EURANDOM PREPRINT SERIES
2013-011

May 8, 2013

Design and evaluation of overloaded service systems with skill based routing, under FCFS policies

Ivo Adan, Marko Boon, Gideon Weiss
ISSN 1389-2355

Design and evaluation of overloaded service systems with skill based routing, under FCFS policies

Ivo Adan* Marko Boon† Gideon Weiss‡

May 8, 2013

Abstract

We study an overloaded service system with servers of types $\mathcal{S} = \{s_1, \ldots, s_J\}$, serving customers of types $\mathcal{C} = \{c_1, \ldots, c_I\}$ under FCFS. Customers arrive in Poisson streams, join the queue and then abandon or get served. Service is skill based, which is described by a compatibility graph G, where $(i, j) \in G$ if server type s_j is trained to serve customer type c_i. The service duration depends on both server and customer type. This system is motivated by applications in areas as diverse as manufacturing, call centers, housing assignment, health care and data servers. At this level of generality, the design in terms of staffing and cross-training decisions is a challenging problem. Based on recent results in [1, 2] and on some asymptotic assumptions, we propose an algorithm to determine, for given data, the required levels of staffing to meet target levels of service quality and labor division. The algorithm is validated through a systematic simulation study, showing that it is remarkably robust and accurate. As such, we believe that the algorithm will prove to be useful in aiding the design and effective operation of complex systems with skill based routing.

Keywords: Service system; first come first served policy; multi type customers and servers; infinite bipartite matching; matching rates; overloaded queues; skill-based routing.

2000 Mathematics Subject Classification: Primary 60J10; Secondary 90B22; 68M20.

1 Introduction

In this paper we consider a service system with several types of customers and several types of servers, each with a specific set of skills. Each customer requires a single service from a single server, and service is skill based. Such systems are very common in all walks of life, and make up a large part of day to day interactions; for example, applications can be found in flexible manufacturing and assembly [20, 12, 13, 8], call centers [7, 6, 18, 4], public housing assignment [14, 15] and multimedia servers [3].

We assume that the system is *overloaded*, and that, as a result, servers are always busy and a fraction of the customers are forced to abandon. We consider the operation of this

*Department of Mechanical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands; email i.j.b.f.adan@tue.nl

†Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands; email marko@win.tue.nl

‡Department of Statistics, The University of Haifa, Mount Carmel 31905, Israel; email gweiss@stat.haifa.ac.il.
Research supported in part by Israel Science Foundation Grants 454/05 and 711/09.

system under FCFS, i.e., servers give priority to earlier, long waiting customers. FCFS is very commonly used, because it is fair to the customers, it is simple to implement, it requires little information about the parameters and the current state of the system, and it is robust under time varying conditions. Systems with skill-based routing have been studied under various routing rules, including FCFS [16], fixed-queue ratio [9] or static priority [18], but, as suggested by [18], to optimize performance, determining the right *level of cross-training* seems more important than the choice of routing rule.

We specify the system as follows: Customers are of types $\mathcal{C} = \{c_1, \ldots, c_I\}$, servers are of types $\mathcal{S} = \{s_1, \ldots, s_J\}$, and a bipartite graph G of compatible matches between $\mathcal{C}, \mathcal{S}$. Graph G specifies the level of cross-training: it has arc $(i, j) \in G$ if server type s_j has the skills to serve customer type c_i. In addition we assume the following data: Customers of type c_i arrive in independent Poisson streams of rate λ_i, and have patience distribution F_i which is absolutely continuous. There are n_j servers of type s_j, and the service times are *customer-server-type dependent*, i.e., the service of a customer of type c_i by a server of type s_j has a random duration distributed as G_{ij}, with rate μ_{ij} and average $m_{ij} = 1/\mu_{ij}$. Service is FCFS: whenever a server becomes available, he will go to the longest waiting compatible customer. In the event that a customer arrives to find several idle servers, he will be assigned to the longest idle compatible server. In the sequel, we adopt the notational convention to use subscript i for customers of type c_i and subscript j for servers of type s_j.

Important performance aspects of such a system, with given data λ_i, F_i, G, n_j, G_{ij}, under FCFS policy, include waiting times, abandonment rates, routing flows between customer types and server types, and workload of each server type. The performance can be measured from actual operation of the system or by simulation. However, the problem of obtaining *analytic closed form expressions* for these performance measures is quite intractable. To mention just one intractable problem: there are no expressions available to evaluate the workload of each type of server. We note that, in case of server-type dependent service times $G_{ij} = G_j$ and some special structured compatibility graphs G, the workload and routing flows can be determined for the approximating deterministic fluid model [16]. However, the other cases remain unsolved.

The design and effective operation of systems with skill based routing involve many decisions on structure, quality of service, and dimensioning, on a *multi-variate level*: In the design of the system, one can redefine some of the skills, increase or decrease the level of cross-training (i.e., add or subtract arcs from G), improve or degrade the service duration. On the operational side, one needs to find the right balance of staffing given by the n_j. One also needs to control the service quality level for each of the customer types, measured in waiting times and abandonment rates. Adequate methods to do this at the level of generality considered in this paper are currently not available.

The goal of this paper is to present a *structured method* to evaluate such systems, with the aim to support their design and efficient operation. We present an algorithm which, given the service requirements (summarized by λ_i, F_i, G_{ij}), proceeds in the following steps: We first decide on the service quality level for each customer type, we then decide on a balanced division of labor for the various server types, and finally we compute the required staffing n_j. Once this algorithm is available, it can support the design process: It can be used to re-specify service quality levels, re-dimension the system, re-adjust service durations, and re-design the compatibility graph until, ideally, the system design meets all requirements.

Our algorithm is based on recent results in [1, 2], and on some assumptions on the asymptotic behavior of the system under many server scaling. Our main assumption is that, under

many server scaling, the sequence of types are i.i.d. for customers that are served and for servers that start service, and that the two sequences are independent. In this paper we do not prove the validity of these assumptions, nor derive the conditions under which they might hold. Instead, we show experimentally, by simulation, that they provide good enough approximations for systems with a realistic number of servers.

The rest of the paper is structured as follows: In Section 2 we outline the theory behind our approach and in Section 3 we demonstrate the algorithm on two illustrative examples, with three types of customers and with three types of servers, each of which is skilled to serve two types of customers. In one design all the customers receive the same level of service, whereas in the other design, the customers are partitioned into two sets, with different levels of service. For a system with as few as eight servers of each type, we obtain excellent agreement between the desired performance, as calculated from our algorithm, and the actual performance as obtained from simulation. In Section 4 we perform a systematic exploration of the performance of our approach, via simulation. Motivated by the *principle of limited flexibility* [12, 8, 13, 18], the exploration in Section 4 is restricted to systems with servers having at most two skills. In Section 4.3 we experimentally investigate the validity of the assumptions on asymptotic behavior of the system under many server scaling. We conclude in Section 5 with a short discussion of our approach, and some promising directions for further research.

2 Theoretical results

In this section we outline the theory behind our approach in three parts: In Section 2.1 we describe the model of FCFS infinite matching of [1], yielding a formula to calculate the matching rates r_{ij}, i.e., the long-run fraction of customers of type c_i that are served by servers of type s_j. Then, in Section 2.2 we state our assumptions on many server scaling, which enable us to use the formula for r_{ij}. Finally, in Section 2.3 we describe our algorithm for dimensioning the system to achieve desired service quality levels and desired division of labor.

