Management Science

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

Managing Appointment Booking Under Customer Choices

Nan Liu, Peter M. van de Ven, Bo Zhang

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

MANAGEMENT SCIENCE
Vol. 65, No. 9, September 2019, pp. 4280–4298
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

Managing Appointment Booking Under Customer Choices

Nan Liu,[a] Peter M. van de Ven,[b] Bo Zhang[c]

[a] Operations Management Department, Carroll School of Management, Boston College, Chestnut Hill, Massachusetts 02467; [b] Centrum Wiskunde & Informatica, 1098 XG Amsterdam, Netherlands; [c] IBM Research AI, Yorktown Heights, New York 10598

Contact: nan.liu@bc.edu, http://orcid.org/0000-0001-7644-7341 (NL); p.m.van.de.ven@cwi.nl (PMvdV); zhangbo@us.ibm.com (BZ)

Received: September 15, 2016
Revised: December 20, 2017; May 28, 2018
Accepted: June 9, 2018
Published Online in Articles in Advance: April 26, 2019

https://doi.org/10.1287/mnsc.2018.3150

Copyright: © 2019 INFORMS

Abstract. Motivated by the increasing use of online appointment booking platforms, we study how to offer appointment slots to customers to maximize the total number of slots booked. We develop two models, *nonsequential* offering and *sequential* offering, to capture different types of interactions between customers and the scheduling system. In these two models, the scheduler offers either a single set of appointment slots for the arriving customer to choose from or multiple sets in sequence, respectively. For the nonsequential model, we identify a static randomized policy, which is asymptotically optimal when the system demand and capacity increase simultaneously, and we further show that offering all available slots at all times has a constant factor of two performance guarantee. For the sequential model, we derive a closed form optimal policy for a large class of instances and develop a simple, effective heuristic for those instances without an explicit optimal policy. By comparing these two models, our study generates useful operational insights for improving the current appointment booking processes. In particular, our analysis reveals an interesting equivalence between the sequential offering model and the nonsequential offering model with perfect customer preference information. This equivalence allows us to apply sequential offering in a wide range of interactive scheduling contexts. Our extensive numerical study shows that sequential offering can significantly improve the slot fill rate (6%–8% on average and up to 18% in our testing cases) compared with nonsequential offering. Given the recent and ongoing growth of online and mobile appointment booking platforms, our research findings can be particularly useful to inform user interface design of these booking platforms.

History: Accepted by Gad Allon, operations management.
Supplemental Material: The online appendix is available at https://doi.org/10.1287/mnsc.2018.3150.

Keywords: service operations management • customer choice • appointment scheduling • Markov decision process • asymptotically optimal policy

1. Introduction

Appointment scheduling is a common tool used by service firms (e.g., tech support, beauty services, and healthcare providers) to match their service capacity with uncertain customer demand. With the widespread use of the internet and smartphones, customers often resort to online channels when searching for information and reserving services. To keep up with customers' preferences and needs, many service organizations have developed online appointment scheduling portals. For instance, Teachers Insurance and Annuity Association of America (TIAA) allows its clients to book appointments with their financial consultants online. There is also a rising number of online service reservation companies that offer online appointment booking software or applications as a service for (small) businesses. Examples include zocdoc.com for medical appointments, opentable.com for dinner reservations, mindbodyonline.com for fitness classes, booker.com for spa services, and salonultimate.com for haircuts.

The interfaces of these online appointment booking systems vary. Some are more toward one-shot offering (i.e., a single list of available appointments is shown on a single screen for customers to choose from). Others offer a small number of options to start, and customers must press "more" or "next" to view additional appointments that are available. This way of scheduling resembles the traditional telephone-based scheduling process, in which the scheduling agent may reveal availability of appointment slots in a sequential manner. Such a sequential way of displaying options is often seen on mobile devices with a small screen as well.

Our research is motivated by these various ways of appointment booking, and we seek to understand how a service provider can best use these (online) appointment booking systems. In scheduling practice, service providers first predetermine for each day an appointment template, which specifies the total number of slots, the length of each slot, and characteristics of customers (e.g., nature of the visit) to be scheduled for each slot. For instance, in a gym setting, one has to determine the number of classes and their capacity, and in healthcare, the service provider first determines the number of patients that a clinician will see that day and at what times. With an appointment template in place, service providers then decide how to assign incoming customer requests to the available slots—nowadays, this process is often done via online appointment scheduling as mentioned above. The relevant performance metric for this process is the *fill rate* (i.e., the fraction of slots in a template booked before the scheduling process closes). Although the fill rate is not equivalent to the eventual capacity utilization because of various postscheduling factors (e.g., cancellations, no shows, and walk ins), it is the first, and in many cases, the most important step to achieving a high utilization (and thus, a high revenue), and it is the objective of the research presented in this paper.

Our focus is on modeling the scheduling process and developing stochastic dynamic optimization models to inform appointment scheduling decisions in the presence of customer choice behavior. Notwithstanding the surge of interest in service operations management in the past decade, basic single-day, choice-based dynamic decision models are absent for a broad class of real-world scheduling systems. To our knowledge, the existing operations research and management literature on this type of dynamic appointment scheduling is very limited; most, if not all, related research assumes that customers reveal their preferences first and that the scheduler decides to accept or reject (e.g., Gupta and Wang 2008, Wang and Gupta 2011). However, as discussed above, in many real-world scheduling platforms, the system (i.e., the scheduler) offers its availability to customers to choose from either in a one-shot format or in a sequential manner, with no explicit knowledge on customer preferences. Customers interact with the scheduler in ways that have not been fully explored in the literature. This paper fills a gap in the literature by proposing the first choice-based dynamic optimization models for making scheduling decisions in systems where customers are allowed to choose among offered appointment slots from an established appointment template. We show how the current appointment booking processes can be improved by developing optimality results, heuristics, and managerial insights in the context of the proposed models.

We propose and study two models for the interaction between customers and the service provider. The first one is referred to as the *nonsequential offering* model. In this model, the scheduler offers a single set of appointment slots to each customer. If some of the offered slots are acceptable to the customer, she chooses one from them; otherwise, she does not book an appointment. This simple, one-time interaction resembles the mechanism of many online appointment systems that provide one-shot offerings, and our results on this model have direct implications on how to manage these systems. Our second model is a *sequential offering* model, in which the scheduler may offer several sets of appointment choices in a sequential manner. This is motivated by (1) web-based appointment applications designed to reveal only a small number of appointment options one web page at a time (e.g., mobile-based appointment applications) and (2) the traditional telephone-based scheduling process, in which the scheduler offers appointment slots sequentially. This second model is stylized in the sense that it does not incorporate customer recall behavior (i.e., a customer choosing a previously offered slot after viewing more offers), which is allowed in both online and phone-based scheduling. Our goal here is to glean insights on how the fill rate can be improved by "smarter" sequencing when sequential offering is part of the scheduling process.

For both cases, we are interested in which slots to offer to improve and maximize the fill rate. We answer this question by investigating the optimal offering policy using Markov decision processes (MDPs) as well as by discussing heuristics. Intuitively, sequential offering should lead to a higher fill rate than nonsequential offering, because sequential offering gives the scheduler more control over the service capacity. We are also interested in how much improvement a service provider can get by switching from nonsequential scheduling to sequential scheduling. We answer this question by comparing the fill rates resulting from these two models, and the gap in the fill rates represents the "value" of sequential offering.

We make the following main contributions to the literature.

To the best of our knowledge, our paper is the first to study and compare two main scheduling paradigms, nonsequential (online) and sequential (mobile or telephone based), used in the service industries.

For the nonsequential offering model, we characterize the optimal policy for a few special instances and show that the optimal policy can be highly complex in general. We then identify a static randomized policy (arising from solving a single linear program), which is *asymptotically optimal* when the system demand and capacity increase by the same factor. We further show that the offering-all policy (i.e., offering all available capacity throughout) has a *constant factor of two performance guarantee*.

For the sequential offering model, we show that there exists an optimal policy that offers slot types one at a time based on their marginal values. We are able to determine these values for a broad class of model instances, which leads to a *closed form* optimal policy in these cases. For model instances without an explicit optimal policy, we develop a simple, effective heuristic.

We show that a sequential offering model is equivalent to a nonsequential offering model with perfect customer preference information. This equivalence ensures that sequential offering can be optimally applied in various interactive scheduling contexts, in particular when customer-scheduler interaction can (partially) reveal customer preference information during the appointment booking process.

Via extensive numerical experiments, we show that the offering-all policy and the heuristic developed for sequential offering work remarkably well in their respective settings, and thus, they can serve as effective approximate scheduling policies for practical use. We also show that, by switching from nonsequential to sequential offering, the slot fill rate can be significantly improved (6%–8% on average and up to 18% in our testing cases).

The remainder of the paper is organized as follows. Section 1.1 briefly reviews the relevant literature. Section 2 introduces the common capacity and demand model that will be used in both the nonsequential and sequential settings. Sections 3 and 4 discuss the nonsequential offering case and the sequential offering case, respectively. Section 5 presents an extensive numerical study that complements our analytic work. In Section 6, we make concluding remarks. All proofs of our technical results can be found in the online appendix.

1.1. Literature Review

From an application perspective, our work is related and complementary to the literature on appointment template design, a topic that has been studied extensively (Cayirli and Veral 2003, Gupta and Denton 2008, Ahmadi-Javid et al. 2016). Our work departs from this literature in that we start from an established template and then study how to manage the interaction between the customers and the scheduler to best direct customers to various slots. Among the existing work on dynamic appointment scheduling, Feldman et al. (2014) is the only study, other than the few papers mentioned in the previous section, that explicitly models customer choice behavior. However, Feldman et al. (2014) focus on customer choices across different days and use a newsvendor model to capture the use of daily capacity; this aggregate daily capacity model does not allow them to consider (allocating customers into) detailed appointment time slots within a daily template.

From a modeling perspective, Zhang and Cooper (2005) look at a similar choice model to ours in the context of revenue management for parallel flights. In contrast to this paper, their approach focuses on deriving bounds on the value function of the underlying MDP and using them to construct heuristics. A few recent studies on assortment optimization are particularly relevant to our paper: Golrezaei et al. (2014), Bernstein et al. (2015), Gallego et al. (2016a), and Gallego et al. (2016b). Bernstein et al. (2015) study a dynamic assortment customization problem, which is mathematically similar to our nonsequential appointment offering problem, assuming multiple types of customers, each of which has a multinomial logit choice behavior over all product types. They assume that the customer type is observable to the seller (corresponding to our scheduler), which differs from our setting. Golrezaei et al. (2014) adopt a general choice model and also allow an arbitrary customer arrival process. Gallego et al. (2016b) extend the work by Golrezaei et al. (2014) to allow rewards that depend on both the customer type and the product type. In addition, Gallego et al. (2016a) study assortment optimization in an online retail setting, where each customer picks the number of pages to view according to a fixed distribution and then chooses among all of the products offered on those pages following some choice model. The last three studies also assume that the customer type is known to the seller, and their focus is on developing control policies competitive with respect to an offline optimum, a different type of research question from ours. The other distinguishing feature of our research from all previous work is that we consider sequential offering, an offering paradigm that has not been studied before.

