Original Article

POMS

Production and Operations Management
2024, Vol. 33(1) 184–204

DOI: 10.1177/10591478231224926
journals.sagepub.com/home/pao

Sage

Nonsequential Appointment Scheduling With a Random Number of Requests

Yan Zhu[1], Zhixin Liu[2] and Xiangtong Qi[3]

Abstract

This article studies an appointment scheduling problem where a service provider dynamically receives appointment requests from a random number of customers. By leveraging the randomness of the number of potential customers, we develop a nonsequential appointment scheduling policy as an alternative to the conventional first-come-first-served (FCFS) policy. This allows for more flexibility in managing appointment scheduling. To calculate the optimal policy, we develop a branch-and-bound algorithm in which the lower bound is estimated using multiple approaches, such as optimality conditions, dynamic programming for calculating FCFS policy, and the shortest path reformulation. Through numerical studies, we observe that nonsequential appointment scheduling is particularly advantageous in systems characterized by highly fluctuating customer numbers or low congestion. In such cases, leaving gaps between appointments for potential future arrivals proves to be a more appropriate strategy. We also evaluate the performance of heuristics proposed in prior literature and provide insights into situations where these heuristics can be effectively applied.

Keywords

Appointment scheduling, dynamic, nonsequential, branch and bound, shortest path

Date received 25 March 2022; accepted 21 October 2023 after three revisions

Handling Editor: Michael Pinedo

1 Introduction

1.1 Background

Appointment scheduling plays a vital role in achieving operational excellence within a service system by minimizing customer waiting time and maximizing the utilization of system resources. In practice, appointment scheduling can be carried out through various approaches, including offline and online scheduling and inter and intraday scheduling.

In offline appointment scheduling, the system provides appointment schedules after all requests have arrived, whereas, in online appointment scheduling, the system offers an appointment upon receiving a customer's request, regardless of the exact number of upcoming customers (Ahmadi-Javid et al., 2017). In interday scheduling, the system aims to assign appointment requests to future days, taking into account customer preferences and indirect waiting costs (the time between the appointment request date and the appointment date provided) (Gupta and Wang, 2008; Keyvanshokooh et al., 2021; Liu et al., 2019). In intraday scheduling, the system schedules specific service start times for customers with predetermined appointment dates, aiming to strike a balance between waiting times, idle times, and overtime within a single day's operation (Chen and Robinson, 2014; Erdogan et al., 2015).

This article investigates an online intraday appointment scheduling problem with a random number of customers. Such problems commonly arise in healthcare systems, particularly when certain doctors are only available for duty on specific days of the week, as illustrated, for example, by the timetable of specialists of a hospital in Hong Kong.[1] While our research primarily focuses on intraday scheduling, it is also relevant to interday scheduling. In certain interday appointment scheduling systems, customers are initially granted the opportunity to select an available service day, after which the system assigns them a specific start time for their appointment. This is illustrated by the appointment system of MaxHealth,[2] where

[1] International Institute of Finance, School of Management, University of Science and Technology of China, Hefei, P.R. China
[2]Department of Management Studies, College of Business, University of Michigan – Dearborn, Dearborn, MI, USA
[3]Department of Industrial Engineering and Decision Analytics, The Hong Kong University of Science and Technology, Kowloon, Hong Kong, Hong Kong SAR

Corresponding author:
Xiangtong Qi, Department of Industrial Engineering and Decision Analytics, The Hong Kong University of Science and Technology, Kowloon, Hong Kong SAR.
Email: ieemqi@ust.hk

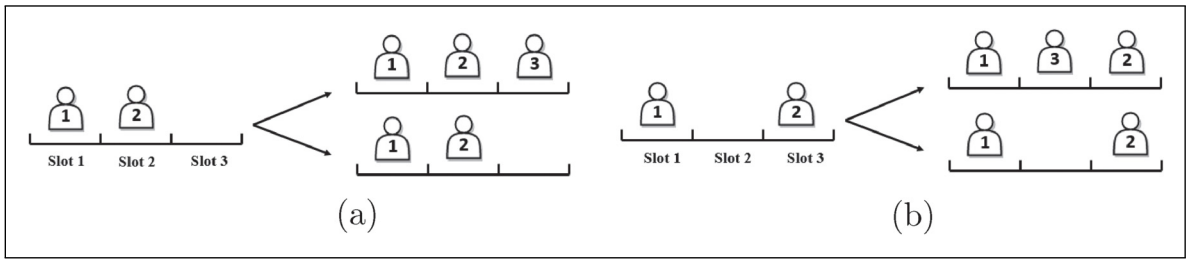

Figure 1. Sequential scheduling and nonsequential scheduling for Example 1: (a) sequential scheduling; (b) nonsequential scheduling.

customers are given the option to choose between morning, lunchtime, or afternoon medical appointments on a particular day. Since customers call in dynamically, the number of upcoming customers on an arbitrary day fluctuates.

1.2 Major Issues

The general difficulty of the online intraday appointment scheduling problem stems from the uncertainty of the number of customers, which leads to the sequencing issue of the customers. In certain systems, customers are scheduled on a first-come-first-served (FCFS) basis, where those submitting requests earlier will be assigned earlier service start times. We refer to such a process as *sequential appointment scheduling*. An example of such a system can be observed in HA Go,[3] an APP used by the Hong Kong public hospital system to provide various services to patients. One particular function of the app is to manage appointments. Once a patient selects a clinic and submits an appointment request, the patient will be provided with the earliest available slot. In some other systems, *nonsequential appointment scheduling* is adopted in which customers submitting requests early may be given later service start times. To illustrate this disparity, we provide Example 1, depicted in Figure 1, to highlight the distinction between sequential and nonsequential appointment scheduling.

EXAMPLE 1. Consider a single server with a service horizon equally divided into three time slots. Each slot can be offered to only one customer. There are two or three potential customers whose service times are identically and independently distributed, and the mean of the service times is equal to the length of a time slot.

In sequential appointment scheduling, time slot i is offered to the i-th customer who submits the request for $i = 1, 2, 3$. The presence or absence of a request from a third customer leads to two possible final appointment schedules, as shown in Figure 1(a). In particular, in the scenario where there are only two customers, if the actual service time for the first customer exceeds the duration of a time slot, it will cause a delay for the second customer.

In nonsequential scheduling, one possible scheduling principle is to assign slot 3 to the second customer submitting the request, thereby reserving slot 2 for the potential request from a third customer. This strategy leads to two possible final appointment schedules, as shown in Figure 1(b). By comparing the overall expected waiting times for customers in Figure 1(a) and (b), we can see that the second customer in Figure 1(b) experiences a much shorter waiting time when there are only two customers. This is because, in this case, the empty slot 2 absorbs the impact of the first customer's randomly varying service time. Although this alone may not be sufficient to firmly conclude that schedule (b) is superior to schedule (a), as other factors such as system idle costs and overtime costs also need consideration, it does highlight the potential advantages of nonsequential scheduling when dealing with an uncertain number of customers.

Such an online intraday nonsequential appointment scheduling problem has been studied in the literature (Chen and Robinson, 2014; Erdogan et al., 2015). These studies reveal the potential benefits of considering sequencing issues when dealing with a random number of customers. However, their focus primarily revolves around proposing heuristics. There is still no algorithm for determining the optimal schedule.

In practice, in addition to uncertainty regarding the number of customers, the task of determining optimal appointments is further complicated by uncertainties in customer behavior. In many appointment service systems, cancellations, late arrivals, and no-shows occur frequently. These behaviors disrupt schedules and increase system idle time (Kong et al., 2020; Liu et al., 2010; Pan et al., 2021). While research has examined the influence of these behaviors on optimal schedules, a comprehensive investigation of these factors in online appointment scheduling problems is lacking.

Our work simultaneously considers three potential customer behavior cases, namely last-minute cancellations, late arrivals, and no-shows. First, if a customer cancels their appointment and notifies the system prior to the scheduled appointment time, the system can bypass them and proceed to serve the next customer. Consequently, there is no waiting time cost incurred for these customers who cancel at the last minute. This is referred to as a *last-minute cancellation* (Liu et al., 2010). Second, in the event of a customer arriving late without prior notification, the system will wait for the customer instead of serving the next customer. The duration of the customer's lateness will not be counted as part of their

waiting time. This is referred to as a *late arrival* (Samorani and Ganguly, 2016). Third, the system cannot confirm whether a customer is a *no-show* until η minutes after their scheduled appointment time, following Deceuninck et al. (2018). Instead of continuing to wait, the system will skip the customer and proceed to serve the next customer. Compared to cancellations, no-shows can result in up to η minutes of wasted system time. We note that there are different definitions of cancellations and no-shows in the existing literature, which will be discussed in the literature review.

1.3 Contribution

We summarize the main results and contributions of our work from three perspectives, namely, the positioning of our model, the algorithms for problem-solving, and the managerial insights for operational guidelines.

For modeling, we first study a basic online appointment scheduling problem with a stochastic number of customers. This particular problem was previously explored by Chen and Robinson (2014) and Erdogan et al. (2015). These studies primarily tackled cases involving continuous appointment times and proposed heuristic approaches for determining appointment sequences. In contrast, our work addresses the discrete appointment times scenario and is the first to optimally solve the online appointment scheduling problem. Significantly, our model can be expanded to incorporate last-minute cancellations, late arrivals, and no-shows by customers, thereby introducing additional complexities to the online scheduling problem beyond the scope of Chen and Robinson (2014) and Erdogan et al. (2015).

With regard to problem-solving, we develop a novel branch-and-bound (B&B) algorithm that integrates two perspectives for this particular problem. First, our problem can be perceived as a dynamic assignment problem, in which the decision is an appointment scheduling principle that specifies the time slot assigned to each customer upon request. Second, our problem can also be formulated as a shortest path problem on a layered directed network, in which each vertex corresponds to a possible state (a term that is formally defined below). Consequently, an appointment scheduling principle can be modeled by the path from a specified source to a specified sink, with the shortest path providing the optimal solution. Both perspectives possess inherent advantages and some disadvantages. To mitigate these drawbacks, we explore various properties of these two models and subsequently develop a set of rules specifically designed to speed up the search process for optimal policies. Hence, the distinctiveness of our algorithm lies in its ability to leverage the strengths of both perspectives while circumventing their respective limitations.

Numerical studies have also yielded several managerial insights, which are presented as follows:

- In scenarios where the number of customers fluctuates significantly or the system is not heavily congested, it is advisable to adopt nonsequential appointment scheduling, that is, leaving some time slots open in between for future arrivals.
- Patient lateness will substantially increase system costs, whereas cancellations and no-shows can partially offset system costs for that particular day.
- For an optimal online policy, we can always find an implemented online policy that is close to or even identical to the corresponding optimal offline policy. The point at which this correspondence appears is closely related to factors such as the expected number of customers, the customer number with the highest probability, and the maximum weighted offline cost. This correspondence pattern motivates us to utilize several offline optimal policies as approximations for an online policy.
- Regarding the heuristics, we compare two approaches: a pairwise interchange proposed in the literature and a greedy algorithm proposed in this study. Our comparison reveals that the pairwise interchange performs better in congested systems. Conversely, the greedy algorithm demonstrates good performance within relatively idle systems. The combination of the two heuristics results in a well-performing heuristic that is suitable for general service systems.

The remainder of this article is organized as follows. In Section 2, we review the relevant literature. In Section 3, we describe the problem, derive optimal properties, and extend our model. In Section 4, we investigate special cases of the sequential scheduling principle. In Section 5, we develop a B&B algorithm. In Section 6, we present the computational results. We conclude this article in Section 7. The proofs of all theorems are available in the E-Companion of the Supplemental Material.

2 Literature Review

Appointment scheduling problems have been investigated in many service industries. Most studies are driven by healthcare management (Deng et al., 2019; Zacharias and Yunes, 2020; Bandi and Gupta, 2020), and some studies contribute to general service systems (Mancilla and Storer, 2012; Kemper et al., 2014; Mak et al., 2014b; Qi, 2017). For comprehensive reviews, refer to Cayirli and Veral (2003), Gupta and Denton (2008), Ahmadi-Javid et al. (2017), and Youn et al. (2022). To keep the review concise, we concentrate on two streams of literature, one on nonsequential appointment scheduling and the other on cancellations, late arrivals, and no-shows.

