Design heuristic for parallel many server systems

Citation for published version (APA):

Document license:
TAVERNE

DOI:
10.1016/j.ejor.2018.08.042

Document status and date:
Published: 16/02/2019

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Stochastics and Statistics

Design heuristic for parallel many server systems

Ivo J. B. F. Adan, Marko A. A. Boon, Gideon Weiss

Abstract

We study a parallel queueing system with multiple types of servers and customers. A bipartite graph describes which pairs of customer-server types are compatible. We consider the service policy that always assigns servers to the first, longest waiting compatible customer, and that always assigns customers to the longest idle compatible server if on arrival multiple compatible servers are available. For a general renewal stream of arriving customers, general service time distributions that depend both on customer and on server types, and general customer patience distributions, the behavior of such systems is very complicated. Key quantities for their performance are the matching rates, the fraction of services for each pair of compatible customer-server. Calculation of these matching rates in general is intractable, it depends on the entire shape of service time distributions. We suggest through a heuristic argument that if the number of servers becomes large, the matching rates are well approximated by matching rates calculated from the tractable bipartite infinite matching model. We present simulation evidence to support this heuristic argument, and show how this can be used to design systems with desired performance requirements.

1. Introduction

We consider designs for a parallel service system with customers of types \( C = \{c_1, \ldots, c_i\} \), servers of types \( S = \{s_1, \ldots, s_j\} \), and a bipartite compatibility graph \( \mathcal{G} \subseteq C \times S \), where \( (c_i, s_j) \in \mathcal{G} \) if servers of type \( s_j \) can serve customers of type \( c_i \). We focus on the policy of first come first served (FCFS), where whenever a server is available he will take the longest waiting compatible customer if any, and assign longest idle server (ALIS), where whenever a customer arrives he will be assigned to the longest idling compatible server if any. This policy of FCFS-ALIS has several advantages and is widely used.

We are given the total arrival rate \( \lambda \), of which a fraction \( \alpha_c \) are customers of type \( c_i \). We assume that customers of type \( c_i \) have a patience probability distribution \( F_c(t) \). We are also given average service times \( m_{c_i,s_j} \) and service rates \( \mu_{c_i,s_j} = 1/m_{c_i,s_j} \). Our goal is to determine the total number of servers \( n \) and the numbers \( n_{ij} \) of servers of each type \( s_j \), which are necessary to serve these customers so as to satisfy some given quality of service requirements. We denote \( \theta_{ij} = n_{ij}/n \).

In our designs we consider the following quality of service determinations:

Complete resource pooling designs In these designs the servers under FCFS-ALIS achieve complete resource pooling, which means that different types of customers have the same average waits, and different types of servers have the same average idleness periods. We then have:
- Quality driven (QD) service, where most customers are served immediately, and all types of servers will have the same pre-specified target average idle time \( T \) between services.
- Efficiency driven (ED) service, where servers will almost never idle, and all types of customers will have the same target average waiting time \( W \), and approximate fraction \( F_i(W) \) of customers of type \( c_i \) will abandon the system without service.
- Quality and efficiency driven (QED) service, where almost no customers abandon, and both waiting times of customers and idling times of servers are short.

Designs with differentiated service levels In these designs the system decomposes automatically, while implementing FCFS-ALIS policy, into subsets of customer types, arranged by

© 2018 Elsevier B.V. All rights reserved.
order of priority. Each of these subsets will then be served almost exclusively by a subset of the server types. We can then specify quality of service requirements individually for each subset of customer types according to priority, so that

- High priority customer types: Will have QD service, with pre-specified target average idle times.
- Medium priority customer types: Will have QED service.
- Low priority customers will have ED service, with pre-specified target average waiting times, and corresponding abandonment rates.

To obtain the required designs we are free to choose the server configuration, which we pre-specify either by the fractions $\beta_{ij}$ of number of services performed by servers of type $s_j$, or by the fractions $\theta_{ij}$ of numbers of servers of type $s_j$. Under both specifications we then determine $n$, which will be proportional to $\lambda$.

The difficulty in such designs is to determine the matching rates $r_{c,i,j}$: the fraction of services for each pair $(c_i, s_j) \in \mathcal{G}$. Exact determination of $r_{c,i,j}$ is, in most cases of interest, completely intractable. In our designs we use approximate estimates of these matching rates. These estimates are calculated from a corresponding FCFS infinite bipartite matching model (FCFS-IBM). We conjecture that the actual matching rates for our parallel service systems converge to the matching rates of FCFS-IBM as $\lambda$ and the corresponding $n$ go to infinity. Hence, all our designs may only achieve their targets approximately, but we conjecture that for large $\lambda$ and $n$ they will be accurate.

We derive and implement all of these designs, and perform an extensive simulation study to assess their performance. The most impressive part of this paper is in these simulation studies. First, the simulations seem to confirm our conjecture. But much more important, the designs achieve the desired performance requirements, in all of the designs which we implemented, already for relatively small, realistic $\lambda$ and $n$.

The structure of the paper is as follows: In Section 2 we discuss background and motivation for our various choices of approach. In Section 3 we survey results on the FCFS-IBM model, and in Section 4 we formulate our many server scaling conjecture of the link between our FCFS-ALIS parallel service system and the FCFS-IBM model. In Section 5 we derive the algorithms for complete resource pooling designs, followed by an extensive simulation study in Section 6. In Section 7 we derive the algorithms for differentiated service and assess their performance. Section 8 discusses the numerical procedure to determine design parameters when fractions $\theta_{ij}$ of numbers of servers of type $s_j$ are specified. We end the paper with a discussion in Section 9.

This paper is an extension and elaboration of some preliminary results in Adan, Boon, Bušić, Mairesse, and Weiss (2013); Adan, Boon, and Weiss (2013).

2. Preliminaries

In this section we make some comments on our motivation and discuss some of the concepts that are used in the paper.

2.1. Parallel service systems

Parallel service systems with server types $S = \{s_1, \ldots, s_j\}$, customer types $C = \{c_1, \ldots, c_l\}$, and bipartite compatibility graph $G \subseteq C \times S$, are typical in situations where a large volume of service requests of various types are channeled to a central facility, where they are attended by a large number of agents differentiated by skill. Such situations commonly occur in manufacturing, transportation, service contact centers, health systems, communications, internet data exchange, computing, and various other areas of applications.

Our queuing model for this type of system has a general renewal stream of arriving customers with rate $\lambda$, where successive arrivals are of i.i.d types, $c_i$ with probability $\alpha_{c_i}$, and there is a total of $n$ servers, $n_{c,i,j}$ of which are of type $s_j$. Service times are independent, distributed according to general distributions $G_{c_i,s_j}$, with mean $m_{c_i,s_j}$ and service rate $\mu_{c_i,s_j} = 1/m_{c_i,s_j}$. Customers have finite patience, with independent patience time distributions $F_i$, and a customer abandons if he does not start service by the time his patience is exhausted.

Parallel server systems are widely discussed in the literature. An incomplete list would include an early study Green (1985), applications to manufacturing and supply chain management Rubino and Ata (2009); Veeger, Etman, and Rooda (2010), applications to call centers and internet service systems Gans, Koole, and Mandelbaum (2003); Harchol-Balter, Crovella, and Murta (1999); Squillante, Xia, Yao, and Zhang (2001); Wallace and Whitt (2005); Zeltyn and Mandelbaum (2005), attempts to find optimal policies, mainly for small graph systems Armony and Ward (2010, 2013); Bell and Williams (2001); Chlamtac and Ward (2013); Tesican and Dai (2010); Williams (2000), heavy traffic and fluid approximations Harrison and Lopez (1999); Harrison and Zeevi (2005), and many server scaling Gurvich and Whitt (2009, 2010). Surprisingly, not many papers have considered FCFS for these systems. Most relevant to our current paper are Adan and Weiss (2014); Foss and Chernova (1998); Nov, Weiss, and Zhang (2016); Talreja and Whitt (2007).

2.2. Objectives for design of parallel service systems

In assessing such systems there are various objectives that may be of importance, on the customer side they include waiting times and abandonment rates as well as consideration of fairness to customers of various types or priorities for some types. In conflict with those, on the server side there is the objective of maximum utilization of the servers, minimizing their number, and reaching a balanced work division between the various types. Each of these may carry a different weight in different application contexts. Often $\lambda$, $\alpha_{c_i}$, and $F_i$ are given, together with some form of quality of service requirements. All the other parameters of the system can be adjusted to achieve the requirements in an optimal way: One can redesign the bipartite compatibility graph, change the service rates, change the workforce mix, decide on $n$, and decide on the service policy. It should perhaps be pointed out that changing the service policy may be as hard and costly as adjusting any of the other system parameters. The level of generality of the queuing models that we study here does not allow a complete exact analysis, and performance is often evaluated in practice by simulation. However, any methods for calculating approximate performance measures or supporting design without the need to use simulation should be quite valuable. It is the aim of this paper to deliver such methods.

2.3. First come first served, assign to longest idle server policy

FCFS-ALIS in a parallel service system has several advantages: It attempts to achieve resource pooling Tsitsiklis and Xu (2012), i.e., all the servers share the same degree of utilization, and it attempts to give all customers the same service level, so that global FCFS is achieved, irrespective of types of customers Talreja and Whitt (2007).

One notable property of FCFS is the following: Assume that arriving customers can choose the server they wish to go to, and each server then serves his queue FCFS. If each arrival has complete information on the schedule of all the servers at his moment of arrival, then to minimize his waiting time he will join the compatible server that has the shortest workload (JSW). Thus JSW is
the Nash equilibrium of fully informed customers minimizing waiting times. But this policy of JSW is automatically achieved when customers queue up in a single queue and the servers are using FCFS. FCFS can then serve as a benchmark, and comparison of the costs under FCFS with other policies will provide an estimate of the price of anarchy.

The ALIS part of the policy is also fair to the servers, by imposing equal idle times on servers irrespective of server type.

