	0.52
2	NP1	0.38
1	PP1	0.20

Fig. 3. Average waiting times for different scenarios.

	Scenario	Specification
NP1	Non-prmp-Priority-1	As in the pooled case and with priority for type 1 jobs when a server idles. Type 2 jobs are served only if there is no type 1 job waiting
PP1	Preemptive-Priority-1	As scenario NP1. In addition: when a type 1 job arrives, a type 2 job is preempted. When no more type 1 jobs are waiting, type 2 jobs are resumed

The possible improvement is illustrated in Fig. 3 for the situation with $k = 10$, $\rho = 0.9$ and $s = 10$ (20 servers in total). (To focus on type 1 jobs the two-way scenario, which would rank in between the unpooled and one-way scenario for the all-over average, is left out.) The results for this specific case (as well as the corresponding specific results for $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$) lead to the following:

3.1. Conclusions

- *The 1WO-1 scenario can still be improved substantially.*
- *'Prioritization' of type 1 jobs when a server idles is therefore required.*
- *'Prioritization' of type 1 jobs by preemption upon arrival leads to further improvement.*
- *Under the preemptive rule waiting times seem to have completely vanished.*
- *Both the NP1 and PP1 rule still improve the completely unpooled (U) or one-way overflow (1WO-1) scenario for type 2 jobs.*

The same pattern also applies for other values of ρ, like 70% or 80%, and similar results for varying number of servers as illustrated by Fig. 4, thus seem to justify the following main conclusions:

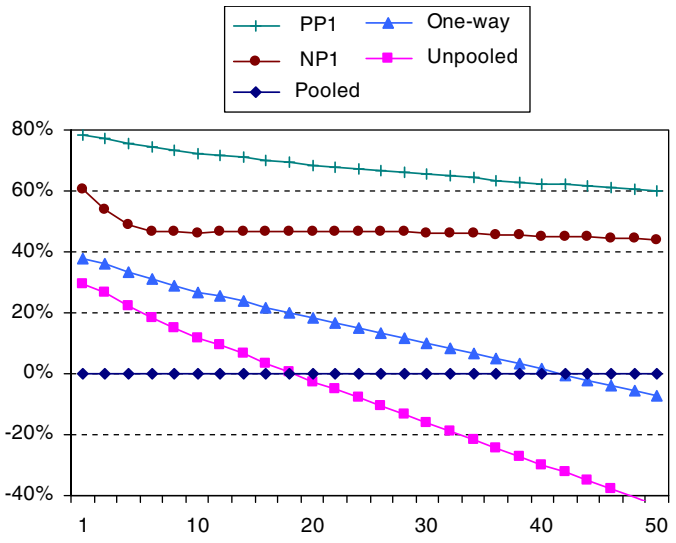


Fig. 4. Reduction in average waiting time compared to pooling for different scenarios and number of servers s.

1. *The improvement patterns seem rather robust for the number of servers except that the unpooled scenario and one-way scenario are eventually outperformed by pooling.*
2. *The type 1 priority rules appear to provide a consistent improvement.*

4. Threshold rules

As shown in Section 3, a simple priority rule, particularly the preemption scenario, for short (type 1) jobs generally, seems to perform quite well and to be 'optimal' among simple scenarios. Unfortunately, preemption (interruption) of service will generally be unacceptable or impractical. For practical interest, therefore, in this section we aim to investigate whether an improvement over a simple overflow or non-preemptive rule can be obtained by more sophisticated non-preemptive rules using threshold values on the queue lengths.