In offline appointment scheduling, several studies investigate the relationship between the optimal sequence and the distribution of service time. They propose certain heuristic rules, such as the smallest-variance-first rule, which prove to be optimal under some conditions (Weiss, 1990; Wang, 1999; Denton et al., 2007; Mak et al., 2014a; Kemper et al., 2014; Kong et al., 2016; Guda et al., 2016; de Kemp et al., 2021). Moreover, the impact of various customer attributes

on the optimal sequencing decision has been explored, such as customers with different probabilities of being a no-show (LaGanga and Lawrence, 2012; Zacharias and Pinedo, 2014) and different delay-unpleasantness measures (Qi, 2017).

In online appointment scheduling, most studies in this stream of the literature are centered around healthcare applications, where customers are classified into two types, that is, routine customers and same-day (or walk-in) customers. Routine customers are scheduled with sufficient prior information, whereas same-day customers and walk-in customers introduce a source of uncertainty into the system. Throughout the existing literature, the treatment of same-day and walk-in customers differs markedly.

Same-day customers must submit an appointment request before the start of the working horizon in order to receive service. Our work is closely related to such problems, which were initially explored by Chen and Robinson (2014) and Erdogan et al. (2015). Chen and Robinson (2014) considered sequencing and scheduling appointments for both homogeneous routine patients and heterogeneous same-day patients. The critical difference between our work and theirs is that Chen and Robinson (2014) employed heuristic rules to identify appointment sequences, whereas we design an algorithm to determine the optimal sequence. Furthermore, Chen and Robinson (2014) assumed that a doctor's day ends once he or she finishes attending to the last scheduled patient. They acknowledge that their heuristic structure ceases to be applicable if the doctor is obligated to remain until the end of the working horizon. This particular scenario is the focus of our investigation. Erdogan et al. (2015) delved into the same model and develop a stochastic program. They also design heuristic sequencing rules to solve this problem. What distinguishes our work from Erdogan et al. (2015) is that they enumerate appointment sequences using an integer program, whereas we implicitly enumerate them using a B&B strategy. Another technical difference is that Chen and Robinson (2014) and Erdogan et al. (2015) assumed that appointment times are continuous, for which a mathematical program is a natural solution tool. In contrast, we consider appointment times to be discrete (e.g., multiples of 10 min), a more practical approach, and subsequently employ a tailored B&B algorithm.

Distinct from same-day customers, walk-in customers arrive unpredictably during the service horizon and have the opportunity to receive service without a prior appointment. Green et al. (2006) established dynamic priority rules for admitting customers to service, particularly when two types of customers are waiting simultaneously. Freeman et al. (2016) examined the scheduling of routine customers, emphasizing the need to serve potential walk-in customers within a specified waiting time upon arrival. Zacharias and Yunes (2020) studied an online scheduling problem, in which general stochastic service times, no-shows, nonpunctuality, and walk-ins are jointly considered. They investigate the multimodularity of the objective function in a general setting. Wang et al. (2020) studied a similar problem to that of Zacharias and Yunes (2020) but from a fundamentally different angle, reformulating the challenging problem to a tractable mathematical program.

In addition, online appointment scheduling problems can also be categorized into intraday (i.e., at what time) and interday (i.e., on which day) scheduling. Most of the above online scheduling literature focuses on intraday scheduling. In interday appointment scheduling, there exists a concept analogous to nonsequential scheduling known as the "leaving holes" strategy (Patrick et al., 2008; Sauré and Puterman, 2014; Izady, 2019). This strategy involves reserving some capacity to accommodate potential same-day demand while allocating daily capacity for advance booking demand. Patrick et al. (2008) explored this strategy in the context of scheduling surgical patients with varying priorities. Truong (2015) characterizes an optimal policy for the dynamic interday problem and derives an efficient algorithm for policy computation. Wang and Truong (2018) investigated a multi-priority interday scheduling problem that incorporates cancellations, with the competition ratio serving as the optimization criterion. Wang et al. (2019) considered sequential appointment scheduling in a network of stations with exponential service times, no-shows, and overbooking. Keyvanshokooh et al. (2021) studied an online resource allocation problem in which arrivals with different service times and rewards make appointments on several servers. Keyvanshokooh et al. (2022) addressed healthcare coordination aimed at setting appointments for pairs of sequential visits and minimizing the indirect access time. Zacharias et al. (2022) simultaneously considered both inter and intraday scheduling in dynamic scheduling decisions.

In addition to offline and online scheduling problems, there is another category of appointment scheduling problem known as "myopic scheduling." In myopic scheduling, the system immediately schedules an appointment as soon as a customer submits a request, without considering potential future requests (Muthuraman and Lawley, 2008; Zacharias and Pinedo, 2014). The key distinction between myopic and online scheduling lies in their consideration of future requests, with online scheduling taking into account such possibilities while myopic scheduling does not.

The second body of literature pertains to the challenges posed by last-minute cancellations, late arrivals, and no-shows. Past research has made significant contributions to appointment scheduling with no-shows and minimizing their impact on schedules, thus achieving a better balance between demand and capacity. Some studies have focused on homogeneous no-show probabilities (Green et al., 2006; Hassin and Mendel, 2008), and others have delved into heterogeneous no-show rates (Zeng et al., 2010; Chen and Robinson, 2014) and time-dependent no-show rates (Kong et al., 2020; Wang et al., 2020; Zacharias and Yunes, 2020). Additionally, some papers consider the rescheduling of a no-show or canceled appointment within the context of interday scheduling problems (Xie et al., 2021). However, relatively few papers have focused on

cancellations (Liu et al., 2010) and late arrivals (Deceuninck et al., 2018; Pan et al., 2021).

It is worth mentioning that the existing literature presents various definitions of cancellations and no-shows. In the study by Liu et al. (2010), the focus is on interday cancellations, where customers can cancel their appointments prior to the scheduled day and the vacant slots resulting from these cancellations can be allocated to future customers. In addition, Chen and Robinson (2014) assumed that if a patient fails to arrive promptly for their appointment, the system can immediately recognize the customer as a no-show, skip them, and serve the next customer, which is equivalent to a last-minute cancellation in our work. Regarding no-shows, according to Zacharias and Yunes (2020), the system does not need to wait for a specific patient at their appointment time. Instead, the system continues to serve customers who have already arrived and are waiting in the queue. Consequently, the actual order of service may differ from the scheduled sequence. In the study of Xie et al. (2021), cancellations and no-shows are considered synonymous and referred to as "missed appointments." Some papers consider two features simultaneously, including both no-shows and late arrivals (Zacharias and Yunes, 2020; Jiang et al., 2019; Pan et al., 2021; Jouini et al., 2022), as well as the combination of no-shows and cancellations (Liu et al., 2010; Feldman et al., 2014; Diamant, 2021). However, there are limited studies that comprehensively address the triad of last-minute customer cancellations, late arrivals, and no-shows in the context of online appointment scheduling. Furthermore, Zacharias and Yunes (2020) shed light on the exponential complexity of objective function evaluations when nonpunctual patients are taken into account. To the best of our knowledge, the simultaneous consideration of last-minute customer cancellations, late arrivals, and no-shows in online appointment scheduling is novel in the existing literature.

3 Model and Properties

3.1 Problem Description and Dynamic Assignment Model

To enhance comprehension of the model, we initially present the problem formulation without incorporating last-minute cancellations, late arrivals, or no-shows. Subsequently, in Section 3.4, we extend this model to encompass the simultaneous consideration of these three stochastic features.

We consider a scenario where there is a single server attending to a random number of customers. The server is available for a specified time interval, denoted as $[0, T]$, where T represents the given horizon (e.g., a morning or an afternoon). Prior to the start of the service interval, each customer is required to schedule an appointment with the server. The number of customers, denoted as $\boldsymbol{n}$, follows a distribution $(p_1, \ldots, p_{n^+})$, where p_i specifies the probability of $\boldsymbol{n} = i$, and $\sum_{i=1}^{n^+} p_i = 1$. Here, n^+ represents the maximum possible number of customers within the specified interval. The service times for all potential customers, denoted by $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{n^+}$, are independently and identically distributed random variables. These service times are integer values with a mean of $\bar{s}$, a minimum of s^-, and a maximum of s^+. When there is no ambiguity, we use $\boldsymbol{s}$ to represent the service time for a customer.

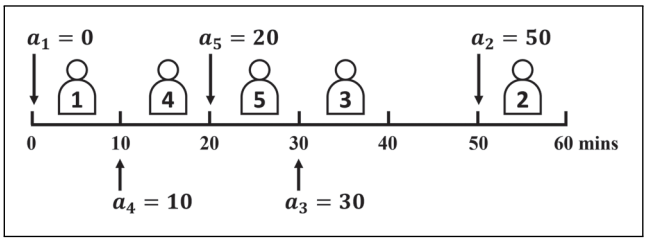

Figure 2. The assigned time slots and corresponding service start times for Example 2.

The service horizon is divided into N evenly spaced time slots, with each slot having a length of $t = T/N$, where t is an integer, for example, $t = 5$ or $t = 10$ min. The server determines the service start time of a customer by assigning a time slot to the customer. Consequently, the customer assigned to the i-th time slot has a service start time $(i - 1)t$. Some studies consider overbooking in the appointment scheduling problem. Zacharias and Yunes (2020) discussed two types of overbooking appointments: (i) double booking certain slots or (ii) allowing service times of appointments to overlap. Our work adopts the second type of overbooking. No two customers will be assigned the same time slot, that is, the same service start time. However, the service times of appointments can overlap. If the length of a time slot t is shorter than the expected service time $\bar{s}$, customers assigned to consecutive time slots may experience overlapping appointments in terms of service duration. To ensure that each customer has a unique service start time, it is necessary for N to be greater than or equal to n^+. This condition can be satisfied by adjusting the time slot length t.

In the online appointment scheduling problem, the decision can be represented by an appointment policy, formally defined as follows.

DEFINITION 1. An appointment policy is represented by a vector $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_{n^+}]$, where $\pi_i \in \{1, 2, \ldots, N\}$, and $\pi_i \neq \pi_j$ for $i \neq j$. The value of π_i specifies the time slot assigned to the i-th customer with a service start time of $a_i = (\pi_i - 1)t$.

EXAMPLE 2. A system has $N = 6$ time slots with $t = 10$, and the maximum number of customers is $n^+ = 5$. Consider an appointment policy $\boldsymbol{\pi} = [1, 6, 4, 2, 3]$. The assigned time slots and corresponding service start times are shown in Figure 2.

Henceforth, for clarity and to distinguish between various policies, we employ additional boldface Greek letters, such as $\boldsymbol{\tau}$ and $\boldsymbol{\nu}$, to denote different strategies.

As the number of customers $\boldsymbol{n}$ is a random variable, it is necessary to assess a policy $\boldsymbol{\pi}$ for each potential value of $\boldsymbol{n}$.

Let us consider the scenario where $\boldsymbol{n} = j$. In this case, the actual schedule for the first j customers, governed by $\boldsymbol{\pi}$, is represented by the sub-vector $\boldsymbol{\pi}^j = [\pi_1, \ldots, \pi_j]$. This can be described by a concept referred to as a "partial policy."

DEFINITION 2. For a policy $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_{n^+}]$, a partial policy $\boldsymbol{\pi}^j$ denoted by sub-vector $[\pi_1, \ldots, \pi_j]$ specifies the schedule of the first j customers. Accordingly, $\boldsymbol{\pi}$ is said to be derived from $\boldsymbol{\pi}^j$. More generally, for any $i < j$, partial policy $\boldsymbol{\pi}^j = [\pi_1, \ldots, \pi_i, \pi_{i+1}, \ldots, \pi_j]$ is said to be derived from $\boldsymbol{\pi}^i = [\pi_1, \ldots, \pi_i]$, which can also be written as $\boldsymbol{\pi}^j = [\boldsymbol{\pi}^i; \pi_{i+1}, \ldots, \pi_j]$.