Finally, our work is related to two other branches of literature. The first is on online bipartite matching (Mehta 2013), and the second is on general stochastic dynamic optimization, in particular stochastic depletion problems (e.g., Chan and Farias 2009) and submodular optimization (e.g., Golovin and Krause 2011). These two lines of research mainly aim to obtain performance guarantee results with respect to offline optimums, which is not our research goal.

2. Capacity and Demand Model

We consider a single day in the future that has just opened for appointment booking. The day has an established appointment template, but none of the slots are filled yet. We divide the appointment scheduling window (i.e., the time between when the day is first opened for booking and the end time of this booking process) into N small periods. Specifically, we consider a discrete time N period dynamic optimization model with I customer types (that may come) and J appointment slot types (in the template), where customer types are characterized by their set of *acceptable* slot types. Denote by Ω_{ij} the zero-one indicator of whether slot type j is acceptable by customer type i, and therefore, the $I \times J$ *choice matrix* $\Omega := [\Omega_{ij}]$ consists of distinct row vectors, each representing a unique customer type. Such a customer type structure is similar to those in the literature that model customer segments characterized by different product preferences (e.g., Bernstein et al. 2015).

We now present the details of our customer arrival and choice model. In each period, at most one customer arrives. The customer is type i with probability $\lambda_i > 0$, and with probability $\lambda_0 := 1 - \sum_{i=1}^{I} \lambda_i$, no customer arrives. On a customer arrival, the scheduler offers her a set $S \subseteq \{1, \ldots, J\}$ of slot types without knowledge of the customer type. When *offer set* S contains one or more acceptable slot types, the customer chooses one uniformly at random. If no type in S is acceptable to this customer, we distinguish two possibilities. Either we use a *nonsequential* model, where the scheduler can only offer a single set and the customer immediately leaves if none of the offered slots are acceptable (Section 3), or we use a *sequential* model, where the scheduler may offer any number of sets sequentially until either the customer encounters an acceptable slot or the customer finds no acceptable slots in any offer set and leaves without booking a slot (Section 4). We start from an initial capacity of b_j slots of type j at the beginning of the reservation process and denote $\mathbf{b} := (b_1, \ldots, b_J)$. Every time a customer selects a slot, the remaining slots of this type are reduced by one. The scheduler aims to maximize the fill rate at the end of the reservation process by deciding on the offer set(s) in each period. This is also equivalent to maximizing the *fill count* (i.e., the total number of slots reserved at the end of the booking process), because the initial capacity $\mathbf{b}$ is fixed.

Our capacity and demand model generalizes that of Wang and Gupta (2011) in the following sense. Our notion of "slot type" can be viewed as an abstraction of the service provider and time block combination in their model, and thus, we allow a generalization of using other attributes of a slot that may affect its acceptability to customers, such as duration. Wang and Gupta (2011) consider distinct customer panels, each characterized by a possibly different acceptance probability distribution over all possible combinations of service providers and time blocks and a set of revenue parameters. In contrast, we define the notion of customer type and identify it with a unique set of acceptable slot types. Their arrival rate (probability) parameters are associated with each customer panel, whereas we directly have the demand rate for each of the I customer types as model primitives.

Our choice model assumes that, for a particular customer type, slot types are either "acceptable" or "unacceptable." This dichotomized classification of slots closely mimics the decision process on whether a time slot works for one's daily schedule. For instance, such a slot-choosing process is seen at the popular polling website www.doodle.com, where each participant responds to a poll by indicating whether a particular time works (i.e., is acceptable) for him or her. This relatively parsimonious choice model enables a tractable analysis of the interplay between appointment booking and customer choice. Its parameters may, for instance, be estimated by conducting a market survey on customers' acceptance on various slot types.

As discussed earlier, the distinction between the nonsequential and sequential customer-scheduler interactions reflects the differences present in various real-life appointment scheduling systems. The nonsequential model is best suited for web-based appointment scheduling systems, such as www.zocdoc.com. In such systems, the customer is presented with a list of time slots to choose from, which corresponds to a single offer set. In contrast, sequential scheduling reflects the iterative nature of, for instance, telephone-based appointment scheduling. Here, the scheduler may propose one or more slots initially and may present more if these are rejected by the customer. Although allowing an unlimited number of offer sets in sequence does not conform with many real-world systems, the sequential model is a valuable object of study, because the scheduler in this setting enjoys the greatest flexibility, and hence, the resulting optimal fill rate serves as an upper bound for that in both the nonsequential model and some intermediate paradigms, such as those allowing a limited number of offer sets or with customer reneging.

The assumption that the customer type is unobservable is unique in our work, and it is present to some degree in all real-world systems that we consider. Users of web- and mobile-based appointment scheduling systems often prefer a simple interface soliciting no or minimal personal information before displaying availabilities, and telephone-based schedulers may only know some basic information about the customers. Even if these collected data are useful in predicting customer preferences, not all service firms have the necessary resources (e.g., human, technology, and software) to make such predictions and then use them in scheduling decisions. Another important motivation for why we choose to assume that the exact customer type is unknown to the scheduler is that this case can be considered a harder problem than the setting with full customer information, because the latter would require only offering a single slot type. We will touch on the intermediate case, in which partial knowledge on customer types is available, in Section 4.6 on interactive scheduling. There, we see that incorporating partial knowledge on customer preference does not necessarily lead to better fill rates than sequential offering (Theorem 4), but it may improve the ease of use for sequential offering (e.g., fewer offers to make).

Our objective is to maximize the fill rate (or equivalently, fill count), thereby assuming that each customer contributes to the objective equally. We choose this objective for a few reasons. First, fill rate is a widely used reporting metric by service firms for their operational and financial performance. The simplicity of this metric also makes it more tractable for analysis. Second, fairness may carry more weight than profitability in the vision of a service firm (e.g., a healthcare delivery organization). Third, although different customers may bring different rewards (e.g., revenues) to the service firm, how to associate such rewards with customer (preference) types is not well understood in the literature.

Finally, our discrete time customer arrival model with, at most, one arrival per period is widely accepted and used by many operations management studies, including those on healthcare scheduling (e.g., Green et al. 2006) and revenue management (e.g., Talluri and Van Ryzin 2004 and Bernstein et al. 2015). One could set N, the total number of time periods, sufficiently large so that the probability of multiple customers arriving during a single period is negligible (and thus, such as is the probability of more than N customers arriving in total). This demand model can be used to approximate an inhomogeneous Poisson arrival process (Subramanian et al. 1999).

In the following sections, we focus on analyzing the models described above. Note that our models do not explicitly capture the rolling horizon feature of the appointment scheduling practice, in which customers may book appointments in future days and unused capacity in a day is wasted when the day is past. However, the rolling horizon multiday scheduling model is known for its intractability (Liu et al. 2010, Feldman et al. 2014), whereas the single-day model is more tractable and often used in the literature to generate useful managerial insights (e.g., Gupta and Wang 2008 and Wang and Gupta 2011). In Section C of the online appendix, we will introduce a rolling horizon version of our model and argue why it is intractable. Moreover, we numerically show how our single-day models can inform decision making in a rolling horizon multiday setting.

3. Nonsequential Offering

We first consider the nonsequential offering model, in which only one offer set S is presented to each arriving customer. Denote by $\mathbf{m} \le \mathbf{b}$ a J-dimensional, nonnegative integer vector that represents the current number of remaining slots of each type and by $\mathbf{e}_j$ the J-dimensional unit vector with its jth entry being one and all others being zero. Define $\bar{S}(\mathbf{m}) := \{j = 1, \ldots, J : m_j > 0\}$, the set of slot types with positive capacity, and $V_n(\mathbf{m})$ as the maximum expected number of appointment slots that can be booked from period n to period 1 with $\mathbf{m}$ slots available at the beginning of period n. Note that we count time backward.

Furthermore, denote by $q_{ij}(S)$ the probability that slot type j is chosen conditional on a type i customer arrival and an offer set $S \in \bar{S}(\mathbf{m})$. We have, for any j,

$$q_{ij}(S) = \begin{cases} \dfrac{\Omega_{ij}}{\sum_{k\in S}\Omega_{ik}}, & \text{if } \sum_{k\in S}\Omega_{ik} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Then, the probability that slot type j is chosen when offer set S is given is

$$q_j(S) = \sum_{i=1}^{I} \lambda_i q_{ij}(S), \tag{2}$$

and the no booking probability is $q_0(S) = 1 - \sum_{j=1}^{J} q_j(S)$. The optimality equation is

$$V_n(\mathbf{m}) = \max_{S\subseteq\bar{S}(\mathbf{m})}\left[\sum_{j\in S} q_j(S)(1 - \Delta^j_{n-1}(\mathbf{m}))\right] + V_{n-1}(\mathbf{m}),$$
$$\text{for } n = N, N-1, \ldots, 1, \tag{3}$$

where $V_0(\cdot) = 0$ and $\Delta^j_{n-1}(\mathbf{m}) := V_{n-1}(\mathbf{m}) - V_{n-1}(\mathbf{m} - \mathbf{e}_j)$ denotes the marginal benefit owing to the m_jth unit of slot type j at period $n-1$.

We first analyze the nonsequential offering model for a few specific instances and show that, in general, the optimal nonsequential offering policy seems to have no appealing structural properties. Thus, characterizing the optimal policy for general large-scale nonsequential offering models is very challenging, if not impossible. We then focus our efforts on constructing simple scheduling policies that have performance guarantees and may perform well in practice. We first consider a limited class of policies (called static randomized offering policies) and identify one such policy that is asymptotically optimal when we increase the system demand and capacity simultaneously. We further show that a simple policy that offers all available slots at all times has a constant ratio of two performance guarantee, independent of all model parameters. In Section 5, we show via extensive numerical instances that this offering-all policy significantly outperforms its theoretical bound. It may thus serve as a simple, effective heuristic offering rule for many practitioners in the nonsequential offering context.

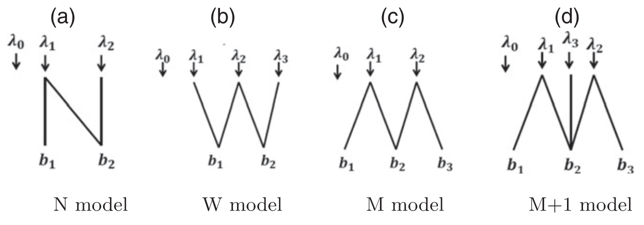
3.1. Results for Specific Model Instances

When there are $J = 2$ slot types, the choice matrix Ω has two possible nontrivial values:

$$\Omega = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \text{and} \quad \Omega = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

These we refer to as the N model instance (Figure 1(a)) and the W model instance (Figure 1(b)), respectively. These two model instances are, for example, applicable to the popular Chinese scheduling system www.guahao.com.cn, which allows customers to book either a morning or an afternoon (medical) appointment for a certain day without providing more granular time

Figure 1. The N, W, M, and M + 1 Model Instances


(a)
(b)
(c)
(d)
λ_0 λ_1 λ_2
b_1 b_2
N model
λ_0 λ_1 λ_2 λ_3
b_1 b_2
W model
λ_0 λ_1 λ_2
b_1 b_2 b_3
M model
λ_0 λ_1 λ_3 λ_2
b_1 b_2 b_3
M+1 model

interval options. In both model instances, we show that it is optimal to offer all available slots at all times (which we call the *offering-all* policy in the rest of this article), because not doing so would unnecessarily risk sending away certain customers. This is formalized in the following result.