The length of both queues determines whether an idling server takes a job from its own queue or instead takes a job from the other queue (even though there are jobs in it own queue). With m_i the number of type i jobs waiting, $i = 1, 2$, the threshold rule is

$$\mathbf{Thr}(\theta_1^1, \theta_2^1; \theta_1^2, \theta_2^2; \Omega_1, \Omega_2)$$
$$: \begin{cases} \text{A server of type 1 serves jobs from queue 2 if either} \\ (i)\ m_2 \geqslant \theta_2^1 \wedge m_1 < \theta_1^2 \quad \text{or} \quad (ii)\ m_1 = 0 \wedge m_2 \geqslant \Omega_2; \\ \quad \text{otherwise, it serves jobs from queue 1} \\ \text{A server of type 2 serves jobs from queue 1 if either} \\ (i)\ m_1 \geqslant \theta_1^1 \wedge m_2 < \theta_2^2 \quad \text{or} \quad (ii)\ m_2 = 0 \wedge m_1 \geqslant \Omega_1; \\ \quad \text{otherwise, it serves jobs from queue 2} \end{cases}$$

By the θ_i-values, calls are thus given priority in case their queue length becomes too large. In case a queue becomes empty, overflow to an idling server is limited for both types. The threshold rule is illustrated in Fig. 5. The red (blue) area consists of states where type 1 (2) jobs are given priority. In states covered by the dashed areas, an idling server takes a job from its own queue. Note that

$$\begin{aligned} \text{Pooled} &= \text{Thr}(\infty, \infty, \times, \times, \infty, \infty), \\ \text{1WO} - 1 &= \text{Thr}(\infty, \infty, \times, \times, 1, \infty), \\ \text{2WO} &= \text{Thr}(\times, \times, 1, 1, 1, 1) = \text{Thr}(1, 1, 1, 1, \times, \times) \\ \text{NP1} &= \text{Thr}(1, \infty, \times, \infty, 1, 1) \end{aligned}$$

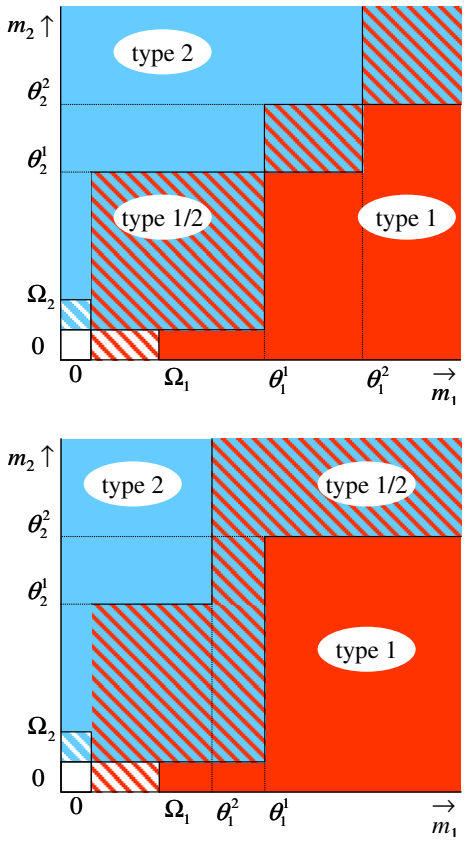


Fig. 5. Queue length dependent priorities, with $\theta_1^1 < \theta_1^2$ (top), $\theta_1^1 > \theta_1^2$ (bottom).

with $\times$ indicating that any arbitrary number can be chosen. In order to find a Thr-rule with minimal overall average waiting time, a search has to be executed to find optimal threshold numbers. As can be concluded from Table 2, for the optimal values it often holds that $\theta_1^2 = 1, \theta_2^1 = \theta_2^2 = \infty$ and $\Omega_1 = \Omega_2 = 1$. In these cases, the rule than reduces to the single threshold rule:

$$\mathbf{T}(\theta_1): \begin{cases} \text{server of either type serves jobs from queue 1 if either} \\ (i)\ m_1 \geqslant \theta_1 \quad \text{or} \quad (ii)\ m_2 = 0 \wedge m_1 \geqslant 1; \\ \quad \text{otherwise, it serves jobs from queue 2} \end{cases}$$

4.1. Conclusions

- $T(\theta_1)^* \approx \text{Thr}(\theta_1^1, \theta_2^1; \theta_1^2, \theta_2^2; \Omega_1, \Omega_2)^*$ for small s. For larger s ($s > 10$) the simple NP1 rule, which is easy to implement in practice, generally performs quite well.
- For all s, a simple single threshold rule $T(\theta_1)^*$ will be nearly optimal.
- $T(\theta_1)^* = \text{NP1}$. For k and s sufficient large (e.g. $k = 10$ and $s > 10$).