2.1 FCFS infinite matching model

In [1] the following FCFS infinite matching model is analyzed: There are two random sequences, $c^1, c^2, \ldots$ of customers and $s^1, s^2, \ldots$ of servers, where c^m are i.i.d. chosen from $\mathcal{C}$ with probabilities $\alpha_1, \ldots, \alpha_I$ and s^n are i.i.d. chosen from $\mathcal{S}$ with probabilities $\beta_1, \ldots, \beta_J$. The two sequences are matched on a FCFS basis, according to the compatibility graph G: the first server s^1 is matched to the first customer in the sequence compatible with him, and server s^n is matched to the first customer in the sequence which is compatible with him, and has not been matched to any of the previous servers $s^1, \ldots, s^{n-1}$.

The matching process can be described by a Markov chain (the precise formulation of which is not relevant here). Denote by $\mathcal{S}(C)$ the set of server types compatible with the subset of customer types C, by $\mathcal{C}(S)$ the set of customer types compatible with the subset of server types S, and by $\mathcal{U}(S)$ the set of customer types which are *unique* to S, i.e., these customer types can be served only by server types S. Let $\alpha_C = \sum_{c_i \in C} \alpha_i$, and $\beta_S = \sum_{s_j \in S} \beta_j$. The Markov chain is ergodic if and only if the following three equivalent conditions hold, referred to as

Complete Resource Pooling:

$$\begin{array}{lll} (i) & \alpha_C < \beta_{\mathcal{S}(C)}, & \text{for all non trivial subsets } C, \\ (ii) & \beta_S < \alpha_{\mathcal{C}(S)}, & \text{for all non trivial subsets } S, \\ (iii) & \alpha_{\mathcal{U}(S)} < \beta_S, & \text{for all non trivial subsets } S. \end{array} \tag{1}$$

Under complete resource pooling, the long-run fraction of matches of customers of type c_i to servers of type s_j exists almost surely. We call it *The Matching Rate* and denote it by r_{ij}. An explicit formula for the matching rates is derived in [1]. The formula is quite complex, and summarized below.

Define for a given pair c_i, s_j, and a given permutation $S_1, S_2, \ldots, S_J$ of the server types:

$$\alpha_{(k)} = \alpha_{\mathcal{U}\{S_1,\ldots,S_k\}}, \qquad \beta_{(k)} = \beta_{\{S_1,\ldots,S_k\}}$$

and

$$\phi_k = \frac{\alpha_{\mathcal{U}\{S_1,\ldots,S_k\}\cap\{c_i\}}}{\alpha_{\mathcal{U}\{S_1,\ldots,S_k\}}}, \qquad \psi_k = \frac{\alpha_{\mathcal{U}\{S_1,\ldots,S_k\}\cap(\mathcal{C}(s_j)\setminus\{c_i\})}}{\alpha_{\mathcal{U}\{S_1,\ldots,S_k\}}}, \qquad \chi_k = 1-\phi_k-\psi_k\ .$$

Then the matching rate r_{ij} is:

$$\begin{aligned} r_{ij} \quad = \quad & \beta_j \sum_{\mathcal{P}_J} B \prod_{k=1}^{J-1} (\beta_{(k)} - \alpha_{(k)})^{-1} \\ & \left(\sum_{k=1}^{J-1} \phi_k \frac{\alpha_{(k)}}{\beta_{(k)} - \alpha_{(k)}\chi_k} \prod_{l=1}^{k-1} \frac{\beta_{(l)} - \alpha_{(l)}}{\beta_{(l)} - \alpha_{(l)}\chi_l} + \frac{\phi_J}{\phi_J + \psi_J} \prod_{l=1}^{J-1} \frac{\beta_{(l)} - \alpha_{(l)}}{\beta_{(l)} - \alpha_{(l)}\chi_l} \right), \end{aligned} \tag{2}$$

where the summation is over $\mathcal{P}_J$, the set all the permutations of $\mathcal{S}$. The normalizing constant is:

$$B^{-1} = \sum_{\mathcal{P}_J} \frac{1}{(\beta_{\{S_1\}} - \alpha_{\mathcal{U}\{S_1\}})(\beta_{\{S_1,S_2\}} - \alpha_{\mathcal{U}\{S_1,S_2\}}) \cdots (\beta_{\{S_1,\ldots,S_{J-1}\}} - \alpha_{\mathcal{U}\{S_1,\ldots,S_{J-1}\}})}.$$

When there is no complete resource pooling (1), the system decomposes in a unique way to L subsystems consisting of $\mathcal{C}^{(1)}, \mathcal{S}^{(1)}, \ldots, \mathcal{C}^{(L)}, \mathcal{S}^{(L)}$. This is discussed in [2], where the subsystems are recursively determined as follows:

$$\mathcal{C}^{(l)} = \arg\min_{C \subseteq \mathcal{C}\setminus\bigcup_{k<l}\mathcal{C}^{(k)}} \frac{\beta_{\mathcal{S}(C)}}{\alpha_C}, \qquad \mathcal{S}^{(l)} = \mathcal{S}(\mathcal{C}^{(l)})\setminus\bigcup_{k<l}\mathcal{S}^{(k)}, \qquad l = 1,\ldots,L. \tag{3}$$

Each of the sub-systems $\mathcal{C}^{(l)}$, $\mathcal{S}^{(l)}$, with its sub-graph of G, has complete resource pooling, and matching rates calculated from (2) with α_i and β_j replaced by

$$\alpha_i^{(l)} = \frac{\alpha_i}{\sum_{c_k\in\mathcal{C}^{(l)}}\alpha_k}, \qquad \beta_j^{(l)} = \frac{\beta_j}{\sum_{s_k\in\mathcal{S}^{(l)}}\beta_k}. \tag{4}$$

2.2 Many server scaling

We now return to the system as described in the introduction, and scale the system as follows: We use a *common scaling factor*, and we let λ_i and n_j increase by that scaling factor, leaving all other quantities (i.e., graph G and distributions F_i, G_{ij}) unchanged. Under this scaling, we make the following assumptions on the performance of the system. We believe that these

assumptions hold asymptotically under appropriate conditions, and that they provide good approximations for actual systems of moderate size. We do not attempt to prove the asymptotic validity, but below we will provide some intuitive plausibility arguments to support our assumptions on the asymptotic behavior and in the next section we will experimentally investigate the quality of the approximation.

(i) There will be a *decomposition of the customer and server types* into subsets of types $\mathcal{C}^{(l)}$, $\mathcal{S}^{(l)}$, $l = 1, \ldots, L$, where customers of types in $\mathcal{C}^{(l)}$ will be served exclusively by servers of types in $\mathcal{S}^{(l)}$. The special case $L = 1$ is the complete resource pooling case, with equal service quality levels for all customers.

(ii) There will be *critical patience times* $w^{(1)} > \cdots > w^{(L)}$, such that customers of type $c_i \in \mathcal{C}^{(l)}$ with patience less than $w^{(l)}$ will abandon, and those with patience exceeding $w^{(l)}$ will be served after a wait of exactly $w^{(l)}$. Let:

$$\begin{aligned} w_i &= w^{(l)}, \quad c_i \in \mathcal{C}^{(l)}, \\ \widehat{\lambda} &= \sum_{c_i \in \mathcal{C}} \lambda_i (1 - F_i(w_i)), \\ \alpha_i &= \lambda_i (1 - F_i(w_i)) / \widehat{\lambda}. \end{aligned} \tag{5}$$

Then $\widehat{\lambda}$ will be the rate at which the system is serving, and α_i the fraction of services that are given to type c_i customers. The sequence, in order of arrivals, of customers that will be served $\ldots, c^m, \ldots$ will be of types drawn i.i.d. with probabilities α_i.

(iii) The instants of service completions (which are also service starts, since servers will always be busy), by the servers of the different types, will form *independent Poisson processes*. There will be probability values β_j such that the sequence of successive servers completing service $\ldots, s^n, \ldots$ will be of types drawn i.i.d. with probabilities β_j, and this sequence is independent of the sequence in (ii) of types of customers that are served.