Proposition 1. *For the N and W model instances, the offering-all policy is optimal.*

When there are $J = 3$ slot types, the simplest nontrivial choice matrix is the M model instance in Figure 1(c) with

$$\Omega = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

It turns out that, in this case, the offering-all policy is not always optimal; rather, rationing of the versatile type 2 slot is needed. We define policy π_1 according to its offer set:

$$S^{\pi_1}(\mathbf{m}) := \begin{cases} \{1,3\} & \text{if } m_1 > 0 \text{ and } m_3 > 0, \\ \bar{S}(\mathbf{m}) & \text{otherwise.} \end{cases} \tag{4}$$

Therefore, policy π_1 proposes to hold back on offering type 2 slots until either type 1 or type 3 slots are used up. We now formalize that one cannot do better than this.

Proposition 2. *For the M model instance, π_1 is optimal.*

The intuition behind Proposition 2 is that blocking slot type 2 does not lead to any immediate loss of customer demand compared with offering it, while forcing early customers into *less versatile* slot types (types 1 and 3). This preserves the *versatile* slots (type 2) for later arrivals when slots run low. For convenience of discussion, we say a slot type is more versatile if this slot type is accepted by a superset of customer types compared with its counterpart.

Following from Proposition 2, we know that a versatile type 2 slot is at least as valuable as one of the other two less versatile slot types at all times; otherwise, it would be better to offer type 2 slots but not to offer the more valuable, less versatile slot type. To be more specific, we have the following corollary.

Corollary 1. *In the M model instance, for $j = 1$ or 3 or both,*

$$V_n(\mathbf{m} - \mathbf{e}_2) \le V_n(\mathbf{m} - \mathbf{e}_j), \ \forall \mathbf{m} > 0, \ n \in \{1, \dots, N\}. \tag{5}$$

However, it is important to note that one of two less versatile slot types (1 and 3) may be strictly more valuable than the versatile type 2. For example, for $\lambda_1 = \lambda_2 = 0.5$, it is easy to verify that $V_2(2,1,0) = 1.625 < 1.75 = V_2(2,0,1)$. The reason here is the following. With $m_1 = 2$ and $n = 2$, sufficient capacity is available for all potential type 1 customer demand (i.e., at most two units), but in both cases ($\mathbf{m} = (2,1,0)$ and $\mathbf{m} = (2,0,1)$), at most one slot is available for type 2 customers. If $(m_2, m_3) = (1,0)$, the one unit of type 2 slot has a positive probability of being taken by a type 1 customer (which would essentially be a waste, because type 1 slots are sufficient to satisfy all possible demand from type 1 customers). In contrast, if $(m_2, m_3) = (0,1)$, the one unit of type 3 slot can only be exclusively used by type 2 customers, and thus, this is more efficient. This simple example shows that, because of customers' ability to (randomly) choose from their offer set, less versatile slots may be more valuable than versatile slots because of resource imbalance. This observation implies that the (future) value of keeping a slot type cannot be viewed solely based on the number of accepting customer types, irrespective of the arrival probabilities or slot capacities. This complication renders the optimal policy for a general model instance quite complex as we show now.

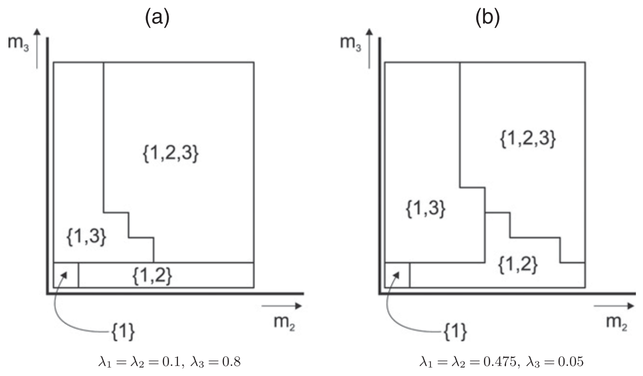
The next model instance that we focus on is the M + 1 model instance shown in Figure 1(d), with choice matrix

$$\Omega = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Note that the only difference between the M + 1 and M model instances is the additional customer type 3 that only accepts type 2 slots. It turns out that the simple, elegant form of the optimal policies in the previous cases does not carry over to the M + 1 model instance.

To illustrate the complexity of the M + 1 model instance, consider the case with $m_1 = 4$ and $n = 5$. Figure 2 shows the *unique* optimal offer set, identified with $S \subset \{1,2,3\}$, as a function of m_2 and m_3. (For instance, if $S = \{1,3\}$, it means offering slot types 1 and 3 but not slot type 2.) Consider $\lambda_1 = \lambda_2 = 0.1$, $\lambda_3 = 0.8$ (Figure 2(a)) or $\lambda_1 = \lambda_2 = 0.475$, $\lambda_3 = 0.05$ (Figure 2(b)). As discussed earlier, resource imbalance can make a less versatile slot more valuable than a more versatile one, which naturally would suggest an action of saving the less versatile slot by only offering the versatile slot.

Figure 2. The Optimal Policy for an M + 1 Model Instance, with $m_1 = 4$, $n = 5$



Indeed, in Figure 2(b), we see that action $\{1, 2\}$ can be the unique optimal action, even when $\mathbf{m} > 0$. This is true, because m_3 is relatively small (equal to one or two in this case), whereas $m_1 = 4$ is ample given $n = 5$ and the symmetric arrival rates of type 1 and type 2 customers. Blocking type 3 and offering the versatile type 2 earlier rather than later can help to resolve the resource imbalance by maximizing the total expected amount of type 2 slots taken by type 2 customers (and thus, saving type 3 slots that can only serve type 2 customers for the future).

In addition, we see that the arrival rate now has a strong impact on the optimal policy in contrast to the other cases that we discussed so far: when λ_3 is large, it is often optimal to include type 2 slots in the offer set, whereas when λ_3 is small, this is not the case. The reasoning here is that, for λ_3 small, the model is very close to the M model instance, for which we know it is optimal to save versatile type 2 slots for later in the booking process. However, not offering type 2 slots also implies turning away all type 3 customers, which explains why this slot type should be offered when λ_3 is large.

These observations we make on the M + 1 model instance suggest that the optimal policy depends on the customer preference profiles, arrival rates, and available slot capacity of the specific model instance under consideration. The optimal policy for a general model can be quite complex and may have no straightforward structural properties. Thus, we shall focus our efforts on identifying simple scheduling policies that have performance guarantees and perform well in practice.

3.2. Asymptotically Optimal Policy

In this section, we construct a *static randomized policy* that is asymptotically optimal when we increase the system demand and capacity simultaneously. We first introduce the class of static randomized policies. At any decision epoch, there are altogether 2^J possible actions in terms of which slot types to offer. Here, we use a binary vector to denote the offer set with a one at position j, meaning that slot type j is offered, and zero otherwise. For example, we denote by the action of closing all slots as $\mathbf{w}^1 := (0, \ldots, 0)$, the J-dimensional zero vector, and the action of opening all slots as $\mathbf{w}^{2^J} := (1, \ldots, 1)$, the J-dimensional one vector. We call the set of all 2^J J-dimensional zero-one vectors as set $\mathcal{W} := \{0, 1\}^J$ and name the elements of the set as $\mathbf{w}^1, \mathbf{w}^2 \ldots, \mathbf{w}^{2^J}$. Define $\mathcal{K} := \{1, \ldots, 2^J\}$ as the action index set, and therefore, $\mathcal{K}$ and $\mathcal{W}$ have the same cardinality.

A policy π^p is a static randomized policy if π^p offers $\mathbf{w}^k$ with some fixed probability p_k, independent of the system state and the time period.[1] The class of static randomized policies contains all π^p values, such that the vector $\mathbf{p} = \{p_k\}_{k=1}^{2^J}$ is a probability vector. For instance, the offering-all policy is a special case in this class with $p_{2^J} = 1$ and $p_k = 0$ for all $k \neq 2^J$.

We show that there exists a vector $\mathbf{p}^*$, such that π^{p^*} is asymptotically optimal when the demand and capacity are scaled up simultaneously. The choice of p_k^* relies on the fluid model corresponding to the stochastic model (3) considered above, in which we can readily determine the optimal offering policy. We choose p_k^*, such that it represents the fraction of the time in which the action $\mathbf{w}^k$ is used in a fluid model under optimal control. Below, we construct this asymptotically optimal policy π^{p^*} and defer more technical details to the online appendix.

3.2.1. Fluid Model.

We first introduce our fluid model. To differentiate from the notation in the stochastic model formulation (3), we shall put the time index n in parentheses instead of as a subscript. Instead of discrete customers arriving in each slot, we represent a customer by a unit of fluid. In total, one unit of demand arrives in each time period, a fraction λ_i of which corresponds to customer type i. This fluid is distributed evenly among all available slots that are offered and accepted by the corresponding customer type.

For each $n = 1, \ldots, N$, the decision vector in the fluid model is $\mathbf{z}(n) = (z_1(n), \ldots, z_{2^J}(n))$, which is a 2^J-dimensional vector, each component $z_k(n) \in [0, 1]$ representing the time during which action $k \in \mathcal{K}$ is being used in period n. Note that each action can be used for any fractional unit of time. Thus, we require that

$$0 \leq z_k(n) \leq 1, \forall k \in \mathcal{K}, n = 1, \ldots N; \tag{6}$$

$$\sum_{k \in \mathcal{K}} z_k(n) = 1, \forall n = 1, \ldots N. \tag{7}$$

Constraint (7) ensures that the total time spent on all possible actions (including the one that closes all slot types) in one period adds up to one.

Let $\boldsymbol{\tau}(n) = [\tau_{k,j}(n)]$ be a $2^J \times J$ matrix, each row of which corresponds to one of the 2^J possible actions. We use $\tau_{k,j}(n)$ to indicate the amount of time for which type j slots are offered during the time when the kth action is taken in period n. We have that

$$\tau_{k,j}(n) = z_k(n)\mathbf{w}_j^k, \ \forall k \in \mathcal{K}, j = 1, 2, \ldots, J, n = 1, \ldots N, \tag{8}$$

where $\mathbf{w}_l^k$ denotes the lth coordinate of vector $\mathbf{w}^k$. Constraint (8) is presented mainly to make the formulation clearer and easier to understand. It ensures that slot type j can be open when action k is chosen only if action k offers slot type j. If action k does not offer slot type j, then $\mathbf{w}_j^k = 0$, and $\tau_{k,j}(n)$ is zero by (8). Let $\mathcal{J}_k = \{j : \mathbf{w}_j^k = 1, j = 1, 2, \ldots, J.\}$ be the full set of slot types offered by action k. Note that (8) implies that

$$\tau_{k,j_1}(n) = \tau_{k,j_2}(n), \ \forall k \in \mathcal{K}, n = 1, \ldots N, j_1, j_2 \in \mathcal{J}_k.$$

That is, if an action k offers multiple slot types, the offering durations of these slot types are the same.