Remark 2 (*References*). A recent reference that may seem highly related is Gans and Zhou (2007). The model in this reference is quite different in that a pooled/dedicated service group is considered with the possibility of outsourcing for one job type. The objective is a cost minimization, rather than a performance optimization. As major difference, however, the effect of a mix ratio and its role for 'optimization', as essential in the present note, is not considered at all. In addition, the optimal threshold policy mentioned in this reference is different in that it specifies a value (for only one job type) at which a randomization for outsourcing takes place.

Table 2
Optimal threshold values

s	NP1	$Thr(\theta_1^1,\theta_2^1;\theta_1^2,\theta_2^2;\Omega_1,\Omega_2)$		$T(\theta_1)$	
	W_A	W_A	$(\theta_1^1,\theta_2^1;\theta_1^2,\theta_2^2;\Omega_1,\Omega_2)^*$	W_A	θ_1^*
1	4.58	4.36	3, ∞, 3, ∞, 1, 5	4.37	3
2	2.44	2.24	3, ∞, 3, ∞, 1, 2	2.27	4
3	1.63	1.49	4, ∞, 4, ∞, 1, 2	1.51	4
4	1.19	1.10	4, ∞, 4, ∞, 1, 3	1.12	4
5	0.93	0.86	4, ∞, 4, ∞, 1, 3	0.88	5
10	0.38	0.37	4, ∞, 4, ∞, 1, 2	0.38	1
15	0.21	0.21	1, ∞, 1, ∞, 1, 1	0.21	1
20	0.14	0.14	1, ∞, 1, ∞, 1, 1	0.14	1
30	0.07	0.07	1, ∞, 1, ∞, 1, 1	0.07	1

Clearly, a variety of threshold policies will have been used in the literature for specific cases and purposes (e.g. Bell and Williams, 2001; Squillante et al., 2001; Osogami et al., 2004). However, for the 'simple' situation as considered in this note, no general conclusion or extensive numerical support for threshold policies has been found.

The paper by Osogami et al. (2004) is the most related one to our note. However, in this paper a donor-beneficiary model was studied with a T1T2 policy for one a single overflow server. (The $Thr(\theta_1^1,\theta_2^1;\theta_1^2,\theta_2^2;\Omega_1,\Omega_2)$-rule can be regarded as a mixture of T1T2 policies, one for each server groups, combined with limitations on overflow in case of an empty queue.) Moreover, only preemption priority is considered.

5. A strict improvement

In the previous sections, improvements of the overall average waiting time were obtained by particularly improving the mean waiting time for type 1 jobs. However, in these scenarios so far also a price had to be paid by type 2 jobs (even though small and for just a small percentage of the jobs).

As another objective, this section addresses the question whether a scenario can be found that strictly improves the pooled scenario, that is, in mean waiting time, for both types 1 and 2 jobs. As shown in Section 3, for the overall average mean waiting time, an optimization by threshold values basically boiled down to just one threshold value θ_1 (to prioritize type 1 jobs). For the present purpose of a strict improvement, in contrast, also a prioritization of type 2 jobs might thus be expected and be required. Instead of one threshold value θ_1, we will therefore consider threshold rules with one threshold value θ_1 and θ_2 for either type of jobs. More precisely, let the threshold rule $\mathbf{S}(\theta_1, \theta_2) = \mathbf{Thr}(\theta_1, \theta_2, \theta_1, \theta_2, 1, 1)$, i.e. as specified by:

$$\mathbf{S}(\theta_1,\theta_2): \begin{cases} \text{A server of type 1 serves jobs from queue 2 if either} \\ (i)\ m_2 \geqslant \theta_2 \wedge m_1 < \theta_1 \quad \text{or} \quad (ii)\ m_1 = 0 \wedge m_2 \geqslant 1; \\ \quad \text{otherwise, it serves jobs from queue 1} \\ \text{A server of type 2 serves jobs from queue 1 if either} \\ (i)\ m_1 \geqslant \theta_1 \wedge m_2 < \theta_2 \quad \text{or} \quad (ii)\ m_2 = 0 \wedge m_1 \geqslant 1; \\ \quad \text{otherwise, it serves jobs from queue 2.} \end{cases}$$