The evaluation of a policy $\boldsymbol{\pi}$ includes three commonly considered costs, namely, customer waiting time costs, system idle time costs, and system overtime costs (Begen and Queyranne, 2011; Ahmadi-Javid et al., 2017). We define the cost of a policy $\boldsymbol{\pi}$ as $G(\boldsymbol{\pi})$ and the cost of a corresponding partial policy $\boldsymbol{\pi}^j$ as $g_j(\boldsymbol{\pi}^j)$. Consequently, the objective function $G(\boldsymbol{\pi})$, derived from different partial policies under the different realizations of $\boldsymbol{n} = j$, can be formulated as:

$$G(\boldsymbol{\pi}) = \sum_{j=1}^{n^+} p_j \, g_j(\boldsymbol{\pi}^j). \tag{1}$$

We also denote the partial cost for the first k possible realized numbers of $\boldsymbol{n}$ by $G_k(\boldsymbol{\pi}^k)$ as follows:

$$G_k(\boldsymbol{\pi}^k) = \sum_{j=1}^{k} p_j \, g_j(\boldsymbol{\pi}^j). \tag{2}$$

The scheduling policy we consider is nonsequential, meaning that in the scenario where $\boldsymbol{n} = j$, customers may not necessarily be served in the same order as in the scenario where $\boldsymbol{n} = j + 1$. This nonsequential nature makes calculating the expected cost of a policy $\boldsymbol{\pi}$ difficult. However, given that the service time distributions are independent and identical, it is more convenient to utilize the positions of assigned time slots rather than the sequence of appointment times when calculating the expected cost of a policy. In the case of $\boldsymbol{n} = j$, we define the state of a partial policy by representing the positions of occupied time slots.

DEFINITION 3. For a partial policy $\boldsymbol{\pi}^j = [\pi_1, \ldots, \pi_j]$, we define state $\boldsymbol{\pi}^{(j)} = [\pi_1^{(j)}, \ldots, \pi_j^{(j)}]$ as a rearrangement of $\pi_1, \ldots, \pi_j$ such that $\pi_1^{(j)} < \ldots < \pi_j^{(j)}$, and each $\pi_i^{(j)}$ specifies the i-th occupied time slot in the state $\boldsymbol{\pi}^{(j)}$.

The state vector $\boldsymbol{\pi}^{(j)}$, indicating the positions of all occupied slots, can also be represented by a 0-1 vector, where a value of 1 specifies an occupied slot and a value of 0 specifies an empty slot.

EXAMPLE 3. Consider two appointment policies $\boldsymbol{\pi} = [1, 6, 4, 2, 3]$ and $\boldsymbol{\tau} = [1, 6, 2, 4, 3]$. The partial policies $\boldsymbol{\pi}^4$ and $\boldsymbol{\tau}^4$, along with their corresponding states, are presented below.

	Partial Policy	State	0-1 Vector
$\boldsymbol{\pi}^4$	[1, 6, 4, 2]	[1, 2, 4, 6]	[1, 1, 0, 1, 0, 1]
$\boldsymbol{\tau}^4$	[1, 6, 2, 4]	[1, 2, 4, 6]	[1, 1, 0, 1, 0, 1]

In Example 3, the two partial policies share the same state, indicating that they yield the same performance when there are exactly four customers. However, when there are three customers, the state $\boldsymbol{\pi}^{(3)} = [1, 4, 6]$ is different from the state $\boldsymbol{\tau}^{(3)} = [1, 2, 6]$. Hence, due to the randomness of the number of customers, the overall performance of these two policies may diverge as a result of the distinct appointment arrangements for the scenario involving exactly three customers.

Given a partial policy $\boldsymbol{\pi}^j$, we define the cost of the corresponding state $\boldsymbol{\pi}^{(j)}$ as $g_j(\boldsymbol{\pi}^{(j)})$. According to the definition of the state for a partial policy, we have $g_j(\boldsymbol{\pi}^{(j)}) = g_j(\boldsymbol{\pi}^j)$. Subsequently, we evaluate the performance of a state $\boldsymbol{\pi}^{(j)}$ with respect to customer waiting time costs, system idle time costs, and system overtime costs.

First, we model the waiting time of a customer. Similar to the definition of a_i, for a customer assigned to the i-th occupied slot $\pi_i^{(j)}$ in state $\boldsymbol{\pi}^{(j)}$, the appointment time is denoted as $a_i^{(j)} = (\pi_i^{(j)} - 1)t$. We use $\boldsymbol{z}_i^{(j)}$, $\boldsymbol{e}_i^{(j)}$, and $\boldsymbol{w}_i^{(j)}$ to represent the actual start time, service completion time, and waiting time of the corresponding customer, respectively. Taking into account the potential delay caused by the immediately preceding customer, the actual start time of a customer's service depends on the customer's appointment time and the completion time of the previously served customer. Note that $\boldsymbol{z}_1^{(j)} = 0$, $\boldsymbol{e}_1^{(j)} = \boldsymbol{s}$, and $\boldsymbol{w}_1^{(j)} = 0$. For $i = 2, \ldots, j$, these random variables exhibit the following relationship:

$$\boldsymbol{z}_i^{(j)} = \max\{a_i^{(j)}, \boldsymbol{e}_{i-1}^{(j)}\}, \quad \boldsymbol{e}_i^{(j)} = \boldsymbol{z}_i^{(j)} + \boldsymbol{s}, \quad \boldsymbol{w}_i^{(j)} = (\boldsymbol{z}_i^{(j)} - a_i^{(j)})^+.$$

Second, we introduce $\boldsymbol{I}_i^{(j)}$ as the idle time of the server immediately following the completion of service for the customer assigned to the i-th occupied slot $\pi_i^{(j)}$ within state $\boldsymbol{\pi}^{(j)}$. For $i = 1, \ldots, j-1$, we have $\boldsymbol{I}_i^{(j)} = (a_{i+1}^{(j)} - \boldsymbol{e}_i^{(j)})^+$ and $\boldsymbol{I}_j^{(j)} = (T - \boldsymbol{e}_j^{(j)})^+$.

Finally, we consider the system overtime cost, which can be represented by the waiting time of a dummy customer who is the first to call and is scheduled to start service at time T. Let us denote this overtime as $\boldsymbol{w}_0^{(j)}$. It is determined by the completion time of the last served customer within state $\boldsymbol{\pi}^{(j)}$, where $\boldsymbol{w}_0^{(j)} = (\boldsymbol{e}_j^{(j)} - T)^+$. We compute the distribution of $\boldsymbol{w}_i^{(j)}$, $\boldsymbol{I}_i^{(j)}$, and $\boldsymbol{w}_0^{(j)}$ according to the definition of convolution, that is, the direct method.

Then, the state cost of $\boldsymbol{\pi}^{(j)}$, $g_j(\boldsymbol{\pi}^{(j)})$, can be formulated as follows:

$$g_j(\boldsymbol{\pi}^{(j)}) = \sum_{i=1}^{j} \left(c^w \mathbf{E}[\boldsymbol{w}_i^{(j)}|\boldsymbol{\pi}^{(j)}] + c^I \mathbf{E}[\boldsymbol{I}_i^{(j)}|\boldsymbol{\pi}^{(j)}]\right) + c^o \mathbf{E}[\boldsymbol{w}_0^{(j)}|\boldsymbol{\pi}^{(j)}], \tag{3}$$

where c^w, c^I, and c^o are the waiting time cost coefficient, the idle time cost coefficient, and the overtime cost coefficient, respectively.

Overall, the objective function $G(\boldsymbol{\pi})$, as an evaluation of an appointment policy $\boldsymbol{\pi}$, can be formulated as follows:

$$G(\boldsymbol{\pi}) = \sum_{j=1}^{n^+} p_j \Bigg[\sum_{i=1}^{j} \left(c^w \mathbf{E}[\boldsymbol{w}_i^{(j)}|\boldsymbol{\pi}^{(j)}] + c^I \mathbf{E}[\boldsymbol{I}_i^{(j)}|\boldsymbol{\pi}^{(j)}]\right) + c^o \mathbf{E}[\boldsymbol{w}_0^{(j)}|\boldsymbol{\pi}^{(j)}] \Bigg]. \tag{4}$$

Furthermore, we can simplify the objective function (4) by representing the idle time of the server as a function of the server's overtime, as shown in Remark 1. This approach is similar to the work of Denton and Gupta (2003), who tackled the offline appointment scheduling problem.

REMARK 1. When the number of customers is $\boldsymbol{n} = j$, the difference between the total expected idle time and the expected server overtime is equal to the difference between the length of the horizon T and the number of customers $\boldsymbol{n} = j$ times the expected service time $\mathbf{E}[s]$. Mathematically, this can be written as $\sum_{i=1}^{j} \mathbf{E}[\boldsymbol{I}_i^{(j)}|\boldsymbol{\pi}^{(j)}] - \mathbf{E}[\boldsymbol{w}_0^{(j)}|\boldsymbol{\pi}^{(j)}] = T - j\mathbf{E}[s]$.

Thus, the objective function $G(\boldsymbol{\pi})$ can also be written as follows:

$$G(\boldsymbol{\pi}) = \sum_{j=1}^{n^+} p_j \Bigg(\sum_{i=1}^{j} c^w \mathbf{E}[\boldsymbol{w}_i^{(j)}|\boldsymbol{\pi}^{(j)}] + (c^I + c^o)\mathbf{E}[\boldsymbol{w}_0^{(j)}|\boldsymbol{\pi}^{(j)}] \Bigg) + c^I(T - \mathbf{E}[\boldsymbol{n}]\mathbf{E}[s]). \tag{5}$$

Note that the last term in equation (5), $T - \mathbf{E}[\boldsymbol{n}]\mathbf{E}[s]$ is a constant that is independent of scheduling policies and represents the partial idle time associated with the system utilization. Thus, minimizing equation (5) is equivalent to minimizing the following equation:

$$G(\boldsymbol{\pi}) = \sum_{j=1}^{n^+} p_j \Bigg(\sum_{i=1}^{j} c^w \mathbf{E}[\boldsymbol{w}_i^{(j)}|\boldsymbol{\pi}^{(j)}] + c^\ell \mathbf{E}[\boldsymbol{w}_0^{(j)}|\boldsymbol{\pi}^{(j)}] \Bigg), \tag{6}$$

where $c^\ell = c^I + c^o$. In addition, we present a property regarding the relationship between idle time and overtime costs in Remark 2.

REMARK 2. When the total cost per unit of idle time plus per unit of overtime $c^I + c^o$ is provided, the ratio of costs between idle time and overtime c^I/c^o does not influence the optimal policy.

3.2 Optimal Properties

We introduce two properties of optimal appointment policies that outline the circumstances in which a specific partial policy does not result in an optimal policy.

The first property focuses on the state costs of two partial policies derived from a common partial policy. We refer to it as the *property on state cost*.

THEOREM 1 (Property on State Cost). *Given a partial policy $\boldsymbol{\pi}^{i-1}$, consider two partial policies $\boldsymbol{\pi}^i = [\boldsymbol{\pi}^{i-1}; a]$ and $\boldsymbol{\tau}^i = [\boldsymbol{\pi}^{i-1}; b]$. If $g_i(\boldsymbol{\tau}^i) > g_i(\boldsymbol{\pi}^i)$, then any policy derived from $\boldsymbol{\tau}^{i+1} = [\boldsymbol{\tau}^i; a]$ is not optimal.*

Theorem 1 indicates that when scheduling the i-th customer based on the partial policy $\boldsymbol{\pi}^{i-1}$, if we choose the one with a large state cost $g_i(\boldsymbol{\tau}^i)$, then there must exist a partial policy $\boldsymbol{\tau}^{i+1}$, from which any derived policy cannot be optimal.