(iv) There will be complete resource pooling as in (1) in each of the sub-systems $\mathcal{C}^{(l)}$, $\mathcal{S}^{(l)}$, and the matching rates between customers and servers will be given by (2), with α_i and β_j replaced by $\alpha_i^{(l)}$ and $\beta_j^{(l)}$, and the following relation will hold:

$$n_j = \widehat{\lambda}^{(l)} \sum_{c_i \in \mathcal{C}^{(l)}} r_{ij} m_{ij}, \tag{6}$$

where

$$\widehat{\lambda}^{(l)} = \sum_{c_i \in \mathcal{C}^{(l)}} \lambda_i (1 - F_i(w_i)). \tag{7}$$

(v) In the decomposition with $w^{(1)} > \cdots > w^{(L)}$, the sets of servers $\mathcal{S}^{(l)}$ will satisfy the right-hand part of (3) for all $l = 1, \ldots, L$.

To justify (ii), we note first that it holds for single-type systems, and for single-server multi-type customer systems [19, 11]. Intuitively, in an overloaded system with abandonments, when arrival rates and service rates are scaled up and patience is unchanged, then variability of the waiting times decreases. As a result, in the limit all customers of the same type will wait the same time before reaching service. This justifies the existence of w_i. Since

patience of successive customers are independent, removing the customers which abandon (i.e. those with patience $< w_i$) will leave the types of the customers in the remaining sequence independent and drawn with probabilities α_i.

To justify (iii), each server of type j will work all the time, serving a stationary sequence of customers which are compatible with him, so that his service start times will in the limit form a stationary point process. Because there are so many other servers, this point process will be largely independent of the other servers. The total sequence of instants of service completions for servers of type j will then be a superposition of many almost stationary, almost independent point processes, which should converge to an independent homogeneous Poisson process.

Once we accept that customers which are served arrive in a sequence of i.i.d. types drawn from probabilities α_i, servers arrive in a sequence of i.i.d. types drawn from probabilities β_j, and that the two sequences are independent, the decomposition and matching rates follow. Complete resource pooling will occur if there is a unique value of w which allows for complete resource pooling with the resulting β_j and n_j. If that is not the case, there will be a subset $\mathcal{C}^{(1)}$ for which the number of servers of types in $\mathcal{S}(C^{(1)})$ is not sufficient to provide service for all customers in $\mathcal{C}^{(1)}$ with patience w. This will define the first sub-system, $\mathcal{C}^{(1)}$, $\mathcal{S}^{(1)}$, which will have the least preferential service (and thus greatest critical patience time). Continuing with the remaining customer and server types, the unique decomposition, will emerge as in [2]. The values of the matching rates, in each of the sub-systems of the decomposition, are then given by (2). Expression (6) for n_j simply equates the amount of work done per time unit (assuming that servers are always busy) and the expected amount of work offered per time unit by the fraction r_{ij} of the customers that are served over all $c_i \in \mathcal{C}(s_j)$.

2.3 Algorithm for design and evaluation

In the previous section we argued that, as arrival rates and staffing increase by a common scaling factor, the system naturally decomposes into subsystems $\mathcal{C}^{(l)}$, $\mathcal{S}^{(l)}$ with increasing service quality levels for its customers. In this section we propose an algorithm that takes this decomposition as starting point, and we will act as if arrival rates and staffing levels are sufficiently large such that the assumptions (i)-(v) are reasonable. For given input data, the algorithm proceeds by first deciding on quality of service for each customer type and division of labor for each server type, and then computes the required staffing n_j to meet these specifications.

Input: The compatibility graph G, the arrival rates λ_i, the patience distributions F_i, and the service-time distributions G_{ij}, of which we only need the means m_{ij}.

Quality of service decision: We specify the partition of the customer types into $\mathcal{C}^{(l)}$, $l = 1, \ldots, L$, where higher l implies more preferential service, and construct the corresponding decomposition of server types $\mathcal{S}^{(l)}$, using the right-hand part of (3). For each class of preference, we specify the cut-off waiting time $w^{(l)}$, decreasing in l.

Division of labor decision: For each subsystem $\mathcal{C}^{(l)}$, $\mathcal{S}^{(l)}$ we specify the fraction of services performed by each type of server, $\beta_j^{(l)}$.

Calculations: Compute, for each subsystem, $\widehat{\lambda}^{(l)}$ and $\alpha_i^{(l)}$, using (4), (5) and (7). Check that $\alpha^{(l)}$ and $\beta^{(l)}$ provide complete resource pooling. Calculate the matching rates r_{ij} using (2),

with α_i and β_j replaced by $\alpha_i^{(l)}$ and $\beta_j^{(l)}$. Compute the staffing n_j using (6) for all server types $s_j \in \mathcal{S}^{(l)}$.

Output: Number of servers of each type, n_j, the matching rates r_{ij}, the cut-off waiting times w_i, the abandonment rates $F_i(w_i)$.

Note that, if in the calculation step, complete resource pooling does not hold, then the division of labor, specified by $\beta_j^{(l)}$ should be adapted such that it does (which is always possible). In the following sections we will investigate, by simulation experiments, whether the desired performance is indeed achieved by the staffing levels proposed by our algorithm.

3 First glance on the algorithm

In this section we first illustrate the algorithm for a simple network with three customer types and three server types. We refer to this network as an "almost complete" network because all possible arcs (i, j) are included, except for the ones with $i = j$; see Figure 1. We first

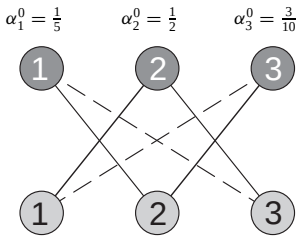

Figure 1: The "almost complete" network. Nodes on the top row represent customer types, and the ones on the bottom row server types. The dashed arcs are present in the case of complete resource pooling, but vanish in the case of incomplete resource pooling.

specify the input data. Let α_i^0 denote the portion of all arrivals which is of type c_i. In this example we set $\alpha^0 = (0.2, 0.5, 0.3)$. For all customers, patience is exponential with mean 10, so $F_i(t) = 1 - e^{-0.1t}$. The service-time distributions $G_{i,j}$ are as follows:

Service-time distributions			
$G_{i,j}$	s_1	s_2	s_3
c_1		pareto(3, 3)	pareto(2, 3)
c_2	exp(0.2)		exp(0.125)
c_3	uniform(2, 6)	uniform(1, 5)	

Mean service times			
$m_{i,j}$	s_1	s_2	s_3
c_1		4.5	3
c_2	5		8
c_3	4	3	

In the following sections we will investigate two cases: Equal service quality for all customer types and service quality differentiation among customer types.

