Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Appointment Scheduling for Multiple Servers

Alex Kuiper, Robert H. Lee

To cite this article:
Alex Kuiper, Robert H. Lee (2022) Appointment Scheduling for Multiple Servers. Management Science

Published online in Articles in Advance 04 Feb 2022

. https://doi.org/10.1287/mnsc.2021.4221

http://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Articles in Advance, pp. 1–19
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

Appointment Scheduling for Multiple Servers

Alex Kuiper,[a,*] **Robert H. Lee**[a]

[a] Department of Business Analytics, Amsterdam Business School, University of Amsterdam, 1001 NL Amsterdam, Netherlands
*Corresponding author
Contact: a.kuiper@uva.nl, https://orcid.org/0000-0001-8408-4018 (AK); r.h.lee@uva.nl, https://orcid.org/0000-0001-6447-7746 (RHL)

Received: February 1, 2020
Revised: August 31, 2020; May 22, 2021
Accepted: July 3, 2021
Published Online in Articles in Advance: February 4, 2022

https://doi.org/10.1287/mnsc.2021.4221

Abstract. Appointment schedules, in essence, balance supply and demand and are often employed in settings where resources are scarce and thus a high utilization is realized (e.g., healthcare). Whereas most of the existing literature focuses on the single-server case, a framework is developed to study appointment scheduling in multiserver settings. Relying on phase-type approximations, general service-time distributions are modeled, which are fed into a recursive approach allowing evaluation and optimization of an objective function that balances expected waiting times and idle times. Studying optimized schedules for multiple servers reveals that the start and end of a session can deviate greatly from the dome-shaped pattern as established for the single-server case. Furthermore, a comparison of various multiserver setups shows that significant performance gains can be achieved when servers are pooled. This allows an explicit quantification of the cost of continuity of care. In addition, session overtime as well as early finish of servers can be incorporated in the approach; the benefits of the additional flexibility that a multiserver setting provides are summarized. For the stationary plateau of the dome, to which the optimal interarrival times converge, steady-state appointment schedules are obtained by exploiting the embedded Markov chain; these schedules are shown and argued to converge quickly to optimal solutions obtained in a heavy-traffic regime. In this regime, algebraic solutions are derived, which provide interesting managerial guidelines when the pooling of servers is considered in appointment scheduling.

History: Accepted by Barış Ata, stochastic models and simulation.
Supplemental Material: Data are available at https://doi.org/10.1287/mnsc.2021.4221.

Keywords: appointment scheduling • parallel servers • pooling • phase-type distributions • heavy-traffic approximations

1. Introduction

Appointment schedules are often used in settings where resources are scarce; for that reason, appointment schedules are used in healthcare. Current literature mainly focuses on the single-server setting. In healthcare, such a setting is often appropriate as continuity of care is obeyed: patients see the same physician during the course of their treatments.

Some settings, however, do not fit well into this framework; these include magnetic resonance imaging (MRI), X-ray facilities, and operating rooms. Each of these cases has multiple resources that are present, and it is logical that the next patient will be served by the first resource to become available. Other examples can be found in legal counselling, technical support appointments, visa application processes, dental hygiene services, and medical rehabilitation services (see, e.g., El-Sharo et al. (2015) and Soltani et al. (2019)). Especially in healthcare, our analysis will demonstrate the benefits when relaxing the continuity of care restraint in situations where multiple service providers are able to provide the same service. In fact, Green et al. (2013) conclude that, to fulfil the growing need of primary care in the near future, the pooling of physicians is inevitable.

The benefits of increased flexibility by pooling resources are assumed reduction of patient or client waiting times and increased utilization. The decision of whether to pool resources, such as MRIs or physicians, is of a *tactical* nature, as it concerns the assignment of clients to resources; see Hulshof et al. (2012) and Ahmadi-Javid et al. (2017).

A complicating factor in appointment scheduling is random service times (Ho and Lau 1992, Cayirli and Veral 2003). This randomness is reflected in a combination of the following:

- Idling resources as a result of having excessive capacity, resulting in *idle time*
- Session overruns as a result of insufficient capacity toward the end of the schedule, creating *overtime*
- Waiting clients as a result of insufficient capacity, resulting in *waiting time*

An appointment schedule tries to find a balance by minimizing a weighted sum of these ramifications of under- and overcapacity.

To define the problem in a mathematical framework, let there be s servers and $n > s$ clients to be scheduled. Let t_i and B_i be the arrival and service times of the ith client, respectively. Assume that each server starts by serving one of the initial s clients, so $t_1 = t_2 = \ldots = t_s = 0$. Thus for each subsequent client $i \in \{s+1, \ldots, n\}$, W_i denotes the waiting time and I_i the idle time, which is the time that resources are idle before the ith client's arrival. In addition, define the overtime O as the time that the session runs over the scheduled session end time T. The objective is to determine a schedule, given by arrival epochs $(t_{s+1}, \ldots, t_n)$ of the $n - s$ subsequent clients, so as to minimize the total expected idle time, waiting time, and overtime over the session. By considering the interarrival time between two appointment epochs $x_{s+i} := t_{s+i} - t_{s+i-1}$, the problem can equivalently be formulated as

$$\min_{(x_{s+1}, \ldots, x_n)} \sum_{i=s+1}^{n} (c_I \mathbb{E}I_i + c_W \mathbb{E}W_i) + c_O \mathbb{E}O, \tag{1}$$

where c_I, c_W, and c_O are cost parameters to be chosen at the discretion of the practitioner.

The computation of these metrics is typically done by considering the evolution of the schedule as a queue. In our case, the resulting queueing system is, using Kendall's notation, a $D/G/s$ queue: deterministic interarrival times (not necessarily uniformly spaced), general service times, and s servers.

Our methodology relies on the fact that service times can be approximated well by mixtures of exponentials—that is, phase-type distributions, wherein each exponential distribution can be thought of as a state of the system. Such a description can be extended to the multiserver setting for which we derive a tractable recursive procedure by exploiting its semi-Markovian nature to keep track of the system, allowing evaluation of the objective function. For optimized multiserver appointment schedules, we find that the resulting interarrival times feature singular patterns that do not appear in the single-server setting. Furthermore, comparing equivalent configurations in the number of servers, the potential gains of pooling in appointment scheduling with random service times are quantified as well as the impact of various environmental and free parameters, such as randomness of the service times, the cost parameters, and the occurrence of no-shows.

2. Literature Review

We divide literature on appointment scheduling into two streams: single-server systems and multiserver systems. The former class has been studied extensively; or comprehensive reviews on these efforts, refer to Cayirli and Veral (2003), Gupta and Denton (2008), and Ahmadi-Javid et al. (2017). Study on multiserver systems has usually been restricted to the domain of multistage settings. Few works have focused on the single-stage, multiserver setting for appointment scheduling (i.e., the $D/G/s$ queue), which is analytically explored in this paper. Next we highlight work that relates to our research.

2.1. Single-Server, Single-Stage Environments

The single-server setting—that is, $s = 1$—is naturally applicable to the single-stage case. This does not, however, constrain the framework from being applicable to a multistage setting—for example, when other stages have more than sufficient capacity not to be a bottleneck (e.g., a reception). In Welch and Bailey (1952), the single-server environment was first formulated. These authors also formulated the well-known Bailey–Welch appointment rule that assigns multiple clients to the first slot to circumvent possible idle time in early stages of the schedule. Ho and Lau (1992) study variations on this appointment rule and find that among important environmental factors affecting the performance of an appointment schedule, the most important are the number of clients to be scheduled, service-time variability, and no-shows.

Another stream of research aside from the study of appointment rules is that of developing methods for finding optimal arrival epochs. An example is the work by Denton and Gupta (2003), who introduce a sequential bounding approach in which the problem is framed as a linear program. Using the L-shaped algorithm, they successively partition the outcome space to approximate an optimal solution. Klassen and Yoogalingam (2009) use simulation in conjunction with optimization to address the single-stage appointment scheduling problem. Another paradigm is to solve this problem over a discrete grid, such as in Kaandorp and Koole (2007), who assume exponential service times. Zacharias and Yunes (2020) show the concept of multimodularity to hold for general stochastic service times, which guarantees the success of efficient optimization algorithms.

A common method to obtain tractability is the use of phase-type distributions (Asmussen et al. 1996), which have proven to provide good levels of accuracy. In the context of appointment scheduling, Wang (1997) is the first work to employ phase-type distributions to derive a recursive system. In the same stream, Bosch and Dietz (2001) use phase-type distributions to analyze the waiting time and overtime over a grid of schedules and show submodularity to ensure convergence. Kuiper et al. (2015) introduce a general method to approximate service times by a phase-type counterpart, allowing computation of relevant queue metrics and facilitating steady-state analyses. They show that it provides good approximations for both the log-normal and Weibull distributions.

Another approach is to discretize time; such an approach is followed by De Vuyst et al. (2014) and Begen and Queyranne (2011) to facilitate evaluation and optimization. Finally, we show that the results of Mak et al. (2015), who study appointment scheduling considering worst-case distributions, are closely related to the results we obtained in steady state.

Focusing on the solutions that these approaches produce, many have reported that the optimized interarrival times depict a *dome-shaped pattern* (Wang 1997, Denton and Gupta 2003, Kaandorp and Koole 2007, Hassin and Mendel 2008, Klassen and Yoogalingam 2009, Kuiper et al. 2015). Appointments early in the session and toward the end are more condensed, whereas in the middle, the interarrival times between appointments are lengthier.

2.2. Multistage Environments

As noted in Cayirli and Veral (2003) and Ahmadi-Javid et al. (2017), the majority of the literature focuses on single-server appointment scheduling. However, multiserver settings are nevertheless prevalent in healthcare. The first extension to consider is the addition of servers in series, creating a multistage environment. For example, a client may first have an X-ray and then have an appointment with a specialist.

Rising et al. (1973) study a system of multiple stages at a university outpatient clinic by means of Monte Carlo simulation. Cox et al. (1985) develop and simulate a queueing model for the multistage setting found in an ear, nose, and throat outpatient clinic. Also relying on simulation, White et al. (2011) study a system in which—besides introducing capacity constraints—they distinguish between two patient types, one of whom requires an X-ray before appointment. In surgery scheduling, Saremi et al. (2013) use simulation optimization in order to address a multistage operating room scheduling problem, incorporating the availability of surgeons.

Another sequential service setting is studied in Zhou and Yue (2019), in which they introduce a stochastic linear program, which is solved by combining a sample average approximation and linear programming (see Denton and Gupta 2003). A two-stage tandem setting is studied analytically in Kuiper and Mandjes (2015). Klassen and Yoogalingam (2019) study, by means of simulation, the clinic's effectiveness when part of the physician's work is taken over in an earlier stage by assistants.

Finally, we mention research that considers systems with multiple stages and servers. Most of this work is case specific and relies on simulation (see, e.g., Côté and Stein (2007)). Another example is the work by Alvarez-Oh et al. (2018), who study a simple system in which patients have to be seen by one of two nurses and then a dedicated provider (single server). Mandelbaum et al. (2020) develop a data-driven robust optimization approach based on uncertainty sets that accommodates a multiserver setting with various patient flows. They apply their model on a cancer center's infusion units and report a 15%–40% reduction of waiting and overtime costs.

2.3. Multiple Parallel Servers

Another extension of the single-server framework is that of servers in parallel. In the specific setting where there is server preference, we refer to section 5.1 in Ahmadi-Javid et al. (2017) and references therein. Here, we focus on the case where clients are indifferent to servers.

Denton et al. (2010) study the problem of scheduling surgeries to multiple operating rooms (parallel servers), where in a first stage it is decided how many servers to open such that in the second stage, surgeries are assigned to specific operating rooms. Once assigned, each operating room acts as a single-server system. El-Sharo et al. (2015) consider a model to decide how many clients should be overbooked to slots in a multiserver setting. For each server, a separate appointment schedule is made. Only when a patient becomes an overflow patient, or a patient is failed to be served in his or her initially assigned slot, will that patient be allocated to any other server.

As these two examples still make a schedule assigned to a specific resource, we find a closer resemblance to our setting in Swisher et al. (2001). They apply discrete event simulation to a clinic to study its performance in a steady state; in this setting, the patient does not go to a specific physician. Furthermore, Harper and Gamlin (2003) study, by means of simulation, the impact of various appointment rules. Sickinger and Kolisch (2009) study an appointment schedule with two computer tomography (CT) scanners. These scanners serve the same queue, which consists of three patient classes—namely, outpatients, for whom the schedule is built, and inpatient and emergency patients, who provide the randomness to be tackled by the design of an appointment schedule. They find that a generalized Bailey–Welch rule performs well.

Zacharias and Pinedo (2017) offset no-shows by providing a recursive method to compute various performance metrics that are optimized by a local search algorithm. Soltani et al. (2019) focus on a legal counselling center where service times are random and model this randomness by matching the first two moments by a discrete service-time distribution, assigning probabilities to multiples of the slot size. More important, as optimization turns out to be computationally intensive, a load-based appointment scheduling heuristic is proposed, which provides a performance increase of 16%.