Some further advantages of FCFS is that it is easy to implement, as it does not require any online calculations or knowledge of system parameters. It is also sometimes required by law Kaplan (1988). Finally it is indeed a policy very commonly used in practice. On the negative side, FCFS may waste resources by letting servers serve customers for which they are not efficient, and it may cause long delays to customar types that have a limited number of compatible serv servs. However, some of these short comings can be avoided by redesigning the compatibility graph. It is safe to say that in practice most service systems use FCFS for a large proportion of their operation, even if they implement some more sophisticated policies in some of their service decisions.

2.4. Matching rates and a conjecture

Unfortunately, analysis of parallel service systems under FCFS-ALIS is very hard. Foss and Chernova (1998) provided an example of a symmetric system with three types of customers and three servers, and just two service distributions with fixed fast and slow service rates, where they conjectured that stability of the system depends on the entire shape of the service time distributions and not just on the service rates. This conjecture is confirmed experimentally by a simulation study in Nov et al. (2016).

The difficulty is, that we do not a-priori know the type of customer that a server of a given type will serve, if he has to choose customers according to FCFS-ALIS. In a stationary system we define the matching rates \( r_{c,i} \) as the long term average fraction of services of customers of type \( c \) which are served by servers of type \( i \), so that \( \sum_{(c,i) \in \mathcal{G}} r_{c,i} = 1 \). If we know the matching rates we can calculate the average amount of work per customer \( \sum_{(c,i) \in \mathcal{G}} r_{c,i} m_{c,i} \). We then have that with \( n \) servers the service capacity of the system is:

\[
\mu = n \left( \sum_{(c,i) \in \mathcal{G}} r_{c,i} m_{c,i} \right)^{-1}.
\]

We can then conclude that stationarity is possible if \( \lambda < \mu \), and if \( \lambda > \mu \) then the system may be stabilized if a fraction of \( 1 - \frac{\mu}{\lambda} \) of the customers abandon the system without service.

This expression of service capacity is meaningless, as the calculation of matching rates for a stationary parallel service queueing system under FCFS-ALIS is in general intractable. The few known cases where one can calculate matching rates are when the bipartite compatibility graph is a tree (i.e., it has no loops) Nov et al. (2016); Tale and Whitt (2007), and when service rates depend only on the server, arrivals are Poisson, and service times are exponential Adan and Weiss (2014). However, matching rates can be calculated for the much simpler and very tractable FCFS infinite bipartite matching (FCFS-IBM) model Adan, Bušić, Mairesse, and Weiss (2018); Adan and Weiss (2011); Caldentey, Kaplan, and Weiss (2009). We discuss the FCFS-IBM model in Section 3.

It is our conjecture that when \( \lambda \) and \( n \) are large, the matching rates of the general parallel service system under FCFS-ALIS are approximated by those of a corresponding FCFS-IBM model. This conjecture is at the basis for our current paper.

A first step towards verifying this conjectured many server scaling limit is made in Zhan and Weiss (2018) by studying the many server ‘N’-system. Even for this very simple system, the derivations of the limits are quite laborious, emphasizing the difficulty in verifying the conjecture.

Our results in this paper use the ability to calculate matching rates for FCFS-IBM, and our conjecture on the many server behavior, in order to design FCFS-ALIS parallel service systems.

2.5. Resource pooling and differentiated service

An exact definition of complete resource pooling in a parallel service system under FCFS-ALIS is given in Nov et al. (2016), in terms of the fluid limit model of the system. Essentially, it says that if we look at a fluid approximation of the system, the deterministic fluid paths of all the servers converge to a single path for arbitrarily large \( \lambda \).

What that means for the stochastic system is that if we consider the stream of arriving customers, and think of the positions of the servers in this sequence at some time \( t \), these positions will remain not far apart, and servers of all types will continuously overtake each other, so that the queues between the servers remain stable for arbitrary total \( \lambda \).

It is well known that resource pooling is extremely desirable in queuing systems: combining \( n \) separate identical single server queues into a single queue with an \( n \)-fold faster server will reduce average queue length and average waiting time by a factor of \( n \), and when traffic intensity is high similar savings are achieved by a single queue with \( n \) servers, and by join shortest queue. While no complete exact results on the effect of pooling in parallel service system are available, Tsitsiklis and Xu (2012) indicate that a similar effect should exist.

In our designs in Section 5 we achieve complete resource pooling. Under complete resource pooling we also obtain that all types of customers receive the same level of service.

In some service systems it is desired not to pool all resources, but to differentiate between the service level of various types of customers, i.e., some types of customers have priority over other types of customers. To achieve this one may assign more servers to higher priority customer types, and one may wish not to use servers that are compatible with high priority customers, to serve customers with lower priority.

In our designs in Section 7 we achieve differentiation of service levels. This differentiation in service levels is achieved without imposing strict priorities: the system is still operated by FCFS-ALIS, but all the links in the compatibility graph that allow servers of high priority customers to serve customers with lower priority will almost never be used.

2.6. Some further design parameters

Once we are given the data \( \lambda, \alpha_{c,i}, \mu_{c,i,s} \) and the quality of service requirement, there may be many ways of achieving this, by using different configurations of servers. We will consider two methods to pre-specify the configuration of servers of the various types. We will first consider pre-specifying the fraction of all the services that will be performed by servers of type \( s_j \), which we denote as \( \beta_{s_j} \). The other method is to pre-specify the fraction \( \theta_{ij} = n_{ij}/n \) of number of servers of type \( s_j \). While specifying \( \theta_{ij} \) seems more natural, the specification of \( \beta_{s_j} \) leads to a more direct implementation. Both methods lead to the same configurations and the same system performance. We describe the first in Sections 5–7, and show how to use the second method in Section 8.
3. FCFS infinite bipartite matching

In this section, we survey results for the FCFS-IBM model, taken from Adan et al. (2018); Adan and Weiss (2011); Caldentey et al. (2009). For this system we can calculate matching rates \( r_{c,s} \). In Section 4 we formulate a conjecture that links this model to parallel server FCFS-ALUS systems.

We now consider a system with customer types \( C \) and server types \( S \), with a bipartite compatibility graph \( G \), and a much simplified stochastic model: We have infinite sequences of customers \( c^1, c^2, \ldots, c^m, \ldots \) where \( c^m \in C \) and of servers \( s^1, s^2, \ldots, s^k, \ldots \) where \( s^k \in S \) (note, superscript \( c^m \) denotes the item in position \( m \), which is one of \( c_1, \ldots, c_l \), similarly for \( s^k \)). We assume that \( c^m \) are drawn according to probabilities \( \alpha = (\alpha_1, \ldots, \alpha_l) \) and \( s^k \) are drawn according to probabilities \( \beta = (\beta_1, \ldots, \beta_j) \), and they are all independent. For each realization of the sequences we match customers and servers according to a FCFS policy: \( s^k \) is matched to the earliest compatible \( c^m \) in \( c^1, c^2, \ldots \), which has not yet been matched to \( s^1, \ldots, s^{k-1} \). The matching process, for given graph \( G \), is illustrated in Fig. 1. This model is much simpler than a queueing model, since it involves no arrival times, no service times, no busy or idle servers, and since it treats customers and servers in an entirely symmetric way. It is shown in Adan and Weiss (2011) that the matching is uniquely determined for any two sequences and that all customers and servers are matched almost surely. Furthermore, the system demonstrates dynamic reversibility, and is associated with a Markov chain that has a product form stationary distribution. The stationary distribution is used to obtain explicit expressions for the matching rates. While this FCFS-IBM model turns out to be quite simple, the actual formula for the matching rates is rather complex. We describe the calculation of the matching rates now.

We use the following notations: we let \( C(s_j) \) be the set of customer types compatible with server type \( s_j \), and \( S(c_i) \) be the set of server types compatible with customer type \( c_i \). For a subset of customer types \( C \) we let \( S(C) = \bigcup_{c_i \in C} S(c_i) \), and for a subset of server types \( S \) we let \( C(S) = \bigcup_{s_j \in S} C(s_j) \). We also let \( \mu(S) = \mu(S) \) be the customer types that can only be served by servers of type \( S \) (note that \( S \) means complement of \( S \), so \( \mu(S) \) is the complement of the set \( \mu(S) \) of the complement \( \mu(S) \) of \( S \) for subsets \( C, S \) we define \( \alpha_C = \sum_{c_i \in C} \alpha_{c_i} \) and \( \beta_S = \sum_{s_j \in S} \beta_{s_j} \).

**Definition 3.1.** For given \( \alpha, \beta, G \) we say that there is complete resource pooling in the FCFS infinite bipartite matching system if the following three equivalent conditions hold:

\[
\alpha_C < \beta_S(\alpha_C), \quad \beta_S < \alpha_{C(S)}, \quad \alpha_C > \beta_S(\alpha_S), \quad S \subset C, S \neq \emptyset, C \subset C, C \neq \emptyset, C.
\]  

(1)

The three versions of this condition say that there are enough servers to match all the customers of a subset of customer types, and there are enough customers to match all the server of a subset of server types. The conditions are equivalent, partly because the \( \alpha_{c_i} \) as well as the \( \beta_{s_j} \) add up to 1.

**Theorem 3.2** (from Adan & Weiss, 2011). (i) Let \( r_{c,s}(n) \) be the (random) number of \( c_i, s_j \) matches between \( c^1, \ldots, c^n \) and \( s^1, \ldots, s^n \) in the FCFS infinite bipartite matching of the two sequences. If complete resource pooling holds, then almost surely \( \lim_{n \to \infty} r_{c,s}(n)/n = r_{c,s} \).