3.1 Complete resource pooling

In this case we decide to provide equal service quality to all customers, corresponding to complete resource pooling. The target waiting times are $w_i = 1.0$, yielding approximate abandonment rates $\theta_i = F_i(w_i) = 1 - e^{-0.1} \approx 0.095$ for all customer types. Hence $\alpha = \alpha^0 =$

(0.2, 0.5, 0.3). The next step is to divide labor: $\beta = (0.3, 0.4, 0.3)$. From (1) it is readily verified that these values of α and β provide complete resource pooling. For this network type, the matching rates have also been derived in [5]:

$$r_{i,j} = \frac{\alpha_i \beta_j [(1-\alpha_i)(1-\beta_j) - \alpha_j \beta_i]}{(1-\alpha_i-\beta_i)(1-\alpha_j-\beta_j)} \left[1 + \sum_{i=1}^{3} \frac{\alpha_i \, \beta_i}{1-\alpha_i-\beta_i} \right]^{-1},$$

yielding

$$[r_{ij}] = \begin{bmatrix} 0 & \frac{176}{1115} & \frac{47}{1115} \\ \frac{54}{223} & 0 & \frac{115}{446} \\ \frac{129}{2230} & \frac{54}{223} & 0 \end{bmatrix}.$$

So

$$[r_{ij} m_{ij}] = \begin{bmatrix} 0 & \frac{792}{1115} & \frac{141}{1115} \\ \frac{270}{223} & 0 & \frac{460}{223} \\ \frac{258}{1115} & \frac{162}{223} & 0 \end{bmatrix}$$

and thus, from (6), we get the staffing levels:

$$n(\lambda) = \lambda e^{-0.1} \left(\frac{1608}{1115}, \frac{1602}{1115}, \frac{2441}{1115} \right) \approx \lambda (1.305, 1.300, 1.981),$$

where λ is the total arrival rate (and scaling parameter). The actual number of servers of each type is of course integer, and obtained by rounding to the nearest one. This concludes the setup of the experiment. In the next paragraph we present the simulation results and investigate whether the service quality level in terms of *expected waiting time*, abandonment rates and matching rates are indeed as predicted by the algorithm.

Simulation results: For $\lambda = 10, 30, 50, 100$, we have performed 100 simulation runs, each of 100,000 matches, from which the first 25,000 have been discarded as warm-up interval; this setup has been used for *all simulation experiments in this paper*. The simulated mean waiting times $E[W_i]$, abandonment rates θ_i and matching rates r_{ij} are summarized in Table 1. We note that the half-width of the 95% confidence intervals for *all simulation results in this paper* are less than 2%.

The simulation results show that, by scaling up λ, the performance of the staffing levels proposed by the algorithm gets closer to its theoretical target. This suggests that the asymptotic assumptions are correct. Note that the performance, and especially the matching rates are already rather close to their target values for moderate systems of around 50 servers. This is remarkable, since the waiting time density, although narrowing, is not yet concentrated at the asymptotic value $w_i = 1.0$ (even for $\lambda = 100$). This indicates that the algorithm is fairly robust, in the sense that, although the asymptotic assumptions do not quite hold yet, the proposed staffing closely meets the desired performance.

3.2 Incomplete resource pooling

Now we decide to give preferential treatment to customers of type c_1 and c_3. Type c_2 customers will have to wait longer for service, and so more will abandon. We use service levels $w_2 = 2$ and $w_1 = w_3 = \frac{1}{2}$, corresponding to abandonment rates $\theta \approx (0.05, 0.18, 0.05)$ and, by (5), we get $\alpha = (0.215, 0.463, 0.322)$. Our decision of preferential treatment results in the

Simulated mean waiting times and abandonment rates							
λ	n	$E[W_1]$	$E[W_2]$	$E[W_3]$	θ_1	θ_2	θ_3
10	(13, 13, 20)	0.870	1.226	0.884	0.086	0.119	0.088
30	(39, 39, 59)	0.924	1.110	0.928	0.089	0.105	0.089
50	(65, 65, 99)	0.929	1.053	0.932	0.089	0.100	0.089
100	(130, 130, 198)	0.956	1.025	0.958	0.090	0.096	0.090
Target		1.000	1.000	1.000	0.095	0.095	0.095

Matching rates									
	Simulated $\lambda = 10$			Simulated $\lambda = 100$			Target		
$r_{i,j}$	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3
c_1		0.153	0.050		0.158	0.043		0.158	0.042
c_2	0.234		0.257	0.240		0.258	0.242		0.258
c_3	0.067	0.238		0.060	0.242		0.058	0.242	

Simulated densities of the waiting times

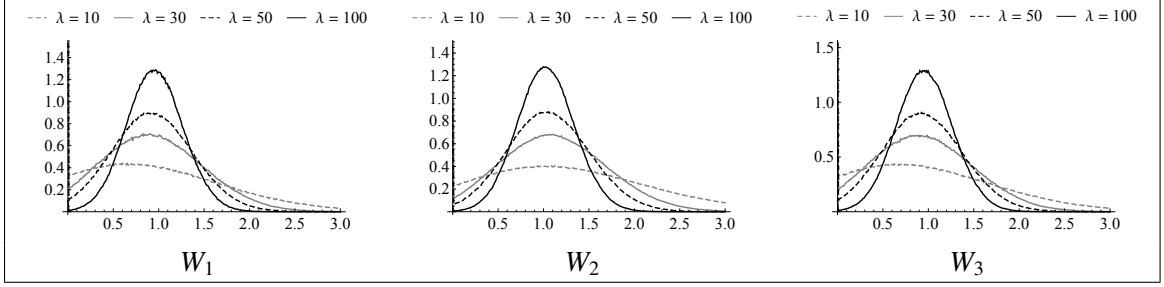

Table 1: Simulation results for the case of complete resource pooling.

partitioning $\mathcal{C}^{(1)} = \{c_2\}$, $\mathcal{S}^{(1)} = \{s_1, s_3\}$ and $\mathcal{C}^{(2)} = \{c_1, c_3\}$, $\mathcal{S}^{(2)} = \{s_2\}$. For each subsystem, we choose appropriate values $\beta_j^{(l)}$ resulting in complete resource pooling: $\beta^{(1)} = (0.5, 0.5)$ and $\beta^{(2)} = (1)$. Note that $\beta_1 = \beta_3 = 0.5 \times \alpha_2 = 0.232$ and $\beta_2 = \alpha_1 + \alpha_3 = 0.537$.

A graphical representation of the original system (with incomplete resource pooling) and the two subsystems (each with complete resource pooling) is depicted in Figure 2. We now calculate the staffing for both subsystems. The matching rates for the first subsystem are $r_{2,1} = r_{2,3} = 0.5$ and for the second one, $r_{1,2} = 0.4$, $r_{3,2} = 0.6$. This leads to the following staffing for the first subsystem:

$$n_1(\lambda) = \lambda\alpha_2^0(1 - F_2(w_2))r_{2,1}m_{2,1} = 1.023\lambda, \qquad n_3(\lambda) = \lambda\alpha_2^0(1 - F_2(w_2))r_{2,3}m_{2,3} = 1.637\lambda,$$

and for the second subsystem:

$$n_2(\lambda) = \lambda\big(\alpha_1^0(1 - F_1(w_1)) + \alpha_3^0(1 - F_3(w_3))\big)\big(r_{1,2}m_{1,2} + r_{3,2}m_{3,2}\big) = 1.712\lambda.$$

Simulation results: The simulated mean waiting times $E[W_i]$, abandonment rates θ_i and matching rates r_{ij} are summarized in Table 2. The results confirm the ones in the case of

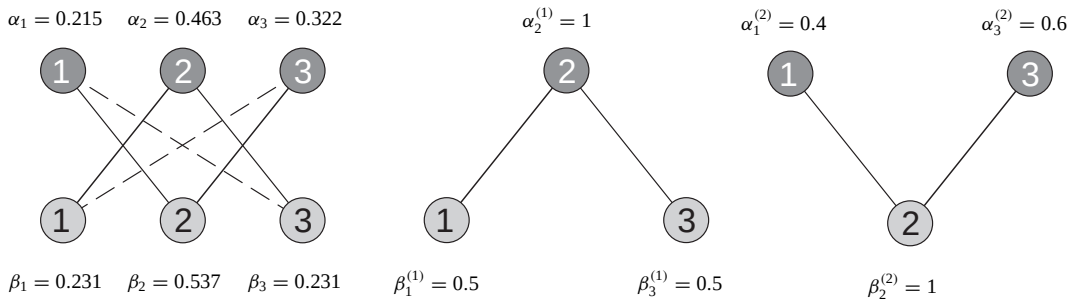

Figure 2: The two subsystems in case of incomplete resource pooling. The nodes on the top row represent the customer types, and the nodes on the bottom row represent the server types. The dashed arcs vanish (i.e., are not used) in the decomposed system.

complete resource pooling: Already for moderate values of λ, the predicted performance of the staffing levels gets fairly close to its theoretical target. Note that, for $\lambda = 10$, the decomposition is not fully realized yet, as 2% of the matches are on links between the two subsystems, but for $\lambda \geq 50$ such matches are less than 0.1% (although not shown in Table 2).