2.4. Contribution and Organization

Our approach augments the current literature on multiserver appointment scheduling by providing a computational approach in continuous time that incorporates service-time variability and the occurrence of no-shows, which are considered the major sources of variation that affect the performance of an appointment schedule (Ho and Lau 1992, Hassin and Mendel 2008).

Relying on phase-type approximations, we extend the phase-type recursion introduced for the single-server setting (Wang 1997) to the multiserver setting by compressing the state space. For performance measures of interest, such as idle and waiting time, we obtain semianalytical derivations. After optimization, the interarrival times exhibit some striking patterns at the beginning and end of the session that deviate from the dome-shaped pattern reported in literature. Furthermore, the approach enables us to quantify the benefits of pooling in appointment scheduling, which addresses the tactical decision on how to allocate resources—an unchallenged question in the literature on appointment scheduling in healthcare (Ahmadi-Javid et al. 2017).

We extend the work to steady state, which enables the evaluation of the performance gain for large numbers of clients and servers effectively. Interestingly, because appointment schedules are often employed in high utilization environments, we find that the problem cast in the heavy-traffic regime captures the steady state accurately. This insight emphasizes that performance improves by a factor of $\sqrt{s}$ when pooling s servers.

The structure of the rest of this paper is as follows. In Section 3 we state the general scheduling problem for the multiserver setting and show the intrinsic complexity of the problem compared with the single-server setting, and we demonstrate our approach to make the problem tractable. In Section 4 we explain in detail how the phase-type distribution of a system with s parallel servers can be obtained and how it facilitates computation of key performance metrics. In Section 5, we use the methodology to compute optimal schedules for a given number of clients under various settings so that we can study the form of the optimal solution as well as the gain from combining servers. Then in Section 6 we extend our phase-type methodology to the steady state and consider a corresponding heavy-traffic analysis, which enables us to gain better insight into the benefits of pooling. Finally, we conclude in Section 7.

3. Problem Definition

Empirical research reports that patients arrive early more often than late, and therefore it is typical to assume that patients are punctual (Cox et al. 1985, Cayirli and Veral 2003). In the interest of generality, we will henceforth refer consistently to *clients* and *servers*. Furthermore, we restrict our model to identical servers and homogeneous clients except for a short discussion on the ramifications of relaxing these assumptions. We assume that servers start nonempty and that there are n clients to be scheduled. Furthermore, in line with appointment scheduling literature, clients are served according to a first-come, first-served discipline, and there is no preemption nor sharing of servers.

3.1. Single-Server Performance Metrics

In the single-server setting, assuming punctuality, we start a session with the first client to arrive at time $t_1 = 0$. Define the interarrival time between the ith client and his predecessor as $x_i := t_i - t_{i-1}$ for $i = 2, \ldots, n$. Furthermore, let $x_1 = 0$. Then it is standard by the *Lindley* recursion (Lindley 1952) that the waiting times are defined recursively by the following equation:

$$W_i = \max\{B_{i-1} + W_{i-1} - x_{i-1}, 0\} = \max\{S_{i-1} - x_{i-1}, 0\}, \tag{2}$$

where the sojourn time $S_i = W_i + B_i$. Obviously, $S_1 = B_1$, as the first client does not have to wait. The session end time, also known as the *makespan*, is defined as the moment when the final client leaves the system, which is at $t_n + S_n$, which is equal to the sums of idle and service times; that is, $\sum_{i=1}^{n}(B_i + I_i)$. Thus the sum of idle times and overtime are deduced from

$$\sum_{i=1}^{n} I_i = t_n + S_n - \sum_{i=1}^{n} B_i \quad \text{and} \quad O = \max\{t_n + S_n - \mathrm{T}, 0\}, \tag{3}$$

where T is the predefined targeted session end time. There have been many methods proposed to compute these metrics (Ahmadi-Javid et al. 2017). Once a method is found, these metrics can be used to evaluate the objective function in Display (1).

3.2. Complexity of a Multiserver System

Unfortunately, in a multiserver setting, despite the fact that the waiting queue is shared, these recursions do *not* apply, as noted in the seminal work by Kiefer and Wolfowitz (1955). One of the critical issues is that the ith departure is not necessarily by the ith client. Because clients are served by several servers in parallel, there can be overtaking; a client is still in service while another server can become available to serve the subsequent client, so that eventually the subsequent client can leave the system earlier than his predecessor. Thus the waiting time of one client does not exclusively depend on the sojourn time of its predecessor, which

makes the problem considerably more challenging and the Lindley recursion inapplicable.

As phase-type distributions have a state-space representation, they permit for keeping track of the *system*. In detail, it is possible to keep track of which client is being served by which server; for this purpose, we need to consider tuples of s dimensions, where 0 denotes a server as empty. The domain of each element in the tuple is in the simplest case just the number of possible clients. For example, in Table 1 we show the number of configurations when there are two servers and five clients to be served. The shaded empty cells cannot be reached as they contain cases where both servers are serving the same client (black) or that a client jumps from one server to another (gray). A simple computation of the number of possible client configurations reveals that this number increases by $n^2 - n + 2$, where n is the number of clients. In general, in a similar way, it can be shown that for s servers, the number of possible configurations is $\mathcal{O}(n^s)$.

3.3. Compressing the State Space

In line with the literature, we assume homogeneous servers and clients. First, without loss of generality, we normalize the mean service time to one for all clients—that is, $\mathbb{E}B_i = \mu_i = 1$. Second, we express the service-time variability in terms of the squared coefficient of variation:

$$\text{scv} = \frac{\mathbb{V}\text{ar}B_i}{(\mathbb{E}B_i)^2} = \mathbb{V}\text{ar}B_i. \tag{4}$$

As clients are served according to a first-come, first-served discipline, it suffices for the ith client to keep track of the work ahead of him—that is, the *number of clients* in and the *status* of the system upon and after his arrival; for these purposes, define the variables Y_i and $\boldsymbol{Z}_i$, respectively. Because the status of the system is given by the current phase(s) of the client(s) in service, $\boldsymbol{Z}_i$ is often multidimensional. Its dimension depends on the number of clients who are currently in service. Keeping track of these variables allows the computation of waiting times and also session end time. As an example, client i's *waiting time* W_i can be inferred from the moment a server comes available for the ith client (i.e., when there are fewer than s clients in service):

$$W_i = \inf\{t \geq 0 \mid Y_i(t) \leq s\} = \inf\{t \geq 0 \mid Y_{i-1}(t + x_i) < s\}. \tag{5}$$

Idle time requires more thought. The makespan denotes the session end time; multiplying this quantity by s gives the servers' total capacity over the course of the session and evidently contains *all* idle times.

Obviously, in a multiserver setting, it is possible that there is no need to keep all servers active throughout the entire session. In Section 5.2 we discuss the impact of this additional feature and why not to include this in the objective function. Last, referring to Equation (3), the notion of session overtime will be carried over to the multiserver case. In the next section we propose a method that enables tracking of the system in continuous time and thus computation of these performance metrics in expectation.

Table 1. The 22 Possible Configurations of Five Clients on Two Servers

(0,0)		(0,2)	(0,3)	(0,4)	(0,5)
(1,0)		(1,2)*	(1,3)	(1,4)	(1,5)
(3,0)		(3,2)		(3,4)	(3,5)
(4,0)		(4,2)	(4,3)		(4,5)
(5,0)		(5,2)	(5,3)	(5,4)	

*Indicates the starting state, client 1 on the first server and client 2 on the second server.

4. Methodology

In this section we outline the method that enables computation of the system. We do so by relying on phase-type distributions, which has been a widely accepted method in queueing (Neuts 1981, Tijms 1986, Asmussen et al. 1996) and, specifically, appointment scheduling (Wang 1997, Bosch and Dietz 2001, Kuiper et al. 2015). To keep track of the multiserver system we exploit the property that a convolution of multiple phase-type distributions can again be described by a phase-type distribution.

A service-time distribution B_i is approximated by a *phase-type counterpart*

$$B_i \sim \text{PH}(\boldsymbol{\alpha}, \boldsymbol{S}), \tag{6}$$

where $\boldsymbol{\alpha}$ is a row vector describing initial probabilities and $\boldsymbol{S}$ the transition matrix.

Following the standard approach, a mixture of two *Erlang* distributions is advised in cases where the service-time distribution has a SCV smaller than 1—that is, $E_{k-1,k}(\mu; p)$, requiring k phases. Furthermore, a *hyperexponential* distribution, $H_2(\mu_1, \mu_2; p)$, with balanced means is used in cases where SCV is larger than 1; this has $k = 2$ phases. The middle case, when SCV = 1, corresponds to the exponential distribution and has just one phase. Without loss of generality, the service times are set to 1, as in the single-server case in Kuiper et al. (2015).

4.1. Phase-Type Recursion

We are interested in the bivariate process $(Y_{s+i}(t), \boldsymbol{Z}_{s+i}(t))$ for $i = 0, \ldots, n - s$ that describes the full evolution of the system, where we have

- $Y_{s+i}(t) \in 0, 1, \ldots, s+i$ clients in the system, as we start with s clients in service, and
- $\boldsymbol{Z}_{s+i}(t) = (Z_1(t), \ldots, Z_\xi(t))$, where $\xi = \min\{Y_{s+i}(t), s\}$, and for each $\ell = 1, \ldots, \xi$, $Z_\ell(t) \in 1, \ldots, k$.

Note that k denotes the number of phases of the phase-type counterpart. Without loss of generality, Z_ℓ can be seen as the ℓ th server, because of homogeneous servers. There are at most ξ servers to record as either there are Y_{s+i} clients to be served or all s servers are active.

Because there is no distinction between which server serves which client, the number of unique combinations of states depends only on the number of servers and possible phases. The maximum number of states required turns out to be $\sum_{i=1}^{n} k^{\min\{i,s\}}$, which is $\mathcal{O}(n)$—a remarkable reduction compared with naively considering all unique routings (cf. $\mathcal{O}(n^s)$ as in Section 3.2).

Corresponding to the bivariate process, we define the probabilities of finding j clients in the system, $j \in \{0, \ldots, s+i\}$, and the server(s) in phase(s) $m_\ell \in \{1, \ldots, k\}$ for $\ell \in \{1, \ldots, \xi\}$:

$$p^{(s+i)}_{j,(m_1,\ldots,m_\xi)}(t) = \mathbb{P}\big[(Y_{s+i}(t), \boldsymbol{Z}_{s+i}(t)) = \big(j, (m_1, \ldots, m_\xi)\big)\big].$$

Define the row vector that contains all possible phases for j clients in service by

$$\begin{aligned}\boldsymbol{p}^{(s+i)}_j(t) = \Big(&p^{(s+i)}_{j,(k,\ldots,k)}(t), \ldots, p^{(s+i)}_{j,(k,\ldots,1)}(t), \ldots,\\ &p^{(s+i)}_{j,(1,\ldots,k)}(t), \ldots, p^{(s+i)}_{j,(1,\ldots,1)}(t)\Big),\end{aligned} \tag{7}$$

which is a vector of size $k^{\min\{j,s\}}$. The quantity $\boldsymbol{p}^{(s+i)}_j(t)\mathbf{1}$ is the probability that j clients remain in the system t time units after client $(s+i)$'s arrival, where $\mathbf{1}$ is a column vector of appropriate size.

In the special case that all service times are exponentially distributed, the workload vector $Y_{s+i}(t)$ is only needed to describe the evolution of the system, because each client's service comprises just a single phase. Consequently, the $\boldsymbol{p}^{(s+i)}_j(t)$ become singletons that describe the probabilities that j clients remain t amount of time after the arrival of the ith client.

For the general case, the probabilities by Equation (7) describe the evolution of the system. Because each server starts by serving a client and no new clients have yet arrived, the initial probability vector of the system is given by the following concatenation: $\boldsymbol{\alpha}_s = (\boldsymbol{\alpha} \otimes \cdots \otimes \boldsymbol{\alpha}, \mathbf{0}_{\sum_{j=1}^{s-1} k^j})$; the Kronecker product in this vector is applied exactly $(s-1)$ times. The transition matrix is given by

$$\boldsymbol{S}_s = \begin{pmatrix} \boldsymbol{S}^{(s)} & \boldsymbol{U}^{(s)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{S}^{(s-1)} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \boldsymbol{U}^{(3)} & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{S}^{(2)} & \boldsymbol{U}^{(2)} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \boldsymbol{S}^{(1)} \end{pmatrix}, \tag{8}$$

where $\boldsymbol{S}^{(\ell)}$ (diagonal) and $\boldsymbol{U}^{(\ell)}$ (upper diagonal) are defined recursively ($1 < \ell \le s$) by

$$\boldsymbol{S}^{(\ell)} = I_{|\boldsymbol{S}|} \otimes \boldsymbol{S}^{(\ell-1)} + \boldsymbol{S} \otimes I_{|\boldsymbol{S}^{(\ell-1)}|}, \tag{9}$$

$$\boldsymbol{U}^{(\ell)} = I_{|\boldsymbol{U}^{(\ell-1)}|} \otimes \boldsymbol{U}^{(1)} + \boldsymbol{U}^{(\ell-1)} \otimes I_{|\boldsymbol{U}^{(1)}|}, \tag{10}$$

with $\boldsymbol{S}^{(1)} = 1$, $\boldsymbol{U}^{(1)} = -\boldsymbol{S}\mathbf{1}$, $|\boldsymbol{A}|$ being the number of rows in matrix $\boldsymbol{A}$, and $I_{|\cdot|}$ is an identity matrix with $|\cdot|$ rows and columns. In fact, $\boldsymbol{U}^{(1)}$ is the traditional phase-type exit vector that corresponds to service completion. The elements of the transition matrix in Equation (8) can be understood as $\boldsymbol{S}^{(\ell)}$, describing the transitions between states in which ℓ servers are busy, and $\boldsymbol{U}^{(\ell)}$, the *exit matrix* that defines the transitions to only $\ell - 1$ servers being busy and thus one server becoming idle.