Let $y_{i,j}(n)$ denote the amount of type j slot's capacity filled by type i customers during period n and $\mathcal{K}_j = \{s : \mathbf{w}_j^s = 1, s \in \mathcal{K}\}$ be the index set of actions that offer type j slots. If $\Omega_{i,j} = 1$,

$$y_{i,j}(n) = \sum_{k \in \mathcal{K}_j} \tau_{k,j}(n) \cdot \frac{\lambda_i}{\sum_{l=1}^{J} \min\{\Omega_{i,l}, \mathbf{w}_l^k\}}, \; i = 1, 2, \ldots, I, \quad j = 1, 2, \ldots, J, \; n = 1, \ldots, N; \tag{9}$$

otherwise, if $\Omega_{i,j} = 0$, then

$$y_{i,j}(n) = 0, \; i = 1, 2, \ldots, I, \; j = 1, 2, \ldots, J, \; n = 1, \ldots, N. \tag{10}$$

Note that all terms in (9), except $\tau_{k,j}(n)$, are constants, and therefore (9), as a set of constraints for the optimization problem, they are linear in the decision variables $\tau_{k,j}(n)$.

Let $M_j(t)$ be the amount of type j slots left with t time periods to go, and let $Z_N(\mathbf{m})$ be the optimal amount of (fluid) customers served with initial capacity vector $\mathbf{m}$ and N periods to go. The goal is to choose $z_k(n)$ (and $\tau_{k,j}(n)$) to solve for

$$Z_N(\mathbf{m}) = \max \sum_{n=1}^{N} \sum_{j=1}^{J} \sum_{i=1}^{I} y_{i,j}(n), \tag{P1}$$

subject to: (6), (7), (8), (9), and (10), and

$$M_j(N) = m_j, \qquad j = 1, 2, \ldots, J, \tag{11}$$

$$M_j(n-1) = M_j(n) - \sum_{i=1}^{I} y_{i,j}(n), \quad j = 1, 2, \ldots, J, \quad n = 1, \ldots, N, \tag{12}$$

$$M_j(n) \geq 0, \quad j = 1, 2, \ldots, J, \; n = 0, 1, \ldots, N-1. \tag{13}$$

In (P1), constraint (11) specifies the initial capacity vector, (12) updates the capacity vector for each period, and (13) ensures that all slot types have nonnegative capacity throughout. We remark that, in our formulation, control can be exerted anytime continuously throughout the horizon, but the system is observed only at discrete time epochs $0, 1, 2, \ldots, N$ to match the stochastic model formulation (3).

3.2.2. Choice of $\mathbf{p}^*$.

Let p_k^* be the fraction of the time in which the optimal policy chooses action k in the fluid model (P1). That is,

$$p_k^* = \frac{\sum_{n=1}^{N} z_k^*(n)}{N}, \tag{14}$$

where $z_k^*(n)$ is the optimal solution to (P1). We now translate this optimal policy for the fluid model to our original discrete and stochastic setting by defining a policy π^{p^*}, such that, in each period n, this policy offers $\mathbf{w}^k$ with probability p_k^*, independent of everything else.

The intuition behind choosing $\mathbf{p}^*$ as the offering probability vector is that, if we scale up the system demand (i.e., N) and capacity (i.e., $\mathbf{m}$) in the stochastic model, using π^{p^*} makes the proportion of total customer demand going to each slot type in the stochastic model approximately match that in the fluid model. Thus, the total fill counts in the stochastic model are similar to those of the fluid model. Because the fluid model is a deterministic model that provides an upper bound on the objective value of the stochastic model (more on this below), we know that $\mathbf{p}^*$ is (close to) optimum in the stochastic model as the system becomes large. We formalize this intuition in the next section.

3.2.3. Main Result.

Consider a sequence of problems indexed by $K = 1, 2, 3 \ldots$. The problems in this sequence are identical except that, for the Kth problem, the number of total periods is NK and the capacity vector is $\mathbf{m}K$. We call the problem instance with $K = 1$ the base problem instance. Let $V_n^{\pi^p}(\cdot)$ be the total expected number of slots filled under a policy π^p with the offering probability vector $\mathbf{p}$ in the stochastic model. The main result is shown in the following theorem.

Theorem 1.
(i) $K^{-1}V_{NK}(\boldsymbol{m}K) \leq K^{-1}Z_{NK}(\mathbf{m}K) = Z_N(\mathbf{m})$, $\forall \mathbf{m} \geq 0$, $K = 1, 2, 3, \ldots$;
(ii) $\lim_{K\to\infty} K^{-1}V_{NK}^{\pi^{p^*}}(\boldsymbol{m}K) = Z_N(\mathbf{m})$.

Recall that $V_n(\cdot)$ is the optimal value of the stochastic model defined in (3). Thus, Theorem 1(i) says that the "normalized" optimal value of the nonsequential offering stochastic model (i.e., the original value divided by K) is bounded from above by that of the corresponding fluid model and that the normalized objective value of the fluid model is the same as the objective value of the base fluid model with $K = 1$. Theorem 1(ii) states that, as the system grows large, the normalized objective value in the stochastic system under policy π^{p^*} converges to this constant upper bound, and thus, π^{p^*} is asymptotically optimal.

The proof of Theorem 1 entails a few key steps, which are outlined below (full details can be found in the online appendix). We first show that the optimal objective value of the fluid model is an upper bound of the optimal value of the stochastic model (i.e., $Z_N(\mathbf{m}) \geq V_N(\mathbf{m})$) for any given set of model parameters. Then, based on any static randomized policy π^p, we construct a lower bound for $V_n^{\pi^p}(\cdot)$, and this lower bound is naturally a lower bound for the optimal value of the stochastic model $V_N(\mathbf{m})$ (because π^p is not necessarily optimal). Finally, we show that, when p is chosen as p^* defined in (14), the normalized lower bound converges to $Z_N(\mathbf{m})$ when the system grows in both demand and capacity. Thus, π^{p^*} is asymptotically optimal.

Our findings build on the early classic results in the revenue management literature, which show that allocation policies arising from a single linear program make the normalized total expected revenue converge to an upper bound on the optimal value (Cooper 2002). Our results are different and new in several important aspects. The model in Cooper (2002) can designate/allocate a particular product (slot) type on a customer arrival (because he assumes that customer preference is known on arrival), whereas our model offers multiple product (slot) types for customers to choose from (because the customer preference is not known). Using the offer set as a decision in the model creates significant new challenges. Our fluid model formulation needs to explicitly take care of customer choice processes and is much more complicated than that in Cooper (2002). Leveraging the fluid model formulation, the asymptotically optimal policy in Cooper (2002) accepts customer requests up to some customer type-specific thresholds, because the optimal solution in the fluid model of Cooper (2002) prescribes such thresholds for each customer type. As a result, the asymptotic policy of Cooper (2002) leads to a closed form expression for each type of the customer demand served, allowing him to show that the normalized demand served converges in distribution to a constant that matches the optimal fluid model decision, by directly using a result in Billingsley (1968). However, because of customer choices, our fluid model cannot give rise to such a simple policy. Our fluid model informs the optimal duration in which a particular offer set is used, and we use this information to construct our policy, which has a completely different form compared with the policy of Cooper (2002). Because we cannot control the exact product (slot) type in an offer set that will be chosen by an arriving customer, we do not have a closed form expression as in Cooper (2002) for the total demand that goes into each product (slot) type and eventually gets served. To deal with this difficulty, we construct a (very) tight lower bound on the objective value and show that this lower bound, after being normalized, converges to the optimal objective value of the fluid model. The idea of our proof may be useful to identify effective approximate policies in other capacity management contexts when the manager cannot directly control the product that a customer may pick.

3.3. Constant Performance Guarantee of the Offering-All Policy

In this section, we focus on a simple scheduling policy: the offering-all policy. Let π_0 represent this policy, and the offer set under π_0 is the full set of all slot types, irrespective of the period n. Note that the effective offer set at state $\mathbf{m}$ is $\bar{S}(\mathbf{m})$. That is, when customers arrive, they only consider those slot types with positive capacity when making a choice. We denote by $V_n^{\pi_0}(\mathbf{m})$ the expected fill count attained by applying the offering-all policy π_0 throughout.

Indeed, this simple policy has a constant performance guarantee that states that, for any set of parameters, the offering-all policy π_0 achieves at least one-half of the optimal fill count.

Theorem 2. *For any* Ω, n, *and* $\mathbf{m}$, $V_n(\mathbf{m}) \leq 2V_n^{\pi_0}(\mathbf{m})$.

It is worth noting that Theorem 2 in fact holds more broadly for all so-called *myopic policies*, which at each period, offer a set maximizing the expected number of filled slots for that period. Myopic policies, however, do not have to offer all slot types in all periods. For instance, offering slot types 1 and 3 in the M model instance would constitute a myopic policy.

Performance guarantee results on myopic policies exist in various dynamic optimization settings, and a ratio of two is often the best provable performance bound (e.g., Chan and Farias 2009 and Mehta 2013). Although this performance bound may seem a little loose, we shall see empirically in Section 5.1 that the offering-all policy performs very well and much better than this lower bound; in finite regimes, the offering-all policy also seems to perform better than the asymptotically optimal policy constructed in Section 3.2.

4. Sequential Offering

We now present our second scheduling paradigm, which allows the scheduler to offer multiple sets of slots sequentially. Recall that this way of offering slots may represent, for instance, web-based scheduling where available slots are not revealed simultaneously as well as telephone-based scheduling. Intuitively, having the scheduler offer slots sequentially instead of all at once will be able to steer customers into selecting more favorable slots from the perspective of system optimization. The questions that we address in this section are then how many and what sets of slots to offer to maximize the fill rate. We start by introducing the sequential offering model next.

4.1. Model Outline

On customer arrival, the scheduler chooses a $K, 1 \leq K \leq J$, and sequentially presents the customer with K mutually exclusive subsets $S_1, S_2, \ldots, S_K \subseteq \bar{S}(\mathbf{m})$. We denote this action as $\mathbf{S} := S_1 - S_2 - \cdots - S_K$. Denote by $\mathcal{S}(\mathbf{m})$ the set of all possible such actions at state $\mathbf{m}$ and by $I_k(\mathbf{S}) := \{i : \sum_{j \in S_k} \Omega_{ij} \geq 1, i \notin \cup_{l=1}^{k-1} I_l(\mathbf{S})\}$, $k = 1, \ldots, K$, the set of customer types who do not accept any slot from the first $(k-1)$ offer sets but encounter at least one acceptable slot in S_k. Therefore, $I_k(\mathbf{S})$ represents the set of customers who, given sequential offering $\mathbf{S}$, accept some slot on arrival into the system. Moreover, the slot chosen by these customers belongs to the kth offer set S_k. The probability that slot type j is chosen under action $\mathbf{S}$ may then be written as $q_j(\mathbf{S}) := \sum_{k=1}^{K} \sum_{i \in I_k(\mathbf{S})} \lambda_i q_{ij}(S_k)$, with $q_{ij}(\cdot)$

as in (1). The assumption that the sets $S_1, S_2, \ldots, S_K$ are mutually exclusive is made from a practical rather than mathematical standpoint: there is simply no reason to offer the same slot type in two or more sets, because the customer will book a slot as soon as she is offered a set with at least one acceptable slot. Thus, only the first set in which such a slot is included is relevant.