4 Systematic exploration

In this section we will explore, more systematically, the performance of the approximations generated by our algorithm.

4.1 Exponential systems

In this section we study a system with five customer types and five server types. Motivated by the principle of limited flexibility [12, 8, 13, 18], we only consider servers having two skills. We assume that a server of type s_j is *specialized* in handling customers of type c_j, and can also serve customers of type c_{j+1}, but at a *slower speed*; see Figure 3. Arrivals are distributed

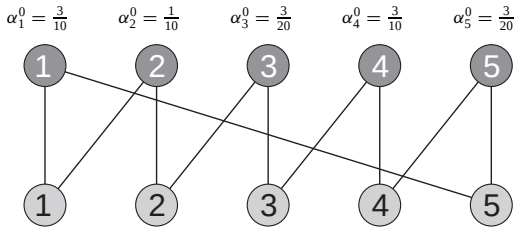

Figure 3: The network of Section 4.

over the five types as follows: $\alpha^0 = (0.3, 0.1, 0.15, 0.3, 0.15)$. For all customer types, patience is exponential with mean 5. The service times are also exponential (but this assumption will

Simulated mean waiting times and fractions of abandonment							
λ	n	$E[W_1]$	$E[W_2]$	$E[W_3]$	θ_1	θ_2	θ_3
10	(10, 17, 16)	0.685	2.404	0.687	0.068	0.221	0.069
30	(31, 51, 49)	0.596	2.011	0.596	0.059	0.184	0.059
50	(51, 86, 82)	0.504	2.017	0.505	0.049	0.183	0.050
100	(102, 171, 164)	0.525	2.010	0.526	0.051	0.180	0.051
Theoretical		0.500	2.000	0.500	0.050	0.180	0.050

Matching rates									
	Simulated $\lambda = 10$			Simulated $\lambda = 100$			Theoretical		
$r_{i,j}$	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3
c_1	0.000	0.209	0.008	0.000	0.215	0.000	0.000	0.215	0.000
c_2	0.225	0.000	0.231	0.231	0.000	0.233	0.231	0.000	0.231
c_3	0.011	0.316	0.000	0.001	0.321	0.000	0.000	0.322	0.000

Simulated densities of the waiting times

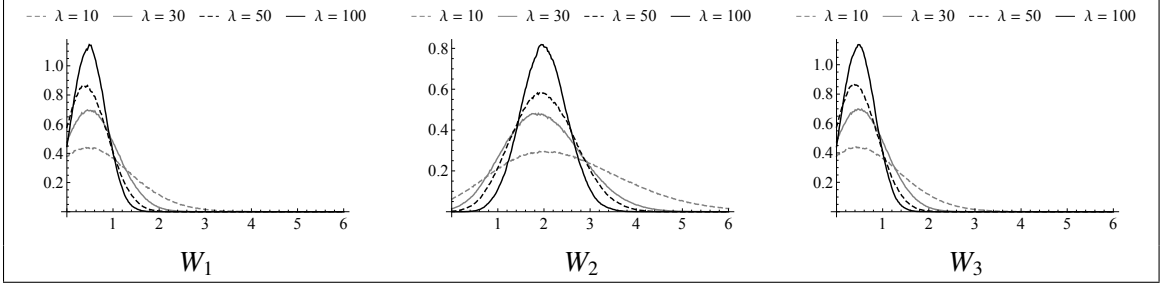

Table 2: Simulation results for the case of incomplete resource pooling.

be relaxed in Section 4.2). The mean service times m_{ij} are calculated as follows:

$$m_{i,j} = \frac{m_i}{v_j} p_{i,j}, \qquad i, j = 1, \ldots, 5,$$

with m_i the mean amount of work of a type c_i customer, v_j the service speed of a type s_j server and p_{ij} the *slow-down factor* of service speed, where

$$m = (2, 3, 4, 3, 4), \qquad v = (1, 0.8, 1.1, 0.9, 1), \qquad p_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 1.25 & \text{if } i = j \,(\bmod J) + 1, \\ 0 & \text{otherwise.} \end{cases}$$

This results in the following values for m_{ij}:

i	1	2	3	4	5
$m_{i,i}$	2.00	3.75	3.64	3.33	4.00
$m_{i+1,i}$	3.75	6.25	3.41	5.56	2.50

Below we present three cases: complete resource pooling and incomplete resource pooling with the system decomposing into two and three subsystems, respectively. The simulation

setup is as before: 100 runs, each of 100, 000 matches with a warm-up of 25,000. The resulting half-width of the 95% confidence intervals are less than 1% for all simulations results.

Complete resource pooling: We set $w_i = 1$, yielding an abandonment rate of 0.181. We choose $\beta = (0.2, 0.2, 0.3, 0.15, 0.15)$, which together with $\alpha = \alpha^0$, guarantees complete resource pooling according to (1). For the staffing we obtain $n(\lambda) = \lambda(0.339, 0.835, 0.845, 0.606, 0.359)$. We simulate this system for $\lambda = 20, 40, 60, 100, 200$. The results are listed in Table 3. Clearly the system performance gets closer to its target values as λ increases, but this seems to happen at a slower pace compared to the (smaller) system in the previous section. This may be due to the greater variation among m_{ij}, as shown in the table above. Note that for all λ, even for $\lambda = 10$, the matching rates are quite close to their target values.

Two-subsystem decomposition: We now impose two service levels. The target waiting times are $w_1 = w_2 = 1$ (low service quality), and $w_3 = w_4 = w_5 = 0.5$ (high service quality). This leads to abandonment rates of 0.181 for customer types c_1 and c_2, and 0.095 for customer types c_3, c_4 and c_5, and thus $\alpha = (0.282, 0.094, 0.156, 0.312, 0.156)$. The system decomposes into two subsystems: $\mathcal{C}^{(1)} = \{c_1, c_2\}$, $\mathcal{S}^{(1)} = \{s_1, s_2, s_5\}$ and $\mathcal{C}^{(2)} = \{c_3, c_4, c_5\}$, $\mathcal{S}^{(2)} = \{s_3, s_4\}$. We have $\alpha^{(1)} = (0.75, 0.25)$ and choose $\beta^{(1)} = (0.6, 0.1, 0.3)$, ensuring complete resource pooling for the first subsystem. For the second one, we have $\alpha^{(2)} = (0.25, 0.5, 0.25)$ and set $\beta^{(2)} = (0.5, 0.5)$. Note that, for the *whole network*, work is divided according to $\beta = (0.226, 0.038, 0.312, 0.312, 0.113)$, for which, by (1), there is no complete resource pooling. For the staffing we obtain $n(\lambda) = \lambda(0.479, 0.123, 0.956, 1.206, 0.246)$. The simulation results are reported in Table 4, showing similar performance as in the complete resource pooling case. Note that, for "high service" customers the mean waiting times and abandonment rates are much closer to their target values than for "low service" customers.