The vector $\boldsymbol{p}^{(s)}(t)$ is fully described by a phase-type distribution $\mathrm{PH}(\boldsymbol{\alpha}_s, \boldsymbol{S}_s)$:

$$\boldsymbol{p}^{(s)}(t) = \Big(\boldsymbol{p}^{(s)}_s(t), \boldsymbol{p}^{(s)}_{s-1}(t), \ldots, \boldsymbol{p}^{(s)}_1(t)\Big) = \boldsymbol{\alpha}_s \exp(\boldsymbol{S}_s t). \tag{11}$$

For a phase-type representation of the system after the arrival of all other clients, a recursive procedure will be proposed. In the same fashion as for the initialisation, we are interested in the vector after the $(s+i)$ th client has entered the system, with $i = 1, \ldots, n-s$; using Equation (7), we find

$$\begin{aligned}\boldsymbol{p}^{(s+i)}(t) = \Big(&\boldsymbol{p}^{(s+i)}_{s+i}(t), \boldsymbol{p}^{(s+i)}_{s+i-1}(t), \ldots, \boldsymbol{p}^{(s+i)}_{s+1}(t), \boldsymbol{p}^{(s+i)}_s(t),\\ &\boldsymbol{p}^{(s+i)}_{s-1}(t), \ldots, \boldsymbol{p}^{(s+i)}_1(t)\Big).\end{aligned} \tag{12}$$

Furthermore, the probability of being in the absorbing state of an empty system is found by

$$p^{(s+i)}_0(t) = 1 - \boldsymbol{p}^{(s+i)}(t)\mathbf{1}.$$

To find an expression that tracks these probabilities over time, we introduce a phase-type distribution, $(\boldsymbol{\alpha}_{s+i}, \boldsymbol{S}_{s+i})$, for each client $i \in \{1, \ldots, n-s\}$. The transition matrix $\boldsymbol{S}_{s+i}$ is found by extending $\boldsymbol{S}_s$ by i times, adding $\boldsymbol{S}^{(s)}$ along the diagonal. In addition, we also need to describe the flow from one client being finished to the next one being served. For this purpose, we place the transition matrix $\boldsymbol{T}^{(s)}$ along the upper diagonal, which will be defined as

$$\boldsymbol{S}_{s+i} = \left(\begin{array}{ccccc|cc} \boldsymbol{S}^{(s)} & \boldsymbol{T}^{(s)} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{S}^{(s)} & \boldsymbol{T}^{(s)} & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \ddots & \ddots & \mathbf{0} & & \\ \vdots & \ddots & \ddots & \boldsymbol{S}^{(s)} & \boldsymbol{T}^{(s)} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{S}^{(s)} & \boldsymbol{T}^{(s)} & \mathbf{0} \\ \hline \mathbf{0} & & & & & \boldsymbol{S}_s & \end{array}\right), \tag{13}$$

$$=: \left(\begin{array}{c|c} \boldsymbol{S}^{\text{wait}}_i & \boldsymbol{T}^{(s)}_s \\ \hline \mathbf{0} & \boldsymbol{S}_s \end{array}\right), \tag{14}$$

where $\boldsymbol{T}^{(s)}$ follows from the recursion

$$\boldsymbol{T}^{(\ell)} = I_{|\boldsymbol{T}^{(\ell-1)}|} \otimes \boldsymbol{T}^{(1)} + \boldsymbol{T}^{(\ell-1)} \otimes I_{|\boldsymbol{T}^{(1)}|}, \quad \text{with}$$
$$\boldsymbol{T}^{(1)} = -\boldsymbol{S}\mathbf{1} \otimes \boldsymbol{\alpha}.$$

Thus each block $\boldsymbol{S}^{(s)}$ added to the diagonal in Equation (13) corresponds to states where one might find an additional client in the waiting queue. For example, after the $(s+i)$th client's arrival, there can be at most $i-s$ clients waiting. Analogously, the exit matrix $\boldsymbol{T}^{(s)}$ describes transitions from a saturated system with $j > s$ clients in the system to one with precisely $j-1 \geq s$ (cf. $\boldsymbol{U}^{(\ell)}$ for $\ell \in 2,\dots,s$); see Equation (10), which correspond each time to the ℓ th server becoming available.

The initial probability vector $\boldsymbol{\alpha}_{s+i}$, for $i = 1,\dots,n-s$, captures the intrinsic *recursivity* of the approach. Because on arrival of a subsequent client the system can be saturated, all s servers are busy, and the client enters the waiting queue or is accepted by an available server. The corresponding vector $\boldsymbol{\alpha}_{s+i}$ can be derived from $\boldsymbol{p}^{(s+i-1)}(x_{s+i})$ as defined in Equation (11) and reads for client $s+i$ who arrives x_{s+i} time after his predecessor:

$$\begin{aligned}\boldsymbol{\alpha}_{s+i} &= f\left(\boldsymbol{p}^{(s+i-1)}(x_{s+i}), \boldsymbol{\alpha}\right)\\ &:= \Big(\boldsymbol{p}_{s+i-1}^{(s+i-1)}(x_{s+i}),\dots,\boldsymbol{p}_{s+1}^{(s+i-1)}(x_{s+i}),\boldsymbol{p}_{s}^{(s+i-1)}(x_{s+i}),\\ &\quad \boldsymbol{\alpha}\otimes\boldsymbol{p}_{s-1}^{(s+i-1)}(x_{s+i}),\dots,\boldsymbol{\alpha}\otimes\boldsymbol{p}_{1}^{(s+i-1)}(x_{s+i}),\\ &\quad \boldsymbol{\alpha}\otimes p_{0}^{(s+i-1)}(x_{s+i})\Big),\end{aligned} \tag{15}$$

wherein the states in the first line of (15) correspond to saturation and in the second line of (15) to the start of service for the new client. Furthermore, note that this vector has k^s entries more than its predecessor. Thus after arrival of the $(s+i)$ th client, the evolution of the system as in Equation (12) is described by $\mathrm{PH}(\boldsymbol{\alpha}_{s+i}, \boldsymbol{S}_{s+i})$.

4.2. Computation of Performance Metrics

Knowing the phase-type representation of the system after each client's arrival, we can compute his waiting time by considering an embedded phase-type distribution. For this purpose, we specifically look at the probabilities that correspond to instances in which clients are waiting. Thus for clients $i = 1,\dots,n-s$,

$$\boldsymbol{p}_{\text{wait}}^{(s+i)}(t) := \left(\boldsymbol{p}_{s+i}^{(s+i)}(t), \boldsymbol{p}_{s+i-1}^{(s+i)}(t),\dots,\boldsymbol{p}_{s+1}^{(s+i)}(t)\right), \tag{16}$$

these probabilities adhere to the recursion earlier, and naturally, one can define a start vector on arrival by $\boldsymbol{\alpha}_i^{\text{wait}} = \boldsymbol{p}_{\text{wait}}^{(s+i)}(0)$. Furthermore, the transitions between these probabilities over time are described by $\boldsymbol{S}_i^{\text{wait}}$, as defined in Equation (14):

$$F_{W_{s+i}}(t) = 1 - \boldsymbol{p}_{\text{wait}}^{(s+i)}(t)\mathbf{1} = 1 - \boldsymbol{\alpha}_i^{\text{wait}} \exp\left(\boldsymbol{S}_i^{\text{wait}} t\right)\mathbf{1},$$

where $\mathbf{1}$ is a column vector of appropriate size. Also, for phase-type distributions, the moments can readily be obtained by using its representation, so that

$$\sum_{i=s+1}^{n} \mathbb{E}W_i = \sum_{i=1}^{n-s} -\boldsymbol{\alpha}_i^{\text{wait}} (\boldsymbol{S}_i^{\text{wait}})^{-1}\mathbf{1}. \tag{17}$$

For idle and overtime, define $F_{M_{s+i}}(t)$ as the cumulative distribution function of the makespan of finishing the first $s+i$ clients t time units after t_{s+i} $(i = 1,\dots,n-s)$; this is given by

$$F_{M_{s+i}}(t) = p_0^{(s+i)}(t) = 1 - \boldsymbol{p}^{(s+i)}(t)\,\mathbf{1} = 1 - \boldsymbol{\alpha}_{s+i} \exp\left(\boldsymbol{S}_{s+i}\, t\right)\mathbf{1},$$

so that $\mathbb{E}M_{s+i} = -\boldsymbol{\alpha}_{s+i}\boldsymbol{S}_{s+i}^{-1}\mathbf{1}$. In particular, the makespan corresponding to having all n clients served demarcates the session, and so the sum of all idle times can be described as the total time available in the system minus time spent in service. Similarly, overtime is incurred for all servers if a client remains in service after the targeted session end time, T. Thus the respective metrics for the servers' idle times and overtimes are given by

$$\mathbb{E}I^{(s)} = s\left(\mathbb{E}M_n + t_n\right) - \sum_{i=1}^{n} \mathbb{E}B_i, \tag{18}$$

$$\mathbb{E}O^{(s)} = s\int_0^\infty \max\left\{t + t_n - \mathrm{T}, 0\right\} \mathrm{d}F_{M_n}(t). \tag{19}$$

The targeted session end time in the multiserver case is set to the sum of mean service times divided by the number of servers; as such, it becomes equivalent to the single-server case. Note that the performance measures are superscript so as to indicate that in a multiserver setting, idle time and overtime are incurred by all servers until the last server finishes the last client.

4.3. Convexity

For the single-server appointment scheduling problem in continuous time, strong stochastic convexity arguments can be invoked to prove that the waiting times are convex in the interarrivals. This follows from the fact that the Lindley recursion of Equation (2) consists of convex operators; see theorem (2.15) of Shanthikumar and Yao (1991). Other performance metrics can be expressed as convex functions of waiting times to establish convexity of the objective function (1); see, for example, Wang (1993) and Kuiper et al. (2021) for a proof that does not rely on stochastic convexity arguments.

For the multiserver queue, recursive systems exist that keep track of the workload per server (e.g., Kiefer and Wolfowitz 1955). Unfortunately, not all operators in this system are convex. As a consequence, strong stochastic convexity arguments cannot be applied to show convexity for general service-time distributions.

Indeed, Harel (1990) found a counterexample that shows in stationarity that expected waiting times as a function of the interarrival time in the $D/G/s$ queue are not convex, using a *bimodal* service time distribution: with 2/3 probability, the service time equals 5, and with 1/3, it equals 11 time units.

For phase-type distributions it is widely known that they can be used to approximate any nonnegative distribution arbitrarily closely. The counterexample of Harel (1990) can easily be replicated by using a combination of two Erlang distributions with appropriate means and many phases. Hence waiting times are not convex in interarrival times for the subclass $D/PH/s$ either.

However, our phase-type distributions are unimodal, and we have strong reason to believe that our solutions are global optima, as various starting points led to the same solutions. Finally, as considered in Section 6, we show that the optimization problem considered in the heavy-traffic regime—to which, in essence, many of the problems converge—is convex.

For the discrete analogue of the single-server appointment scheduling problem, Zacharias and Yunes (2020) establish multimodularity to guarantee that a global minimum is found. Our methodology can be used to show that multimodularity does not extend to the multiserver setting. For this purpose, consider a schedule of equal slots: the first s clients arrive at the first slot starting at time 0, and all subsequent clients arrive according to an equidistant schedule, $x_{s+i} = \mathbb{E}B/s$. Then, choosing $n = 10$ and considering phase-type distributed service times with (1) SCV = 1.5 and the number of servers $s = 2$, 3, or 4 or (2) SCV = 0.75 and the number of servers $s = 3$ or 4, we find that neither the expected idle times of Equation (18) nor the ones corrected for *early leave* (see Equation (22) in Section 5.2) adhere to the first property as stated in lemma 1 of Zacharias and Yunes (2020). Indeed, upon inspection, the proofs of multimodularity of the single-server performance metrics rely on keeping track of the workload per slot by means of the Lindley recursion, but this principle fails as the variables for clients' waiting times and servers' workload do not coincide in a multiserver setting (Daley 1997).

5. Multiserver Appointment Scheduling in Transient Settings

In this section we employ our methodology to compute appointment schedules for the multiserver setting and contrast that with implementing a complementary setup of optimized single-server appointment schedules. In addition, the impact of service-time variability, no-shows, and some relevant modifications to the objective functions, such as early leave of servers, are studied.