Among these dynamic (or queue length dependent) rules $\mathbf{S}(\theta_1, \theta_2)$ a rule is sought which strictly improves the pooled scenario. An $\mathbf{S}$(Opt)-rule is determined that takes into account the waiting times of both job types by:

$$\textit{Step } 1: \min_{\theta_1,\theta_2} \max\{\mathbf{W}_1[S(\theta_1,\theta_2)], \mathbf{W}_2[S(\theta_1,\theta_2)]\}$$
This leads to an optimal threshold combination $(\theta_1,\theta_2)^*$
and overall average waiting time under $(\theta_1,\theta_2)^*$: $\boldsymbol{W}_A[S(\theta_1,\theta_2)^*]$. (2.1)

$$\textit{Step } 2: \boldsymbol{W}_A[S(\theta_1,\theta_2)^{**}] = \min_{\theta_1,\theta_2} \mathbf{W}_A[S(\theta_1,\theta_2)]$$
$$\text{s.t. } \max\{\boldsymbol{W}_1[S(\theta_1,\theta_2)], \boldsymbol{W}_2[S(\theta_1,\theta_2)]\} < \boldsymbol{W}_{\text{Pooled}}. \quad (2.2)$$

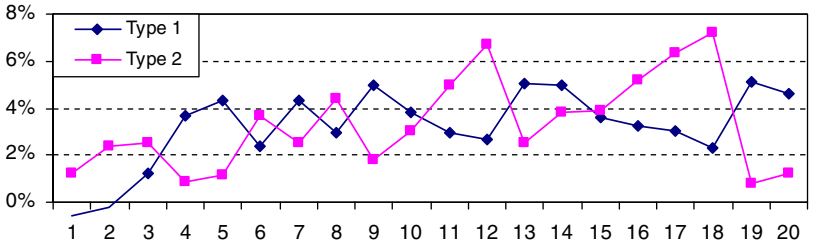


Fig. 6. Relative improvements over the pooled scenario.

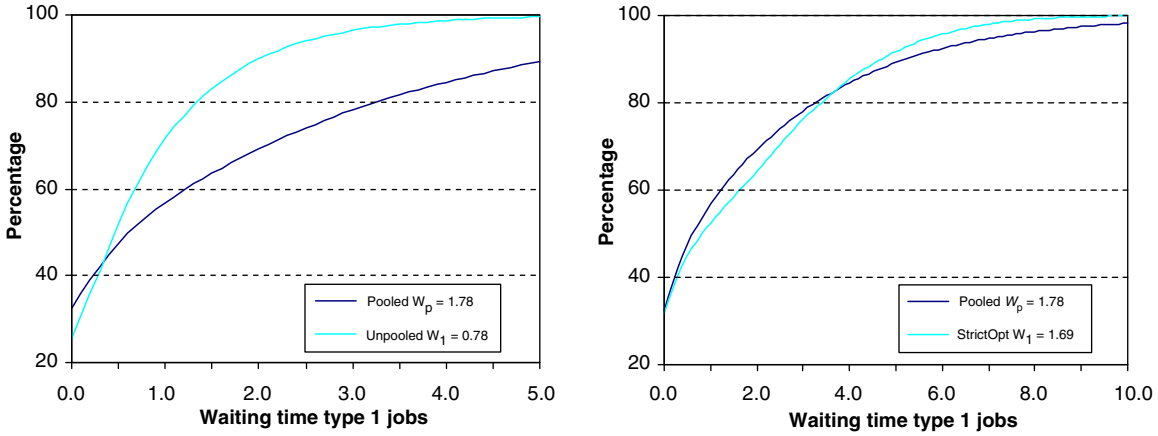


Fig. 7. Cumulative waiting time distributions for type 1 jobs. Pooled scenario and S(Opt) rule (right) pooled and unpooled scenario (left) ($s = 5$; $k = 10$; $\rho = 0.9$).