Three-subsystem decomposition: Now we select *three* service levels, $w = (2, 1, 1, 0.5, 0.5)$. The corresponding abandonment rates are $\theta = (0.330, 0.181, 0.181, 0.095, 0.095)$. Accordingly, the system decomposes into three subsystems, namely $\mathcal{C}^{(1)} = \{c_1\}$, $\mathcal{S}^{(1)} = \{s_1, s_5\}$, $\mathcal{C}^{(2)} = \{c_2, c_3\}$, $\mathcal{S}^{(2)} = \{s_2, s_3\}$, and $\mathcal{C}^{(3)} = \{c_4, c_5\}$, $\mathcal{S}^{(3)} = \{s_4\}$. For the first subsystem, we have $\alpha^{(1)} = (1)$ and set $\beta^{(1)} = (0.7, 0.3)$, for the second subsystem, $\alpha^{(2)} = (0.4, 0.6)$ and $\beta^{(2)} = (0.5, 0.5)$, and finally, $\alpha^{(3)} = (0.5, 0.5)$ and $\beta^{(3)} = (1)$. It is readily verified that there is complete resource pooling in each subsystem, but *no* complete resource pooling in the whole network with $\beta = (0.173, 0.126, 0.126, 0.501, 0.074)$. By (5), we get the staffing levels $n(\lambda) = \lambda(0.282, 0.435, 0.372, 1.659, 0.151)$. The simulation results can be found in Table 5. The conclusions are similar the ones for the other two cases: There is a clear distinction between the mean waiting times of customer types, which gets sharper as the arrival rate increases.

4.2 Non-exponential systems

To investigate sensitivity with respect to service-time distributions, we consider the system with complete resource pooling of Section 4.1, but now with non-exponential service-time distributions. We use the following four distributions, matched to the mean service times

$m_{i,j}$:

$$\exp(1/m_{i,j}), \qquad \text{Gamma}(2, m_{i,j}/2), \qquad \text{Uniform}(0, 2m_{i,j}), \qquad \text{Pareto}(\frac{2}{3}m_{i,j}, 3).$$

In all simulations we have chosen $\lambda = 60$. From the previous section, we know that our algorithm prescribes the staffing $n = (20, 50, 51, 36, 22)$. The results in Table 6 suggest that the service-time distribution has hardly any effect on the system performance. This implies that the performance of the staffing proposed by our algorithm does not require a priori knowledge of the service-time *distributions* – having good estimates of the means is sufficient.

4.3 Validation of asymptotic assumptions

The aim of this section is to experimentally investigate the assumptions on the asymptotic behavior of the system, under many server scaling. First, we investigate the Poisson assumption by considering the sequence of service starts and testing whether the inter-occurrence times are independent and exponentially distributed. Second, we validate whether the sequences of customer types and server types are i.i.d. sequences. We perform a simulation of $35,000$ matches (the first $10,000$ of which are considered as warm-up) of the model with complete resource pooling, with arrival rate $\lambda = 100$ and uniform service times, described in Section 4.2.

Poisson assumption: Assumption (iii) in Section 2.2 states that the sequences of service starts for each of the server types s_j form independent Poisson processes. To verify the Poisson assumption, we first consider the sequence of service starts of *all* server types together. In Figure 4(a) we plot the cumulative number of service starts versus the time. From this plot we can conclude that the process of service starts is homogeneous with constant rate 82.1, being close to $\lambda(1 - F_i(w_i)) = 100(1 - 0.181) = 81.9$. Figure 4(b) shows the inter-occurrence times of the last $1,000$ service starts. The abundance of observations makes it difficult to draw conclusions, but the plot suggests that there is no autocorrelation. This is confirmed by Figure 4(c), showing the estimated autocorrelation function for lags 1 - 30, which lies between the critical values (using a significance level of 0.05). Common goodness-of-fit tests clearly indicate that the null hypothesis, stating that the inter-occurrence times of service starts are exponentially distributed, cannot be rejected (P-values greater than 0.2). This supports the Poisson assumption. Additional tests indicate that the process of service starts *for each of the server types* s_j is also Poisson.

Server and customer sequences: We now test whether the sequences of types of servers that start service, and customers that receive service, are i.i.d. To do so, we first consider the simulated sequence $s^1, s^2, \ldots, s^N$ of server types corresponding to these matches. We use two methods to test the i.i.d. assumption. First, we estimate the autocorrelation coefficients of this sequence at lags $1, 2, 3, \ldots$ and test whether they are significantly different from zero. The estimated autocorrelation coefficients for the first five lags are $-0.001, 0.006, -0.007, 0.005$, and 0.000, all between the common thresholds of $\pm z_{1-\alpha/2}/\sqrt{N} \approx \pm 0.012$, supporting the hypothesis of independent server types for $\alpha = 0.05$. Conducting the same test to the sequence $c^1, c^2, \ldots, c^N$ of types of customers that will be served, in order of arrivals, yields autocorrelation coefficients -0.010, -0.010, -0.005, -0.006, and -0.008, also not significantly different from zero.

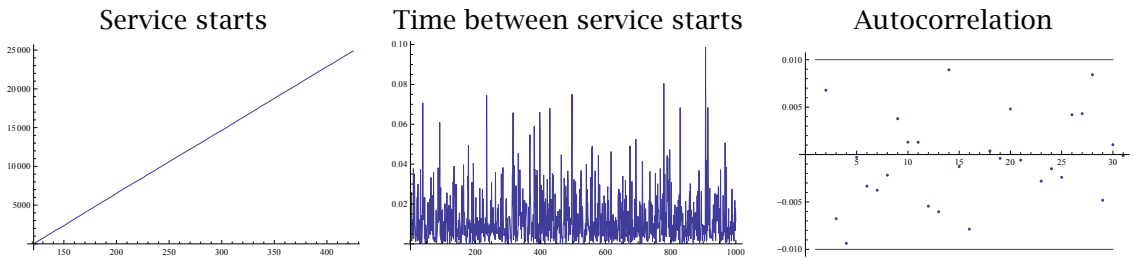

Figure 4: Simulated service starts. Figure (a) plots the cumulative number of service starts versus the time. Figure (b) shows the inter-occurrence times of 1,000 successive service starts. Figure (c) shows the estimated autocorrelation function for lags 1 - 30.

The second method considers all successive *pairs* in the sequence of server types, i.e., (s^1, s^2), $(s^2, s^3), \ldots, (s^{N-1}, s^N)$ and compares the number of occurrences of each pair to its expected value. If there is no dependence between the server types of two successive service starts, the fraction of (s_j, s_k) pairs in this list should be the product of the product of the fraction of $(s_j, \cdot)$ pairs and the fraction of $(\cdot, s_k)$ pairs. Tables 7(a) and 7(b) show the fraction of occurrences of each pair, and the expected values under the null hypothesis. The tables indicate that the simulated fractions are close to the expected fractions, implying that the sequence is indeed i.i.d. Pearson's χ^2 test strongly confirms this assumption. We applied the same method to all pairs in the sequence of customer types, and the results are shown in Tables 7(c) and 7(d). Also in this case the statistical tests do not reject the i.i.d. hypothesis.

Finally, we have tested the correlation between the sequence of server types and the sequence of customer types. The estimated correlation coefficient is 0.008, which lies between the bounds ± 0.012. The estimated cross-correlation function also shows no significance for the lags $-5, -4, \ldots, 4, 5$.

To summarize, all tests clearly indicate that the i.i.d. assumption is not violated.

Remark 1. *For illustration purposes, we used a single simulation run with uniform service times to motivate the conclusions in this section. However, the conclusions are based on multiple simulation runs, for* all *service-time distributions considered in Section 4.2, and multiple statistical tests for goodness of fit and correlation.*

5 Concluding remarks

In this paper we developed an algorithm for the staffing of many-server systems with skill-based routing in *overload*, i.e., in the *efficiency driven* regime. The algorithm is based on recent exact closed-form results for the infinite-matching model, and on some, intuitively appealing asymptotic assumptions. These assumptions have been experimentally validated, but they still lack a rigorous justification. The algorithm has been explored through simulation, showing that it is remarkably robust and accurate. Hence, we believe that it provides a useful tool to support the design and effective operation of complex systems with skill based routing.