In our analyses we examine the optimal solutions found when appointments are scheduled for multiple servers. For our computations we relied on a standard machine, and our programs are written in `MATLAB`. For minimization, `MATLAB`'s built-in routine `fmincon` is employed. We use the metrics as defined in Section 4.2 and set the cost of waiting time to 1 ($c_W = 1$) so that the cost ratio of idle to waiting time simplifies to c_I. Hence, our minimization becomes

$$\min_{x_{s+1},\dots,x_n} c_I \mathbb{E}I^{(s)} + \sum_{i=s+1}^{n} \mathbb{E}W_i + c_O \mathbb{E}O^{(s)}, \tag{20}$$

so that the nontrivial arrival epochs follow from $t_{s+i} = \sum_{j=1}^{i} x_{s+j}$.

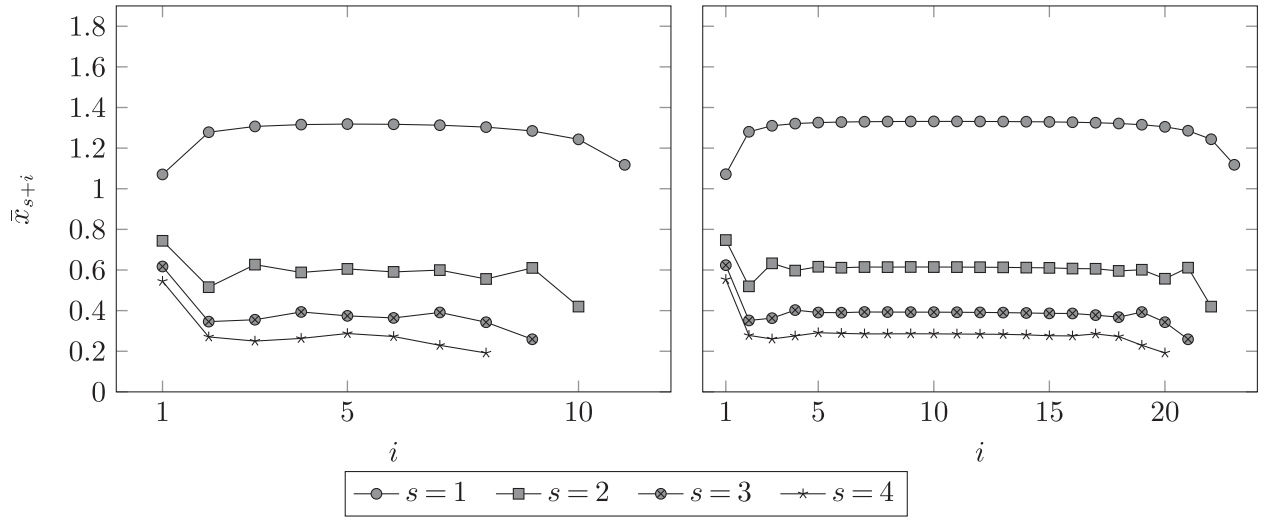
The explicit expressions for the metrics given in Section 4.2 facilitate computation. Note that if $c_O = 0$, the objective function reduces to a closed-form expression. Research by Klassen and Yoogalingam (2014) suggests that including overtime into the objective function has roughly the same impact as increasing the weight put on idle time. As a consequence, most of our experiments concentrate on the case of idle and waiting time only, although in Section 5.2 we specifically study the inclusion of overtime. We verified the optimizations by choosing as different starting points vectors consisting of only zeros, ones, average service rates ($1/s$), and the heavy-traffic solution as obtained in Section 6.2.

5.1. Structure of the Optimal Solution

Studying the patterns of the optimal interarrival times of multiserver appointment schedules for various numbers of clients, the characteristic dome-shaped plateau to which solutions converge can be identified. For example, in Figure 1 for $s = 1$ clear dome shapes appear. For the pooled schedules, where $s = 2$, 3, or 4, the middle solutions converge to steady-state values, encompassing the long-term balance between idling and waiting. Note that because the number of clients to be scheduled is fixed, there is one interarrival fewer to be determined if the number of servers s increases.

At the start of a session, as seen in Figure 1 and also in Figure 2, we observe a *reversed bullwhip* in the interarrival times; a steep decline in the interarrival times is followed by a damping pattern of iteratively increasing and decreasing interarrival times. The reason for this pattern is that the synchronized start of service is completely absorbed by the randomness in the system if there are sufficient clients to be scheduled. Comparing Figures 1 and 2, the extent of this effect is amplified for lower values of SCV. In the extreme case of no uncertainty (i.e., the $D/D/s$ queue), the optimal schedule for s servers would be the arrival of a batch of s clients after each mean service time.

Figure 1. Patterns of Interarrival Times for SCV = 0.25 and $c_I = 1$



Note. From top to bottom, the number of servers increases from one to four with (left) 12 clients and (right) 24 clients to be scheduled.

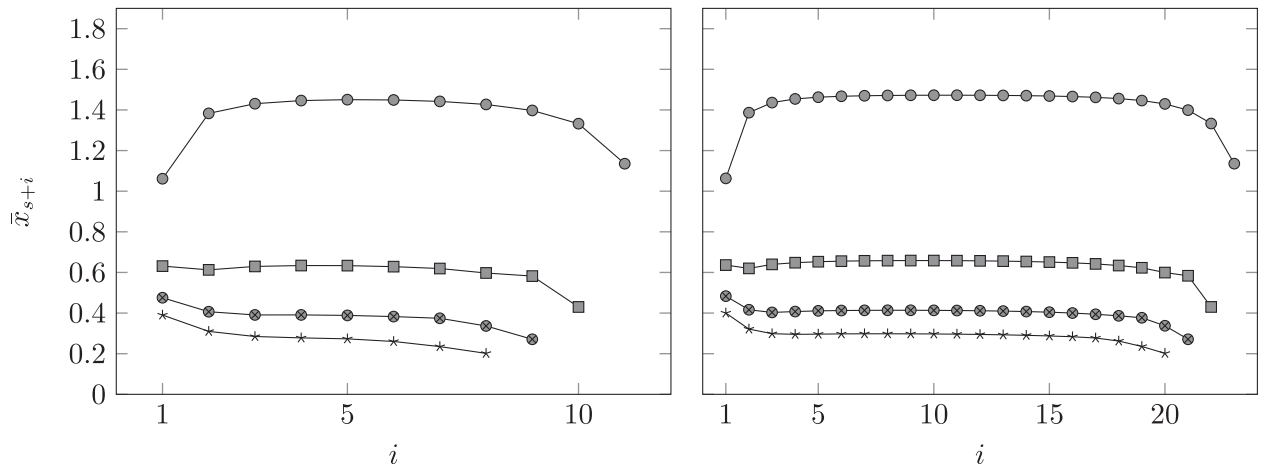
If SCV equals 1, these patterns disappear (see Figure 3), possibly as a result of *memorylessness* of the exponential service times that are used to model the service times: aside from the number of clients in the system at each arrival, no additional information is revealed about how far along each service time is. In the case of a SCV greater than 1, a mixture of exponential service times is used, which contains even greater variability than the exponential case, and so as in Figure 4, the dome-shaped pattern stands out.

Focusing on the session end, fluctuating interarrival times are apparent in a bullwhip pattern for low SCV values. In particular, analogous to the start-up, these patterns are stronger for lower SCV cases, and evidently, the effect disappears for SCV ≥ 1. The explanation for this behavior is that unused capacity on other servers is penalized as idle time, so the optimization tries to synchronize the servers' end times at the expense of reducing waiting times. Waiting time becomes less important toward the end of the session, as there will be fewer clients who would be affected by a tight schedule.

5.2. Session End Revisited

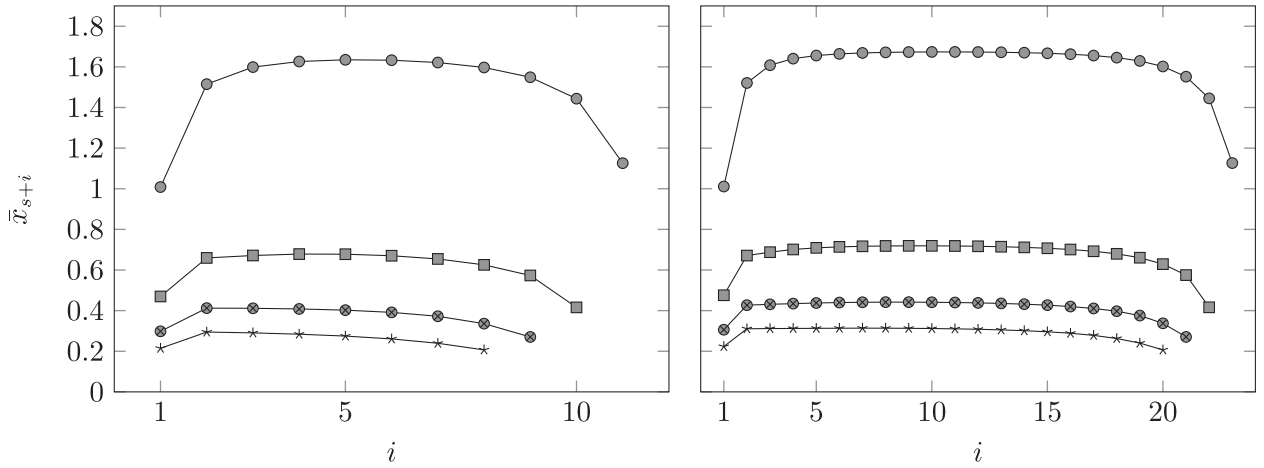
One can opt to include session overtime in the framework, which is computed by Equation (19). In Figure 5 we report the optimized interarrival times that result from an objective function that is composed of only waiting time and overtime. Here, we

Figure 2. Patterns of Interarrival Times for SCV = 0.5 and $c_I = 1$



Note. From top to bottom, the number of servers increases from one to four (see the legend in Figure 1) with (left) 12 clients and (right) 24 clients to be scheduled.

Figure 3. Patterns of Interarrival Times for SCV = 1 and $c_I = 1$

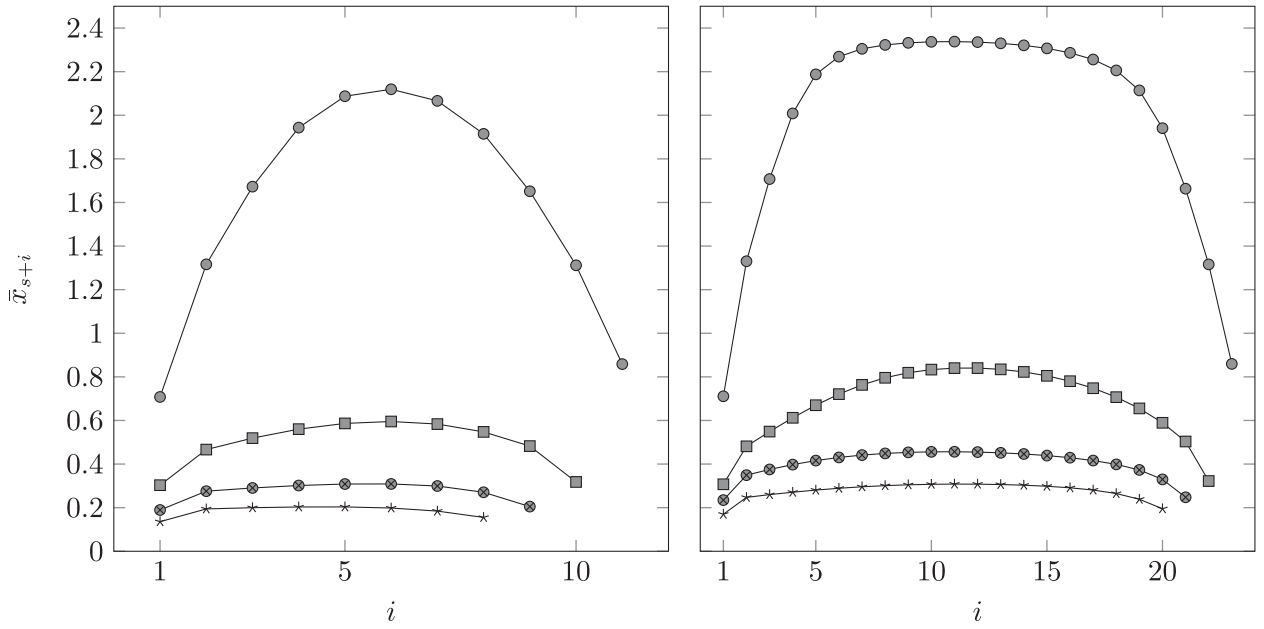


Note. From top to bottom, the number of servers increases from one to four (see the legend in Figure 1) with (left) 12 clients and (right) 24 clients to be scheduled.

choose $c_O = 1.5$—1.5 times the value chosen for c_I in Figure 2—which is typical, as argued in Cayirli et al. (2012). In Figure 5 we added to the optimal solutions the solutions from Figure 2, which incorporated idle time instead of overtime. Remarkably, for any number of servers, a comparison of the shape of the curves reveals that including overtime has a similar impact as idle time, which echoes the conclusions of Klassen and Yoogalingam (2014) for the single-server case.