Second, we study partial policies from the aspect of the marginal state cost. For this purpose, we introduce the concept of the expected marginal cost brought by the i-th customer, denoted as $\Delta g_i(\boldsymbol{\pi}^i) = g_i(\boldsymbol{\pi}^i) - g_{i-1}(\boldsymbol{\pi}^{i-1})$, where $\boldsymbol{\pi}^i$ represents a partial policy derived from $\boldsymbol{\pi}^{i-1}$. The following theorem can be referred to as the *property on marginal cost*.

THEOREM 2 (Property on Marginal State Cost). *Given a partial policy $\boldsymbol{\pi}^{i-1}$ and a partial policy $\boldsymbol{\pi}^i$ derived from $\boldsymbol{\pi}^{i-1}$, if $\Delta g_{i-1}(\boldsymbol{\pi}^{i-1}) > \Delta g_i(\boldsymbol{\pi}^i)$, then any policy derived from $\boldsymbol{\pi}^i$ cannot be optimal.*

Theorem 2 shows that the new state cost should not be relatively too low; otherwise, the corresponding new partial policy cannot be optimal. It establishes a minimum threshold for the marginal cost when scheduling an additional customer. Moreover, Theorem 2 implies that, in an optimal appointment policy, the marginal cost should increase with i. This phenomenon is referred to as the *increasing marginal state cost*, as formally shown in Corollary 1.

COROLLARY 1 (Increasing Marginal State Cost). *There exists an optimal appointment policy $\boldsymbol{\pi}$, in which $\Delta g_i(\boldsymbol{\pi}^i) \leq \Delta g_{i+1}(\boldsymbol{\pi}^{i+1})$, $\forall\, i = 2, \ldots, n-1$.*

In Section 5, based on Theorems 1 and 2, and Corollary 1, we propose a set of elimination rules and establish a lower bound to speed up the search process within our B&B algorithm.

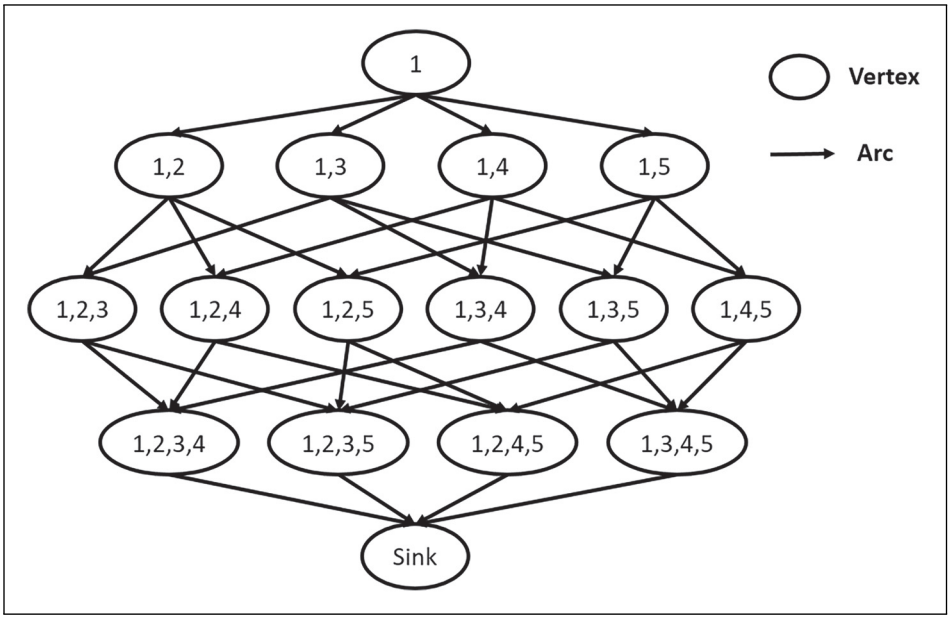

Figure 3. Shortest path model of the scheduling problem with $N = 5$, $n^+ = 4$, and $n^- = 3$ for Example 4.

3.3 Shortest Path Model

Our appointment scheduling problem can also be modeled as a shortest path problem on a directed network with $n^+ + 1$ levels. Within this framework, level 1 comprises a single source vertex representing the state $\boldsymbol{\pi}^{(1)} = [1]$. In level $i, i = 2, \ldots, n^+$, there exist C_{N-1}^{i-1} vertices, each representing a distinct state $\boldsymbol{\pi}^{(i)}$. For conciseness, we denote a vertex with state $\boldsymbol{\pi}^{(i)}$ as a *state* $\boldsymbol{\pi}^{(i)}$ *vertex*. Between consecutive levels $i-1$ and i, a directed arc is present from a state $\boldsymbol{\pi}^{(i-1)}$ vertex to a state $\boldsymbol{\pi}^{(i)}$ vertex if, and only if, the latter can be obtained by adding an unassigned time slot to the former. The cost associated with such an arc is denoted as $p_i g_i(\boldsymbol{\pi}^{(i)})$. In level $n^+ + 1$, there is a sink vertex. An arc with a cost of 0 connects each vertex in level n^+ to the sink vertex.

To represent a partial policy $\boldsymbol{\pi}^i$, we consider the subpath $\boldsymbol{\pi}^{(1)} \to \boldsymbol{\pi}^{(2)} \to \cdots \to \boldsymbol{\pi}^{(i)}$, where each $\boldsymbol{\pi}^{(j)}$ corresponds to the state of partial policy $\boldsymbol{\pi}^j, j = 1, \ldots, i$. Due to the network construction, the cost of the subpath is $G_i(\boldsymbol{\pi}^i)$, which also serves as the partial cost of policy $\boldsymbol{\pi}^i$. Hence, we establish a one-to-one correspondence between a partial policy and a subpath. For clarity, we denote the subpath as $\boldsymbol{\pi}^i$, bearing a path cost of $G_i(\boldsymbol{\pi}^i)$. In such a way, our appointment scheduling problem is equivalent to finding the shortest path from the source vertex to the virtual sink vertex.

EXAMPLE 4. A shortest path network with $N = 5$ and $n^+ = 4$ is demonstrated in Figure 3. For example, a partial policy $\boldsymbol{\pi}^3 = [1, 5, 2]$ can be represented by a subpath $[1] \to [1, 5] \to [1, 2, 5]$.

In problems with a general scale, the constructed network exhibits an exponential size, where the number of vertices is in $O(2^{N-1})$ and the number of arcs is in $O(N2^{N-2})$. As such, directly constructing this network poses a considerable challenge, let alone solving the shortest path problem to obtain an optimal appointment policy. The value of the shortest path model lies in offering a fresh perspective for understanding an optimal appointment policy. For example, Theorem 1, in the context of the shortest path network, can be perceived as a subpath from a vertex in level $i-1$ to a vertex in level $i+1$. When viewed through this lens, we can extend Theorem 1 to encompass a path across multiple levels. This leads to the following Theorem 3, which is a clear result from the perspective of the shortest path model.

THEOREM 3 (Property on Path Cost). *Given two subpaths $\boldsymbol{\pi}^i$ and $\hat{\boldsymbol{\pi}}^i$ starting from the source vertex and ending at a state $\boldsymbol{\pi}^{(i)}$ vertex, if $G_i(\boldsymbol{\pi}^i) < G_i(\hat{\boldsymbol{\pi}}^i)$, then subpath $\hat{\boldsymbol{\pi}}^i$ is not on an optimal path, hence any policy derived from $\hat{\boldsymbol{\pi}}^i$ is not an optimal policy.*

Note that Theorem 1 is a special case of Theorem 3. Both theorems serve as valuable tools for evaluating the optimality of partial policies. Theorem 1 specifically focuses on the comparison between two partial policies derived from a shared partial policy. Theorem 3 is applicable to a larger set of partial policies. It is worth mentioning that Theorem 3 requires additional computational effort, as it entails the calculation of the path cost.

Besides the new optimal condition, this shortest path model will contribute further to our B&B algorithm, which is discussed in Section EC.1 of the Supplemental Material.

3.4 Last-Minute Cancellations, Late Arrivals, and No-Shows

Up to now, we assume that all customers will arrive at the system punctually. In this section, we extend the scope of our model by incorporating all three random features into our online appointment scheduling problem.

Prior to the scheduled appointment, there is a probability that a customer may cancel their appointment and notify the system, denoted as ϕ_c. For customers who do not inform the system, there are two possible scenarios: they might either arrive late or not show up at all. The system cannot confirm the no-show behavior of a customer until η minutes after the scheduled appointment time. Besides, if a customer arrives late, the duration of the lateness is represented by a random variable $\boldsymbol{r}$ with an upper limit of η. In particular, if $\boldsymbol{r} = 0$, it represents that the customer arrives punctually. The probability of a customer not showing up is denoted as ϕ_n.

Subsequently, we denote the status of all potential customers by $X_1, \ldots, X_{n^+}$, which are independent and identically distributed random variables. When there is no ambiguity, we directly use X to represent the status of a customer. The distribution of X is as follows:

$$X = \begin{cases} 0, & \text{the customer cancels, w.p.} \quad \phi_c \\ 1, & \text{the customer arrives with} \\ & \text{possible lateness, w.p.} \quad 1 - \phi_c - \phi_n \\ 2, & \text{the customer does not show up, w.p.} \quad \phi_n \end{cases}$$

The evaluation of a state $\boldsymbol{\pi}_i^{(j)}$ needs to be modified with respect to the relationship between $\boldsymbol{z}_i^{(j)}$ and $\boldsymbol{e}_i^{(j)}$, and with regard to the new formulation of $\boldsymbol{w}_i^{(j)}$.

Redefine $\boldsymbol{z}_i^{(j)}$ as the time point that the system is available for the customer assigned to the slot $\pi_i^{(j)}$. The revised relationship between $\boldsymbol{z}_i^{(j)}$ and $\boldsymbol{e}_i^{(j)}$ is shown as follows:

$$\begin{aligned} \boldsymbol{e}_i^{(j)} &= (X-1)\left(\tfrac{X}{2}-1\right)\boldsymbol{z}_i^{(j)} + X(2-X)(\max\{a_i^{(j)} + \boldsymbol{r}, \boldsymbol{z}_i^{(j)}\} + \boldsymbol{s}) \\ &\quad + \tfrac{X(X-1)}{2}\max\{a_i^{(j)} + \eta, \boldsymbol{z}_i^{(j)}\}, \\ \boldsymbol{z}_{i+1}^{(j)} &= \max\{\boldsymbol{e}_i^{(j)}, a_{i+1}^{(j)}\}. \end{aligned}$$

The waiting time of the customer assigned to slot $\pi_i^{(j)}$ should be modified as follows:

$$\begin{aligned} \boldsymbol{w}_i^{(j)} &= (X-1)\left(\tfrac{X}{2}-1\right)\times 0 + X(2-X)\max\{\boldsymbol{z}_i^{(j)} - \boldsymbol{r} - a_i^{(j)}, 0\} \\ &\quad + \tfrac{X(X-1)}{2}\times 0. \end{aligned}$$

Within the expressions of $\boldsymbol{e}_i^{(j)}$ and $\boldsymbol{w}_i^{(j)}$, the first term represents the scenario in which the customer cancels the appointment, the second term pertains to the situation in which the customer arrives late, and the third term corresponds to the case where the customer does not show up.

Fundamentally, due to the assumption that customers are homogeneous, albeit with additional diversity in their behaviors, all theorems remain valid, as confirmed by the following corollary.

COROLLARY 2. *With the consideration of last-minute cancellations, late arrivals, and no-shows, Remark* 1 *and Theorems* 1–3 *hold.*

4 FCFS Scheduling

In this section, we first develop a dynamic programming (DP) algorithm for dealing with online FCFS appointment scheduling. Subsequently, we simplify the DP algorithm to address offline problems. These two cases can provide some basic bounds for solving online nonsequential scheduling problems. In this section, we employ $\boldsymbol{\pi}$ to represent the FCFS policy while adding the supplementary constraint $\pi_1 < \pi_2 < \cdots < \pi_{n^+}$.