A hopeful direction for further research is to consider other regimes of interest: The quality driven (QD) regime (in which there is *no overload*), and the critical, quality and efficiency driven (QED) regime (in which the load is exactly one). Preliminary results indicate that the

techniques used in this paper may also work for the QD and QED regimes.

References

[1] Adan, I.J.B.F., Weiss, G. (2012) Exact FCFS matching rates for two infinite multi-type sequences. *Operations Research* 60:475–489.

[2] Adan, I.J.B.F., Weiss, G. (2012) A queue with skill based service under FCFS-ALIS: steady state, overloaded system, and behavior under abandonments. *Eurandom Report* 2012-011.

[3] Aerts, J., Korst, J., Verhaegh, W., (2007) Load balancing for redundant storage strategies: Multiprocessor scheduling with machine eligibility. *Journal of Scheduling* 4:245–257.

[4] Aksin, Z., Armony, M., Mehrotra, V., (2007) The modern call-center, a multi-disciplinary perspective on operations management research. *Production and Operations Management* 16:665–688.

[5] Caldentey, R., Kaplan, E.H., Weiss, G., (2009) FCFS infinite bipartite matching of servers and customers. *Advances in Applied Probability* 41:695-730.

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

[7] Garnett, O., Mandelbaum, A. (2000) An introduction to skill-based routing and its operational complexities. http://iew3.technion.ac.il/serveng/Lectures/SBR.pdf

[8] Gurumurthi, S., Benjaafar, S. (2004) Modeling and analysis of flexible queueing systems. *Naval Research Logistics* 51:755–782.

[9] Gurvich I., Whitt, W. (2010) Service-level differentiation in many-server service system via queue-ratio routing. *Operations Research* 58:316–328.

[10] Green, L. (1985) A queueing system with general-use and limited-use servers. *Operations Research* 33:168–182.

[11] Jennings O.B., Reed, J.E. (2011) An overloaded multiclass FIFO queue with abandonments. *Operations Research* 60:1282–1295.

[12] Jordan, W.C., Graves, S.C. (1995) Principles on the benefits of manufacturing process flexibility. *Management Science* 41:577–594.

[13] Jordan, W.C., Inman, R.R., Blumenfeld D.E. (2004) Chained cross-training of assembly line workers. *International Journal of Production Research* 42:1899–1910.

[14] Kaplan, E.H. (1984) Managing the demand for public housing. ORC technical report # 183, MIT.

[15] Kaplan, E.H. (1988) A public housing queue with reneging and task-specific servers. *Decision Sciences* 19:383–391.

[16] Talreja, R., Whitt, W. (2007) Fluid Models for Overloaded Multi-class many-service queueing systems with FCFS routing. *Management Science* 54:1513–1527.

[17] Visschers, J., Adan, I.J.B.F., Weiss, G. (2012) A product form solution to a system with multi-type customers and multi-type servers. *Queueing Systems* 70:269–298.

[18] Wallace R.B., Whitt, W. (2005) A staffing algorithm for call centers with skill-based routing. *Manufacturing and Service Operations Management* 7:276–294.

[19] Whitt, W. (2006) Fluid models for multiserver queues with abandonments. *Operations Research* 54:37–54.

[20] Zijm, W.H.M., Laarhoven, P.J.M. (1993) Production preparation and numerical control in PCB assembly. *International Journal of Flexible Manufacturing Systems* 5(3):187–207.

Simulated mean waiting times						
λ	n	$E[W_1]$	$E[W_2]$	$E[W_3]$	$E[W_4]$	$E[W_5]$
20	(7, 17, 17, 12, 7)	1.351	0.631	0.561	0.869	1.242
40	(14, 33, 34, 24, 14)	1.252	0.729	0.683	0.905	1.159
60	(20, 50, 51, 36, 22)	1.206	0.773	0.740	0.921	1.116
100	(34, 84, 85, 61, 36)	1.121	0.813	0.792	0.919	1.053
200	(68, 167, 169, 121, 72)	1.082	0.903	0.893	0.966	1.042
Target		1.000	1.000	1.000	1.000	1.000

Simulated abandonment rates						
λ	n	θ_1	θ_2	θ_3	θ_4	θ_5
20	(7, 17, 17, 12, 7)	0.245	0.125	0.111	0.165	0.227
40	(14, 33, 34, 24, 14)	0.225	0.139	0.131	0.168	0.210
60	(20, 50, 51, 36, 22)	0.216	0.145	0.140	0.169	0.202
100	(34, 84, 85, 61, 36)	0.200	0.150	0.147	0.168	0.190
200	(68, 167, 169, 121, 72)	0.192	0.163	0.162	0.173	0.186
Target		0.181	0.181	0.181	0.181	0.181

Target matching rates					
	s_1	s_2	s_3	s_4	s_5
c_1	0.192				0.108
c_2	0.008	0.092			
c_3		0.108	0.042		
c_4			0.258	0.042	
c_5				0.108	0.042

Simulated matches $\lambda = 20$					
	s_1	s_2	s_3	s_4	s_5
c_1	0.184				0.095
c_2	0.016	0.091			
c_3		0.108	0.056		
c_4			0.244	0.064	
c_5				0.094	0.048

Simulated matches $\lambda = 100$					
	s_1	s_2	s_3	s_4	s_5
c_1	0.187				0.104
c_2	0.010	0.093			
c_3		0.108	0.048		
c_4			0.252	0.050	
c_5				0.103	0.044

Table 3: Simulation results for the network in Figure 3 with complete resource pooling.

Simulated mean waiting times						
λ	n	$E[W_1]$	$E[W_2]$	$E[W_3]$	$E[W_4]$	$E[W_5]$
20	(10, 2, 19, 24, 5)	1.366	1.362	0.533	0.417	0.515
40	(19, 5, 38, 48, 10)	1.277	1.230	0.490	0.422	0.487
60	(29, 7, 57, 72, 15)	1.200	1.164	0.483	0.432	0.480
100	(48, 12, 96, 121, 25)	1.117	1.085	0.452	0.419	0.451
200	(96, 25, 191, 241, 49)	1.045	1.023	0.489	0.472	0.489
Target		1.000	1.000	0.500	0.500	0.500

Simulated abandonment rates						
λ	n	θ_1	θ_2	θ_3	θ_4	θ_5
20	(10, 2, 19, 24, 5)	0.248	0.248	0.106	0.084	0.102
40	(19, 5, 38, 48, 10)	0.230	0.223	0.095	0.083	0.095
60	(29, 7, 57, 72, 15)	0.215	0.210	0.093	0.084	0.093
100	(48, 12, 96, 121, 25)	0.201	0.196	0.087	0.081	0.087
200	(96, 25, 191, 241, 49)	0.187	0.184	0.092	0.089	0.092
Target		0.181	0.181	0.095	0.095	0.095

Target matching rates					
	s_1	s_2	s_3	s_4	s_5
c_1	0.169				0.113
c_2	0.056	0.038			
c_3			0.156		
c_4			0.156	0.156	
c_5				0.156	

Simulated matches $\lambda = 20$					
	s_1	s_2	s_3	s_4	s_5
c_1	0.170				0.100
c_2	0.069	0.021			
c_3		0.007	0.150		
c_4			0.160	0.170	
c_5				0.150	0.011

Simulated matches $\lambda = 100$					
	s_1	s_2	s_3	s_4	s_5
c_1	0.160				0.110
c_2	0.060	0.033			
c_3		0.002	0.160		
c_4			0.160	0.160	
c_5				0.150	0.003

Table 4: Simulation results for the network in Figure 3 with two-subsystem decomposition.