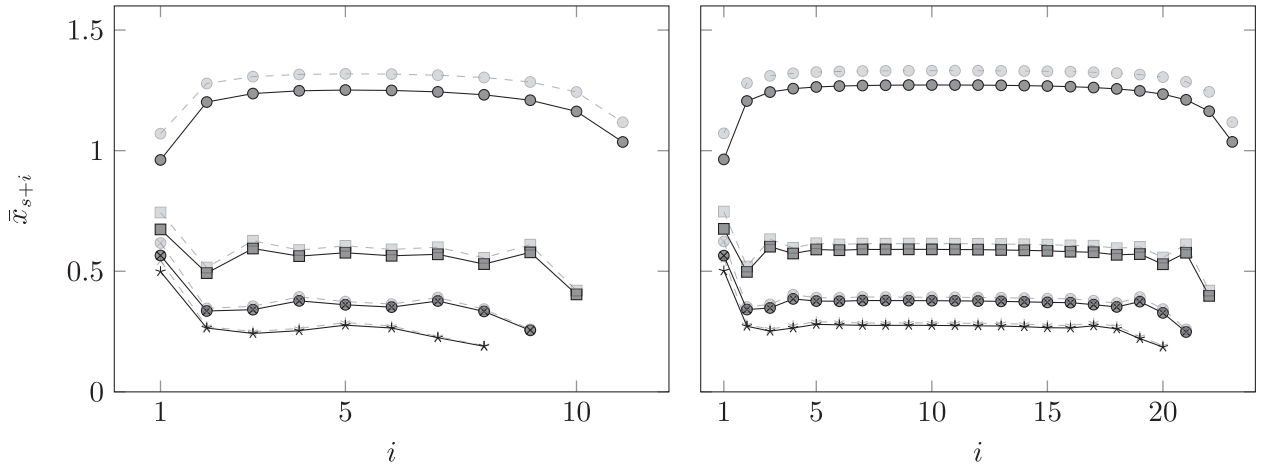
Another salient feature of a multiserver appointment schedule is that servers can finish earlier when there is insufficient work left—that is, there are fewer clients in the process and in the appointment schedule than there are servers. For example, in Figure 6(b) servers 2 and 3 can finish earlier. By allowing this early leave, the idle time of servers waiting until the last client has finished service can be reduced; this time is indicated by the diagonal lines. So far, this feature is not incorporated in the objective function, as it would cause unsynchronized endings to not be penalized and thus lead to unnecessary underutilization of a session. Naturally, in

Figure 4. Patterns of Interarrival Times for SCV = 4 and $c_I = 1$



Note. From top to bottom, the number of servers increases from one to four (see the legend in Figure 1) with (left) 12 clients and (right) 24 clients to be scheduled.

Figure 5. Patterns of Interarrival Times for SCV = 0.25 and $c_O = 1.5$ ($c_I = 0$)



Notes. From top to bottom, the number of servers increases from one to four (see the legend in Figure 1) with (left) 12 clients and (right) 24 clients to be scheduled. The new patterns are represented by the fully opaque marks, in contrast to the original patterns from Figure 1 in which c_I=1 (c_O=0), which are represented by semitransparent marks.

a system of single-server appointment schedules, a server leaves when there is no client left, as seen in the system in Figure 6(a). So in order to have a balanced comparison (also in terms of idle times) between a system of single-server systems and a multi-server setting, a correction for early leave of servers is necessary.

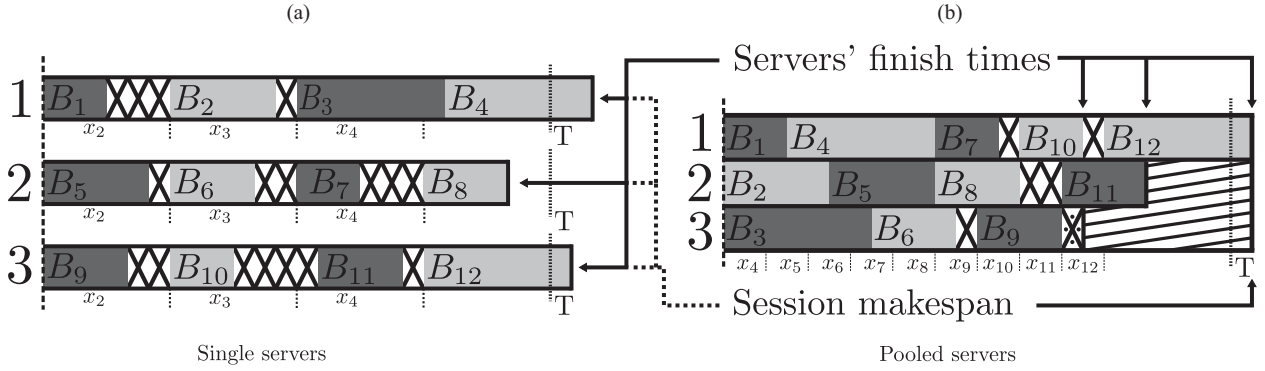
After optimizing over the objective function in Display (20), the additional gain of allowing early leave of servers can be computed. Obviously, a server can only finish if it is certain that the server will not be required in the future. So if there are ξ servers busy ($\xi \in \{1,2,\dots,s\}$) and one finishes, it can be released if and only if there are exactly $\xi - 1$ clients remaining to be served—that is, those currently in service plus the ones still scheduled. To highlight the subtlety, note that the third server in Figure 6(b) can only leave after the 10th client has left and no sooner, which is after the *ex with dots*. At that time, there is one client yet to *start* and one still *in* service.

The expected finish time of a server in a pooled system (numbered in reverse order of leaving) can be computed by constructing elements of a cumulative distribution function. The ℓ th server ($\ell \in \{1,\dots,\xi\}$) can finish in a time $t \in [0, x_{n-\ell+j+1})$ after the $(n-\ell+j)$ th arrival for $j \in \{1,\dots,\ell\}$ with $x_{n+1} := \infty$, so there are $\ell - j$ clients yet to arrive, if there are fewer than j clients in service. This leads to

$$F_{E_{\ell,j}}(t) = 1 - \sum_{i=j}^{n-\ell+j} \boldsymbol{p}_i^{(n-\ell+j)}(t)\mathbf{1}.$$

Recall that $\boldsymbol{p}_i^{(n-\ell+j)}$ describes the phases after the arrival of the $(n-\ell+j)$ th client for which i clients remain in the system; see Equation (7). The expected

Figure 6. A Visualization of Two Appointment Systems That Serve 12 Clients on Three Servers



Notes. Singly operating servers, each with the same schedule (a) and in parallel (b). Exes, idle time; diagonal lines, the gain won by allowing early leave. Service times are denoted by B_i and the session end time by T.

finish time of server ℓ and server-specific overtime can be computed accordingly via the numerical integration of

$$\mathbb{E}E_\ell = \sum_{j=1}^{\ell} \int_0^{x_{n-\ell+j+1}} (t + t_{n-\ell+j})\, \mathrm{d}F_{E_{\ell,j}}(t),$$

$$\mathbb{E}O_\ell = \sum_{j=1}^{\ell} \int_0^{x_{n-\ell+j+1}} \max\{t + t_{n-\ell+j} - \mathrm{T}, 0\}\, \mathrm{d}F_{E_{\ell,j}}(t). \quad (21)$$

The session end metrics defined in Equations (18) and (19) relate accordingly: $\mathbb{E}I^{(s)} \equiv s\,\mathbb{E}E_1 - \sum_{i=1}^{n} \mathbb{E}B_i$ and $\mathbb{E}O^{(s)} \equiv s\,\mathbb{E}O_1$. Because servers leave when no longer needed, the expected idle times throughout the schedule are computed by

$$\sum_{i=s+1}^{n} \mathbb{E}I_i = \sum_{\ell=1}^{s} \mathbb{E}E_\ell - \sum_{i=1}^{n} \mathbb{E}B_i. \quad (22)$$

This revisitation of the session end extends Equation (3) to the multiserver setting, allowing a comparison with equivalent systems of single servers (i.e., to study the impact of pooling).

5.3. Benefits of Pooling

Besides studying the structure of the optimal solutions, we are also interested in a comparison of performance between having server-dedicated appointment schedules versus a pooled multiserver appointment schedule. As reported for call centers (e.g., Van Dijk and Van der Sluis 2008), we anticipate that the pooling of resources will be highly beneficial.

To understand the merits of pooling, we compare the performance of our multiserver appointment schedules to those in which an equivalent system of single-server schedules are employed. In order to have a balanced comparison, we compute the expected overtimes per server and idle times throughout the session by using Equations (21) and (22) after the optimization as to examine how an individual server benefits from being in a pooled system. Varying the number of clients in multiples of 12, divisible by 2, 3, or 4 (servers), we report the expected performance in Table 2.

The performance improvement is striking, and significant reductions in each performance dimension appear, up to about 55% when four servers are pooled. Note that in some cases the sum of overtimes and idle times is the same, which naturally occurs when the last client is scheduled after the targeted session end time: $\sum_{i=1}^{n-s} \bar{x}_{s+i} = t_n > \mathrm{T}$.

In appointment scheduling, the servers' time is often considered to be more valuable than that of clients—for example, in healthcare, where waiting time is valued considerably less than idle time (Robinson and Chen 2011). Therefore we experiment here with settings in which the cost parameter c_I takes a high value in Objective (20). Besides depicting the baseline schedule of Figure 1, where $c_I = 1$, the optimal schedules are shown in Figure 7 for $c_I = 5$, in accordance with the middle setting of Cayirli et al. (2012), and $c_I = 20$ as an extreme case, albeit in line with the observations of Robinson and Chen (2011). Besides moving the dome-shaped pattern down (i.e., tightening the schedule), placing a lower value on waiting time damps the distinctive start and end patterns of a multiserver schedule.

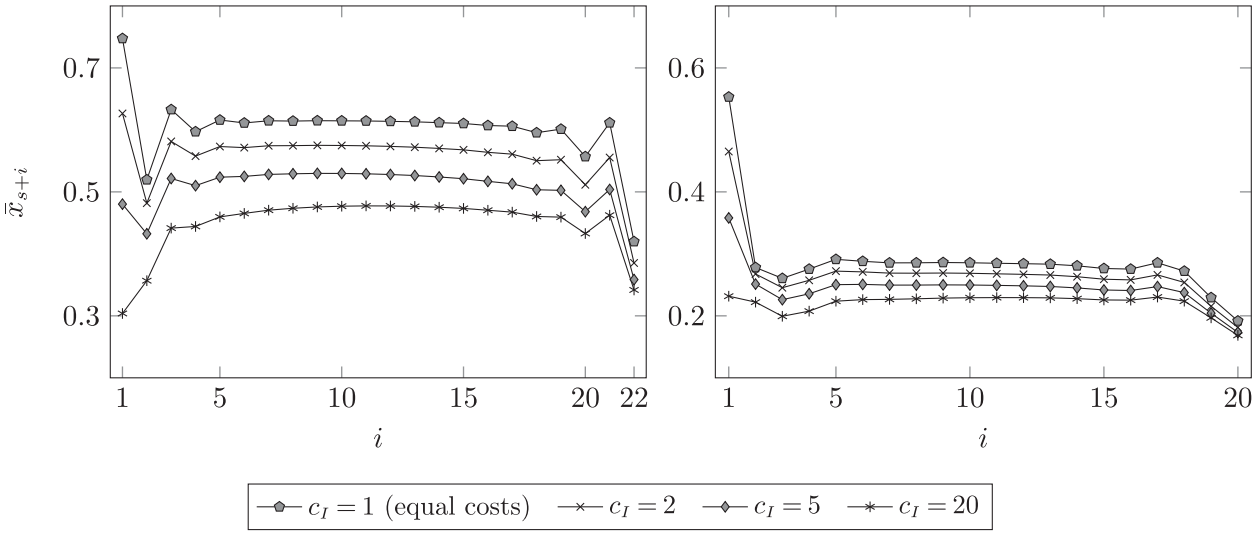
Considering the gains achieved by pooling in Table 3 in the case of $c_I = 5$, we observe that the improvements on the servers' account lag behind in the smaller instances. This effect is due to the imbalanced effort that is now put on reducing idling, and consequently overtime, resulting in tight schedules in the corresponding system of s single servers. Contrariwise, the expected waiting times decrease greatly; see Table 2. For $n = 48$,

Table 2. Comparison of Single Server and Pooled Server System Performance for SCV = 0.5 and $c_I = 1$

	$n = 12$			$n = 24$			$n = 48$		
	Expected costs in an optimized system of s single servers (a)								
s	$\sum \mathbb{E}I_i$	$\sum \mathbb{E}W_i$	$\sum \mathbb{E}O_\ell$	$\sum \mathbb{E}I_i$	$\sum \mathbb{E}W_i$	$\sum \mathbb{E}O_\ell$	$\sum \mathbb{E}I_i$	$\sum \mathbb{E}W_i$	$\sum \mathbb{E}O_\ell$
2	3.4118	3.4602	3.4118	8.9153	7.2742	8.9153	20.2858	14.6852	20.2858
3	2.6000	3.1740	2.7369	7.8055	7.1316	7.8055	19.0337	14.6337	19.0337
4	1.9706	2.8264	2.5451	6.8236	6.9204	6.8236	17.8307	14.5483	17.8307
	Expected costs in an optimized system of s pooled servers (b)								
2	2.4315	2.3272	2.4415	6.2922	4.8752	6.2922	14.2322	9.8381	14.2322
3	1.5421	1.6861	1.8252	4.5390	3.7538	4.5393	10.9165	7.6884	10.9165
4	1.0314	1.2768	1.7097	3.4807	3.0745	3.5521	8.8986	6.4358	8.8986
	Performance gains ($a - b/a$) (%)								
2	28.73	32.74	28.44	29.42	32.98	29.42	29.84	33.01	29.84
3	40.69	46.88	33.31	41.85	47.36	41.84	42.65	47.46	42.65
4	47.66	54.83	32.82	48.99	55.57	47.94	50.09	55.76	50.09

Notes. SCV is set to 0.5 and c_I=1, that is, idle and waiting time are valued equally importantly. Overtimes are computed after optimization with the aforementioned settings using $\mathrm{T} = n/s$.