4.1 DP for Online FCFS Scheduling

Under the FCFS rule, the expected waiting time of the i-th customer solely relies on the schedule of the first $i-1$ customers. Therefore, we can assess the expected waiting time of the i-th customer by the partial FCFS policy $\boldsymbol{\pi}^j$, that is, $\mathbf{E}[\boldsymbol{w}_i|\boldsymbol{\pi}^j]$, without resorting to the state $\boldsymbol{\pi}^{(i)}$, that is, $\mathbf{E}[\boldsymbol{w}_i|\boldsymbol{\pi}^{(j)}]$. In addition, since $\pi_1 < \cdots < \pi_{n^+}$ in the FCFS policy $\boldsymbol{\pi}$, we have $\mathbf{E}[\boldsymbol{w}_i|\boldsymbol{\pi}^j] = \mathbf{E}[\boldsymbol{w}_i|\boldsymbol{\pi}^i]$, $\forall j \geq i$.

Then, we can rewrite the objective function (6) as an expression of the expected waiting cost contributed by each customer, as follows:

$$\begin{aligned} G(\boldsymbol{\pi}) &= \sum_{j=1}^{n^+} p_j \left(\sum_{i=1}^{j} c^w \mathbf{E}[\boldsymbol{w}_i^{(j)}|\boldsymbol{\pi}^{(j)}] + c^\ell \mathbf{E}[\boldsymbol{w}_0^{(j)}|\boldsymbol{\pi}^{(j)}] \right) \\ &= \sum_{j=1}^{n^+} \left[\left(1 - \sum_{i=1}^{j-1} p_i\right) c^w \mathbf{E}[\boldsymbol{w}_j|\boldsymbol{\pi}^j] + p_j c^\ell \mathbf{E}[\boldsymbol{w}_0|\boldsymbol{\pi}^j] \right]. \end{aligned} \tag{7}$$

The formulation (7) can be further expressed as a recursive equation as shown below, in which $F'_j(\pi_1, \ldots, \pi_j)$ represents the summation of expected waiting costs contributed by the first j customers plus the summation of expected overtime costs for the first j realizations of $\boldsymbol{n}$.

$$\begin{aligned} F'_j(\pi_1, \ldots, \pi_j) &= F'_{j-1}(\pi_1, \ldots, \pi_{j-1}) + \left(1 - \sum_{i=1}^{j-1} p_i\right) c^w \mathbf{E}[\boldsymbol{w}_j|\boldsymbol{\pi}^j] \\ &\quad + p_j c^\ell \mathbf{E}[\boldsymbol{w}_0|\boldsymbol{\pi}^j]. \end{aligned} \tag{8}$$

Since the waiting time of the j-th customer is affected by the appointment times of all previous $j-1$ customers, we need to enumerate all possible $C_N^{n^+}$ policies to minimize $F'_{n^+}(\pi_1, \ldots, \pi_{n^+})$. However, this enumeration process can be

time-consuming, especially for large values of N. To address this, we introduce a lower bound on $\mathbf{E}[w_j|\boldsymbol{\pi}^j]$ by constructing a partial policy $\boldsymbol{\nu}^j$, which will be illustrated in the following lemma.

DEFINITION 4. In the FCFS scheduling problem, given a partial FCFS policy $\boldsymbol{\pi}^j$ and a particular $\kappa \in \{1, \dots, j-2\}$, we define a partial policy $\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa)$, in which $\nu_i = i$ for $i = 1, \dots, j-\kappa-1$ and $\nu_i = \pi_i$ for $i = j-\kappa, \dots, j$. We can also refer to $\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa)$ as $\boldsymbol{\nu}^j(\pi_{j-\kappa}, \dots, \pi_j)$.

EXAMPLE 5. Consider a partial FCFS policy $\boldsymbol{\pi}^6 = [1, 3, 6, 9, 13, 16]$. If $\kappa = 2$, we have $\boldsymbol{\nu}^6(\boldsymbol{\pi}^6, 2) = [1, 2, 3, 9, 13, 16]$. If $\kappa = 3$, we have $\boldsymbol{\nu}^6(\boldsymbol{\pi}^6, 3) = [1, 2, 6, 9, 13, 16]$.

The value κ represents the difference between $\boldsymbol{\pi}^j$ and $\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa)$. More specifically, the larger the value of κ, the smaller the difference between $\boldsymbol{\pi}^j$ and $\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa)$.

LEMMA 1. *In the FCFS scheduling problem, given a partial policy $\boldsymbol{\pi}^j$ and a particular $\kappa \in \{0, \dots, j-2\}$, we have $\mathbf{E}[w_j|\boldsymbol{\pi}^j] \geq \mathbf{E}[w_j|\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa)]$.*

Lemma 1 can be enhanced as the following corollary, which implies that the larger the value of κ, the tighter the constraint becomes.

COROLLARY 3. *In FCFS scheduling, given a partial policy $\boldsymbol{\pi}^j = [\pi_1, \dots, \pi_j]$ and a particular $\kappa \in \{2, \dots, j-1\}$, we construct two partial policies $\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa)$ and $\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa+1)$. Then we have*

$$\mathbf{E}[w_j|\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa+1)] \geq \mathbf{E}[w_j|\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa)].$$

The purpose of constructing the policy $\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa)$ serves not only to provide a lower bound on a customer's waiting time, but also to develop a DP algorithm. Based on the special structure of $\boldsymbol{\nu}^j(\boldsymbol{\pi}^j, \kappa)$, after replacing $\mathbf{E}[w_j|\boldsymbol{\pi}^j]$ and $\mathbf{E}[w_0|\boldsymbol{\pi}^j]$ in formulation (8) as $\mathbf{E}[w_j|\boldsymbol{\nu}^j(\pi_{j-\kappa}, \dots, \pi_j)]$ and $\mathbf{E}[w_0|\boldsymbol{\nu}^j(\pi_{j-\kappa}, \dots, \pi_j)]$, we can write the DP recursions as follows:

$$\begin{aligned}
&F_j(\pi_{j-\kappa+1}, \dots, \pi_j) \\
&\quad = \min \Bigg\{ F_{j-1}(\pi_{j-\kappa}, \dots, \pi_{j-1}) \\
&\qquad + \left(1 - \sum_{i=1}^{j-1} p_i\right) c^w \mathbf{E}[w_j|\boldsymbol{\nu}^j(\pi_{j-\kappa}, \dots, \pi_j)] \\
&\qquad + p_j c^\ell \mathbf{E}[w_0|\boldsymbol{\nu}^j(\pi_{j-\kappa}, \dots, \pi_j)] \Big| \pi_{j-\kappa} : \pi_{j-\kappa} \\
&\qquad \in \{j-\kappa, \dots, \pi_{j-\kappa+1} - 1\} \Bigg\}.
\end{aligned}$$

Lemma 2 shows that $F_j(\pi_{j-\kappa+1}, \dots, \pi_j)$ serves as a lower bound for the summation of expected waiting costs contributed by the first j call-in customers, plus the summation of expected overtime costs associated with the first j possible realizations of $\boldsymbol{n}$. This lower bound is based on the schedule from the $j-k+1$-st customer to the j-th customer, that is, $(\pi_{j-\kappa+1}, \dots, \pi_j)$.

LEMMA 2. *Given the schedule from the $j-k+1$-st customer to the j-th customer $(\pi_{j-\kappa+1}, \dots, \pi_j)$, for any $(\pi_1, \dots, \pi_{j-\kappa})$ satisfying $1 \leq \pi_1 \leq \cdots \leq \pi_{j-\kappa} \leq \pi_{j-\kappa+1}$, we have $F_j(\pi_{j-\kappa+1}, \dots, \pi_j) \leq F'_j(\pi_1, \dots, \pi_{j-\kappa}, \pi_{j-\kappa+1}, \dots, \pi_j)$.*

The boundary condition of the DP is

$$F_\kappa(1, \dots, \pi_\kappa) = \sum_{i=1}^{\kappa} \mathbf{E}[w_i|\boldsymbol{\pi}^i].$$

In addition, the optimal value of the DP is

$$\min \left\{ F_{n^+}(\pi_{n^+-\kappa+1}, \dots, \pi_{n^+}) \middle| \nabla_{n^+-\kappa+1} \right\},$$

where $\nabla_{n^+-\kappa+1} = \{(\pi_{n^+-\kappa+1}, \dots, \pi_{n^+}) : n^+ - \kappa + 1 \leq \pi_{n^+-\kappa+1} < \cdots < \pi_{n^+} \leq N\}$.

In the above DP formulation, there are $n^+ - \kappa$ stages. Each stage i requires $O((N-n^+)C^\kappa_{N-n^+})$ time to compute all the $F(\pi_{i-\kappa+1}, \dots, \pi_i)$ values. Therefore, the overall time complexity of the DP algorithm is $O((n^+-\kappa)(N-n^+)C^\kappa_{N-n^+})$.

The DP algorithm yields a cost denoted as $\bar{F}^{DP}(\kappa)$ and a feasible policy termed $\boldsymbol{\pi}^{DP}(\kappa)$ through backtracking. These can be referred to as the DP value and the DP policy, respectively. Next, we introduce Lemma 3.

LEMMA 3. *For a specific choice of κ, assuming the DP policy of an online FCFS problem is $\boldsymbol{\pi}^{DP}(\kappa)$, then we have $G(\boldsymbol{\pi}^{DP}(\kappa)) \geq \bar{F}^{DP}(\kappa)$.*

Lemma 3 shows that the DP value is a lower bound for the actual expected cost of the DP policy.

In addition, we note that the optimal cost of the online FCFS problem can be constrained by the DP value and the actual cost of the DP policy, as demonstrated in Theorem 4. By leveraging these upper and lower bounds, we can assess the performance of the DP algorithm. Please refer to Appendix EC.2 of the Supplemental Material for a detailed evaluation.

THEOREM 4. *For a specific choice of κ, assuming the DP policy of an online FCFS problem is $\boldsymbol{\pi}^{DP}(\kappa)$ and the optimal policy of the online FCFS problem is $\boldsymbol{\pi}^*$, then we have $\bar{F}^{DP}(\kappa) \leq G(\boldsymbol{\pi}^*) \leq G(\boldsymbol{\pi}^{DP}(\kappa))$.*

Since the online FCFS problem is a special case of the online nonsequential problem, the actual cost of the DP policy in the online FCFS problem $G(\boldsymbol{\pi}^{DP}(\kappa))$ can also serve as an upper bound for the corresponding nonsequential scheduling problem.

4.2 Offline Problem

In an offline scheduling problem, the number of customers $\boldsymbol{n}$ is a constant. Due to the homogeneity of customers, we can treat it as a special FCFS problem, thereby solving it by a simplified version of the DP algorithm proposed in Section 4.1.

The recursion formulation of the simplified DP algorithm is as follows:

$$\begin{aligned} &F_j(\pi_{j-\kappa+1}, \dots, \pi_j) \\ &\quad = \min \Bigg\{ F_{j-1}(\pi_{j-\kappa}, \dots, \pi_{j-1}) \\ &\qquad + c^w \mathbf{E}[\boldsymbol{w}_j | \boldsymbol{v}^j(\pi_{j-\kappa}, \dots, \pi_j)] \Big| \pi_{j-\kappa} : \pi_{j-\kappa} \\ &\qquad \in \{j-\kappa, \dots, \pi_{j-\kappa+1} - 1\} \Bigg\}. \end{aligned}$$

The boundary condition of the DP is

$$F_\kappa(1, \dots, \pi_\kappa) = \sum_{i=1}^{\kappa} \mathbf{E}[\boldsymbol{w}_i | \boldsymbol{\pi}^i].$$

The optimal cost of the DP $\bar{g}_n(\kappa)$ can be obtained from the following equation:

$$\begin{aligned} \bar{g}_n(\kappa) = \min \Bigg\{ &F(\pi_{n-\kappa+1}, \dots, \pi_n) \\ &+ c^\ell \mathbf{E}[\boldsymbol{w}_0 | \boldsymbol{v}^n(\pi_{n-\kappa+1}, \dots, \pi_n)] \Big| \nabla_{n-\kappa+1} \Bigg\}, \end{aligned}$$

where $\nabla_{n-\kappa+1} = \{(\pi_{n-\kappa+1}, \dots, \pi_n) : n - \kappa + 1 \le \pi_{n-\kappa+1} < \cdots < \pi_n \le N\}$.