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

The improvements are only in the order of a few % but consistently outside 95% confidence intervals with a range of 1/2%. Fig. 6 shows the relative improvements (mean waiting time reduction) that can so be obtained for both types 1 and 2 jobs over the pooling scenario for $k = 10$ and $s = 1$ to 20.

Table 3 lists optimal threshold combinations $(\theta_1, \theta_2)^{**}$ (if existing), 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 exist. 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.

Some noteworthy observations from Table 3 are:

(1) *To optimize the pooled scenario for both types 1 and 2 jobs some more "preference" or "prioritization" appears to be required as indicated by the "optimal" levels* (θ_1, θ_2). *These optimal thresholds seem to prioritize type 2 (long) jobs.* ($\theta_2 = 1$ *or 2*).
(2) *A strict improvement over the pooled scenario for both job types does not only apply to large* ($k = 10$) *but also to small mix ratios* ($k = 2, 3$).
(3) *The optimal strategy appears to be rather robust for mix ratio k and number of agents s.*

Remark 3 (*Stochastic dominance*). One might question whether a strictly improving scenario $(\theta_1, \theta_2)^{**}$ also stochastically dominates (reduces) the pooled scenario instead of just by its means W_1 and W_2. However, there is no rationale for such a strong ordering. In fact, such stochastic ordering results might just as well not be valid for the other scenarios as in Sections 2 and 3 for which an improvement over the pooled scenario seems intuitively obvious. For example, even for a situation in which the unpooled scenario improves the pooled scenario on average, as purely due to the dominance of short (type 1) jobs, the type 1 performance in the unpooled case does not stochastically dominates that for the pooled case. This is illustrated in Fig. 7. (A similar observation can also be made by the waiting probability as reported in Whitt, 1999.) For the one-way scenario, a similar non-ordering result can be provided. And indeed also for a $(\theta_1, \theta_2)^{**}$ scenario that strictly dominates the pooled scenario in average waiting times, stochastic dominance does not generally applies, as shown in Fig. 7.

Nevertheless, similarly to the mean waiting time also for percentiles and service levels (as standardly used for call centers), strict improvements over the pooled scenario might be found. For example, for $s = 5$, $k = 10$, and $(\theta_1, \theta_2)^{**} = (7, 1)$ with w_1, and w_2 the corresponding waiting times for types 1 and 2 jobs and w_p the waiting for the pooled case, we have

$$P\{w_1 < 3.8\} = 83.74\% > P\{w_p < 3.8\} = 83.44\%,$$
$$P\{w_2 < 3.8\} = 83.77\% > P\{w_p < 3.8\} = 83.44\%.$$

In fact, for the different performance criteria and values, different values $(\theta_1, \theta_2)^{**}$ might be found to strictly improve the pooled scenario.

6. Summary

Let us summarize the observations by a schematic ordering of scenarios and rules and some global conclusions.

6.1. A schematic overview

So far, the results have been restricted to the illustrative case of $\rho = 0.9$ and $\boldsymbol{k} = 10$. But while $\rho = 0.7$, 0.8 or 0.9 seem quite realistic, $\boldsymbol{k} = 10$ might be considered as rather extreme. For small k, one may expect less distinctive if not opposite results. Table 4, therefore, provides a rough schematic overview for different agent numbers s and mix ratios $\boldsymbol{k}$. The inequality symbol < indicates a smaller average waiting time. Table 4 illustrates that pooling is 'optimal' only in the situation of identical services ($\boldsymbol{k} = 1$) and otherwise not. Here, the single threshold rule is only included as computed for $\boldsymbol{k} = 10$.