Figure 7. Impact of the Cost Parameter on the Optimal Appointment Schedule



Note. Extending the setting of Figure 1 (n=24 and SCV=0.25), optimal appointment schedules when pooling two (left) or four (right) servers while varying c_I in the objective function of Display (20).

moving away from dominating transient effects, we conclude that the performance improvement for idle and overtime mimics those reported earlier. This is backed by our analysis in Section 6 wherein the performance improvement in the long run turns out to be a factor of $\sqrt{s}$.

5.4. No-Shows

No-shows are recognized as an important environmental factor to be accounted for in appointment schedules (e.g., Ho and Lau 1992 and Cayirli and Veral 2003) and are also studied in detail for multiserver systems in Zacharias and Pinedo (2017). To accommodate for the impact of no-shows, which occur with probability q, in our framework, we only have to adapt the initial probability vectors. With probability $(1-q)$ we have a client whose service time is approximated by a phase-type distribution and with probability q a client who does not show-up at all.

Define $\boldsymbol{\alpha}^q_{s,j} = (\boldsymbol{\alpha} \otimes \cdots \otimes \boldsymbol{\alpha})(1-q)^j q^{(s-j)} \binom{s}{j}$, where the Kronecker product is applied exactly j times. Now the start vector reads as $\boldsymbol{\alpha}^q_s = (\boldsymbol{\alpha}^q_{s,s}, \boldsymbol{\alpha}^q_{s,s-1}, \ldots, \boldsymbol{\alpha}^q_{s,1})$, and when a subsequent client *should* arrive, the lines in Equation (15) that define the recursion are replaced by

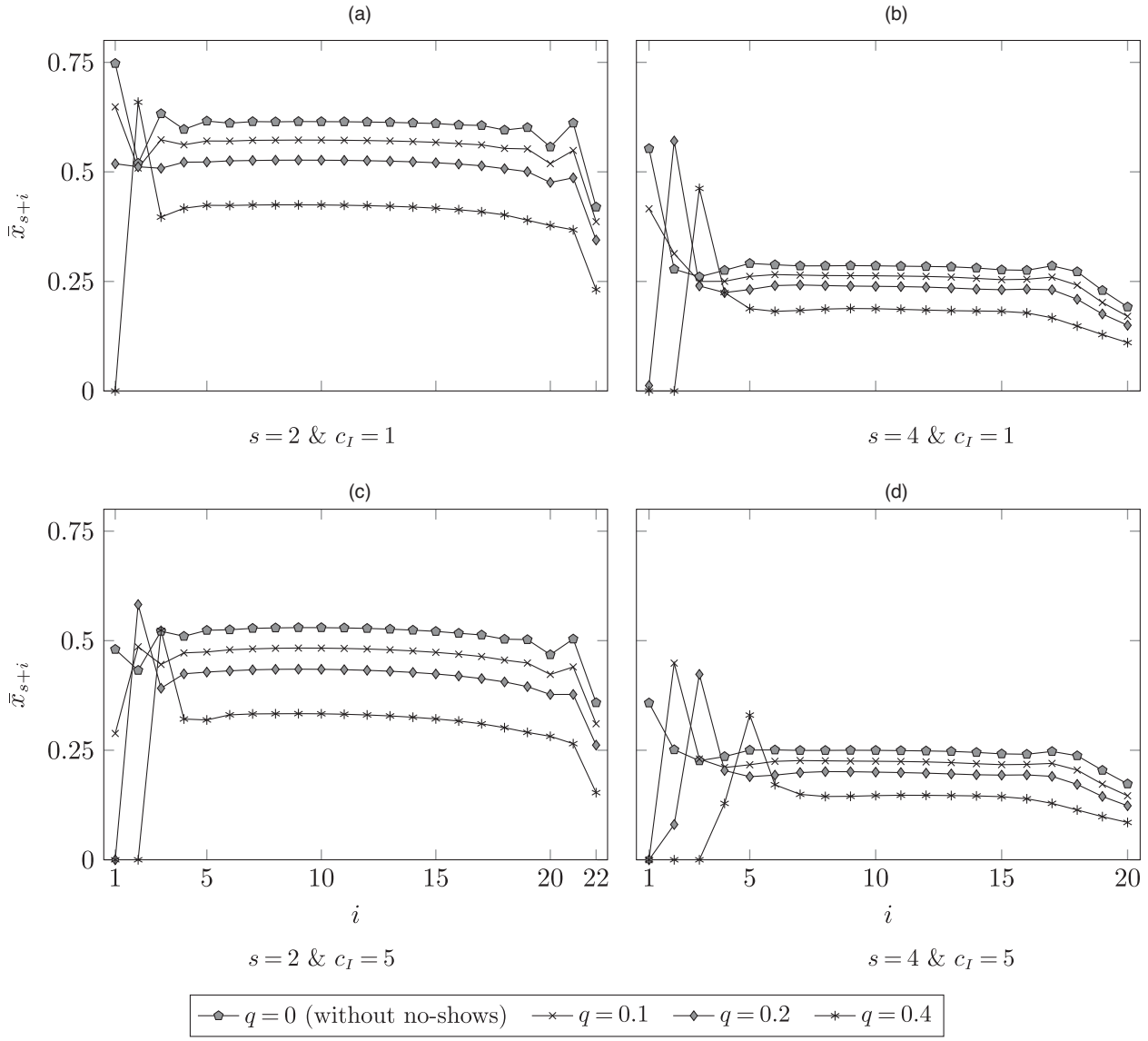
$$\boldsymbol{\alpha}^q_{s+i} = (1-q)\left(f\left(\boldsymbol{p}^{(s+i-1)}(x_{s+i}), \boldsymbol{\alpha}\right)\right) + q\left(\mathbf{0}_{k^s}, \boldsymbol{p}^{(s+i-1)}(x_{s+i})\right), \tag{23}$$

Table 3. Comparison of Single Server and Pooled Server System Performance for SCV = 0.5 and $c_I = 5$

	n = 12			n = 24			n = 48		
	Expected costs in an optimized system of s single servers (a)								
s	$\sum \mathbb{E}I_i$	$\sum \mathbb{E}W_i$	$\sum \mathbb{E}O_\ell$	$\sum \mathbb{E}I_i$	$\sum \mathbb{E}W_i$	$\sum \mathbb{E}O_\ell$	$\sum \mathbb{E}I_i$	$\sum \mathbb{E}W_i$	$\sum \mathbb{E}O_\ell$
2	0.7593	9.1779	1.5273	2.6319	21.3441	2.8663	7.3178	44.2482	7.3178
3	0.4742	7.6170	1.7530	1.9720	19.9379	2.8714	6.1683	43.7249	6.2061
4	0.3085	6.2159	1.9732	1.5185	18.3558	3.0546	5.2639	42.6882	5.7325
	Expected costs in an optimized system of s pooled servers (b)								
2	0.5876	6.3901	1.1895	1.9504	14.7088	2.1586	5.2899	30.3613	5.2899
3	0.3306	4.3396	1.2471	1.2623	11.1103	1.8984	3.7485	24.1044	3.8288
4	0.2042	3.0992	1.3759	0.8922	8.8466	1.8916	2.8589	20.2124	3.2533
	Performance gains $(a-b/a)$ (%)								
2	22.62	30.38	22.12	25.89	31.09	24.69	27.71	31.38	27.71
3	30.29	43.03	28.86	35.99	44.28	33.89	39.23	44.87	38.31
4	33.81	50.14	30.27	41.25	51.80	38.07	45.69	52.65	43.25

Notes. Idle time is valued as five times more important than waiting. Overtimes are computed after optimization with the aforementioned settings using T = n/s.

Figure 8. Impact of No-Shows on the Optimal Appointment Schedule



Note. Extending the setting of Figure 1 (n = 24 and SCV = 0.25), varying no-show probabilities q for different numbers of pooled servers s and cost parameters c_I

using the $f(\cdot, \boldsymbol{\alpha})$ function as also defined in Equation (15). Analyzing Equation (23) shows that with probability $(1-q)$, the $(s+i)$ th client is added to the system, either in service or in the queue, and with probability q, the system remains unchanged, as the client did not show up. Using the earlier transition matrices, the possible transitions remain unchanged. We conclude that $\text{PH}(\boldsymbol{\alpha}^q_{s+i}, \boldsymbol{S}_{s+i})$ describes the system after the $(s+i)$ th client's arrival on which we apply the developed machinery; Equation (22) should be adapted accordingly to account for no-shows by subtracting $(1-q)\sum_{i=1}^{n} \mathbb{E}B_i$ instead.

On top of the baseline setting of Figure 1, we implemented no-shows to occur with probabilities 10%, 20%, and 40% in Figure 8 for different numbers of servers and cost parameters. The no-show levels chosen cover the range reported in Cayirli et al. (2012). Because of the occurrence of no-shows, a scheduled client effectively brings in less work, so we observe that the dome is pushed downward by approximately the fraction with which no-shows occur. This happens irrespectively of the weight chosen for idle time in the objective function, c_I.

More interesting, with no-shows we see that at the beginning the optimization counteracts the possibility of server idling; the first interarrival(s) decrease and even become 0, which means starting a session with more clients scheduled than the number of servers s. This overbooking will typically be followed by a relatively high interarrival time, as several panels in

Figure 8 clearly show. In single-server equivalents, an overbooking to the first slot only occurs at a no-show probability of 40%, after which the dome-shaped pattern commences. Overbooking is more persistent when more servers are pooled, as for multiple servers it hedges against the possibility of idling without costing more in terms of waiting time, because the queue is serviced by more than one server.

At the end of the schedules in Figure 8, we see that no-shows dampen the idiosyncratic fluctuations. Comparing the top with the bottom panels, we observe that the schedule has tightened, and overbooking has occurred on more occasions and with greater severity. This is the result of valuing idle time more importantly in the objective function. Still, we see that also in these cases a relatively long interarrival time follows, demonstrating that the effect of no-shows cannot be dismissed.

6. Multiserver Appointment Scheduling in Steady-State Settings

As noted in Cayirli and Veral (2003), steady state is never reached in a real clinical session with a small number of clients. However, as seen in the various figures that depict the optimal transient schedule, the middlemost values converge quickly to a stationary plateau. Therefore the steady-state counterparts of these systems provide relevant insight for the transient setting and are appropriate for the majority of clients. Indeed, studying the clinic in steady state has lead to a fruitful stream of research (Lindley 1952, Jansson 1966, Swisher et al. 2001, Kuiper et al. 2017).

The reason why the plateau of each dome converges to the corresponding optimal steady-state interarrival time is intuitive, because transient effects found at the start and end of a session favour service providers. Therefore waiting time is valued less in a transient setting. In steady state these transient effects are neglected, as the session has run forever, so that the extent of waiting time propagates to possibly all clients, which results in the optimal stationary interarrival time bounding the plateau of the dome from above.

In a stationary analysis, the transient effects at the start and toward the end of the session are neglected. As a result, the objective function reduces to an elegant combination of idle time, which can be expressed as excessive capacity, $sx - \mathbb{E}B$, versus clients' waiting times, $\mathbb{E}W$. Let x be the steady-state interarrival time; then, without loss of generality, given $c_W \equiv 1$, we find

$$\arg\min_x c_I\,\mathbb{E}I(x) + c_W\,\mathbb{E}W(x) = \arg\min_x c_I\,sx + \mathbb{E}S(x), \tag{24}$$

with $S(x)$ as the sojourn-time distribution that obviously depends on the stationary interarrival time x. Given the fact that we minimize an objective function with $c_I < \infty$, the utilization will never reach a fully loaded system so that the existence of a steady-state solution is guaranteed (Kiefer and Wolfowitz 1955, Whitt 1982). In the following sections we propose two methods that provide solutions in this limiting regime.