Here, the DP algorithm also provides a DP policy and a DP value. With Lemma 4, we demonstrate that the DP value, denoted as $\bar{g}_n(\kappa)$, serves as a lower bound for the minimum cost of the offline problem with exactly n customers. This can be shown as follows:

LEMMA 4. *For a specific choice of κ, assuming the DP value of an offline problem with n customers is $\bar{g}_n(\kappa)$ and the optimal policy of the offline problem is $\boldsymbol{\pi}^*$, then we have $\bar{g}_n(\kappa) \le G(\boldsymbol{\pi}^*)$.*

Such a bound can be applied to our B&B algorithm, which is discussed in detail in the next section.

5 B&B Algorithm

In this section, we begin by presenting the structure of the B&B tree. Then, in Sections 5.2 and 5.3, we propose a set of elimination rules and a lower bound. Furthermore, we provide a pseudo-code algorithm and an illustrative example to demonstrate the workings of the B&B tree. Lastly, in Section 5.5, we propose a greedy algorithm to address the challenges posed by larger problem instances, which may be difficult to solve using the B&B algorithm. In addition, in Appendix EC.1 of the Supplemental Material, we outline other contributions derived from the shortest path model.

5.1 Algorithm Description

The B&B algorithm searches for an optimal policy over a search tree. Each node in the tree corresponds to a partial policy $\boldsymbol{\pi}^j$, which is used to evaluate all policies derived from $\boldsymbol{\pi}^j$. The root node specifies the arrangement for the initial customer, that is, $\boldsymbol{\pi}^1 = [1]$, and leaf nodes correspond to full policies $\boldsymbol{\pi}$. A node $\boldsymbol{\pi}^j, j < n^+$, has a set of child nodes $\boldsymbol{\pi}^{j+1}$'s which are siblings to one another. Correspondingly, node $\boldsymbol{\pi}^j$ is the parent node to these $\boldsymbol{\pi}^{j+1}$ nodes. The state of each node corresponds to the state of its corresponding partial policy. For each node $\boldsymbol{\pi}^j$, we compute the state cost $g_j(\boldsymbol{\pi}^j)$ and the partial cost $G_j(\boldsymbol{\pi}^j)$.

The B&B algorithm dynamically constructs the tree in the search process, following a *best-and-depth first* strategy. Once a node is generated, it undergoes a check to determine whether it violates any elimination rules. If violations are found, the node is promptly eliminated without exploring its child nodes. In addition, at each node, an evaluation is performed to estimate the lower bound cost of all leaf nodes derived from that specific node. If the lower bound exceeds the current upper bound, the node is immediately fathomed. The initial upper bound of the search tree is set as the actual cost of the DP policy developed in Section 4.1. During the search process, if a leaf node has a cost lower than the current upper bound, the upper bound is updated accordingly.

5.2 Elimination Rules

In order to enhance the efficiency of the search process, we develop a set of rules to eliminate nodes dominated by other nodes so that the remaining nodes can still lead to an optimal policy.

In light of the optimal conditions outlined in Theorem 1, we examine two sibling nodes $\boldsymbol{\pi}^i$ and $\boldsymbol{\tau}^i$, which share the same parent node $\boldsymbol{\pi}^{i-1}$. There must exist two nodes $\boldsymbol{\pi}^{i+1}$ and $\boldsymbol{\tau}^{i+1}$ that are child nodes of $\boldsymbol{\pi}^i$ and $\boldsymbol{\tau}^i$, respectively, and have the same state, that is, $\boldsymbol{\pi}^{(i+1)} = \boldsymbol{\tau}^{(i+1)}$. Without loss of generality, we assume that $g_i(\boldsymbol{\pi}^i) < g_i(\boldsymbol{\tau}^i)$. It can then be proven, as demonstrated in Theorem 1, that none of the leaf nodes $\boldsymbol{\tau}$ derived from $\boldsymbol{\tau}^{i+1}$ can be optimal. Therefore, we present the following elimination rule.

RULE 1. *For any two sibling nodes $\boldsymbol{\pi}^i$ and $\boldsymbol{\tau}^i$, if their respective child nodes $\boldsymbol{\pi}^{i+1}$ and $\boldsymbol{\tau}^{i+1}$ satisfy $\boldsymbol{\pi}^{(i+1)} = \boldsymbol{\tau}^{(i+1)}$, and $g_i(\boldsymbol{\pi}^i) < g_i(\boldsymbol{\tau}^i)$, then the node $\boldsymbol{\tau}^{i+1}$ can be eliminated.*

To employ Rule 1, we sort all the child nodes of a parent node in ascending order according to their state costs. This enables us to eliminate, on average, half of the child nodes, which corresponds to the *best-first* strategy.

In Theorem 2 and Corollary 1, we prove that within an optimal policy $\boldsymbol{\pi}$, it satisfies the condition $\Delta g_i(\boldsymbol{\pi}^i) \leq \Delta g_{i+1}(\boldsymbol{\pi}^{i+1})$ for $i = 2, \ldots, n-1$. Based on this insight, we can eliminate nodes that violate the increasing marginal cost property, as specified in the following rule:

RULE 2. *If the marginal cost of a node $\Delta g_i(\boldsymbol{\pi}^i)$ is less than that of its parent node $\Delta g_{i-1}(\boldsymbol{\pi}^{i-1})$, then node $\boldsymbol{\pi}^i$ can be eliminated.*

In Theorem 3, we show that when considering two subpaths $\boldsymbol{\pi}^i$ and $\boldsymbol{\tau}^i$ leading to the same state $\boldsymbol{\pi}^{(i)}$ vertex, if $G_i(\boldsymbol{\pi}^i) < G_i(\boldsymbol{\tau}^i)$, then any policy derived from $\boldsymbol{\tau}^i$ cannot be optimal. Consequently, for every state $\boldsymbol{\pi}^{(i)}$, we express the cost of the current best subpath leading to that state as $U(\boldsymbol{\pi}^{(i)})$. With this in mind, we propose the subsequent elimination rule.

RULE 3. *For a node $\boldsymbol{\pi}^k$, if $G_k(\boldsymbol{\pi}^k) > U(\boldsymbol{\pi}^{(k)})$, then node $\boldsymbol{\pi}^k$ can be eliminated.*

During the search process, when considering a specific node $\boldsymbol{\pi}^k$, if the cost function $G_k(\boldsymbol{\pi}^k)$ is less than or equal to the current cost $U(\boldsymbol{\pi}^{(k)})$ of the best subpath leading to state $\boldsymbol{\pi}^{(k)}$, we proceed with the search of that node and update $U(\boldsymbol{\pi}^{(k)})$ to $G_k(\boldsymbol{\pi}^k)$. Conversely, if $G_k(\boldsymbol{\pi}^k)$ exceeds $U(\boldsymbol{\pi}^{(k)})$, node $\boldsymbol{\pi}^k$ is eliminated.

5.3 Lower Bound

In addition to the aforementioned node elimination rules, we propose a lower bound on the cost of any full policy derived from node $\boldsymbol{\pi}^k$. This lower bound can be established through two distinct approaches.

First, as per Corollary 1, the condition $g_j(\boldsymbol{\pi}^j) \geq g_{j-1}(\boldsymbol{\pi}^{j-1}) + \Delta g_{j-1}(\boldsymbol{\pi}^{j-1})$ must be satisfied by all nodes that may lead to an optimal policy. Thus, when considering node $\boldsymbol{\pi}^k$, we can take $g_k(\boldsymbol{\pi}^k) + (i-k)\Delta g_k(\boldsymbol{\pi}^k)$ as a lower bound for $g_i(\boldsymbol{\pi}^i)$, if $\boldsymbol{\pi}^i$ is derived from $\boldsymbol{\pi}^k$, for $i = k+1, \ldots, n^+$.

Second, we can derive a lower bound for $g_i(\boldsymbol{\pi}^i)$ from the offline problem with exactly i customers. Recall that the DP value $\bar{g}_i$ mentioned in Section 4.2 can serve as a lower bound for the case involving exactly i customers.

Hence, we can formulate the lower bound of $g_i(\boldsymbol{\pi}^i)$ as $\max\{\bar{g}_i, g_k(\boldsymbol{\pi}^k) + (i-k)\Delta g_k(\boldsymbol{\pi}^k)\}$. Let us denote this lower bound as $L(\boldsymbol{\pi}^k)$, which is as follows:

$$L(\boldsymbol{\pi}^k) = \sum_{i=1}^{k} p_i g_i(\boldsymbol{\pi}^i) + \sum_{i=k+1}^{n^+} p_i \max\{\bar{g}_i, g_k(\boldsymbol{\pi}^k) + (i-k)\Delta g_k(\boldsymbol{\pi}^k)\}.$$

The following theorem shows that $L(\boldsymbol{\pi}^k)$ can serve as the lower bound for the cost of any full policy derived from node $\boldsymbol{\pi}^k$.

THEOREM 5. *For any leaf node $\boldsymbol{\pi}$ derived from $\boldsymbol{\pi}^k$, we have $G(\boldsymbol{\pi}) \geq L(\boldsymbol{\pi}^k)$.*

During our search process, for any node $\boldsymbol{\pi}^k$, if $L(\boldsymbol{\pi}^k)$ exceeds the current upper bound of the B&B tree, it implies that the node $\boldsymbol{\pi}^k$ should be eliminated.

5.4 B&B Algorithm

By integrating the B&B tree structure outlined in Section 5.1, the rules from Section 3.2, and the lower bound proposed in Section 5.3, we demonstrate the entire B&B algorithm with the pseudo-code provided in Appendix EC.10 of the Supplemental Material. In addition, with reference to the shortest path model, we define *second-level nodes* to mitigate certain redundancies associated with the root node, as shown in Appendix EC.1.2.1 of the Supplemental Material.

Throughout the search process, each node undergoes an initial assessment based on Rule 1 and, should it violate this rule, it is promptly eliminated. Among the remaining nodes, including second-level nodes, any node that fails to comply with the increasing marginal cost property specified in Rule 2 is discarded. In addition, if the partial cost $G_i(\boldsymbol{\pi}^i)$ exceeds $U(\boldsymbol{\pi}^{(i)})$ or the lower bound $L(\boldsymbol{\pi}^i)$ exceeds the current upper bound of the B&B tree, the node is eliminated in accordance with Rule 3 and Theorem 5. An example demonstrating the nodes that are searched for and eliminated within a B&B tree is shown below.

EXAMPLE 6. Consider a server with $N = 5$ time slots, where each time slot has a duration of $t = 10$ minutes. The service time $\boldsymbol{s}$ is distributed according to a discrete uniform distribution $U[5, 15]$ and the number of customers follows a discrete uniform distribution $U[2, 5]$. The cost coefficients are $c^w = 1$ and $c^\ell = 3$. The search tree of the B&B algorithm is shown in Figure 4.

In Figure 4, each node is characterized by two descriptions, namely, the partial policy and the 0-1 vector form state. To illustrate, consider a second-level node denoted as $[1, 3]$. Its corresponding 0-1 vector form state is represented as $[1, 0, 1, 0, 0]$, indicating that the first and third time slots have been assigned in this particular node.

Within the search tree, the second-level nodes are $[1, 2]$, $[1, 3]$, $[1, 4]$, and $[1, 5]$, possessing corresponding state values of 1.36, 0, 0, and 4.09, respectively. Following the best-and-depth first strategy, these second-level nodes are sorted in ascending order based on their state values. Furthermore, the sibling relationship between these second-level nodes is maintained during the sorting process.