(ii) The matching rate \( r_{c,s} \) is calculated by

\[
\left\{ \begin{array}{l}
\sum_{k=1}^{l-1} \frac{\alpha_{(k)}}{\beta(k) - \alpha_{(k)}} \prod_{i=1}^{k-1} \beta(i) - \alpha(i)(X_i + \phi_j \prod_{i=1}^{j-1} \beta(i) - \alpha(i)X_i)}
\end{array} \right.
\]

where the summation is over \( \mathcal{J} \), the set of all permutations of the server types \( S \), and for each permutation (\( S_1, \ldots, S_j \)) of the servers \( S_1, \ldots, S_j \), the following notation is used:

\[
\alpha(k) = \alpha_{S_1, \ldots, S_j}, \quad \beta(k) = \beta_{S_1, \ldots, S_j}, \quad k = 1, \ldots, j.
\]

\[
\phi_k = \frac{\alpha_{S_1, \ldots, S_j} \alpha_{C(S_1)} \alpha_{C(S_j)}}{\alpha_{S_1, \ldots, S_j}}, \quad \psi_k = \frac{\alpha_{S_1, \ldots, S_j} \alpha_{C(S_1)} \alpha_{C(S_j)}}{\alpha_{S_1, \ldots, S_j}}, \quad k = 1, \ldots, j.
\]

\[
B = \sum_{k=1}^{j} \left( \beta_{S_1, \ldots, S_j} \alpha_{S_1, \ldots, S_j} \right) - 1.
\]

(2)

An easy example of this formula is for the case that \( I = J \) and \( C(S_j) = \alpha(C_j) \), i.e. the bipartite compatibility graph is almost complete, each server can serve all but one of the customer types. In that case, complete resource pooling holds if and only if \( \alpha_{c_i} + \beta_{s_j} < 1 \) and the matching rates are:

\[
r_{c,s} = \frac{(1 - \alpha_{c_i}) (1 - \beta_{s_j}) - \alpha_{c_i} \beta_{s_j}}{(1 - \alpha_{c_i} - \beta_{s_j}) (1 - \alpha_{c_i} - \beta_{s_j})} \left( 1 + \sum_{i=1}^{l} \frac{\alpha_{c_i} \beta_{s_j}}{(1 - \alpha_{c_i} - \beta_{s_j})} \right).
\]  

(3)

However, for any other bipartite compatibility formula (2) does not seem to simplify, and we suspect that its calculation is \( \mathcal{P} \) hard. We have programmed it to be able to calculate it up to \( I, J \leq 10 \), but it will become hard to compute the matching rates for larger number of types. Recently, Fazel-Zarandi and Kaplan (2017) developed a highly accurate and efficient approximation for the matching rates \( r_{c,s} \) based on Ohn’s Law (which in some cases reduces to exact results). This provides an attractive alternative to the exact solution in case of many customer and server types.

When resource pooling does not hold, it is shown in Adan and Weiss (2014) that there is a unique decomposition \( (C, S) \) into subsystems (\( C(1), S(1) \)), \( (C(2), S(2)) \), such that

\[
\frac{\beta_{(1)}}{\alpha_{(1)}} < \cdots < \frac{\beta_{(j)}}{\alpha_{(j)}}.
\]

(4)

\( (C(1), S(1)) \) has complete resource pooling, \( I = 1, \ldots, L \).

A Mathematica program to exactly calculate the matching rates for given \( \alpha, \beta, G \) is available from the authors.
4. Matching under many server scaling

In this section, we describe the conjectured behavior of the FCFS-ALIS parallel service system when arrival rates and number of servers are increased, and formulate the conjectured link with the infinite bipartite matching model, which enables us to approximate the matching rates.

We first survey the behavior of the GI/G/n queue with abandonments, as \( n \) becomes large. Next we formulate our conjecture regarding matching rates of our parallel service system. We then consider the behavior of the parallel service system when complete resource pooling holds, in three modes of operation, QD, ED and QED. Finally we consider the case that complete resource pooling does not hold, and the system decomposes.

4.1. Many server behavior of GI/G/n queue with abandonments

Consider a queueing system with a single customer type and a single server type, with arrival rate \( \lambda \), and patience distribution \( F \), and with \( n \) servers, each with service rate \( \mu \), so that the traffic intensity is \( \rho = \lambda/n\mu \). We assume that servers operate on the ALIS schedule, i.e., when several servers are idle and a customer arrives, the customer will be assigned to the longest idle server. Many server scaling occurs when we keep \( \mu \) and \( \rho \) fixed and let both \( \lambda \) and \( n \) increase at the same rate. Note that, to increase \( \lambda \), we scale the inter-arrival time distribution, and thus we do not alter its shape. Because of abandonments the system will always be stable. There will be three modes of operation for this system: When \( \rho < 1 \) the system is in QD (quality driven mode). In QD mode, there is almost always a fraction \( \approx (1-\rho) \) of the servers which are idle, and customers almost never wait, and as a result customers almost never abandon. In fact, each server after each service completion will almost always idle, for an approximate time \( T = \frac{1}{\mu} \frac{1}{1-\rho} \). When \( \rho > 1 \) the system is in ED (efficiency driven mode). In ED mode, servers are almost always busy and there is almost always a queue, so almost all customers have to wait. Furthermore, with \( W \) determined by \( F(W) = (\rho - 1)/\rho \), a fraction of \( \approx F(W) \) of the customers abandon without service. In fact, almost all the customers with patience that is \( < W \) abandon the system without service, while almost all the customers with patience that is \( > W \) are served after a wait that is \( \approx W \). When \( \rho \approx 1 \) the system is in QED (quality and efficiency driven mode). In QED mode, servers are busy most of the time and if they idle it is only for a short while, an appreciable fraction of customers do not need to wait, the remaining customers wait a very short time, and very few customers abandon. Whitt (2006).

4.2. A conjecture on matching rates

We now consider our FCFS-ALIS parallel service system. We fix the fractions \( \alpha_i \) and \( \beta_i \), the service time distributions \( G_{\alpha_i,\beta_i} \), and the patience distributions \( F_i \). We then let \( \lambda \) and \( n \) increase at the same rate, so that we get into many server scaling. We cannot directly calculate \( \rho \) for this system, as it depends on the matching rates, which are intractable, and in general depends not only on service rates but also on the shape of the service distributions, but we will try and approximate the matching rates and \( \rho \). We now make the following conjectures regarding the behavior of the system under many server scaling, when complete resource pooling holds.

(i) We conjecture that the order in which customers will reach the head of the line (if they did not abandon previously) will be such that the types of customers will be i.i.d. with some probabilities \( \tilde{\alpha}_i \), approximately.

(ii) We conjecture that the order in which servers will become available at completion of service, will be such that the types of servers will be i.i.d with some probabilities \( \tilde{\beta}_i \), approximately.

The rationale for the first conjecture is that under complete resource pooling, when \( \lambda \) and \( n \) are large, under QD or QED customers will almost never wait and so the successive types will be i.i.d with probabilities \( \alpha_i \), while under ED, all customers will almost always wait for approximately a constant time \( W \) before being served, and so a fraction \( F_i(W) \) of customers of type \( c_i \) will abandon, and successive customer types that reach the head of the line will be i.i.d. with some modified probabilities.

The rationale for the second conjecture is that in steady state the time points at which each individual server will complete a job form a stationary point process, and with many servers these point processes will be nearly independent. It is known that the superposition of many independent stationary point processes, under appropriate scaling, converges to a Poisson process (cf. Khinchine (1960)), which indicates that our conjecture on server availability may in fact be true.

We note that these conjectures are only loosely formulated, and a correct statement may need to be modified and qualified. In the rest of the paper we will assume that these \( \tilde{\alpha}_i \) and \( \tilde{\beta}_i \) exist. We will have similar conjectures on the existence of \( \tilde{\alpha}_i \) and \( \tilde{\beta}_i \) for the decomposed system, when complete resource pooling does not hold.

We now take a leap to connect our parallel service system with the infinite bipartite matching model:

**Conjecture 4.1.** Consider the FCFS-ALIS system, under many server scaling. Assume existence of \( \tilde{\alpha}_i \) and \( \tilde{\beta}_i \). If \( \tilde{\alpha}_i \), \( \tilde{\beta}_i \) satisfy the complete resource pooling condition \( (\rho) \), then the matching rates \( r_{\beta_i,\alpha_i} \) for the service system will converge to the matching rates calculated from \( (\rho) \), in which \( \alpha_i \) is replaced by \( \tilde{\alpha}_i \).

4.3. Three modes of operation under complete resource pooling

The following three figures illustrate the operation of our system under FCFS-ALIS when complete resource pooling holds, in each of the three modes of operation:

In ED mode, all the servers are almost always busy, customers with enough patience wait a time of approximately \( W \), and when they reach the head of the queue, they match with the next compatible server (see Fig. 2). Note that customers entering service are still of i.i.d. types, approximately, but with new probabilities \( \tilde{\alpha}_i \), since they are thinned independently by impatience.

In QD mode, there is a queue of idle servers which is almost never empty, each server, on completing a service, joins the end of this queue (see Fig. 3). A server reaches the head of the queue after an idle time of approximately \( T \), and matches with the first compatible customer. Customers almost never wait and are of i.i.d. types with probabilities \( \tilde{\alpha}_i \).

In QED mode, the system alternates infrequently between periods when there is a short queue of waiting customers, and during those periods servers do not idle, and periods when there is a short queue of idle servers, and in those periods the customers which arrive enter service immediately without waiting (see Fig. 4). In this mode, a characteristic of the system is the fraction of customers that do not wait, and the fractions of service completions which are not followed by idleness.

We do not attempt to control these fractions of no wait and no idle. They depend on finer adjustments of the server allocations, and on coefficients of variation of service and arrival times, which are outside our framework, and should be subject to further research.

In Figs. 2–4 we circle the place in the figure where customers and servers match. The conjecture is that types of servers and types of customers when they reach this point are independent.
i.i.d. (approximately) and so matches occur just as in the infinite matching model.

4.4. Operation of decomposed systems

When there is no resource pooling, the system decomposes into sub-systems as stated in (4).