Simulated mean waiting times						
λ	n	$E[W_1]$	$E[W_2]$	$E[W_3]$	$E[W_4]$	$E[W_5]$
20	(6, 9, 7, 33, 3)	2.483	1.658	1.252	0.449	0.459
40	(11, 17, 15, 66, 6)	2.454	1.548	1.138	0.438	0.443
60	(17, 26, 22, 100, 9)	2.255	1.417	1.096	0.411	0.413
100	(28, 43, 37, 166, 15)	2.150	1.317	1.066	0.443	0.443
200	(56, 87, 74, 332, 30)	2.057	1.188	1.033	0.474	0.474
Target		2.000	1.000	1.000	0.500	0.500

Simulated abandonment rates						
λ	n	θ_1	θ_2	θ_3	θ_4	θ_5
20	(6, 9, 7, 33, 3)	0.403	0.294	0.232	0.090	0.092
40	(11, 17, 15, 66, 6)	0.393	0.272	0.209	0.087	0.088
60	(17, 26, 22, 100, 9)	0.366	0.250	0.200	0.080	0.081
100	(28, 43, 37, 166, 15)	0.351	0.233	0.194	0.085	0.085
200	(56, 87, 74, 332, 30)	0.336	0.210	0.186	0.089	0.089
Target		0.330	0.181	0.181	0.095	0.095

Target matching rates					
	s_1	s_2	s_3	s_4	s_5
c_1	0.173				0.074
c_2		0.101			
c_3		0.025	0.126		
c_4				0.334	
c_5				0.167	

Simulated matches $\lambda = 20$					
	s_1	s_2	s_3	s_4	s_5
c_1	0.156				0.076
c_2	0.020	0.071			
c_3		0.050	0.099		
c_4			0.027	0.326	
c_5				0.175	0.001

Simulated matches $\lambda = 100$					
	s_1	s_2	s_3	s_4	s_5
c_1	0.167				0.075
c_2	0.004	0.091			
c_3		0.031	0.119		
c_4			0.008	0.334	
c_5				0.171	

Table 5: Simulation results for the network in Figure 3 with three-subsystem decomposition.

Simulated mean waiting times					
Service-time dist.	$E[W_1]$	$E[W_2]$	$E[W_3]$	$E[W_4]$	$E[W_5]$
Exponential	1.209	0.774	0.740	0.921	1.118
Gamma	1.197	0.788	0.755	0.930	1.114
Uniform	1.195	0.789	0.756	0.929	1.112
Pareto	1.183	0.793	0.761	0.929	1.107
Target	1.000	1.000	1.000	1.000	1.000

Simulated abandonment rates					
Service-time dist.	θ_1	θ_2	θ_3	θ_4	θ_5
Exponential	0.216	0.145	0.139	0.170	0.202
Gamma	0.215	0.147	0.142	0.171	0.201
Uniform	0.213	0.147	0.142	0.171	0.201
Pareto	0.212	0.148	0.143	0.170	0.200
Target	0.181	0.181	0.181	0.181	0.181

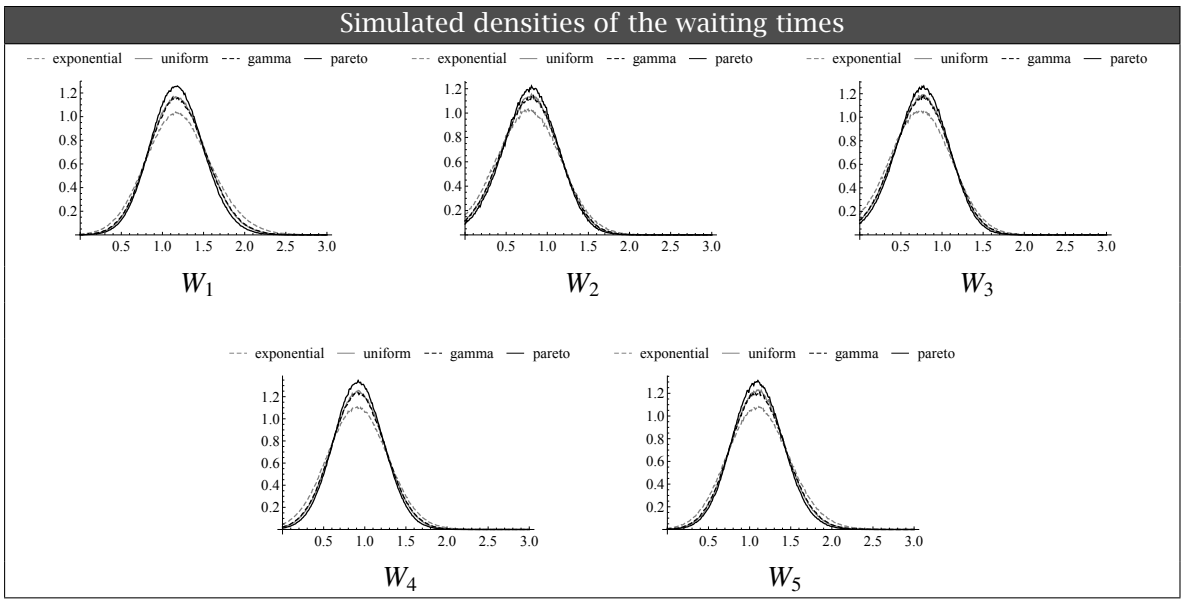

Table 6: Simulation results for the network in Figure 3 with complete resource pooling and general service-time distributions.

Simulated sequence of server types						
	1	2	3	4	5	Total
1	0.038	0.042	0.060	0.028	0.030	0.198
2	0.040	0.040	0.059	0.031	0.030	0.199
3	0.060	0.058	0.090	0.045	0.047	0.300
4	0.031	0.030	0.046	0.023	0.022	0.151
5	0.030	0.030	0.045	0.024	0.022	0.151
Total	0.198	0.199	0.300	0.151	0.151	1.000

(a)

Expected sequence of server types						
	1	2	3	4	5	Total
1	0.039	0.040	0.059	0.030	0.030	0.198
2	0.040	0.040	0.060	0.030	0.030	0.199
3	0.059	0.060	0.090	0.045	0.045	0.300
4	0.030	0.030	0.045	0.023	0.023	0.151
5	0.030	0.030	0.045	0.023	0.023	0.151
Total	0.198	0.199	0.300	0.151	0.151	1.000

(b)

Simulated sequence of customer types						
	1	2	3	4	5	Total
1	0.086	0.030	0.046	0.092	0.042	0.297
2	0.031	0.010	0.014	0.031	0.015	0.102
3	0.044	0.016	0.022	0.047	0.022	0.152
4	0.091	0.031	0.047	0.088	0.046	0.303
5	0.045	0.016	0.022	0.044	0.020	0.147
Total	0.297	0.102	0.152	0.303	0.147	1.000

(c)

Expected sequence of customer types						
	1	2	3	4	5	Total
1	0.088	0.030	0.045	0.090	0.044	0.297
2	0.030	0.010	0.016	0.031	0.015	0.102
3	0.045	0.016	0.023	0.046	0.022	0.152
4	0.090	0.031	0.046	0.092	0.044	0.303
5	0.044	0.015	0.022	0.044	0.022	0.147
Total	0.297	0.102	0.152	0.303	0.147	1.000

(d)

Table 7: The first table shows the fraction of occurrences of each pair (s_i, s_k) in the sequence of server types at each service start. The second table shows the expected fractions under the null hypothesis that the server types of two successive service starts are independent. The third and fourth table show similar results for the sequence of customer types.