In Figure 4, nodes with double lines are eliminated according to Rule 1. For example, the second-level node $[1, 5]$ has no child node because each of its sibling nodes has a smaller state value. Similarly, the eliminated node $[1, 4, 3]$ is not displayed as it has been eliminated due to the smaller state value of node $[1, 3]$ compared to node $[1, 4]$. The node $[1, 2, 5, 3]$, indicated by a dashed line, is eliminated according to Rule 2. Moreover, the shadow nodes, indicated by dashed lines, are eliminated based on the upper bound $U(\boldsymbol{\pi}^{(k)})$ specified in Rule 3, while

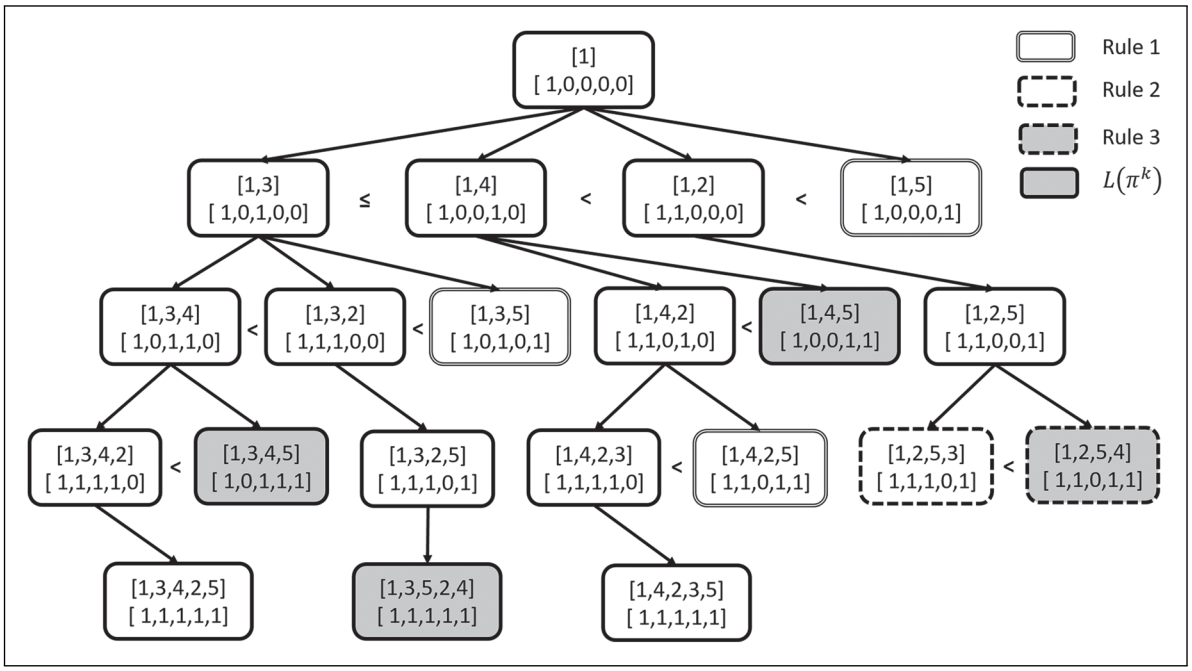

Figure 4. Search tree of the branch-and-bound (B&B) algorithm for Example 6.

the shadow nodes, indicated by the solid lines, are eliminated using the lower bound $L(\boldsymbol{\pi}^k)$. Nodes depicted with solid lines are retained and generate child nodes for further examination. Finally, the optimal policies are characterized by leaf nodes $[1, 3, 4, 2, 5]$ and $[1, 4, 2, 3, 5]$.

5.5 Greedy Algorithm

For large-sized instances, the B&B algorithm may be terminated once the specified time limit is reached. Consequently, we introduce a more time-efficient greedy algorithm based on the B&B algorithm.

Within the greedy algorithm, we selectively retain only the best child node from each explored node within the search tree. Since there are no sibling nodes, the application of Rule 1 is redundant. Nevertheless, we can still employ Rule 2, Rule 3, and the lower bound $L(\boldsymbol{\pi}^i)$ to speed up the search process. In addition, when there are a large number of second-level nodes, we only examine the initial few nodes and disregard the remaining nodes, as long as these nodes are arranged in a nondecreasing order based on their state costs. The pseudocode of the greedy algorithm can be found in Appendix EC.11 of the Supplemental Material.

Consider the instance of Example 6. We involve all the second-level nodes and show the search tree of the greedy algorithm in Figure 5. In Figure 5, the gridded nodes are eliminated via the greedy selection and only the node with the smallest state value among the sibling nodes is retained for further consideration. While Rule 1 does not apply, both Rule 2 and Rule 3 remain effective in the greedy search process. In Example 6, the greedy algorithm successfully identifies the two optimal policies. Furthermore, in Section 6, we evaluate the performance of the greedy algorithm when applied to large-scale instances.

6 Computational Studies

Our computational studies serve several purposes. First, we demonstrate the computational efficiency of our B&B algorithm in online nonsequential scheduling. Second, we assess the performance of our greedy algorithm when confronted with large-scale problems. Third, we study the benefit of nonsequential policies compared to FCFS policies. Fourth, we delve into the impact of last-minute cancellations, late arrivals, and no-shows. Fifth, we investigate the competitive ratio (CR) between implemented online nonsequential policies and offline policies. Sixth, we compare heuristics proposed by existing papers with the optimal nonsequential policies. All the algorithms mentioned above were coded in Python and tested on a desktop computer with an i7-7600 CPU.

The instances used in our computational studies are as follows. By default, the service horizon is designated as $T = 240$ min, and each time slot has a duration of $t = 10$ min. It corresponds to 24 slots of duration $t = 10$ in a 4-hour morning or afternoon session, or 24 slots of length $t = 20$ within an 8-hour day, as per the framework proposed by Chen and Robinson (2014). In accordance with the empirical study of Cayirli et al. (2006), we assume that the service time $\boldsymbol{s}$ follows a lognormal discrete distribution, with a mean of 20 and a standard deviation of 5. To ensure tractability, the minimum and maximum service times are truncated at $s^- = 8$ and $s^+ = 30$, respectively. Without loss of generality, we normalize the cost

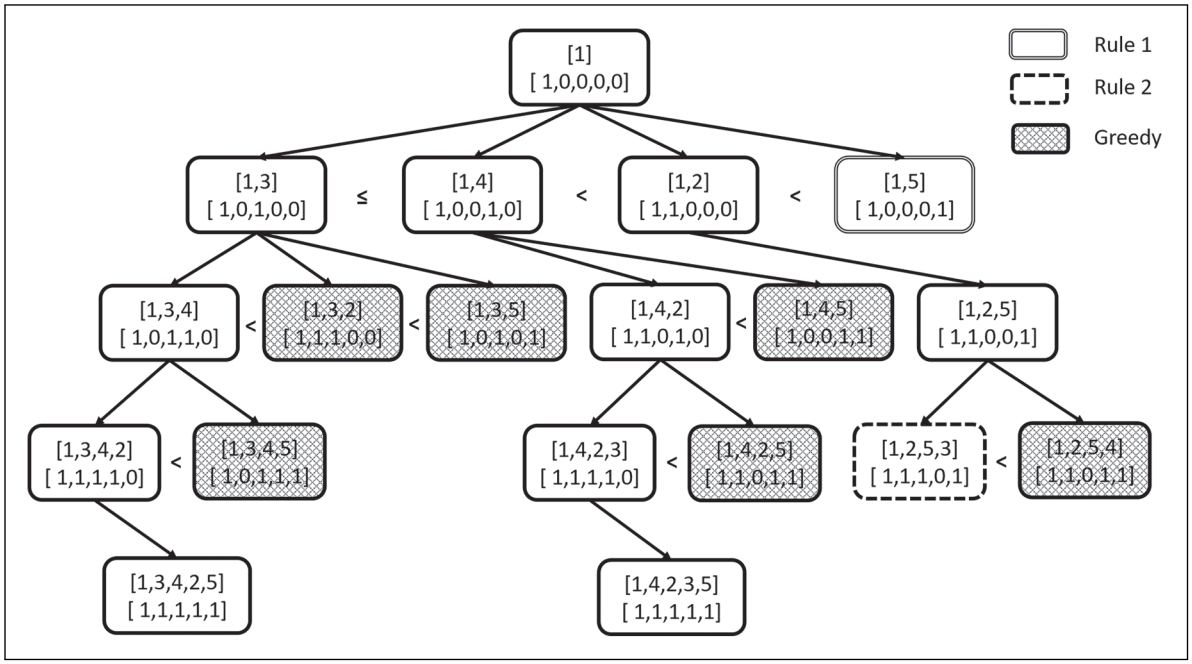

Figure 5. Search tree of the greedy algorithm for Example 6.

coefficient associated with waiting time c^w to 1. Moreover, we employ a cost parameter $c^\ell = c^I + c^o = 3$, which exceeds the aforementioned waiting time cost coefficient c^w. Additionally, the number of customers, n, follows a binomial distribution, specifically, $n \sim n^- + binomial(n_b, P)$. Hereinafter, n^- is set to the default value of 7, while n_b assumes values from the default set $n_b \in \{3, 4, 5, 6, 7\}$ and P takes on values from the default set $P \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$. For convenience, we utilize the notation $[n^-, n^+]$ to denote the range of $\boldsymbol{n}$ and the notation $([n^-, n^+], P)$ to denote a specific instance, where $n^+ = n^- + n_b$.

In our numerical studies, the mean of the number of customers, denoted as $\bar{n}$, and the server utilization are formulated as follows:

$$\bar{n} = n^- + n_b P, \text{ Utilization} = \frac{\bar{s} \times \bar{n}}{T} \times 100\%.$$

6.1 Computational Efficiency of the B&B Algorithm

We investigate the computational efficiency of the B&B algorithm by demonstrating the computation time of the algorithm. According to the computation time presented in Table 1, we observe that our B&B algorithm efficiently solves instances with a service horizon of $T = 240$ min, which means it is well-suited for managing daily appointment operations in a service system, such as scheduling 24 slots, each with a duration of $t = 10$ min, during a 4-hour morning or afternoon session, or scheduling 24 slots, each with a duration of $t = 20$ min, within an 8-hour workday.

The computation time increases as the range of uncertainty in the number of customers $\boldsymbol{n}$, the value of P, and the working horizon T increase. Of these factors, the computation time is most affected by the working horizon T because the total number of possible policies grows exponentially with the number of time slots. Notably, some instances with a service horizon of $T = 360$ min cannot be solved by the B&B algorithm within 2 h. These instances, which we refer to as "large-sized instances," are marked by horizontal lines. In Section 6.2, we revisit some of these large instances using the greedy algorithm.

In addition, we examine the percentage of retained nodes at each level using three specific instances (see Appendix EC.3 of the Supplemental Material). This analysis serves to showcase the notable efficacy of our suggested elimination rules in expediting the search process.

6.2 Performance of the Greedy Algorithm

In Section 5.5, we introduce a greedy algorithm to solve large-sized instances. In this section, we evaluate the performance of the greedy algorithm by measuring the relative difference, referred to as the "relative gap," between the cost of the policy found by the greedy algorithm and the cost generated by the B&B algorithm, with respect to the latter's cost.

Specifically, we analyze nine large-sized instances with a service horizon of $T = 360$ min, $n_b \in \{9, 10, 11\}$, and $P \in \{0.3, 0.5, 0.7\}$, as outlined in Table 2. Among these instances, the B&B algorithm can solve five instances optimally, while for the remaining four instances (marked with an asterisk), the B&B algorithm stops due to the predetermined time limit. The columns labeled as $100, 500, 1,000$, and $1,500$ represent the number of second-level nodes involved in the search process for the greedy algorithm. The *gap* column presents the

Table 1. Running time (in seconds) of the Branch-and-Bound (B&B) algorithms.

		P						P						P					
T	n	0.3	0.4	0.5	0.6	0.7	T	n	0.3	0.4	0.5	0.6	0.7	T	n	0.3	0.4	0.5	0.6	0.7
240	[7,12]	20	32	37	54	65	300	[7, 14]	65	106	434	651	1,024	360	[7,16]	1,643	1,670	1,858	5,706	–
	[7, 13]	26	41	61	77	90		[7, 15]	103	227	690	1,111	1,364		[7, 17]	1,536	1,810	2,599	–	–
	[7, 14]	37	62	81	100	136		[7, 16]	111	582	1,081	1,357	1,900		[7, 18]	1,808	2,678	–	–	–

relative gap, which is recorded as 0 if both algorithms find the same policy; otherwise, it is expressed as a relative percentage. The *time* column displays the computation time required for the greedy algorithm.