6.2. Global conclusions

For situations with mix ratio $\boldsymbol{k} > 1$, some global conclusions are

1. Pooling can be substantially improved by a number of simple and '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 servers,
   - A priority rule (pre-emptive or non-preemptive) for short (type 1) jobs for arbitrary number of servers and with a more or less constant improvement order.
2. From a practical point of view the NP1 scenario is 'nearly' optimal. It can only be improved slightly by more sophisticated threshold priority policies for short jobs.

Table 3
Optimal threshold combinations $(\theta_1, \theta_2)^{**}$ or $(\theta_1, \theta_2)^{*}$

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 4
Ordering of scenarios

s	1 2 3	4 5 .... 10 .... 15 .... 20 .... 30 .... 40 .... 50 ....
$k = 10$ $k = 9$	PP1 < $T(\theta_1)^*$ < NP1 < 2-way < 1-way < unpooled < **pooled**	PP1 < $T(\theta_1)^*$ < NP1 < 1-way < 2-way < unpooled < **pooled** ... ... PP1 < $T(\theta_1)^*$ = NP1 < 2-way < **pooled** < 1-way < unpooled
$k = 8$ $k = 7$	PP1 < NP1 < 2-way < 1-way < unpooled < **pooled**	PP1 < NP1 < 2-way < 1-way < **pooled** < unpooled PP1 < NP1 < 2-way < **pooled** < 1-way < unpooled
$k = 6$	PP1 < NP1 < 2-way < 1-way < **pooled** < unpooled	PP1 < NP1 < 2-way < **pooled** < 1-way < unpooled
$2 \le k \le 5$	PP1 < NP1 < 2-way < **pooled** < 1-way < unpooled	
$k = 1$	**pooled** = 2-way = NP1 ≤ PP1 << 1-way < unpooled	

3. Pooling can also be improved slightly but strictly for both short and long jobs by optimal threshold rules. These optimal scenarios generally tend to prioritize long (type 2) jobs.
4. Simulation is necessarily required to evaluate scenarios.
5. Particularly for more complex pooling questions, such as for multiple types, skills and preferences, a combination of queueing (for insights) and of simulation (for evaluation and optimization) is required.

Acknowledgement

The authors are grateful for the comments of an anonymous referee, which led to a substantial restructuring of the presentation, as well as to Remarks 2 and 3.

References

Bell, S.L., Williams, R.J., 2001. Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: Asymptotic optimality of a threshold policy. Annals of Applied Probability 11, 608–649.

Borst, S.C., Mandelbaum, A., Reiman, M.I., 2004. Dimensioning large call centers. Operations Research 52, 17–34.

Cattani, K., Schmidt, G.M., 2005. The pooling principle. INFORMS Transactions on Education 5. <http://ite.pubs.informs.org/Vol5No2/CattaniSchmidt/>.

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

Gans, N., Zhou, Y.-P., 2007. Call routing schemes for call-center outsourcing. Manufacturing & Service Operations Management 9, 33–50.

Osogami, T. Harchol-Balter, M., Scheller-Wolf, A., Zhang, L., 2004. Exploring threshold-based policies for load sharing. In: 42nd Annual Allerton Conference on Communication, Control, and Computing, University of Illinois, Urbana-Champaign, pp. 1012–1021.

Smith, D.R., Whitt, W., 1981. Resource sharing for efficiency in traffic systems bell system. Tech Journal 60, 39–55.

Squillante, M.S., Xia, C.H., Yao, D.D., Zhang, L., 2001. Threshold-based priority policies for parallel-server systems with affinity scheduling. Proceedings of the American Control Conference 4, 2992–2999.

Van Dijk, N.M., Van der Sluis, E., 2008. To pool or not to pool in call centers. Production and Operations Management 17, 1–10.

Wallace, R.B., Whitt, W., 2005. Resource pooling and staffing in call centers with skill-based routing. Manufacturing and Service Operations Management 7, 276–294.

Whitt, W., 1999. Partitioning customers into service groups. Management Science 45, 1579–1592.

Wolff, R.W., 1989. Stochastic Modelling and the Theory of Queues. Prentice-Hall, Englewood Cliffs, NJ.