6.1. Phase-Type Approach

The transition matrix that is obtained in Section 5 can be used to compute the equilibrium distribution, $\boldsymbol{\pi}$, by considering the system's embedded Markov chain. Consider the system slightly before a new client's arrival: the system jumps to a state with an additional client; then after x amount of time, the system should have returned to its equilibrium distribution. The full system contains an infinite number of states, but because the probability of having n clients in the system goes rapidly to 0 as n grows large, we cut off the number of clients to be allowed in the system at n; in our experiments, $n = 40$ has worked well. This allows the computation of the steady-state distribution for a given x by solving

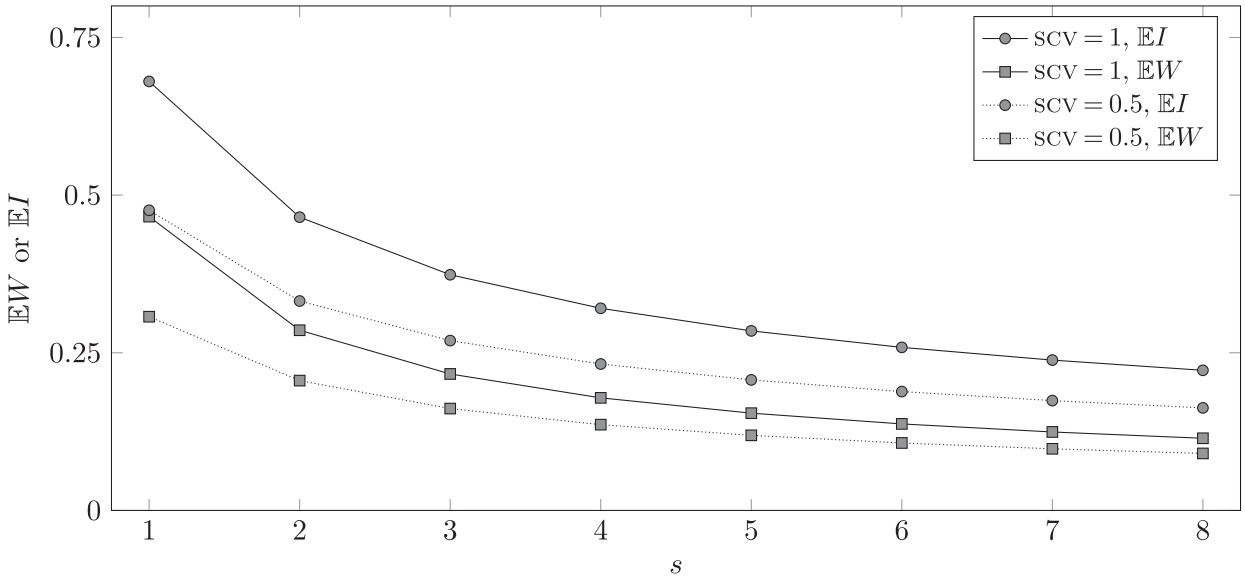
$$\boldsymbol{\pi}_0^{(n)} = f\left(\boldsymbol{\pi}^{(n)}, \boldsymbol{\alpha}\right)\boldsymbol{P}_n, \text{where}$$
$$\boldsymbol{P}_n = \left[\exp\left(\boldsymbol{S}_{n+1}x\right), 1 - \exp\left(\boldsymbol{S}_{n+1}x\right)\mathbf{1}\right],$$

with $\boldsymbol{\pi}_0^{(n)} := \left(\boldsymbol{\pi}^{(n)}, \pi_0^{(n)}\right)$ and thus $\boldsymbol{\pi}^{(n)}$ having similar states as $\boldsymbol{p}^{(n)}(x)$ of Equation (12), so that the function $f(\cdot, \boldsymbol{\alpha})$ as defined in Equation (15) can be applied. The transition matrix extended with the transitions to the empty state can subsequently be used to obtain the steady-state probabilities for each of the states in the vector $\boldsymbol{\pi}_n$ by solving the aforementioned system, cutting off the transitions to states with $n + 1$ clients and imposing the normalization equation $\boldsymbol{\pi}_0^{(n)}\mathbf{1} = 1$.

This nonsingular system for the steady state can solved by exploiting, for example, MATLAB's built-in routines to compute the matrix exponential and solve the system of linear equations. Then $\boldsymbol{\pi}_n$ can be used as the initial probability vector $\boldsymbol{\alpha}_n$, and the transition matrix is just $\boldsymbol{S}_n$, so that the performance metrics of Section 4.2 can be computed and filled into the objective function of (24). Optimization over x provides us with the optimal stationary interarrival time $\bar{x}_{\text{pt}}$.

In Figure 9, where an equal cost ratio is chosen ($c_I = 1$), we observe a decreasing pattern in the expected idle and waiting times when the number of servers increases. As can be seen, the marginal decreases in idle and waiting times are less for higher values of s, which is in line with the theoretical result for waiting times in $G/G/s$ queues derived in Weber (1980). Furthermore, we see that the performance gain achieved by pooling is greater when the SCV is larger; SCV varies from 0.5 (reflecting healthcare environments) to 1 (exponential service times).

Figure 9. Stationary Expected Idle and Waiting Times



Note. Expected idle and waiting times corresponding to the optimal stationary solutions for $s \in \{1, \dots, 8\}$ using the embedded Markov chain from the phase-type approach; $c_I = 1$ and SCV is set to 0.5 or 1.

In the next section we make an interesting connection with appointment scheduling studied in a heavy-traffic regime and compare the stationary solutions obtained by the phase-type approach with those obtained under heavy traffic for which elegant expressions are derived.

6.2. Robust Schedules

The goal in many appointment scheduling problems is to reduce the waiting time while keeping utilization at a high level, so that no capacity is wasted. Consequently, the idle time should be low so that the load of the system is close to 1. This observation warrants consideration of the problem in a heavy-traffic regime. In our cases, this entails a *steady-state* interarrival time x only slightly larger than the mean service time divided by the number of servers (i.e., $\mathbb{E}B/s$).

In fact, when s increases, our steady-state results using the method outlined in the preceding converge to the results obtained in the heavy-traffic regime. Because the variability accrued is spread over multiple servers, expected waiting and idle times will be lower than in an equivalent system of single servers. Consequently, higher utilizations are achieved; the interarrival times are much closer to the service rate. Moreover, as for the single-server case, as idle time is evaluated as being more important, interarrival times tend to the service rate; see also our numerical results in the previous section. So it should generally hold that in these two scenarios, a heavy-traffic approximation will provide an accurate approximation.

Using the steady-state result for the $G/G/s$ under a heavy-traffic regime (see, e.g., theorem 2 in Köllerström (1974) or section 5 in Whitt (1983)), we have

$$\frac{2s(sx - \mathbb{E}B)}{\text{scv}} W(x) \sim \text{Exp}(1), \quad \text{when} \quad x \downarrow \mathbb{E}B/s.$$

Specifically, because the means are normalized, we can rewrite the optimization problem of (24) for given c_I, which is easily solved using straightforward calculus:

$$\bar{x}_{\text{ht}} = \frac{1}{s}\left(1 + \sqrt{\frac{\text{scv}}{2c_I s}}\right) = \underset{x \in (\mathbb{E}B/s, \infty)}{\arg\min}\, c_I(sx - 1) + \frac{\text{scv}}{2s(sx - 1)}.$$

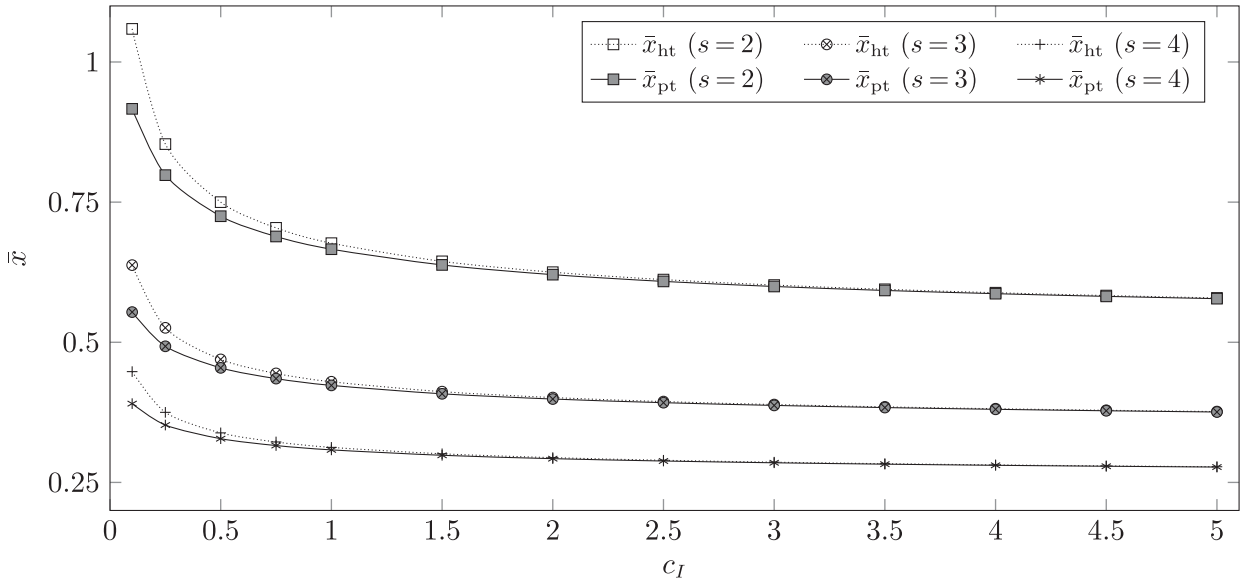
Furthermore, because the second derivative in x is positive, a global optimum is guaranteed. We observe that the margin to account for randomness on top of the average service rate is a multiplication of $1/\sqrt{s}$, which tends to 0 as s increases.

Studying the optimal solution as a function of the interarrival times, we observe in Figure 10 that for c_I values greater than 1, given that SCV is not too large, the heavy-traffic solutions and the solutions obtained by the phase-type approximation are nearly the same. In fact, the graphs depicted in Figure 11 show that for low SCV values—say, SCV < 1—heavy-traffic solutions provide accurate approximations even when c_I equals 1.

In addition, the corresponding expected idle and waiting times provide insight into the patterns observed in Figure 9 and are explicitly given by

$$\mathbb{E}I_{\text{ht}} = \sqrt{\frac{\text{scv}}{2s\, c_I}} \quad \text{and} \quad \mathbb{E}W_{\text{ht}} = \sqrt{\frac{\text{scv}\, c_I}{2s}}. \tag{25}$$

Figure 10. Impact of the Cost Parameter on the Optimal Stationary Solution



Notes. SCV = 0.5, and the cost ratio varies from 5:1 to 1:10. Solutions computed using the phase-type approach $\bar{x}pt$ or the heavy-traffic approximation $\bar{x}ht$.

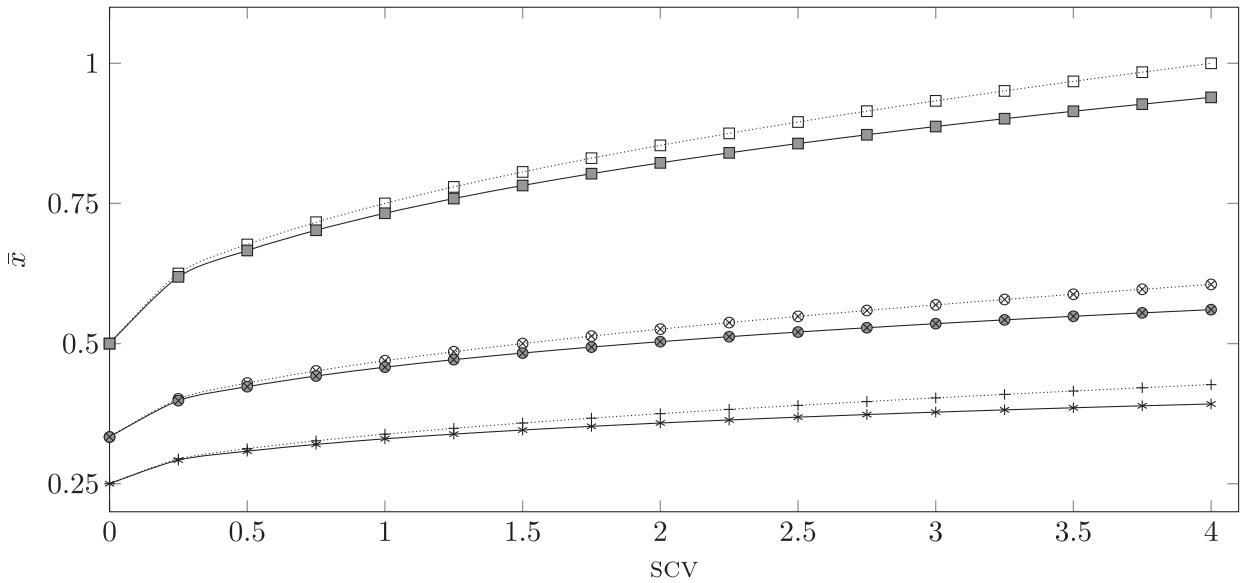
For a decision maker, these expressions reveal how the operational benefits of pooling can be concentrated on either one of the performance dimensions. If utilization is the primary concern, idle times can even be reduced by a factor of s, keeping waiting times the same, by multiplying c_I with s when servers are pooled. Conversely, when waiting times are of paramount importance, c_I should be divided by s.