The way that this decomposition occurs in the FCFS-ALIS parallel service system is as follows: There may be a subset of customer types, \( C(1) \) which are served by servers \( S(1) = S(C(1)) \) for which the total service capacity of \( S(1) \) is low compared to the service capacity of the remaining servers \( S(T) = S\backslash S(1) \). In that case, the servers \( S(T) \) will serve all the customers of types \( C(T) \) before any of the servers \( S(1) \) will reach them, and the servers of \( S(1) \) will serve only customers of types \( C(1) \). That is, for \( (c_i, s_j) \in \mathcal{G} \) for which \( s_j \in S(1), c_i \notin C(1) \), a server of type \( s_j \) will almost never serve a customer of type \( c_i \). Thus the sub-system \( (C(1), S(1)) \) will behave as if it is separated from the rest of the system. This sub-system will have the lowest quality service for the customers, and the highest utilization for the servers.

It is then possible that the remaining customer and server types, \( (C(T), S(T)) \) further decomposes, with a sub-system \( (C(2), S(2)) \) which has better service quality, and less server utilization than \( (C(1), S(1)) \), but lower service quality and higher utilization than the remaining customer and server types. Further decomposition is then possible, with a total of \( l \) sub-systems.

For our parallel server system under FCFS-ALIS this decomposition will be unique. Furthermore, in these cases, under many server scaling, even though the policy will allow service between the sub-systems, say of \( c_i \in C(1), s_j \in S(2), c_i' \in C(2), (c_i, s_j) \in \mathcal{G} \), such services will almost never occur.

When there is no resource pooling we may have that all sub-systems operate in QD mode, or all sub-systems operate in ED mode, or more generally, we may have that some sub-systems operate in ED mode, some in QD mode, and possibly also one sub-system operates in QED mode. In the third most general case we would have:

\[
W_1 > \ldots > W_{l-1} > W_l = 0 = T_l < T_{l+1} < \ldots < T_l
\]

i.e., the limiting average waiting times are longest for sub-system \( (C(1), S(1)) \), and the limiting average idle times are longest for sub-system \( (C(2), S(2)) \). Of course under QED actual waiting or idling is always \( \geq 0 \) but it approaches 0 as \( n \) increases.

To further illustrate this situation, assume that the staffing is such that the system decomposes into 3 sub-systems. Sub-system \( (C(1), S(1)) \) has the least staffing, and so will receive the worst service, while sub-system \( (C(2), S(2)) \) has the most abundant staffing, and will receive the best service, with sub-system \( (C(2), S(2)) \) in-between.

With no abandonments, starting with small \( \lambda \), the system will be stable under FCFS-ALIS. If we let \( \lambda \) increase, servers in \( S(1) \) will become fully utilized, and the queue of customers of types in \( c(1) \) will become unstable, while \( (C(2), S(2)) \) will also become utilized, and the queue of customers of types \( C(2) \) will also become unstable, while \( (C(3), S(3)) \) remain stable, and finally when \( \lambda \) becomes large, all the servers will be fully utilized and the queues of all types will become unstable.

Furthermore, once the whole system is unstable, the longest waiting customers will all be of types in \( c(1) \), waiting for servers \( S(1) \). Servers \( S(2) \) will skip those longest waiting customers and will serve customers of types \( C(2) \) which will have shorter waits, and servers \( S(3) \) will skip all the waiting customers of type \( C(1), C(2) \), and serve customers of types \( C(3) \), which will have the shortest waits.

Under abandonments, for high enough \( \lambda \), customers of types \( C(1) \) will have the highest fraction of abandonments and the longest wait, customers of types \( C(3) \) will have the smallest fraction of abandonments and the shortest wait, and types \( C(2) \) will be in-between.

On the other hand, if the system is stable, for small enough \( \lambda \), under ALIS servers of types \( S(1) \) will have shorter idle periods than servers \( S(2) \), who in turn will have shorter idle periods than servers \( S(3) \).

Fig. 5 shows how such a system will behave under our conjecture.

The three illustrations depict behavior under ED (top figure), QD (middle figure), and in the bottom one, \( (C(1), S(1)) \) is in ED, \( (C(2), S(2)) \) is in QED and \( (C(3), S(3)) \) is in QD.

In ED mode (top of Fig. 5), customers of type \( C(1) \) wait the longest time, before being served by servers of type \( S(1) \). Servers of types \( S(2) \) skip customers of type \( C(1) \) and serve customers of
5. Complete resource pooling designs

In this section, we introduce algorithms to design a parallel service system with complete resource pooling. After discussing the general strategy, we present the algorithms for QD, ED, and QED service, respectively. An algorithm for differentiated service is presented in Section 7.

5.1. General strategy

The setup for the design problem is given by server and customer types and the compatibility graph. To these are added the patience distributions of the customers, and the service time distributions for each customer/server type pair. Of the latter only the average service time and service rate are required for the design. Next the data includes the arrival rate $\lambda$, and the customer type frequencies $\alpha_i$.

The first design decision is whether we wish to provide uniform service levels to all types, or whether we divide them to classes of varying priorities. In this section, we design the system to have complete resource pooling; in the next section we partition the customers and assign subsets of servers to each priority class in such a way that the different priorities will receive different service levels under FCFS-ALIS (details follow later).

Next we need to decide on the mode of operation, ED, QD or QED, and on the parameters of quality of service and of utilization. We consider the case of complete resource pooling. If we decide on ED, then quality of service will be determined by specifying the target waiting time uniformly for all types of customers (which also determines fractions of abandonments). If we decide on QD, then the level of utilization will be determined by specifying an target idle time uniform for all types of servers. If we decide on QED, the required system will need to have servers almost fully utilized and zero or short customer queues.

For given $\lambda$ and $\alpha_i$, having specified the quality parameters, there will be many staffing combinations of servers that will satisfy these requirements. The next design decision will specify which of these staffing designs we are to choose.

As we stated in the introduction, we can do this in two different ways: We can pre-specify the fraction of the total number of services $\beta_i$, which each type of server performs, or we can pre-specify the fraction $\theta_i$ of the total number of servers for each type. The first is easier, and we will describe the algorithms based on pre-specified $\beta_i$, in which we calculate the required $n$ and $n_j$, and determine $\theta_i$. If $\theta_i$ are pre-specified we will need to solve numerically for the $\beta_i$ that will yield these values of $\theta_i$. We outline the numerical procedure to determine the $\theta_i$ in Section 8.

Once we determined the quality parameters, modified $\alpha_i$, can be determined and with pre-specified $\beta_i$, we then use the bipartite infinite matching model, formula (2), to obtain the matching rates $c_{i,j}$. Once we have the matching rates, we can calculate the amount of work required from each type of server, and this determines, by Little's law, the number of servers that are needed of each type in order to meet the requested quality of service and utilization. In the following sub-sections we show how to perform these steps for complete resource pooling under each of the three regimes. We illustrate the calculations for several examples in Section 6, and demonstrate the effectiveness of the heuristics through simulation.
5.2. Design for quality driven service

Here the traffic intensity is < 1, and customers almost never wait, and therefore even more rarely abandon. There are almost always some idle servers waiting for customers, and because of ALIS, servers of different types all have the same idle time distribution. The quality parameter in this case is the value $T$ of the target server idle time. It is a measure of the utilization of the servers. Because there are virtually no abandonments, the patience time distribution is not required as input.

Algorithm for QD

Input:

- Compatibility graph $G$
- Arrival rate $\lambda$
- Fractions of customer types $\alpha_i$
- Mean service times $m_{i,s_j}$

Requested quality of service parameter:

- Target server idle time after each service $T$

Design parameters:

- Fraction of services performed by each server type $\beta_i$

Algorithm:

- Check $\alpha, \beta$ for complete resource pooling
  
  Compute matching rates
  
  $r_{c,s} := \text{use Eq. (2)}$
  
  Compute staffing levels
  
  $n_i := \frac{\lambda}{\sum_{i \in C(s)} r_{c,s}} (m_{i,s} + T)$

Output:

- Required workforce $n_i$

Note that the required workforce $n_i$ calculated from Little’s law may not be integer and needs to be rounded. This can be done in any way chosen by the decision maker.

5.3. Design for efficiency driven service

Here the traffic intensity is > 1, servers are always busy and customers always need to wait, and a certain fraction will abandon. By FCFS, customers of different types all have the same waiting time distribution, and the system demonstrates global FCFS (this term was coined by Talreja & Whitt, 2007). The system is stabilized by abandonments. The quality of service parameter here is the target waiting time $W$, so that customers with patience less than $W$ do not get served, while customers with patience that exceeds $W$ get served after a wait of $W$. This means that approximately a fraction $F_i(W)$ of customers of type $c_i$ will abandon. Since customers are thinned independently by impatience, we need to calculate the total effective arrival rate (of patient customers) and we need to use the adjusted fractions $\tilde{\alpha}_c$ of customer types entering service in formula (2) in the algorithm for ED below.

Algorithm for ED

Input:

- Compatibility graph $G$
- Arrival rate $\lambda$
- Fractions of customer types $\alpha_i$
- Patience distributions $F_i(.)$
- Mean service times $m_{i,s_j}$

Requested quality of service parameter:

- Target customer waiting time $W$

Design parameters:

- Fraction of services performed by each server type $\beta_i$

Algorithm:

- Compute expected fraction of abandonments
  
  $p_c := F_c(W)$
  
- Compute the total effective arrival rate
  
  $\tilde{\lambda} := \sum_{i \in C(s)} \alpha_i \cdot (1 - p_c)$
  
- Adjust fraction of each customer type
  
  $\tilde{\alpha}_c := \frac{\tilde{\lambda} \cdot \tilde{\alpha}_c}{\sum_{i \in C(s)} \tilde{\lambda} \cdot \tilde{\alpha}_c}$

- Check $\tilde{\alpha}, \tilde{\beta}$ for complete resource pooling
  
  Compute matching rates
  
  $r_{c,s} := \text{use Eq. (2)}$
  
  Compute staffing levels
  
  $n_i := \frac{\tilde{\lambda} \cdot \tilde{\alpha}_c}{\sum_{i \in C(s)} r_{c,s} \cdot m_{i,s_j}}$

Output:

- Required workforce $n_i$

5.4. Design for quality and efficiency driven service

Given the arrival rates, there is a unique FCFS system that will supply QED service, with servers almost always busy, most customers either do not wait or wait a short time, and few abandonments. The calculation of QED design follows the same steps as for QD with $T = 0$ and for ED with $W = 0$.