For ease of presentation, we still use $V_n(\mathbf{m})$ to denote the maximum expected number of slots that can be booked with $\mathbf{m}$ slots available and n periods to go in this section. For an action $\mathbf{S}$, we let $\bigcup \mathbf{S} := \bigcup_{i=1}^{K} S_i$ denote the set of all slot types offered throughout action $\mathbf{S}$. Then, for the sequential offering model, we have

$$V_n(\mathbf{m}) = \max_{\mathbf{S}\in\mathcal{S}(\mathbf{m})}\left[\sum_{j\in\bigcup\mathbf{S}} q_j(\mathbf{S})(1-\Delta^j_{n-1}(\mathbf{m}))\right] + V_{n-1}(\mathbf{m}), \quad \text{for } n = N, N-1, \ldots, 1, \quad (15)$$

where $V_0(\cdot) = 0$ and $\Delta^j_{n-1}(\mathbf{m}) := V_{n-1}(\mathbf{m}) - V_{n-1}(\mathbf{m}-\mathbf{e}_j)$ denotes the marginal benefit owing to the m_jth unit of slot type j at period $n-1$. We observe that both the transition probability $q_j(\mathbf{S})$ and the set of feasible actions $\mathcal{S}(\mathbf{m})$ are much more complicated than their counterparts in the nonsequential model.

The sequential offering setting can be viewed as a generalization of nonsequential scheduling to any number $K \geq 1$ of offer sets. Consequently, it stands to reason that the offering-all policy will not perform well in the sequential setting, because this would limit the scheduler to a single offer set ($K = 1$). We indeed numerically confirm this conjecture in Section 5.3. Note that, in contrast to the nonsequential case, an offering-all policy is unlikely to be used in a practical setting, such as telephone scheduling (because it would take too much time for the scheduler to go over every possible appointment option). In the online setting, there is a way to take advantage of sequential offerings by redesigning the customer interface that releases information sequentially.

To provide a roadmap of analyzing the sequential model, we summarize our key findings in this section as follows.

1. We first consider a general setting and derive various structural results that provide more insights; in particular, we show that it is optimal to offer slot types one by one.
2. For a large class of problem instances with nested preference structures (to be discussed later), we derive a closed form optimal sequential offering policy.
3. For problem instances not in this class, we develop a simple and highly effective heuristic based on the idea of balanced resource use and fluid models.
4. We prove that the optimal sequential offering does as well as in the nonsequential case where the scheduler has full information of the customer type on arrival; we argue that this equivalence allows us to apply the idea of sequential offering in various interactive scheduling contexts.

4.2. Results for the General Sequential Offering Model

We now present some properties of the sequential model with general choice matrices. We first derive some structural properties of the value function.

Lemma 1. *The value function $V_n(\mathbf{m})$ satisfies*

(i) $0 \leq V_{n+1}(\mathbf{m}) - V_n(\mathbf{m}) \leq 1$, $\forall \mathbf{m} \geq 0$, $\forall n = 1, 2, \ldots, N-1$; *and*

(ii) $0 \leq V_n(\mathbf{m}+\mathbf{e}_j) - V_n(\mathbf{m}) \leq 1$, $\forall \mathbf{m} \geq 0$, $\forall n = 1, 2, \ldots, N$.

Part (ii) of Lemma 1 implies that $V_n(\mathbf{m}+\mathbf{e}_j) \leq V_n(\mathbf{m}) + 1$ (i.e., it is better to have a slot booked now rather than saving it for future). Therefore, in the context of sequential offering, it is better to keep offering slots if none have been taken so far. This is formalized in the following result, which shows that there exists an optimal sequential offering policy that exhausts all available slot types in each period.

Lemma 2. *For any Ω, $\mathbf{m}$, and n, there exists an optimal action $\mathbf{S}^*$, such that $\bigcup \mathbf{S}^* = \bar{S}(\mathbf{m})$.*

Building on Lemma 2, we are able to characterize the structure of an optimal sequential offering policy described in the theorem below.

Theorem 3. *Let $\mathbf{m} > 0$ be the system state at period $n \geq 1$, and let $j_1, j_2, \ldots, j_J$ be a permutation of $1, 2, \ldots, J$, such that $V_{n-1}(\mathbf{m}-\mathbf{e}_{j_k}) \geq V_{n-1}(\mathbf{m}-\mathbf{e}_{j_{k+1}})$, $k = 1, 2, \ldots, J-1$. Then, the action $\{j_1\} - \cdots - \{j_J\}$ is optimal.*

Theorem 3 implies that there exists an optimal policy that offers one slot type at a time. More importantly, this result shows a specific optimal offer sequence based on the value function to go. To understand this, recall that $V_{n-1}(\mathbf{m}) - V_{n-1}(\mathbf{m}-\mathbf{e}_j)$ can be viewed as the value of keeping the m_jth type j slot from period $n-1$ onward. Because all customers bring in the same amount of reward, it benefits the system the most if an arrival customer can be booked for the slot type with the least value to keep (i.e., the slot type with the largest $V_{n-1}(\mathbf{m}-\mathbf{e}_j)$).

Even if the scheduler does not know the exact customer type, following the optimal offer sequence described in Theorem 3 ensures that the arriving customer takes the "least valuable" slot (as long as there is at least one acceptable slot remaining). Indeed, matching customers with slots in this way would be the best choice for the scheduler, even if she had perfect information about customer type (i.e., she knew exactly the customer type on arrival). Following this rationale, our next result shows an interesting and important correspondence between (1) the sequential offering without customer type information and (2) the nonsequential offering with *perfect* customer type information. To distinguish these two settings, we let $V_n^s(\mathbf{m})$ and $V_n^f(\mathbf{m})$ represent the optimal value functions for settings (1) and (2), respectively, in the next theorem.

Theorem 4. $V_n^s(\mathbf{m}) = V_n^f(\mathbf{m})$, $\forall\, \mathbf{m} \geq 0$, $n = 0, 1, 2, \ldots, N$.

Theorem 4 suggests that the optimal sequential offering can fully exploit the value of customer type information; however, it does not imply that it can fully elicit the customer type. Specifically, optimal sequential offering happens to result in the same system state changes as if the scheduler had full information about customer type but does not let the scheduler know exactly the customer type (Remark 2 in Section 4.4). Theorem 4 suggests that sequential offering is a useful operational mechanism to improve the scheduling efficiency in the absence of customer type information. Our numerical experiments in Section 5 confirm and quantify such efficiency gains.

4.3. Optimal Sequential Offering Policies

In this section, we fully characterize the optimal sequential offering policy for a large class of choice matrix instances, which include the N, M, and M + 1 model instances (Figure 1). To this end, let $I(j)$ be the set of customer types who accept slot type j (i.e., $I(j) = \{i = 1, 2, \ldots, I : \Omega_{ij} = 1\}$, $\forall j = 1, 2, \ldots, J$). It makes intuitive sense that, if $I(j_1) \subset I(j_2)$, then slot type j_2 is more valuable than j_1, and thus, slot type j_1 should be offered first. Combining this observation with Theorem 3 could then help us to design an optimal policy. Let us first introduce a specific class of model instances.

Definition 1. We say that a model instance characterized by Ω is nested if, for all $j_1, j_2 = 1, 2, \ldots, J$ and $j_1 \neq j_2$, one of the following three conditions holds: (i) $I(j_1) \cap I(j_2) = \emptyset$, (ii) $I(j_1) \subset I(j_2)$, or (iii) $I(j_1) \supset I(j_2)$.

Note that not all model instances are nested. One simple example is the W model instance from Figure 1(b), where $I(1) = \{1, 2\}$ and $I(2) = \{2, 3\}$. None of the conditions (i)–(iii) from Definition 1 hold in this case for $j_1 = 1$ and $j_2 = 2$. However, it is readily verified that the N, M, and M + 1 model instances are all nested.

Remark 1. The concept of a nested model instance is related to the *star structure* considered in the previous literature on flexibility design (e.g., Akçay et al. 2010). Consider a system with a certain number of resource types (corresponding to slot types in our context), which can be used to do jobs of certain types (customer types in our context). A star flexibility structure is one such that there are specialized resource types, one for each job type, plus a versatile resource type that can perform all job types. The nested structure generalizes the star structure.

It turns out that we can fully characterize an optimal policy for nested model instances as follows.

Theorem 5. *Suppose Ω is nested; any policy that offers slot type j_1 before offering slot type j_2 for any j_1, j_2 such that $I(j_1) \subset I(j_2)$ is optimal.*

Theorem 5 proposes to offer nested slot types in an increasing order of the accepting customer types. Note that, when two slot types are mutually exclusive (i.e., $I(j_1) \cap I(j_2) = \emptyset$), the order in which they are offered is irrelevant, because customers who would select a slot from one set could never from the other. To give some specific examples, we can fully characterize the optimal policy for the N, M, and M + 1 model instances using Theorem 5.

Corollary 2. *For the N model instance and any n and $\mathbf{m}$, an optimal sequential offering policy is to offer $\mathbf{S} = \{1\} - \{2\}$.*

Corollary 3. *For the M and M + 1 model instances and any n, an optimal sequential offering policy is to offer*

$$\mathbf{S} = \begin{cases} \{1,3\} - \{2\}, & \text{if } m_1, m_2, m_3 \geq 1, \\ \{1\} - \{2\}, & \text{if } m_3 = 0, \\ \{3\} - \{2\}, & \text{if } m_1 = 0. \end{cases}$$

4.4. Beyond Nested Model Instances

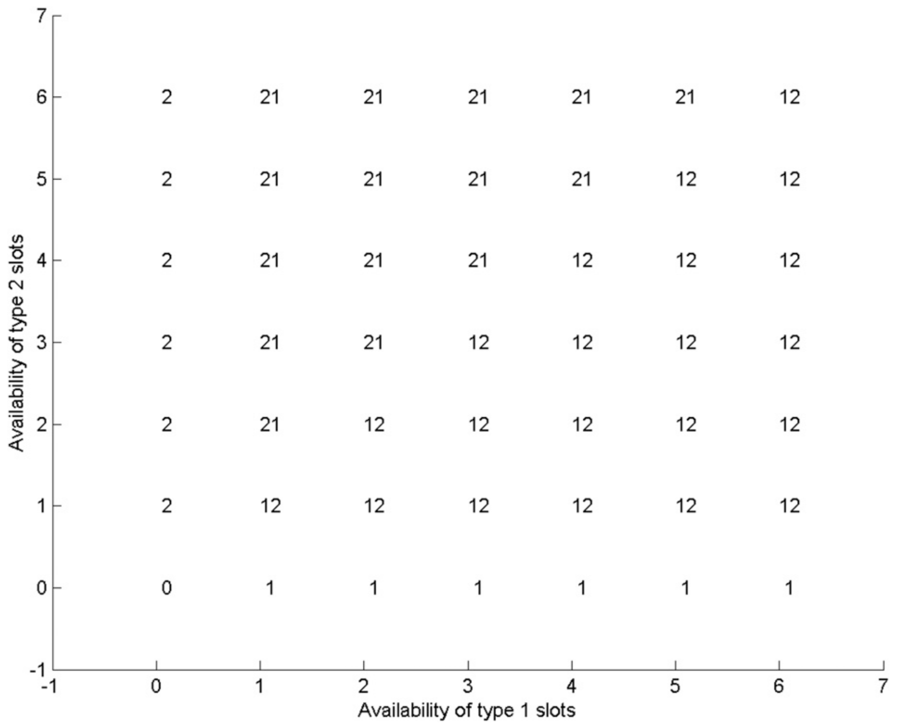
Although Theorem 5 solves a large class of the sequential model instances, not all instances have a nested structure. In this section, we analyze the W model instance (Figure 1) to glean some insights into the instances that are not nested.