Note that the setup discussed here is robust against misspecification of the distribution and only depends on the first two moments. Moreover, the waiting times in heavy traffic coincide with the conjectured upper bound on the waiting times for multiserver queues (Daley 1997), equating the heavy-traffic results to those obtained while minimizing the objective function under the worst-case distribution (cf. Mak et al. (2015)).

7. Conclusion and Discussion

In this work, the intensively studied single-server appointment schedule problem is extended to a multiserver setting. The multiserver setting introduces many

Figure 11. Impact of the SCV on the Optimal Stationary Solution



Notes. c_{1I} = 1 and the SCV ranges from 0 to 4. Solutions computed using the phase-type approach $\bar{x}pt$ or the heavy-traffic approximation $\bar{x}ht$. The solutions are differentiated according to the same legend as in Figure 10.

obstacles to tracking and convexity that are not present when only one server is considered. As a result, this case is not studied in an analytical manner in the literature, although it features in many service systems. We offer a computational approach to the multiserver setting, relying on the tractability of phase-type distributions, which is employed to gain insight in the optimal solution in multiserver appointment schedules.

The impact of deviating from the class of phase-type distributions in the transient setting would be a logical avenue for further research. Furthermore, it remains open whether in each instance a global optimum is found. By the fact that our optimizations converged to the same solution when varying start vectors—and that, in corresponding heavy-traffic regimes, convexity can be shown—we have strong reason to believe that the problems considered are convex. Of course, this remains a challenging line of research for queueing theorists. Finally, confining the solutions over discrete points in time might be an extension that might be of particular value to practice; such a model is studied for a single server in Zacharias and Yunes (2020).

A well-known result of single-server appointment scheduling is the dome-shaped pattern. Contrasting with the multiserver setting, there are some discrepancies that arise at the start and the end of a session for cases in which SCV is below 1—typical in many service systems such as in healthcare (Cayirli and Veral 2003). These patterns arise as a result of multiple servers starting synchronously, each serving one client. The apparent pattern is characterized by a damped bullwhip, which converges in the middle to a steady-state plateau as the randomness in the service times gradually suppresses these effects. At the end of the session, the optimization tries to achieve a synchronous ending of the servers, which culminates in a similar but reversed pattern in the optimal interarrival times.

These patterns gradually decrease when SCV tends to 1; if SCV equals 1 (or is higher), the exponential distribution (or a mixture) is used for which a dome-shaped pattern appears for multiple servers as well. The inclusion of overtime has the same impact as incorporating idle time in the optimization. If idle time is valued more importantly, pooled appointment schedules are tightened, and the atypical start and end patterns of sessions are damped. Experiments further reveal that including no-shows lowers the plateau by the fraction in which they occur, and the striking multiserver patterns at the end are damped. At the start of the session, no-shows result in overbooking, which, in the case of low SCV values, is followed by an extremely large interarrival time. This idiosyncratic initialization behavior persists even for higher cost parameters.

In addition, a relevant selection of cases provides practitioners insight into the decisions and trade-offs that arise. In detail, the performance gains of pooled appointment systems versus equivalent systems of singly operating servers are analyzed in a framework that incorporates service-time variation. Focusing on the waiting times, the analysis shows that the expected waiting times are reduced by about 31% when two servers are pooled and by an astounding 53% for four servers. Similar double-digit reductions are reported for expected idle time and overtime. For example, in healthcare, the comparison unravels the implicit cost of continuity of care.

The optimal stationary schedule is also studied, which approximates the dome-shaped plateau arising in the transient setting. Notice that with more servers, the variation in the system is reduced, and thus a heavy-traffic regime becomes an appropriate modeling framework. The optimal solution in this regime has an algebraic expression. Moreover, this regime elucidates that the expected idle or waiting times decrease by a factor of $\sqrt{s}$ when s servers are pooled. Finally, it is likely that the heavy-traffic solution coincides with the conjectured upper bound (Daley 1997) on the expected waiting time in a multiserver setting and is thus robust. As such, this study provides a comprehensive account of the multiserver appointment scheduling problem, which was as yet unaccounted for in the field.

Acknowledgments

The authors gratefully acknowledge the insightful and constructive remarks from the associate editor and the review team.

References

Ahmadi-Javid A, Jalali Z, Klassen KJ (2017) Outpatient appointment systems in healthcare: A review of optimization studies. *Eur. J. Oper. Res.* 258(1):3–34.

Alvarez-Oh HJ, Balasubramanian H, Koker E, Muriel A (2018) Stochastic appointment scheduling in a team primary care practice with two flexible nurses and two dedicated providers. *Service Sci.* 10(3):241–260.

Asmussen S, Nerman O, Olssen M (1996) Fitting phase-type distributions via the EM algorithm. *Scandinavian J. Statist.* 23(4): 419–441.

Begen MA, Queyranne M (2011) Appointment scheduling with discrete random durations. *Math. Oper. Res.* 36(2):240–257.

Bosch PMV, Dietz DC (2001) Scheduling and sequencing arrivals to an appointment system. *J. Service Res.* 4(1):15–25.

Cayirli T, Veral E (2003) Outpatient scheduling in healthcare: A review of literature. *Production Oper. Management* 12(4):519–549.

Cayirli T, Yang K, Quek S (2012) A universal appointment rule in the presence of no-shows and walk-ins. *Production Oper. Management* 21(4):682–697.

Côté MJ, Stein WE (2007) A stochastic model for a visit to the doctor's office. *Math. Comput. Model.* 45(3–4):309–323.

Cox TF, Birchall JP, Wong H (1985) Optimising the queuing system for an ear, nose and throat outpatient clinic. *J. Appl. Statist.* 12(2):113–126.

Daley DJ (1997) Some results for the mean waiting-time and workload in $GI/GI/k$ queues. Dshalalow JH, ed. *Frontiers in Queueing:*

Models and Applications in Science and Engineering (CRC Press, Boca Raton, FL), 35–59.
Denton B, Gupta D (2003) A sequential bounding approach for optimal appointment scheduling. *IIE Trans.* 35(11):1003–1016.
Denton BT, Miller AJ, Balasubramanian HJ, Huschka TR (2010) Optimal allocation of surgery blocks to operating rooms under uncertainty. *Oper. Res.* 58(4):802–816.
De Vuyst S, Bruneel H, Fiems D (2014) Computationally efficient evaluation of appointment schedules in healthcare. *Eur. J. Oper. Res.* 237(3):1142–1154.
El-Sharo M, Zheng B, Yoon SW, Khasawneh MT (2015) An overbooking scheduling model for outpatient appointments in a multi-provider clinic. *Oper. Res. Health Care* 6:1–10.
Green LV, Savin S, Lu Y (2013) Primary care physician shortages could be eliminated through use of teams, nonphysicians, and electronic communication. *Health Affairs* 32(1):11–19.
Gupta D, Denton B (2008) Appointment scheduling in healthcare: Challenges and opportunities. *IIE Trans.* 40(9):800–819.
Harel A (1990) Convexity results for single-server queues and for multiserver queues with constant service times. *J. Appl. Probab.* 27(2):465–468.
Harper PR, Gamlin H (2003) Reduced outpatient waiting times with improved appointment scheduling: A simulation modelling approach. *OR Spectrum* 25(2):207–222.
Hassin R, Mendel S (2008) Scheduling arrivals to queues: A single-server model with no-shows. *Management Sci.* 54(3):565–572.
Ho C-J, Lau H-S (1992) Minimizing total cost in scheduling outpatient appointments. *Management Sci.* 38(12):1750–1764.
Hulshof PJ, Kortbeek N, Boucherie RJ, Hans EW, Bakker PJ (2012) Taxonomic classification of planning decisions in healthcare: A structured review of the state of the art in OR/MS. *Health Systems* 1(2):129–175.
Jansson B (1966) Choosing a good appointment system—A study of queues of the type (*D*, *M*, 1). *Oper. Res.* 14(2):292–312.
Kaandorp G, Koole G (2007) Optimal outpatient appointment scheduling. *Health Care Management Sci.* 10(3):217–229.
Kiefer J, Wolfowitz J (1955) On the theory of queues with many servers. *Trans. Amer. Math. Soc.* 78(1):1–18.
Klassen KJ, Yoogalingam R (2009) Improving performance in outpatient appointment services with a simulation optimization approach. *Production Oper. Management* 18(4):447–458.
Klassen KJ, Yoogalingam R (2014) Strategies for appointment policy design with patient unpunctuality. *Decision Sci.* 45(5): 881–911.
Klassen KJ, Yoogalingam R (2019) Appointment scheduling in multistage outpatient clinics. *Health Care Management Sci.* 22(2):229–244.
Köllerström J (1974) Heavy traffic theory for queues with several servers. i. *J. Appl. Probab.* 11(3):544–552.
Kuiper A, Mandjes M (2015) Appointment scheduling in tandem-type service systems. *Omega* 57(Part B):145–156.
Kuiper A, Kemper B, Mandjes M (2015) A computational approach to optimized appointment scheduling. *Queueing Systems* 79(1):5–36.
Kuiper A, Mandjes M, de Mast J (2017) Optimal stationary appointment schedules. *Oper. Res. Lett.* 45(6):549–555.
Kuiper A, Mandjes M, de Mast J, Brokkelkamp R (2021) A flexible and optimal approach for appointment scheduling in healthcare. *Decision Sci.*, ePub ahead of print May 4, https://doi.org/10.1111/deci.12517.
Lindley D (1952) The theory of queues with a single server. *Math. Proc. Cambridge Philos. Soc.* 48(2):277–289.
Mak HY, Rong Y, Zhang J (2015) Appointment scheduling with limited distributional information. *Management Sci.* 61(2): 316–334.
Mandelbaum A, Momčilović P, Trichakis N, Kadish S, Leib R, Bunnell CA (2020) Data-driven appointment-scheduling under uncertainty: The case of an infusion unit in a cancer center. *Management Sci.* 66(1):243–270.
Neuts M (1981) *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach* (Johns Hopkins University Press, Baltimore).
Rising EJ, Baron R, Averill B (1973) A systems analysis of a university-health-service outpatient clinic. *Oper. Res.* 21(5):1030–1047.
Robinson L, Chen R (2011) Estimating the implied value of the customer's waiting time. *Manuf. Serv. Oper. Management* 13(1): 53–57.
Saremi A, Jula P, ElMekkawy T, Wang GG (2013) Appointment scheduling of outpatient surgical services in a multistage operating room department. *Internat. J. Production Econom.* 141(2): 646–658.
Shanthikumar JG, Yao DD (1991) Strong stochastic convexity: Closure properties and applications. *J. Appl. Probab.* 28(1):131–145.
Sickinger S, Kolisch R (2009) The performance of a generalized Bailey–Welch rule for outpatient appointment scheduling under inpatient and emergency demand. *Health Care Management Sci.* 12(4):408–419.
Soltani M, Samorani M, Kolfal B (2019) Appointment scheduling with multiple providers and stochastic service times. *Eur. J. Oper. Res.* 277(2):667–683.
Swisher JR, Jacobson SH, Jun J, Balci O (2001) Modeling and analyzing a physician clinic environment using discrete-event (visual) simulation. *Comput. Oper. Res.* 28(2):105–125.
Tijms H (1986) *Stochastic Modelling and Analysis: A Computational Approach*, Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics (John Wiley & Sons, Chichester, UK).
Van Dijk NM, Van der Sluis E (2008) To pool or not to pool in call centers. *Production Oper. Management* 17(3):296–305.
Wang PP (1993) Static and dynamic scheduling of customer arrivals to a single-server system. *Naval Res. Logist.* 40(3):345–360.
Wang PP (1997) Optimally scheduling n customer arrival times for a single-server system. *Comput. Oper. Res.* 24(8):703–716.
Weber RR (1980) On the marginal benefit of adding servers to $G/GI/m$ queues. *Management Sci.* 26(9):946–951.
Welch J, Bailey N (1952) Appointment systems in hospital outpatient departments. *Lancet* 259(6718):1105–1108.
White DL, Froehle CM, Klassen KJ (2011) The effect of integrated scheduling and capacity policies on clinical efficiency. *Production Oper. Management* 20(3):442–455.
Whitt W (1982) Existence of limiting distributions in the $GI/G/s$ queue. *Math. Oper. Res.* 7(1):88–94.
Whitt W (1983) Comparison conjectures about the M/G/s queue. *Oper. Res. Lett.* 2(5):203–209.
Zacharias C, Pinedo M (2017) Managing customer arrivals in service systems with multiple identical servers. *Manufacturing Service Oper. Management* 19(4):639–656.
Zacharias C, Yunes T (2020) Multimodularity in the stochastic appointment scheduling problem with discrete arrival epochs. *Management Sci.* 66(2):744–763.
Zhou S, Yue Q (2019) Appointment scheduling for multi-stage sequential service systems with stochastic service durations. *Comput. Oper. Res.* 112(December):104757.