6. Examples and simulation of pooled service designs

In this section, we describe two examples, for each of which we have prepared several pooled service designs, under several modes of operation, and assuming Poisson arrivals using a range of values for $\lambda$. We have restricted the examples to Poisson arrivals to keep the presentation manageable and also because experiments confirm that the same conclusions remain valid for general renewal streams. We have then performed extensive simulation runs on each of these designs. Our purpose in this section is threefold:

(i) Illustrate the implementation of the algorithms;
(ii) Examine the validity of the matching rates conjecture;
(iii) Evaluate the efficacy of our designs.

The designs depend on the service rates for each link, but not on the actual distributions of service times. To examine the validity of the matching conjecture and to assess the efficacy of the designs, we have chosen to simulate service time distributions that are very different, including uniform in a finite range, exponential and Pareto.

The first example system has 3 customer types and 3 server types with an almost complete bipartite compatibility graph. We have designed operation of this system with complete resource pooling, in ED, QD and QED mode. The purpose of this example is to assess pooled service designs. This example is important, because it is for this system topology that Foss and Chernova (1998) have shown that calculation of exact matching rates depends on the full shape of processing time distributions and is intractable.

The second example has 6 customer types and 6 server types with a symmetric compatibility graph that has degree 3 for all nodes. The purpose of considering this example is to examine validity of the matching conjecture in a complex graph. In the last sub-section, we study the impact of the patience time distribution for this example.

Our main conclusions from the simulations of these examples are:

(i) The matching rates conjecture seems to be valid under ED, QD and QED mode, for the whole range of $\lambda$ values, and under all the different distributions of service times. For small values of $\lambda$ the deviations are slightly larger. This can be partly explained by the fact that the algorithm yields real numbers for $n_i$, while in the implemented design rounded integer values are used.

(ii) In ED mode, for large values of $\lambda$ we get convergence to the exact values of $W$ with very small variability in waiting times, and exact abandonments. Similarly, under QD, for large values of $\lambda$ we get convergence to the exact values of $T$ with very small variability in idle times, and almost all customers are not waiting for service.

(iii) Most important, it seems that for small values of $\lambda$, while waiting times in ED mode and idle times in QD mode are quite variable, the average waiting time in ED and the average idle time in QD are very close to their targets. This indicates that our design heuristic may be effective already for a moderate number of servers.

(iv) Convergence in the QED mode is not appreciably worse than in the ED or QD modes.
The results do not seem to depend on the service time distributions.

We now present the examples with detailed simulation results. The reported simulation results for each design have been obtained as the average of 1,000 runs, where each run consists of 1,250,000 customers. However, the first 250,000 customers have been removed from the results to account for a possible startup effect. This set-up produces simulation results that are with 95% confidence accurate up to at least three digits, as presented in the tables.

6.1. Example 1 – 3 × 3 Almost complete graph with pooled service

The system studied in this example is specified below, where \( \text{Exp}(a) \) denotes the exponential distribution with rate \( a \), \( U(a, b) \) is the uniform distribution on the interval \((a, b)\) and Pareto\((k, a)\) is the Pareto distribution \( F(t) = 1 - (k/t)^a \) for \( t > k \).

\[
\begin{align*}
2\lambda & \quad 5\lambda & \quad 3\lambda \\
(c_1) & \quad (c_2) & \quad (c_3)
\end{align*}
\]

The patience times and service time distributions are given in the tables below.

<table>
<thead>
<tr>
<th>Patience time distributions</th>
<th>( F_{c_1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>\text{Exp}(0.1)</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>\text{U}(0, 10)</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>\text{Exp}(0.2)</td>
</tr>
</tbody>
</table>

The service time distributions are as follows:

<table>
<thead>
<tr>
<th>Service time distributions</th>
<th>( m_{c_1, s_1} )</th>
<th>( m_{c_2, s_1} )</th>
<th>( m_{c_3, s_1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>Pareto(2, 3)</td>
<td>\text{Exp}(0.125)</td>
<td>\text{Pareto}(2, 3)</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>\text{Exp}(0.2)</td>
<td>\text{U}(2, 6)</td>
<td>\text{U}(2, 6)</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>\text{Pareto}(3, 3)</td>
<td>\text{U}(1, 5)</td>
<td>\text{U}(1, 5)</td>
</tr>
</tbody>
</table>

Only the mean service times are used by the design algorithms. The full distributions are used in the simulations.

In the designs for Example 1 we take as service fractions: \( \beta_{c_1} = 0.3, \beta_{c_2} = 0.3, \beta_{c_3} = 0.4 \).

**ED design:** We specify the target waiting time \( W = 1 \), corresponding to approximately 25% of the mean service times. For the given patience distributions this entails abandonment rates of approximately 10% for customers of types \( c_1 \) and \( c_2 \), and of 18% for customers of type \( c_3 \). We calculate the effective arrival rates of customers that do get served after a wait of \( W = 1 \):

\[
1 - F_{c_1}(W) = e^{-0.1W} = 0.905, \quad 1 - F_{c_2}(W) = (10 - W)/10 = 0.9, \quad 1 - F_{c_3}(W) = e^{-0.2W} = 0.819.
\]

so \( \lambda_{c_1}(1 - F_{c_1}(W)) = 0.2\lambda \times 0.905 = 0.181\lambda \), \( \lambda_{c_2}(1 - F_{c_2}(W)) = 0.450\lambda \), \( \lambda_{c_3}(1 - F_{c_3}(W)) = 0.246\lambda \), and thus the effective arrival rate equals \( \lambda = (0.181 + 0.450 + 0.246)\lambda = 0.877\lambda \).

**QD design:** We take a target idle time of \( T = 0.5 \). This corresponds to a utilization of approximately 0.9.

**QED design:** The unadjusted values of \( \lambda, \alpha_{c_i}, m_{c_i, s_j} \) are used.

It is readily verified that in all three regimes (ED, QD and QED), Conditions (1) are satisfied, so complete resource polling holds. From the algorithms we obtain the calculated required workforce for the three designs shown in Table 1 (where we rounded the \( n_j \)'s to the nearest integers).

The simulation results for Example 1 are listed in Tables 2–3 and in Fig. 6. We note that the histograms in Fig. 6 only depict the waiting times and idle times greater than zero (so probability mass at zero is not shown).

Table 1 shows that the theoretical matching rates calculated by the algorithm are quite close to the simulated (actual) matching rates, already for moderate values of \( \lambda \). The results for the waiting times and idle times confirm our intuition that they should converge to the targeted quality of service requirements: for large values of \( \lambda \), the probability mass of the waiting times in ED con-
Table 2
Simulated matching rates for Example 1.

<table>
<thead>
<tr>
<th></th>
<th>ED regime</th>
<th>QED regime</th>
<th>QD regime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$c_1$</td>
<td>$c_2$</td>
<td>$c_3$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>0.038</td>
<td>0.262</td>
<td>0.042</td>
</tr>
<tr>
<td>$s_2$</td>
<td>0.251</td>
<td>0.049</td>
<td>0.242</td>
</tr>
<tr>
<td>$s_3$</td>
<td>0.168</td>
<td>0.232</td>
<td>0.158</td>
</tr>
</tbody>
</table>

$\lambda = 20$

<table>
<thead>
<tr>
<th></th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>0.046</td>
<td>0.262</td>
<td>0.048</td>
<td>0.260</td>
<td>0.047</td>
<td>0.258</td>
<td>0.242</td>
<td>0.241</td>
<td>0.236</td>
</tr>
<tr>
<td>$s_2$</td>
<td>0.241</td>
<td>0.056</td>
<td>0.239</td>
<td>0.064</td>
<td>0.234</td>
<td>0.153</td>
<td>0.238</td>
<td>0.236</td>
<td>0.236</td>
</tr>
<tr>
<td>$s_3$</td>
<td>0.164</td>
<td>0.230</td>
<td>0.155</td>
<td>0.237</td>
<td>0.155</td>
<td>0.238</td>
<td>0.242</td>
<td>0.242</td>
<td>0.242</td>
</tr>
</tbody>
</table>

$\lambda = 60$

<table>
<thead>
<tr>
<th></th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>0.041</td>
<td>0.261</td>
<td>0.045</td>
<td>0.258</td>
<td>0.045</td>
<td>0.257</td>
<td>0.242</td>
<td>0.243</td>
<td>0.238</td>
</tr>
<tr>
<td>$s_2$</td>
<td>0.248</td>
<td>0.051</td>
<td>0.239</td>
<td>0.061</td>
<td>0.237</td>
<td>0.155</td>
<td>0.242</td>
<td>0.243</td>
<td>0.238</td>
</tr>
<tr>
<td>$s_3$</td>
<td>0.167</td>
<td>0.232</td>
<td>0.156</td>
<td>0.237</td>
<td>0.155</td>
<td>0.238</td>
<td>0.242</td>
<td>0.242</td>
<td>0.242</td>
</tr>
</tbody>
</table>

$\lambda = 200$

<table>
<thead>
<tr>
<th></th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>0.039</td>
<td>0.261</td>
<td>0.043</td>
<td>0.259</td>
<td>0.043</td>
<td>0.258</td>
<td>0.242</td>
<td>0.242</td>
<td>0.242</td>
</tr>
<tr>
<td>$s_2$</td>
<td>0.250</td>
<td>0.049</td>
<td>0.242</td>
<td>0.059</td>
<td>0.242</td>
<td>0.059</td>
<td>0.242</td>
<td>0.242</td>
<td>0.242</td>
</tr>
<tr>
<td>$s_3$</td>
<td>0.168</td>
<td>0.232</td>
<td>0.157</td>
<td>0.240</td>
<td>0.157</td>
<td>0.241</td>
<td>0.242</td>
<td>0.242</td>
<td>0.242</td>
</tr>
</tbody>
</table>

Fig. 6. Histograms of the customer waiting times and server idle times, for each customer/server type separately (probability mass at zero is not shown), for Example 1.
centrates near \( W \) and the probability mass of the idle times in QD concentrates near \( T \). In the QED regime, waiting times, idle times and abandonment rates are small (Tables 3 and 4).