To analyze the W model instance, one can formulate an MDP with three possible actions: $\{1, 2\}$, $\{1\} - \{2\}$, and $\{2\} - \{1\}$ (and the corresponding actions at the boundaries). However, there exist no straightforward offering orders for slot types, and the optimal sequential policy turns out to be state dependent. Specifically, we find that the optimal policy is a *switching curve* policy: with the availability of one type of slots held fixed, it is optimal to offer the other type of slots first as long as there is a sufficiently large amount of such slots left.

Figure 3 illustrates the optimal actions for the W model instance at different system states with $\lambda = (0.2, 0.5, 0, 3)$ and $n = 6$. The numerals "0," "1," "2," "12," and "21" correspond to the actions of offering nothing, offering type 1 slots only, offering type 2 slots only, offering type 1 slots and then type 2 slots, and offering type 2 slots and then type 1 slots, respectively. The optimal actions at boundary are obvious. In the interior region of the system states, we can clearly see the switching curve structure. For instance, when the system state is $(3, 3)$, it is optimal to offer $\{1\} - \{2\}$. When the number of type 2 slots increases to four, then it is optimal to offer $\{2\} - \{1\}$.

The intuition behind this is different from that of the model instances considered above where customer preferences are nested (e.g., the N, M, and M + 1 model

Figure 3. Structure of the Optimal Policy Under W Model Instance with Sequential Offers



instances). In the W model instance, type 1 (type 3) customers only accept type 1 (type 2) slots, but type 2 customers accept both types of slots. If there are relatively more type 1 slots than type 2 slots, then it makes more sense to "divert" type 2 customers to choose type 1 slots, thus saving type 2 slots only for type 3 customers. Accordingly, the switching curve policy stipulates that type 1 slots be offered first, ensuring that type 2 customers, if any, will pick type 1 slots. The intuition above is formalized in the proposition below.

Proposition 3. *Consider the W model instance with sequential offers. Given m_2, if there exists an m_1^* such that the optimal action at state (m_1^*, m_2) is $\{1\} - \{2\}$, then $\forall \mathbf{m} \in \{(m_1, m_2), m_1 \geq m_1^*\}$, and the optimal action is $\{1\} - \{2\}$. Similarly, given m_1, if there exists an m_2^* such that the optimal action at state (m_1, m_2^*) is $\{2\} - \{1\}$, then $\forall \mathbf{m} \in \{(m_1, m_2), m_2 \geq m_2^*\}$,* and *the optimal action is $\{2\} - \{1\}$.*

Remark 2. In Section 4.3, we state that sequential offering may not fully reveal exact customer types but allows the system to evolve in the same optimal way as if the scheduler knew exactly the customer type. We use the W model instance to illustrate this point. Consider the W model instance with nonsequential offers and the scheduler knowing the exact type of arriving customers. Suppose that the optimal action is to offer $\{1\}$ when type 1 or type 2 customers arrive and offer $\{2\}$ when type 3 customer arrives. Now, in a sequential offering model where the scheduler does not know the exact type of arriving customers, the scheduler would have offered $\{1\} - \{2\}$ to any arriving customer. If we encountered type 1 or 2 customers, type 1 slot would be taken, but we do not know the exact type of this customer (we know, however, that she must be either type 1 or type 2); if a type 3 customer arrived, she would reject type 1 slot but take type 2 slot. In this way, the system evolves as if the scheduler had perfect information on customer type. The structural properties of the optimal policy described in Proposition 3 are likely the best that we can obtain for the W model instance; the exact form of the optimal policy depends on model parameters and the system state, much like with the M + 1 model instance in the nonsequential case. If customer preference structures become more complicated, it is very difficult, if not impossible, to develop structural properties for the optimal sequential offering policy. Thus, for model instances that do not satisfy the conditions of Theorem 5, we propose an effective heuristic below.

4.5. The Drain Heuristic

If customer preferences are not nested, the analysis of the W model instance suggests that the optimal policy is to offer slots with more capacity relative to its customer demand. Inspired by this observation and using the idea of fluid models, we propose the following heuristic algorithm, which aims to "drain" the abundant slot type first followed by less abundant ones. This heuristic aims to have all slot types emptied simultaneously, thus maximizing the fill rate. That is, this heuristic tries to "balance" the resource use. Specifically, the drain algorithm works in the following simple way. At period n and for each slot type $j \in \bar{S}(\mathbf{m})$, we calculate

$$I_j := \frac{m_j}{n \sum_{i=1}^{I} \lambda_i \frac{\Omega_{ij}}{\sum_{k \in \bar{S}(\mathbf{m})} \Omega_{ik}}}. \tag{16}$$

Note that the fraction term in the denominator of (16) represents the share of type i customers who will choose type j slots, assuming that all available slot types are offered simultaneously. Taking expectation with respect to the customer type distribution and multiplying by n, the number of customers to come, the denominator of (16) can be viewed as the expected load on type j slots in the next n periods. As a result, the index I_j can be regarded as the ratio between capacity left and "expected" load.

The drain algorithm is then to calculate all I_j values at the beginning of each period and offer slots in decreasing order of the I_j. The algorithm calls for offering slot types with larger I_j first, because these slot types have relatively more capacity compared with demand. In other words, a slot type with a larger I_j is likely to have a smaller marginal value to keep and thus, can be offered earlier. We could of course safely remove n in the definition of I_j and obtain the exact same order of slots. However, we leave n in the denominator of (16), because this allows us to interpret I_j as the ratio between capacity left and expected number of requests. Based on this interpretation, it is clear that this heuristic aims to have all slot types emptied simultaneously, thus maximizing the fill rate. We will test the performance of this algorithm in Section 5.2.

4.6. Applications to Interactive Scheduling

In Sections 3 and 4, we discuss two different models of customer-scheduler interactions in the appointment booking practice. In one model, the scheduler makes a one-shot offering, and in the other, the scheduler enjoys the full flexibility of sequential offering. The appointment booking process, however, can fall in between these two models in terms of the degree to which the customer preference information is collected and used during the interaction between the scheduler and each customer. Such interactions may be present both in a traditional setting with human interaction (e.g., a customer, after being offered an appointment at 8 a.m. by a receptionist, may indicate that none of the morning slots are acceptable) or fully digitally (e.g., the Partners HealthCare Patient Gateway online booking website allows patients to indicate their acceptable time slots upfront).

When additional customer preference information is gathered during the appointment booking process, the scheduler can still follow the optimal list of slot types $\{j_1\} - \cdots - \{j_J\}$ obtained from Theorem 3 but simply skip all slots known to be unacceptable either up front or dynamically as additional information is collected. This offering strategy is still optimal, because it would end up with the same system state compared with not skipping those slots indicated as unacceptable before or during the booking process (e.g., directly declared by the customer) and thus, give the exact same fill count that can be obtained if the scheduler had full information about the customer type (Theorem 4). Recall that the order of $\{j_1\} - \cdots - \{j_J\}$ can be readily obtained with nested customer preferences (Theorem 5), or otherwise, an approximate order can be easily formed by the drain heuristic (16).

Although outside the scope of this paper, these considerations on interactive scheduling raise various issues related to the tradeoff between obtaining the best fill rate and providing a convenient experience to the customer. For instance, the scheduler may want to limit the number of sets offered to the customer to provide a smooth user experience. In light of Theorem 3, one potential idea for future study is to group slot types based on the order of $\{j_1\} - \cdots - \{j_J\}$.

5. Numerical Results

In the last two sections, we consider nonsequential offering and sequential offering. For each setting, we derive optimal or near-optimal booking policies. In this section, we run extensive numerical experiments to test and compare these policies and the two scheduling paradigms.

We organize this section as follows. Sections 5.1 and 5.2 discuss the performance of the offering-all policy in the nonsequential model and the drain heuristic in the sequential model, respectively. We show that these two algorithms obtain fill rates that are remarkably close to those of the respective optimal policies, and therefore, they can serve as simple, effective heuristics for practical use. Section 5.3 compares the differences in the expected fill rate under the nonsequential and sequential offering models, where this difference represents the value of sequential offering. Specifically, we evaluate the differences between the optimal policies and those between the heuristics. The former represents the "theoretical" value of sequential offering compared with nonsequential offering, whereas the latter can be thought of as the "practical" value if practitioners adopt the heuristics mentioned above for each setting.

5.1. Performance of the Offering-All Policy

We start our evaluation of the offering-all policy in two specific model instances considered above: M and M + 1 model instances. (We need not to evaluate the offering-all policy in the N and W model instances, because the offering-all policy is optimal there.) Here, we use backward induction to determine the expected performance of the optimal policy and compare it through simulation with that of the offering-all policy π_0. To this

Table 1. Optimality Gap of the Offering-All Policy in the M Model Instance

		$(\lambda_1,\lambda_2)=(1/2,1/2)$, %			$(\lambda_1,\lambda_2)=(1/3,2/3)$, %			$(\lambda_1,\lambda_2)=(1/4,3/4)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
20	45	−4.4	−3.6	−3.6	−4.1	−3.3	−3.3	−3.6	−2.9	−3.0
30	91	−4.8	−3.7	−3.8	−4.5	−3.5	−3.5	−3.8	−3.1	−3.2
40	153	−5.1	−3.8	−3.8	−4.7	−3.6	−3.6	−4.0	−3.2	−3.3
50	231	−5.3	−3.8	−3.8	−4.8	−3.7	−3.7	−4.1	−3.3	−3.4

end, we simulate the offering-all policy for 1,000 days. The performance metric of interest is the percentage optimality gap defined as $(u_g - u_o)/u_o \times 100\%$, where u_o is the expected fill count of the optimal policy and u_g is the average fill count over 1,000 simulated days under the offering-all policy.

Table 1 summarizes the statistics on the optimality gap of the offering-all policy in the M model instance. For each $N = 20, 30, 40, 50$, we evaluate the maximum, average, and median optimality gaps over all possible initial capacity vectors $(b_1, b_2, b_3) \in \mathbb{Z}_+^3$, such that $b_j \geq 0.2N$, $\forall j$ and $b_1 + b_2 + b_3 = N$. The number of initial capacity vectors considered for each N is shown as the number of scenarios in the second column of Table 1. In general, the optimality gap of the offering-all policy in the M model instance is relatively small ($\approx$ 3%–4%) and is not sensitive to model parameters.

Table 2 shows the optimality gap statistics for the M + 1 model instance, and the setup of this table is similar to that of Table 1. When λ_3 is small, the M + 1 model instance is very similar to the M model, and thus, the optimality gaps of the offering-all policy are similar to those observed in Table 1. As λ_3 increases, the performance of the offering-all policy improves, because the offering-all policy becomes more likely to be optimal.

To evaluate the performance of the offering-all policy in settings beyond these two simple instances, we carry out an extensive numerical study using randomly generated customer preference matrices. Fixing the number of slot types J, there are 2^J different possible customer types, including those that accept no slots at all. By allowing any possible combination of these customer types, there could be $2^{2^J} - 1$ possible preference matrices (excluding the empty matrix). To test the performance of the offering-all policy in a robust and yet computationally tractable manner, we compare its performance among many randomly generated such preference matrices.