6.2. Example 2 – 6 \( \times \) 6 Symmetric degree 3 graph with pooled service

We now consider a more complex graph to examine the validity of the matching rates conjecture.

<table>
<thead>
<tr>
<th>Example 2 – System and Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>There are 6 types of customers and 6 types of servers. The total arrival rate is parameterized by ( \lambda ). The graph and the values of ( \lambda_i / \lambda ) are described in the following figure:</td>
</tr>
<tr>
<td><img src="image" alt="Graph" /></td>
</tr>
<tr>
<td>( \lambda_i / \lambda )</td>
</tr>
<tr>
<td>( c_1 )</td>
</tr>
<tr>
<td>( c_2 )</td>
</tr>
<tr>
<td>( c_3 )</td>
</tr>
<tr>
<td>( c_4 )</td>
</tr>
<tr>
<td>( c_5 )</td>
</tr>
<tr>
<td>( c_6 )</td>
</tr>
<tr>
<td>The patience times are all exponentially distributed with mean 10. The service times distributions are given in the table below.</td>
</tr>
<tr>
<td><img src="image" alt="Table" /></td>
</tr>
<tr>
<td><img src="image" alt="Table" /></td>
</tr>
<tr>
<td>Only the mean service times are used by the design algorithms. The full distributions are used in the simulations.</td>
</tr>
</tbody>
</table>

In the designs for Example 2 we take as service fractions: \( \beta_j = 1/6, \ j = 1, \ldots , 6 \). It then follows, by checking Conditions [1], that in each regime (ED, QED and QD) service is pooled. We take \( W = 1 \) in the ED regime and \( T = 0.5 \) in the QD regime. The calculated workforce (where we again rounded the \( n_i \), s to the nearest integers) is given in Table 5, while the matching rates are given in Table 6. Note that these matching rates are the same for all three regimes (ED, QED and QD). This is caused by the fact that all customers have the same patience distribution and the same target waiting times, resulting in equal \( \alpha_i \) for all three regimes. The simulated matching rates are also practically identical to the theoretical ones for all three regimes. In the tables, we depict the averages over the three regimes, but the actual differences between the three simulated values and their averaged
values are less than 0.001. Simulation results for the fraction of customers that do not have to wait and the fraction of idle times that are equal to zero are given in Table 7.

Finally, we depict the abandonment rates in Table 8. Due to the symmetry in the system, we only report abandonment rates for customer types C1 and C2, because the results for customer types C3 and C4 are identical to those of customer type C1, and those of customer types C5 and C6 are identical to those of type C2.

The expected abandonment rates are 0.095 for every customer type (due to symmetry) in the ED regime and zero in the QED and QD regimes. The pattern here is as expected: for $\lambda = 20$ the simulated abandonment rates are quite close to the target values, in particular in the ED and QD regimes. For $\lambda = 200$ we are also very close to achieving our target abandonment rates in the QED regime. In this example, we have taken all patience times to be exponentially distributed with mean 10, which is quite large compared to the target waiting times (no waiting in the QD and QED regimes, and $W = 1$ in the ED regime). In the next sub-section, we conduct a more in-depth study of the impact of the patience time distributions on the abandonment rates.

6.3. Impact of the patience time distribution

So far, we have not studied the impact of the distribution of the customer patience time. In order to gain more insight in this aspect, we take the setting of Example 2 because of its symmetry. This makes it suitable for measuring the impact of the patience time distribution on the various performance measures. Customers of types C1 and C2 have exponentially distributed patience times; types C3 and C4 have a uniform distribution; types C5 and C6 have a Pareto distribution. The means of these patience times are 2.5 for customer types C1, C2, C5 and C6. Table 9 gives a detailed overview of the distributions and the parameters. We also show the expected fractions of abandonments in the ED regime (with $W = 1$). Recall that these expected abandonment rates are zero in the QED regime. For this reason, the required numbers of servers in the QED regime do not depend on the patience time distribution, implying they are identical to the numbers listed in Table 5. The staffing levels in the ED regime are slightly different due to the different distributions. For $\lambda = 20$ these differences are still very small (difference of at most one server), but for $\lambda = 200$ the number of servers of each type are typically smaller than in the previous sub-section:

$$n_{c1} = 115, n_{c2} = 91, n_{c3} = 110, n_{c4} = 85, n_{c5} = 113, n_{c6} = 84.$$

This can be explained by the fact that the abandonment rates for most customer types are much higher in this example than in the previous example. We have deliberately chosen higher values to emphasize the impact of the patience time distributions.

We ran the same number of simulations ($1000 \times 1,000,000$ matches) as before. The simulation results (which we have omitted for reasons of compactness) clearly indicate that the different patience time distributions do not have an impact on the simulated matching rates nor on the mean waiting times, which are still extremely close to the theoretical values – even for small $\lambda$. However, Table 10, with simulated fractions of abandonments, shows some interesting patterns. First, we observe that the simulated arrival rates clearly converge towards their target values. Interestingly, this convergence is slower in those cases where we wish to achieve zero abandonments in the ED regime. The ED regime is not primarily designed to favor customers and a target of 0% abandonments is difficult to achieve in this regime. The second interesting observation, which is in contrast to the previous remark, is that the uniform and Pareto distribution make it relatively easy to achieve 0% abandonments in the QED regime. The reason for this phenomenon is that the support of both distributions has a positive offset. As $\lambda$ increases, the tail of the waiting time distribution becomes so light, that no customer experiences a waiting time that is longer than the smallest possible patience. All in all we can conclude that:

- Customer waiting times and server idle times are (nearly) insensitive to the patience time distributions;
- Abandonment rates are close to their target values irrespective of the patience time distributions, in particular for large $\eta$;
- Abandonment rates of 0% are difficult to achieve in the ED regime, but easy to achieve in the QED regime if the support of the patience time distribution has a positive offset. For
completeness, we stress that the QD regime always results in zero abandonments, even for small $\lambda$.

7. Designs with differentiated service levels

In this section, we discuss the case of decomposition into several classes with different priorities, where the mode of operation and the quality of service parameters are specified for each subclass separately.

### 7.1. General strategy

The main goal is to give customers differentiated service levels, from high priority to standard priority to low priority customer types. We present a design algorithm that will result in a decomposition of the system into sub-systems, each with its own performance characteristics while still using the global FCFS-ALIS policy. The priorities will be translated into quality of service parameters for each of the sub-systems: for low priority customers we have sub-systems operating in the ED regime, but with smaller target waiting times for higher priority customers (within these sub-clases of low priority customers). Similarly, for high priority customers we can design the sub-systems to be in QD mode, but with even larger target server idling times for those with the highest priority. Finally, if desired, there is the option to have one customer class in the QD regime. We now discuss the technical details about how to design these systems with differentiated service.

We have a partition of customer types $C$ into $C^{(1)}, \ldots, C^{(L)}$, where customers of types $C_j \in C^{(1)}$ have highest priority and customers of types $C_j \in C^{(L)}$ have lowest priority. Customers in class $C^{(l)}$ will then have a set of servers $S^{(l)}$, so that each customer type $C_j \in C^{(l)}$ will have at least one compatible server type $s_j \in S^{(l)}$. In this decomposition of sub-systems ($C^{(1)}, S^{(1)}$), we will allow $s_j \in S^{(l)}$ to serve $C_j \in C^{(k)}$, $k \geq l$, but will not allow $s_j \in S^{(l)}$ to serve $C_j \in C^{(k)}$, $k < l$. In other words, we redesign the compatibility graph $G$ by eliminating all links from $S^{(l)}$ to $C^{(k)}$ for $k < l$, but we preserve the links to higher priority customers in $C^{(k)}$ for $k > l$, since when we use FCFS-ALIS, these links will hardly ever be used, because servers in $S^{(k)}$ will be behind all servers in $S^{(k)}$ for $k > l$ almost all the time.

In terms of the quality of service parameters, this translates to specifying target customer waiting times $W_1 > W_2 > \cdots > W_L > 0$ for classes $C^{(1)}, C^{(2)}, \ldots, C^{(L)}$ in ED mode, and specifying target server idle times $T_1 > \cdots > T_{L+2} > T_{L+1} > 0$ for customer classes in QD mode. It is possible to have (at most) one class in QED mode, say $C^{(l)}$, which could be achieved by setting $T_i = W_i = 0$ for that class.
Algorithm for differentiated service

Input:
• Compatibility graph \( \mathcal{G} \)
• Arrival rate \( \lambda \)
• Fractions of customer types \( \alpha_i \)
• Patience distributions \( f_i(\cdot) \)
• Mean service times \( m_{s_i} \)

Requested quality of service parameters:
• Partition of customer types by priority into \( c^{(1)}, \ldots, c^{(k)} \)
• Quality of service parameters:
  • \( W_1 > W_2 > \ldots > W_{k-1} > 0 \) for the ED sub-systems (if any)
  • \( 0 < T_{11} < \ldots < T_{1j} \) for the QD sub-systems (if any)
  • \( T_i - W_i = 0 \) for the QED sub-system (if any)

Design parameters:
• Choose partition of server types \( S^{(1)}, \ldots, S^{(k)} \)
• Eliminate links from \( S^{(m)} \) to \( c^{(k)} \), for \( k \leq m \)
• Assign fraction of services performed by each server type \( \beta_i \), within \( S^{(m)} \)

Algorithm:
• Apply appropriate design algorithm for sub-system \( (c^{(k)}, S^{(k)}) \):
  • ED for \( k \in \{1, \ldots, \ell - 1\} \)
  • QED for \( k = \ell \)
  • QD for \( k \in \{\ell + 1, \ldots, \ell + \ell'\} \)

Output:
• Redesigned compatibility graph \( \mathcal{G} \)
• Required workforce \( n_i \)

7.2. Example 3 – 5 \( \times \) 5 Hamiltonian graph with differentiated service

This example illustrates differentiated service, with the customer types divided into three classes: high, standard and low priority customers.

Example 3 – System and Data

There are 5 types of customers and 5 types of servers. The total arrival rate is parameterized by \( \lambda \). The graph and the values of \( \alpha_i, \lambda \) are described in the following figure:

The fractions \( \alpha_i \) are all equal, i.e., \( \alpha_i = \frac{1}{5} \). The patience times are all exponentially distributed with mean 10. The service times are all uniformly distributed, with parameters as given in the table below:

<table>
<thead>
<tr>
<th>Service time distributions</th>
<th>( G_{c_i, s_j} )</th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( c_3 )</th>
<th>( c_4 )</th>
<th>( c_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>U(2, 6)</td>
<td>U(2, 4)</td>
<td>U(4, 7)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( s_2 )</td>
<td>U(1, 3)</td>
<td></td>
<td>U(3, 6)</td>
<td>U(2, 6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( s_3 )</td>
<td>U(3, 7)</td>
<td></td>
<td>U(1, 5)</td>
<td>U(6, 11)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean service times</th>
<th>( G_{c_i, s_j} )</th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( c_3 )</th>
<th>( c_4 )</th>
<th>( c_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( s_2 )</td>
<td>2</td>
<td>5.5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( s_3 )</td>
<td>4.5</td>
<td>4</td>
<td>3</td>
<td>8.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( s_4 )</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.5</td>
</tr>
</tbody>
</table>

Only the mean service times are used by the design algorithms. The full distributions are used in the simulations.

We consider the following decomposition of the system of Example 3, from high to low priority:

\( c^{(3)} = \{c_1, c_2\} \), \( S^{(3)} = \{s_1\} \), \( c^{(2)} = \{c_3, c_4\} \), \( S^{(2)} = \{s_2, s_3\} \), \( c^{(1)} = \{c_5\} \), \( S^{(1)} = \{s_4, s_5\} \).

The decomposed system is described in Fig. 7. Note that this decomposition results from eliminating three links in the compatibility graph: the link from \( s_2 \) to \( c_2 \), \( s_4 \) to \( c_4 \), and from \( s_5 \) to \( c_1 \). We then have that:

- \( c^{(3)}, S^{(3)} \) in isolation is a “V” system, with arrival rate 0.4\( \lambda \), adjusted fractions \( \alpha_{c_1} = \alpha_{c_2} = \frac{1}{2} \) and \( \beta_{s_1} = 1 \).
- \( c^{(2)}, S^{(2)} \) in isolation is an “N” system, with arrival rates 0.4\( \lambda \), adjusted fractions \( \alpha_{c_3} = \alpha_{c_4} = \frac{1}{2} \) and we take \( \beta_{s_2} = \frac{1}{2}, \beta_{s_3} = \frac{1}{2} \).
- \( c^{(1)}, S^{(1)} \) in isolation is a “Lambda” system, with arrival rate \( \lambda = 0.2\lambda \), \( \alpha_{c_5} = 1 \) and we take \( \beta_{s_4} = \beta_{s_5} = \frac{1}{2} \).

We note that the sub-networks have a simple compatibility structure, so that matching rates are obtained immediately as:

\[
\begin{align*}
\beta_{s_1} &= 1, \\ \beta_{s_2} &= \frac{1}{3}, \\ \beta_{s_3} &= \frac{2}{3}, \\ \beta_{s_4} &= \frac{1}{2}, \\ \beta_{s_5} &= \frac{1}{2}.
\end{align*}
\]
Table 11
Calculated required workforce for Example 3.

<table>
<thead>
<tr>
<th>Design</th>
<th>ED regime</th>
<th>Mixed regime</th>
<th>QD regime</th>
</tr>
</thead>
<tbody>
<tr>
<td>W = 1</td>
<td>$W_1$</td>
<td>$W_2$</td>
<td>$W_3$</td>
</tr>
<tr>
<td>$n_{x_1}$</td>
<td>$n_{x_2}$</td>
<td>$n_{x_3}$</td>
<td>$n_{x_4}$</td>
</tr>
<tr>
<td>$n_{y_1}$</td>
<td>$n_{y_2}$</td>
<td>$n_{y_3}$</td>
<td>$n_{y_4}$</td>
</tr>
<tr>
<td>$n_{z_1}$</td>
<td>$n_{z_2}$</td>
<td>$n_{z_3}$</td>
<td>$n_{z_4}$</td>
</tr>
<tr>
<td>$n_{w_1}$</td>
<td>$n_{w_2}$</td>
<td>$n_{w_3}$</td>
<td>$n_{w_4}$</td>
</tr>
</tbody>
</table>

Table 12
Simulation results of the matching rates for Example 3.

<table>
<thead>
<tr>
<th>Matching rates</th>
<th>ED regime</th>
<th>Mixed regime</th>
<th>QD regime</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{c_1,s_1}$</td>
<td>$r_{c_2,s_1}$</td>
<td>$r_{c_3,s_1}$</td>
<td>$r_{c_4,s_1}$</td>
</tr>
<tr>
<td>$r_{c_2,s_2}$</td>
<td>$r_{c_3,s_2}$</td>
<td>$r_{c_4,s_2}$</td>
<td>$r_{c_5,s_2}$</td>
</tr>
<tr>
<td>$r_{c_3,s_3}$</td>
<td>$r_{c_4,s_3}$</td>
<td>$r_{c_5,s_3}$</td>
<td></td>
</tr>
<tr>
<td>$r_{c_4,s_4}$</td>
<td>$r_{c_5,s_4}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{c_5,s_5}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8. System design when fraction of servers of each type is specified

As discussed before, there is an alternative way to determine the staffing levels, which does not require an explicit choice of the $\beta_j$'s. In some practical cases it may be more natural to specify the desired fractions of the total number of servers of each type instead, denoted by

$$\theta_j = (\theta_{j_1}, \theta_{j_2}, \ldots, \theta_{j_n}) = (n_{j_1}, \ldots, n_{j_n})/n.$$  

In this section, we consider staffing decisions made on the basis of specification of the $\theta_j$'s.

In Section 5 we illustrated how to obtain the vector of values $(n_{j_1}, \ldots, n_{j_n})$ for given values of $\beta_j = (\beta_{j_1}, \ldots, \beta_{j_n})$, which also determines $(\theta_{j_1}, \ldots, \theta_{j_n})$. We therefore have a function $H: \mathbb{R}^n \rightarrow \mathbb{R}^n$, which calculates $(\theta_{j_1}, \ldots, \theta_{j_n}) = H(\beta_{j_1}, \ldots, \beta_{j_n})$. We let $H(\beta) = H(\beta_{j_1}, \ldots, \beta_{j_n})$ denote the jth element of $H$. We now need to perform the inverse calculation, of $(\beta_{j_1}, \ldots, \beta_{j_n}) = H^{-1}(\theta_{j_1}, \ldots, \theta_{j_n})$. We do not know whether $H$ is one-to-one, so $H^{-1}$ may be multivalued, in which case we would like to find just one inverse vector. To obtain such a vector we consider, for given $\theta_1, \ldots, \theta_j$ and some

$$r_{c_1,s_1}^{(3)} = r_{c_2,s_1}^{(3)} = \frac{1}{2}, \quad r_{c_2,s_2}^{(3)} = r_{c_3,s_1}^{(3)} = r_{c_4,s_1}^{(3)} = \frac{1}{3}, \quad r_{c_3,s_2}^{(3)} = r_{c_5,s_1}^{(3)} = \frac{1}{2}.$$  

We make three designs for this network, in which customers $c_1, c_2$ have high priority, $c_3, c_4$ have standard priority, and $c_5$ have low priority:

1. The first design has all sub-systems in the ED regime. Priorities are translated to longer or shorter target waiting times;
2. The second design is for a mixed design with the top priority sub-system in QD, the middle priority sub-system in QED and low priority sub-system in ED;
3. The third design is for all three sub-systems in the QD regime. Different priorities are now achieved by varying the target server idle times.

It is readily verified, by checking Conditions (1), that in each design complete resource pooling holds for each sub-system. Table 11 shows the calculated workforce for the three designs, as a function of $\lambda$.

The simulation results for Example 3 are listed in Tables 12 and 13, and in Fig. 8. The results illustrate that the systems with differentiated service under FCFS-ALIS perform in accordance with the design, for all three design types (ED, mixed, QD).
Table 13
Simulated fractions of customers that do not wait and the fractions of service completions that are not followed by idleness, for Example 3.

<table>
<thead>
<tr>
<th>λ</th>
<th>No waiting</th>
<th>No idling</th>
<th>No waiting</th>
<th>No idling</th>
<th>No waiting</th>
<th>No idling</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.056</td>
<td>0.934</td>
<td>0.578</td>
<td>0.398</td>
<td>0.869</td>
<td>0.124</td>
</tr>
<tr>
<td>60</td>
<td>0.010</td>
<td>0.988</td>
<td>0.570</td>
<td>0.412</td>
<td>0.935</td>
<td>0.063</td>
</tr>
<tr>
<td>200</td>
<td>0.000</td>
<td>1.000</td>
<td>0.556</td>
<td>0.430</td>
<td>0.974</td>
<td>0.025</td>
</tr>
</tbody>
</table>

Customer waiting times and server idle times

<table>
<thead>
<tr>
<th>ED regime Waiting times</th>
<th>Mixed regime Waiting times</th>
<th>Mixed Regime Idle times</th>
<th>QD regime Idle times</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_{c1}$</td>
<td>$T_{s1}$</td>
<td>$T_{s2}$</td>
<td>$T_{s3}$</td>
</tr>
<tr>
<td>$W_{c2}$</td>
<td>$T_{s4}$</td>
<td>$T_{s5}$</td>
<td></td>
</tr>
<tr>
<td>$W_{c3}$</td>
<td>$T_{s6}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$W_{c4}$</td>
<td>$T_{s7}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$W_{c5}$</td>
<td>$T_{s8}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 8. Simulation results: histograms of the customer waiting times and server idle times, for all customer and server types and all three designs for Example 3.
proposed \( \beta_{i_1}, \ldots, \beta_{i_j} \), the squared sum of differences:

\[
\Delta = \sum_{j=1 \ldots J} (H^T(\beta_{i_1}, \ldots, \beta_{i_j}) - \theta_{i_j})^2.
\]  

(5)

For given \( (\theta_{i_1}, \ldots, \theta_{i_j}) \) we wish to find \( (\beta_{i_1}, \ldots, \beta_{i_j}) \) that will minimize \( \Delta \) or that will solve \( \Delta = 0 \). Numerical solution of this problem can be performed by a wide choice of minimization software. We illustrate the solution for the examples of Section 6 below.