We also vary the arrival probability vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_I)$ for each preference matrix. In particular, we test three possible vectors: $\boldsymbol{\lambda}^{(1)}$, such that $\lambda_i^{(1)} = 1/I$; $\boldsymbol{\lambda}^{(2)}$, such that $\lambda_i^{(2)} = 2(I + i - 2)/(3I^2 - 3I)$; and $\boldsymbol{\lambda}^{(3)}$, with $\lambda_i^{(3)} = 2(I + 3i - 4)/(5I^2 - 5I)$. In all three cases, the λ_i values add up to one. For $\boldsymbol{\lambda}^{(2)}$ and $\boldsymbol{\lambda}^{(3)}$, λ_1 is the largest, and each successive λ_i is smaller by a factor two or four, respectively. Note that the value of I depends on the randomly generated preference matrix and may vary from $I = 1$ (because we exclude the empty matrix) to the maximum number of customer types.

Our results are summarized in Table 3, where we show the optimality gap of the offering-all policy. We compute the performance of the offering-all policy through simulation as before, and the performance of the optimal policy through backward induction. We fix J and N, and then, we generate the number of random instances indicated in Table 3 ("number of instances"). For each instance, we also vary the initial capacity vectors similar to what was done for Tables 1 and 2 ("number of scenarios" in Table 3). Fixing the structure of the arrival rate vector, we then report the maximum, average, and median optimality gaps over all instances and scenarios. It is clear from this table that the offering-all policy continues to do very well, and the average gap with the optimal policy is around 0.5% throughout, independent of the size of the matrix and the arrival rates.

Before proceeding to the next section, we briefly discuss the performance of π^{p^*} (i.e., the static randomized policy arising from the fluid model in Section 3.2). We focus on the M model and vary N, the arrival probabilities, and the initial capacity vectors. We report the optimality gap statistics for π^{p^*} in Table 4. We observe that the average optimality gap decreases from

Table 2. Optimality Gap of the Offering-All Policy in the M + 1 Model Instance

		$(\lambda_1,\lambda_2,\lambda_3)=(9/20,9/20,1/10)$, %			$(\lambda_1,\lambda_2,\lambda_3)=(2/5,2/5,1/5)$, %			$(\lambda_1,\lambda_2,\lambda_3)=(3/10,3/10,2/5)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
20	45	−3.1	−2.0	−1.9	−2.0	−1.1	−0.9	−0.7	−0.3	−0.2
30	91	−3.4	−2.1	−2.0	−2.3	−1.1	−1.0	−0.8	−0.3	−0.2
40	153	−3.7	−2.1	−2.0	−2.5	−1.2	−1.0	−0.8	−0.2	−0.1
50	231	−3.9	−2.2	−2.0	−2.6	−1.2	−1.0	−0.8	−0.2	−0.1

Table 3. Optimality Gap of the Offering-All Policy for Random Network Instances

				$\lambda=\lambda^{(1)}$, %			$\lambda=\lambda^{(2)}$, %			$\lambda=\lambda^{(3)}$, %		
J	N	No. of instances	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
3	10	100	36	−3.9	−0.2	−0.0	−3.9	−0.2	−0.0	−3.7	−0.3	−0.0
3	20	80	120	−4.6	−0.3	−0.1	−4.6	−0.3	−0.1	−5.1	−0.4	−0.1
3	30	40	253	−5.2	−0.3	−0.1	−5.1	−0.4	−0.1	−3.6	−0.3	−0.1
3	40	10	435	−3.0	−0.3	−0.0	−4.0	−0.5	−0.1	−4.4	−0.5	−0.2
4	10	100	84	−5.0	−0.3	−0.1	−4.7	−0.4	−0.2	−4.0	−0.3	−0.1
4	20	10	455	−3.7	−0.5	−0.3	−2.8	−0.5	−0.3	−4.7	−0.5	−0.3
4	30	10	83	−3.6	−0.6	−0.3	−3.6	−0.7	−0.2	−3.0	−0.6	−0.4
5	10	100	126	−3.9	−0.4	−0.2	−3.7	−0.4	−0.3	−4.2	−0.5	−0.3
5	20	10	126	−2.1	−0.3	−0.2	−3.9	−1.0	−0.6	−4.8	−0.6	−0.3

about 8% to 5% when N increases from 20 to 50. This is consistent with our theory above that π^{p^*} is asymptotically optimal when the demand and capacity increase simultaneously. Because of space constraints, we shall refrain from further exploring the computational issues of π^{p^*} and leave those for future research.

5.2. Performance of the Drain Heuristic

In this section, we evaluate the performance of our drain heuristic developed in Section 4.5. We focus on the N, M, and W model instances. As in Section 5.1, we vary the mix of customer types, the total number of periods, and the initial capacity vectors. The performance of the optimal sequential offering policy is evaluated by backward induction. The performances of the drain heuristic are evaluated by running a discrete event simulation with 1,000 days replication and then computing the average fill count per day. We present the statistics on the percentage optimality gaps of drain in Tables 5, 6, and 7 for the N, M, and W instances, respectively. In particular, for the N and W model instances, the optimality gap statistics are taken over all initial capacity vectors (b_1,b_2), such that $(b_1,b_2)\in\{(x,y)\in\mathbb{Z}_+^2: x,y\geq 0.2N, x+y=N\}$. The second column of each table shows the number of initial capacity vectors consider for each N.

In the N model, the optimality gap of drain is on average within 0.4% (max 0.8%) in all 264 scenarios that we tested. The performances of drain in the W instance are slightly better than those in the N model. For the M model instance, the optimality gap of drain is on average within 0.7% (max 1.4%) across all 1,560 scenarios that we tested. These observations suggest that the drain heuristic has a remarkable performance. Given its simplicity, it can serve as an effective scheduling rule for practitioners.

5.3. Value of Sequential Offering

5.3.1. Comparison of Optimal Policies.

In this section, we investigate the value of sequential scheduling by comparing the optimal sequential policy with the optimal nonsequential policy. We focus on the N, M, and

Table 4. Optimality Gap of the Static Randomized Policy π^{p^*} in the M Model Instance

		$(\lambda_1,\lambda_2)=(1/2,1/2)$, %			$(\lambda_1,\lambda_2)=(1/3,2/3)$, %			$(\lambda_1,\lambda_2)=(1/4,3/4)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
20	45	−10.7	−7.7	−7.2	−9.8	−7.9	−8.0	−10.8	−8.7	−8.7
30	91	−9.1	−6.4	−5.9	−8.7	−6.7	−6.7	−8.8	−7.2	−7.1
40	153	−8.1	−5.7	−5.2	−7.9	−5.8	−5.8	−7.8	−6.2	−6.2
50	231	−7.5	−5.2	−4.6	−7.1	−5.3	−5.2	−7.0	−5.6	−5.5

Table 5. Optimality Gap of the Drain Heuristic in the N Model Instance

		$(\lambda_1,\lambda_2)=(1/2,1/2)$, %			$(\lambda_1,\lambda_2)=(1/3,2/3)$, %			$(\lambda_1,\lambda_2)=(1/4,3/4)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
20	13	−0.8	−0.4	−0.4	−0.6	−0.1	−0.2	−0.7	−0.2	−0.3
30	19	−0.6	−0.2	−0.4	−0.8	−0.2	−0.1	−0.5	−0.0	−0.1
40	25	−0.6	−0.2	−0.2	−0.8	−0.1	−0.1	−0.5	−0.0	−0.0
50	31	−0.5	−0.1	−0.2	−0.5	−0.2	−0.2	−0.6	−0.0	−0.1

Table 6. Optimality Gap of the Drain Heuristic in the M Model Instance

		$(\lambda_1, \lambda_2) = (1/2, 1/2)$, %			$(\lambda_1, \lambda_2) = (1/3, 2/3)$, %			$(\lambda_1, \lambda_2) = (1/4, 3/4)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
20	45	−1.4	−0.7	−0.8	−1.1	−0.4	−0.4	−0.9	−0.2	−0.2
30	91	−0.9	−0.6	−0.6	−0.8	−0.3	−0.3	−0.7	−0.2	−0.2
40	153	−0.7	−0.5	−0.5	−0.7	−0.3	−0.3	−0.9	−0.2	−0.1
50	231	−0.6	−0.4	−0.5	−0.6	−0.2	−0.3	−0.6	−0.1	−0.2

W model instances. To provide a robust performance evaluation, we vary a range of model parameters, including the mix of customer types, the total number of periods, and the initial capacity vectors, like in earlier sections. Table 8 presents the maximum, average, and median percentage improvements in fill count by following an optimal sequential offering policy compared with the optimal nonsequential policy in the N model instance. Tables 9 and 10 present similar information for the M and W model instances, respectively.

We observe that the efficiency gains in the M and W model instances are robust to the initial customer type mix. The efficiency gain in the W model instance is about 6%–7% on average and can be as high as 13%. The efficiency gain in the M model instance is slightly lower. For the N model instance, the gain is relatively more sensitive to customer type mix and ranges between 6% and 11% on average. In certain cases, the efficiency gain in the N model can be as high as 18%. These numerical findings show that sequential offering holds strong potentials to improve the operational efficiency in appointment scheduling systems.

5.3.2. Comparison of Heuristics.

In this section, we compare the performances of two heuristic scheduling policies discussed above: the offering-all policy and the drain heuristic developed in Section 4.5. We also consider another policy called the *random sequential offering policy*, which offers available slot types one at a time in a permutation chosen uniformly at random. This policy mimics the existing practice of telephone scheduling, which is often done without careful planning. These three policies are used or can be easily used by practice, and therefore, the comparison results in this section reveal the value of sequential scheduling that may be realized by adopting these policies in practice.

We focus on the N, M, and W model instances and use the combinations of parameters as in earlier sections. The performance of these three policies is evaluated by running a discrete event simulation with 1,000 days of replication and then computing the average fill count per day for each policy. We present the percentage improvement in the fill count of drain over the other two policies. Detailed results are shown in Tables 11, 12, and 13.

For the N model instance, we see an average 9%–11% improvement (max 18%) if using drain compared with using random sequential or the offering-all policy. In the M model, the average improvement is around

Table 7. Optimality Gap of the Drain Heuristic in the W Model Instance

		$(\lambda_1, \lambda_2, \lambda_3) = (1/3, 1/3, 1/3)$, %			$(\lambda_1, \lambda_2, \lambda_3) = (1/5, 1/2, 3/10)$, %			$(\lambda_1, \lambda_2, \lambda_3) = (1/10, 3/10, 3/5)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
20	13	−0.2	0.0	0.1	−0.1	0.1	0.1	−0.7	−0.2	−0.1
30	19	−0.7	0.0	0.0	−0.4	0.0	0.0	−0.6	−0.1	−0.1
40	25	−0.2	0.0	0.0	−0.2	0.0	0.0	−0.5	0.1	0.0
50	31	−0.2	0.0	0.0	−0.2	0.0	0.0	−0.4	−0.1	−0.1

Table 8. Fill Count Improvement in the N Model Instance (Opt Sequential vs. Opt Nonsequential)

		$(\lambda_1, \lambda_2) = (1/2, 1/2)$, %			$(\lambda_1, \lambda_2) = (1/4, 3/4)$, %			$(\lambda_1, \lambda_2) = (3/4, 1/4)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
20	13	16.0	10.6	12.4	10.8	9.0	9.6	13.2	6.2	5.5
30	19	16.8	10.9	12.6	11.1	9.3	9.6	14.0	6.3	5.4
40	25	17.2	11.1	12.5	11.2	9.5	9.9	14.5	6.4	5.3
50	31	17.5	11.2	12.9	11.3	9.6	9.8	14.8	6.4	5.3

Table 9. Fill Count Improvement in the M Model Instance (Opt Sequential vs. Opt Nonsequential)

		$(\lambda_1,\lambda_2)=(1/2,1/2)$, %			$(\lambda_1,\lambda_2)=(1/3,2/3)$, %			$(\lambda_1,\lambda_2)=(1/4,3/4)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
20	45	7.4	4.0	3.4	7.8	4.2	3.8	6.9	4.1	4.0
30	91	7.9	3.7	3.3	8.3	4.1	3.8	7.2	4.2	4.2
40	153	8.3	3.5	2.9	8.5	4.1	3.9	7.4	4.3	4.3
50	231	8.5	3.3	2.6	8.7	4.1	4.0	7.5	4.4	4.4

Table 10. Fill Count Improvement in the W Model Instance (Opt Sequential vs. Opt Nonsequential)

		$(\lambda_1,\lambda_2,\lambda_3)=(1/3,1/3,1/3)$, %			$(\lambda_1,\lambda_2,\lambda_3)=(1/5,1/2,3/10)$, %			$(\lambda_1,\lambda_2,\lambda_3)=(1/10,3/10,3/5)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
20	13	8.2	6.1	6.5	10.8	6.6	7.0	11.2	7.7	8.3
30	19	9.0	6.6	7.9	11.8	7.0	7.2	11.6	8.1	9.2
40	25	9.5	6.9	7.9	12.3	7.2	7.3	12.0	8.3	9.4
50	31	9.8	7.1	8.3	12.7	7.3	7.3	12.2	8.4	9.4

Table 11. Comparison of the Drain Heuristic with Other Scheduling Policies (the N Model Instance)

		$(\lambda_1,\lambda_2)=(1/2,1/2)$, %			$(\lambda_1,\lambda_2)=(1/3,2/3)$, %			$(\lambda_1,\lambda_2)=(1/4,3/4)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
% improvement over offering-all policy										
20	13	16.5	10.2	11.6	14.0	10.3	11.6	11.0	9.0	9.6
30	19	17.1	10.8	12.4	14.0	10.6	11.4	11.7	9.1	9.7
40	25	17.8	10.9	12.5	13.9	10.8	11.4	11.1	9.3	9.6
50	31	17.8	11.2	13.3	14.5	10.9	11.5	11.7	9.6	9.8
% improvement over random sequential										
20	13	16.6	10.2	11.9	14.0	10.3	11.2	11.2	8.7	9.0
30	19	16.7	10.8	12.5	13.9	10.5	11.4	11.8	9.3	9.5
40	25	17.4	11.1	12.7	14.0	11.0	11.9	11.9	9.5	9.6
50	31	18.0	11.1	13.2	14.5	10.9	11.2	11.6	9.5	10.0

Table 12. Comparison of the Drain Heuristic with Other Scheduling Policies (the M Model Instance)

		$(\lambda_1,\lambda_2)=(1/2,1/2)$, %			$(\lambda_1,\lambda_2)=(1/3,2/3)$, %			$(\lambda_1,\lambda_2)=(1/4,3/4)$, %		
N	No. of scenarios	Max	Average	Median	Max	Average	Median	Max	Average	Median
% improvement over offering-all policy										
20	45	12.1	7.0	6.8	11.8	7.4	7.1	10.8	6.9	6.9
30	91	13.7	7.1	6.5	13.4	7.6	7.1	11.3	7.4	7.4
40	153	14.0	7.0	6.3	13.8	7.7	7.7	11.5	7.6	7.8
50	231	13.9	7.0	6.4	14.0	7.8	7.9	11.6	7.9	8.0
% improvement over random sequential										
20	45	12.3	7.0	6.9	13.2	7.4	6.8	11.1	6.9	6.7
30	91	13.5	7.1	6.7	13.7	7.6	7.1	11.0	7.4	7.4
40	153	13.8	7.0	6.4	13.7	7.7	7.4	11.4	7.7	7.9
50	231	14.2	7.0	6.4	14.2	7.9	7.7	11.8	7.8	8.0

Table 13. Comparison of the Drain Heuristic with Other Scheduling Policies (the W Model Instance)

N	No. of scenarios	$(\lambda_1, \lambda_2, \lambda_3) = (1/3, 1/3, 1/3)$, %			$(\lambda_1, \lambda_2, \lambda_3) = (1/5, 1/2, 3/10)$, %			$(\lambda_1, \lambda_2, \lambda_3) = (1/10, 3/10, 3/5)$, %		
		Max	Average	Median	Max	Average	Median	Max	Average	Median
% improvement over offering-all policy										
20	13	8.0	6.1	6.9	10.8	6.6	6.7	11.6	7.8	8.5
30	19	9.1	6.5	7.6	11.7	6.9	6.9	11.9	8.1	9.1
40	25	9.7	6.9	7.8	12.5	7.2	7.2	12.3	8.4	9.5
50	31	10.3	7.0	7.9	12.6	7.3	7.3	12.4	8.4	9.3
% improvement over random sequential										
20	13	8.5	6.1	6.8	10.9	6.7	6.9	10.9	7.5	8.1
30	19	9.6	6.5	7.9	11.9	6.9	7.3	11.5	8.0	9.3
40	25	9.8	7.0	8.1	12.5	7.2	7.5	12.3	8.4	9.4
50	31	10.2	7.1	7.9	12.8	7.3	7.4	12.2	8.3	9.4

7%–8%, with max 14%. For the W model, drain makes on average 6%–8% improvement over random sequential or the offering-all policy, with the maximum improvement up to 13%. It is worth remarking on that, in all model instances, the random sequential policy has about the same performance as the offering-all policy. Therefore, although the former is a sequential policy and the latter is not, the potential of sequential offering is not exploited because of the careless choice of the offered slots.

6. Conclusion

Motivated by the increasing popularity of online appointment booking platforms, we study how to offer appointment slots to customers to maximize the total number of slots filled. We consider two models, non-sequential offering and sequential offering, for different customer-scheduler interactions in the appointment booking process. For each model, we develop optimal or near-optimal booking policies.

In our numerical experiments, we find that sequential offering in a proper manner makes a significant improvement over the two benchmark policies: the random sequential offering policy (that mimics the existing practice of telephone scheduling) and the offering-all policy (that resembles many of the current online appointment booking systems). Indeed, carefully designed sequential offerings can achieve the performance of a scheduler with perfect information on each customer but without revealing customer types. These findings suggest substantial potential for improving the current appointment scheduling practice.

Another notable observation from our numerical study is that the two benchmark policies have quite similar performance, which indicates that current online scheduling (that often offers all available slots) and traditional telephone scheduling (without a careful offer sequence) would result in similar fill rates. Thus, one should not expect that implementing an online scheduling system in place of traditional telephone scheduling can automatically lead to more appointments booked. In fact, if the telephone scheduler was (intentionally or unintentionally) offering slots in a "smart" way, moving to online scheduling may hurt the fill rate if the latter is not implemented carefully.

As our research suggests, however, one may improve the performance of online scheduling by designing an interface that uses the idea of sequential offering, collecting information on customer choice behavior, and then, making offers in a smarter way. Given the recent and ongoing growth of online and mobile appointment booking platforms, our research findings can be particularly useful to inform user interface design of these booking platforms.

In summary, our work provides the first analytical framework to model, compare, and improve various appointment booking processes; we develop optimal or near-optimal booking policies for two representative appointment offering schemes. Our study also suggests many possible directions for future research. To name a few, first, we assume a specific model for customer choice, and future research may consider scheduling decisions under different choice models. Second, it would be interesting to consider other customer behaviors (e.g., cancellations, no shows, recall, and renege after a few trials) in the scheduling models. Third, our numerical study of the multiday scheduling presented in the online appendix is by no means exhaustive, and it would be a fruitful direction to investigate the (optimal) joint offering policy for both day and slot choices. Last but not least, asymptotic regimes with different scaling of model parameters may be interesting objects of study both from a stochastic model theoretical perspective and for informing more efficient operations in practical settings.

Acknowledgments

The authors are grateful to the department editor, the associate editor, and all of the referees for their constructive comments throughout the review process.

Endnote

[1] Even if some of the slot types are unavailable, π^p would still offer these slot types according to the probability vector $\mathbf{p}$. However, customers only consider those slot types that are available in the booking process.

References

Ahmadi-Javid A, Jalali Z, Klassen KJ (2016) Outpatient appointment systems in healthcare: A review of optimization studies. *Eur. J. Oper. Res.* 258(1):3–34.

Akçay Y, Balakrishnan A, Xu SH (2010) Dynamic assignment of flexible service resources. *Production Oper. Management* 19(3):279–304.

Bernstein F, Kök AG, Xie L (2015) Dynamic assortment customization with limited inventories. *Manufacturing Service Oper. Management* 17(4):538–553.

Billingsley P (1968) *Convergence of Probability Measures* (John Wiley & Sons, New York).

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

Chan CW, Farias VF (2009) Stochastic depletion problems: Effective myopic policies for a class of dynamic optimization problems. *Math. Oper. Res.* 34(2):333–350.

Cooper WL (2002) Asymptotic behavior of an allocation policy for revenue management. *Oper. Res.* 50(4):720–727.

Feldman J, Liu N, Topaloglu H, Ziya S (2014) Appointment scheduling under patient preference and no-show behavior. *Oper. Res.* 62(4):794–811.

Gallego G, Li A, Truong V-A, Wang X (2016a) Approximation algorithms for product framing and pricing. Working paper, Department of Industrial Engineering and Operations Research, Columbia University, New York.

Gallego G, Li A, Truong V-A, Wang X (2016b) Online resrouce allocation with customer choice. Working paper, Department of Industrial Engineering and Operations Research, Columbia University, New York.

Golovin D, Krause A (2011) Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artificial Intelligence Res.* 42(1):427–486.

Golrezaei N, Nazerzadeh H, Rusmevichientong P (2014) Real-time optimization of personalized assortments. *Management Sci.* 60(6): 1532–1551.

Green L, Savin S, Wang B (2006) Managing patient service in a diagnostic medical facility. *Oper. Res.* 54(1):11–25.

Gupta D, Denton B (2008) Appointment scheduling in health care: Challenges and opportunities. *IIE Trans.* 40(9):800–819.

Gupta D, Wang L (2008) Revenue management for a primary-care clinic in the presence of patient choice. *Oper. Res.* 56(3):576–592.

Liu N, Ziya S, Kulkarni VG (2010) Dynamic scheduling of outpatient appointments under patient no-shows and cancellations. *Manufacturing Service Oper. Management* 12(2):347–364.

Mehta A (2013) Online matching and ad allocation. *Foundations Trends Theoret. Comput. Sci.* 8(4):265–368.

Subramanian J, Stidham S Jr, Lautenbacher CJ (1999) Airline yield management with overbooking, cancellations, and no-shows. *Transportation Sci.* 33(2):147–167.

Talluri KT, Van Ryzin GJ (2004) Revenue management under a general discrete choice model of consumer behavior. *Management Sci.* 50(1):15–33.

Wang WY, Gupta D (2011) Adaptive appointment systems with patient preferences. *Manufacturing Service Oper. Management* 13(3):373–389.

Zhang D, Cooper WL (2005) Revenue management for parallel flights with customer-choice behavior. *Oper. Res.* 53(3):415–431.