8.1. Example 1 revisited: calculations for given \( n_s/n \)

We revisit Example 1 from Section 6.1 and Example 2 from Section 6.2. We note that in Example 3 from Section 7.2, the one to one relation between \( \beta_{i_1} \) and \( \theta_{i_1} \) is immediate and does not require numerical minimization.

In this example, we consider the same system with three customer types and three server types. In Section 6 we have chosen settings that ensure complete resource pooling, namely \( (\beta_{i_1}, \beta_{i_2}, \beta_{i_3}) = (0.3, 0.3, 0.4) \). Instead of specifying \( (\beta_{i_1}, \beta_{i_2}, \beta_{i_3}) \) we now specify desired fractions of server types \( (\theta_{i_1}, \theta_{i_2}, \theta_{i_3}) \), distinguishing between three cases:

- \( \theta_1 = \theta_2 = \theta_3 = 1/3 \), i.e., all fractions are equal;
- \( \theta_1 = (1/6, 1/3, 1/2) \), i.e., half of the servers should be of type \( s_2 \), one third of type \( s_2 \), and the rest of type \( s_1 \);
- \( \theta_1 = (1/2, 1/3, 1/6) \), i.e., half of the servers should be of type \( s_1 \), one third of type \( s_2 \), and the rest of type \( s_3 \).

In order to find a vector \( \beta_{i_1} \) that results in the desired \( \theta_{i_1} \), we numerically minimize the function \( \Delta \) as defined in Eq. (5). When minimizing \( \Delta \), we impose additional restrictions (1) to ensure complete resource pooling. For this simple network, these restrictions can be written as follows:

\[
\text{QD, QED}: 0 < \beta_{i_1} < 0.7, \ 0.5 < \beta_{i_1} + \beta_{i_2} < 1, \ 0 < \beta_{i_3} < 0.8;
\]

\[
\text{ED}: 0 < \beta_{i_1} < 0.720, \ 0.513 < \beta_{i_1} + \beta_{i_2} < 1, \ 0 < \beta_{i_3} < 0.794.
\]

The results of the numerical routine can be found in Table 14 for all three regimes. The first column gives the vector \( \theta_{i_1} \) of desired fractions of servers, the second column indicates whether these desired fractions could actually be attained. The third column gives the vector \( \beta_{i_1} \) that minimizes Eq. (5), and the last column displays the required staffing levels of each server type for \( \lambda = 100 \).

Interestingly, the desired fractions \( n_s/n \) for \( j = 1, 2, 3 \) cannot be achieved for the case where \( \theta_1 = (1/6, 1/3, 1/2) \). In all three regimes, the global minimum of the function \( \Delta \) is not attained within the region of complete resource pooling. This is illustrated in Fig. 9, where contour plots for \( \Delta \) as a function of \( \beta_{i_1} \) and \( \beta_{i_2} \) (with \( \beta_{i_3} = 1 - \beta_{i_1} - \beta_{i_2} \)) are shown for the three different \( \theta_{i_1} \) vectors, for the ED regime. The plots for the QD and QED regimes are omitted, as they are similar.

Since the global minimum of \( \Delta \) for the desired vector \( \theta_1 = (1/6, 1/3, 1/2) \) is located outside the boundaries of the complete resource pooling region, we find that \( \Delta > 0 \) for the vector \( \beta_{i_1} \) that minimizes \( \Delta \) within the complete resource pooling region. As a consequence, the relative staffing levels \( n_s/n \) resulting from the vector \( \beta_{i_1} \) differ from the desired \( \theta_{i_1} \). For example, in the ED regime the recommended staffing levels are \( (88, 171, 155) \), with relative values \( (0.21349, 0.411097, 0.375413) \), whereas the desired \( \theta_1 = (1/6, 1/3, 1/2) \). We now consider the \( 6 \times 6 \) symmetric degree 3 graph from Example 2, with pooled service. As illustration, we distinguish between three combinations of \( \theta_{i_1} \)’s, similar to the previous example:

- \( \theta_1 = \theta_2 = \ldots = \theta_6 = 1/6 \), i.e., all equal,
- \( \theta_1 = (1, 2, 3, 4, 5, 6)/21 \), i.e., increasing in the server type,
- \( \theta_1 = (6, 5, 4, 3, 2, 1)/21 \), i.e., decreasing in the server type.

Using the same numerical optimization routine, we determine the required \( \beta_{i_1} \)’s from which the staffing levels \( n_s / n \) can be determined for the QD, QED, and the ED regime. The results can be found in Table 15. Some interesting conclusions can be drawn from this table. First, we see that all three designs achieve complete resource pooling. This may be explained by the additional flexibility of each server being compatible with 3 types of customers, i.e., large degree of overlap in compatibilities. Second, it is immediately clear that the relative staffing levels \( n_s / n \) resulting from the numerical procedure are all equal to the specified values \( \theta_{i_1} \), meaning that the minimization of \( \Delta \) resulted in a global minimum of 0 satisfying the conditions for complete resource pooling. Third, it can be seen that the vector \( \beta_{i_1} \) that minimizes \( \Delta \) is the same for all three regimes. This can easily be explained from the fact that all patience distributions are the same, meaning that the values of \( \alpha_{i_1} \), \( i = 1 \ldots I \), do not depend on the selected regime and neither does the function \( \Delta \).

9. Discussion

Our purpose in this paper was twofold: To verify a conjecture on matching rates, and to provide a useful tool for design of parallel service systems. We now discuss these two points.

Our computational results seem to indicate that our conjecture on matching rates may be correct. In fact our simulations show much more than that. Our conjecture is that under many server scaling matching rates will converge precisely to those of the infinite matching model. This would in particular imply that for systems operated under our design, when \( n \rightarrow \infty \) the performance should converge to a deterministic limit, in which for ED designs all patient customers wait exactly \( W \), in QD design all servers will
idle exactly $T$, and in QED nobody would idle or wait. Our simulations indicate that this may be true, and leave the question of proof of the limiting result open.

However, the simulations also indicate that this convergence does not require unreasonably large $n$. In fact, for quite realistic values of $\lambda$ and $n$ we find that our designs perform extremely well: The matching rates in the simulations are very close to those predicted by the approximations, but more importantly, the performance of the systems is very close to the required service quality and utilization parameters as specified in the designs.

We have found that in ED mode the fraction of abandonments is almost precisely the pre-specified value, and that the waiting time distribution of patient customers is distributed around the target value $W$ which is the conjectured limiting value for $n \to \infty$. We have also shown that in QD mode the idle times are distributed around the target value $T$ which is the conjectured limiting value for $n \to \infty$. We have also shown that in the QED mode the system works in perfect balance between customers and servers, with complete resource pooling and uniform service level for all customers, with negligible abandonments. All this for the whole range of values of $\lambda$, from 20 to 200.

We note again that our conjecture on matching rates is for general bipartite compatibility graphs, general patience and service time distributions, and general renewal arrivals. Our simulations are of course limited and cannot replace a mathematical proof, and we see no way of making them extensive enough to validate our designs in all situations. However the simulations do cover various bipartite compatibility graphs, a wide range of distributions, and realistic quality of service parameter values, and as can be seen, the results agree with our predictions in all the examples we tried.

It is important to reiterate that these systems under FCFS-ALIS are completely intractable. Without recourse to our matching rate calculations it would be impossible to design a system that would achieve resource pooling and approximate our design parameters. The papers of Foss and Chernova (1998) and Nov, Weiss & Zhang (2016) abundantly illustrate.

Extensive simulation confirmed that the algorithms based on our conjecture are accurate and effective: they produce work force levels and, in case of differentiated service, a redesigned compatibility graph that meets targeted quality of service requirements. Moreover, the heuristic algorithms also appeared to work well when the required work force levels are not so large. As such, these algorithms provide a valuable tool to support decisions on the design of multi-type parallel service systems.

We realize that in real large scale systems there are many specific needs and constraints that will make pure FCFS-ALIS policy impossible to use. However, we believe that even if FCFS-ALIS is not used throughout, it is used for a significant fraction of the scheduling decisions in real large systems. Currently these systems are often evaluated and redesigned based on simulation studies. We believe that our tools that allow reasonably accurate evaluation of system performance under FCFS-ALIS using direct computation, can complement the range of design tools based on simulations, and provide a useful method of design. In particular, our tools can provide quick answers on how system performance will change if we change the compatibility graph, if we redefine service

![Fig. 9. Contour plots of $\Delta$ as a function of $\beta_s$ and $\beta_a$ for Example 1, in the ED regime. The highlighted area indicates the combinations of $\beta_s$ and $\beta_a$ that result in a system with complete resource pooling.](image)
priorities, if we change quality of service parameters, if we reallocate server types, or if we change service rates by redefining tasks.

Much still remains to be done in this stream of research. A proof of the conjecture, in general form or under more limiting assumptions, is the first task. A preliminary attempt to verify the conjecture for the many server N-system in the Poisson-exponential case is presented in Zhan and Weiss (2018). It would also be interesting to evaluate the efficacy of FCFS-ALIS policy in comparison with other policies. Finally, application to a real system with real data will be most illuminating.

Acknowledgments

We are most grateful to the two referees for their very detailed and extensive comments on an earlier version of this paper. Their useful suggestions have substantially improved the paper.

References


Williams, R. J. (2000). On dynamic scheduling of a parallel server system with complete resource pooling. Fields Institute Communications, 28, 5–1