For instances that can be solved optimally by the B&B algorithm, the gap is satisfactorily small even when applying only 100 second-level nodes in the greedy algorithm. Taking the instance $([7, 18], P = 0.3)$ as an illustration, the greedy algorithm successfully reaches the optimal policy in a mere 70 s, which is significantly faster than the $1,808$ s required by the B&B algorithm. In addition, as the number of second-level nodes involved increases, the relative gap approaches zero. This phenomenon is observed in large-sized instances, suggesting that the cost of the best policy obtained by the greedy algorithm is comparable to that obtained by the B&B algorithm. Furthermore, two instances in the table exhibit a negative relative gap: $([7, 17], P = 0.7)$ with 1,500 second-level nodes and the instance $([7, 18], P = 0.7)$ with 1,500 second-level nodes. This implies that the greedy algorithm can swiftly identify a superior policy compared to the B&B algorithm.

In summary, the greedy algorithm is an accurate and efficient heuristic for addressing large-sized online nonsequential appointment scheduling problems.

6.3 Benefit of Nonsequential Policy

We assess the benefit of the nonsequential scheduling policy in comparison to the sequential scheduling policy and quantify it using the following expression:

$$\frac{\text{FCFS} - \text{Nonsequential}}{\text{FCFS}} \times 100\%.$$

Here, "Nonsequential" represents the cost of the optimal nonsequential policy obtained through the B&B algorithm, and "FCFS" corresponds to the DP value of the corresponding FCFS problem. Since the DP value can serve as a lower bound for the minimum cost of the FCFS problem, this measurement provides a lower bound for the actual benefit attained from nonsequential scheduling.

Our initial inquiry explores the correlation between the benefit and two parameters pertaining to the distribution of $\boldsymbol{n}$, namely n_b and P. As depicted in Table 3, with a designated n_b value, the benefit decreases as P increases. Conversely, when P remains constant, the trend in the benefit is not obvious with the increasing n_b. These observations imply that although n_b and P may indeed hold relevance to the benefits of nonsequential scheduling, they cannot individually provide a perfect explanation for the benefits.

Next, we delve deeper into the association between the benefit and the combined impact of n_b and P, specifically, the variance of $\boldsymbol{n}$, denoted as $V = n_b P(1-P)$, as well as the mean, represented by $\bar{n} = n^- + n_b P$.

First, given a fixed variance V, we assess the impact of the mean on the benefit. In Table 3, we consider any pair

Table 2. Performance of the greedy algorithm.

		Number of second-level nodes involved							
		100		500		1000		1500	
n	P	Gap	Time	Gap	Time	Gap	Time	Gap	Time
[7, 16]	0.3	0.01%	44	0.01%	131	0.01%	211	0.00%	292
	0.5	0.07%	92	0.07%	264	0.07%	433	0	584
	0.7 *	3.42%	157	3.42%	531	3.42%	845	0	1,155
[7, 17]	0.3	0.00%	45	0.00%	134	0.00%	217	0.00%	296
	0.5	0.05%	115	0.05%	369	0.05%	591	0	790
	0.7 *	0	209	0	761	0	1,210	−4.73%	1,704
[7, 18]	0.3	0	70	0	203	0	333	0	449
	0.5 *	2.13%	166	2.13%	563	2.13%	901	0.18%	1,245
	0.7 *	0.41%	271	0.08%	1,026	0.08%	1,630	−9.28%	2,274

Table 3. Benefits of nonsequential scheduling compared to sequential scheduling.

	P								
n	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7
[7, 10]	48.0%	40.8%	29.8%	22.9%	21.7%	20.2%	18.4%	17.2%	16.9%
[7, 11]	32.2%	30.2%	27.2%	25.4%	24.8%	20.7%	13.1%	9.9%	7.9%
[7, 12]	33.5%	33.4%	26.0%	19.3%	17.8%	17.6%	16.6%	11.1%	6.2%
[7, 13]	36.6%	30.8%	30.2%	23.3%	17.2%	12.4%	8.7%	6.9%	2.1%
[7, 14]	42.5%	36.9%	32.1%	28.4%	21.0%	14.9%	11.2%	8.5%	6.5%

Table 4. Benefits of nonsequential scheduling under different variances.

	$\bar{n} = 8$		$\bar{n} = 9$		$\bar{n} = 10$		$\bar{n} = 11$		$\bar{n} = 12$	
n	V	Benefit	V	Benefit	V	Benefit	V	Benefit	V	Benefit
[7, 13]	0.83	54.3%	1.33	34.7%	1.50	23.5%	1.33	13.5%	0.83	3.6%
[7, 14]	0.86	66.3%	1.43	47.3%	1.71	35.0%	1.71	19.9%	1.43	12.6%
[7, 15]	0.88	76.3%	1.50	58.7%	1.88	46.0%	2.00	32.8%	1.88	25.5%
[7, 16]	0.89	81.9%	1.56	70.5%	2.00	57.2%	2.22	46.9%	2.22	38.5%
[7, 17]	0.90	88.3%	1.60	78.2%	2.10	68.5%	2.40	59.0%	2.50	49.7%

of instances, namely $([n^-, n^+], P)$ and $([n^-, n^+], 1 - P)$. The variances of the two instances are the same, that is, $V = (n^+ - n^-)P(1-P)$, whereas the means differ, that is, $(n^+ - n^-)P$ and $(n^+ - n^-)(1 - P)$, respectively. By comparison within the pairs, we find that the benefit diminishes as the mean increases. For example, when comparing the instances $([7, 10], 0.3)$ and $([7, 10], 0.7)$, which possess the same variance, we find that the former has a smaller mean and a greater benefit than the latter. Hence, we infer that when the variance is fixed, a smaller mean or, alternatively, a lower utilization rate corresponds to a greater benefit.

Second, we explore the impact of variance on the benefit while considering a fixed mean value $\bar{n}$. As shown in the columns of Table 4, we observe that the benefit increases as the variance V increases, given the mean $\bar{n}$. Remarkably, even when the mean is large, such as $\bar{n} = 12$, the benefit of the nonsequential scheduling remains evident if the variance V is large. Therefore, we infer that a system with a highly fluctuating number of customers can benefit more from nonsequential scheduling compared to a system with a small variance in the number of customers.

In summary, nonsequential scheduling holds a significant advantage over sequential scheduling when the system is not congested or when there is a considerable fluctuation in the number of customers.

6.4 Impact of Cancellations, Lateness, and No-Shows

In Section 3.4, we extended our model to incorporate last-minute cancellations, late arrivals, and no-shows in the context of online intraday appointment scheduling problems. In this section, we further investigate the impact of different sources of uncertainty on online scheduling.

Table 5 presents the system costs in nine scenarios, including cancellations only, no-shows only, lateness only, all possible combinations of pairs, and a full complement of all three factors. The minimum and the maximum number of customers are denoted as $n^- = 7$ and $n^+ = 14$, respectively. First and

Table 5. The total cost of the system when considering cancellations, late arrivals, and no-shows.

Case	$\boldsymbol{n} \sim [7, 14]$		P								
			0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7
1	All customers show up on time		8.4	11.4	15.5	21.0	28.2	36.3	45.9	57.8	72.3
2	Only cancel	$\phi_c = 0.2$	4.9	6.4	8.4	11.0	14.1	17.6	22.1	27.8	34.9
3	Only late	$r \sim U[0, 15]$	16.2	21.5	28.1	35.1	43.7	54.0	65.2	77.4	91.5
4	Only no-show	$\eta = 10, \phi_n = 0.2$	4.9	6.4	8.4	11.0	14.1	17.6	22.1	27.8	34.9
5	Only no-show	$\eta = 15, \phi_n = 0.2$	5.0	6.6	8.6	11.3	14.4	18.1	22.9	28.9	36.6
6	Cancel, late	$\phi_c = 0.2, r \sim U[0, 15]$	9.6	12.6	16.1	19.8	24.0	28.7	33.9	39.5	46.2
7	Cancel, no-show	$\phi_c = 0.2, \eta = 15, \phi_n = 0.2$	2.5	3.2	4.2	5.3	6.5	8.0	10.0	12.5	15.7
8	Late, no-show	$r \sim U[0, 15], \phi_n = 0.2$	9.6	12.6	16.2	19.8	24.1	28.8	33.9	39.6	46.3
9	Cancel, late, no-show	$\phi_c = 0.2, r \sim U[0, 15], \phi_n = 0.2$	4.9	6.3	7.9	9.6	11.4	13.6	16.0	18.6	21.5

foremost, we observe that lateness yields a significant increase in system costs, as evidenced by the contrasts between Cases 1 and 3, Cases 2 and 6, Cases 5 and 8, and Cases 7 and 9. Furthermore, when we compare Cases 2 and 4, cancellations and no-shows can reduce system costs to some extent on that specific day. This can be attributed to the fact that cancellations and no-shows create available capacity in the schedule, thereby decreasing the waiting time for other customers and minimizing the overtime for the system.

In Case 4, where η is equal to the length of a time slot, the system cost remains the same as in Case 2. Moving on to Case 5, where η exceeds the length of a time slot, the additional impact incurred is still relatively insignificant compared to Case 4. When we explore scenarios featuring potential lateness, such as Cases 6 and 8, the distinction between the impact of no-shows and cancellations becomes negligible, despite η still surpassing the length of a time slot. From this observation, we deduce that while a no-show may result in a waste of up to η minutes compared to a cancellation, the potential delay caused by the previous customer can mitigate this difference. Therefore, despite the conceptual difference in the definitions of last-minute cancellations and no-shows, it appears reasonable to model them as a common source of uncertainty. This finding aligns with the assumption made by Xie et al. (2021) that last-minute cancellations and no-shows are treated as interchangeable occurrences referred to as "missed appointments."

In summary, lateness yields a significant increase in system costs, whereas last-minute cancellations and no-shows can reduce system costs on that specific day. Furthermore, it is reasonable to treat last-minute cancellations and no-shows as interchangeable occurrences in some cases.

6.5 CR Analysis

In this article, our objective is to minimize the expected cost of the system, which is a reasonable criterion given that the system knows the customer distribution. In the realm of online appointment scheduling, two additional types of evaluation criteria, namely, (i) regret analysis and (ii) CR analysis, are commonly employed to evaluate the worst-case guarantees of online policies when the distribution of the number of customers is unknown (Zhalechian et al., 2022; Stein et al., 2020). In this section, we compare the implemented optimal online nonsequential policies with their corresponding optimal offline policies, thereby assessing the performance of online policies through CR analysis. We define CR_i and CR as follows:

$$CR_i = \frac{OPT_{online}(\boldsymbol{n} = i)}{OPT_{offline}(\boldsymbol{n} = i)},$$

$$CR = \max_{\boldsymbol{n}=i} CR_i = \max_{\boldsymbol{n}=i} \frac{OPT_{online}(\boldsymbol{n} = i)}{OPT_{offline}(\boldsymbol{n} = i)}.$$

In Table 6, the service horizon is $T = 180$ min, and the number of customers follows the distribution $\boldsymbol{n} \sim 4 + binomial(3, P)$. We analyze nine instances with various values of P. The left part of the table displays the expected number of customers and the optimal online policy. On the right side, we present the optimal offline policy, along with its cost $\bar{g}_i$, when $\boldsymbol{n} = i$. To assess the comparison between the implemented online policy and the offline policy, we provide the value of CR_i for each instance when $\boldsymbol{n} = i$. Additionally, we include the distribution of $\boldsymbol{n}$, that is, $(p_7, \ldots, p_{10})$, and the weighted offline cost $p_i\bar{g}_i$ for further analysis.

From the table, we observe some intriguing phenomena. First, the optimal online policy remains consistent across cases with varying P values, such as $P = 0.1$, 0.2, or 0.3, which shows the robustness of our online policies. Second, the optimal online policy might exhibit subpar performance in the worst-case scenario, illustrated by the instance where $P = 0.9$ and $CR = 3.27$, with the worst case arising at $\boldsymbol{n} = 7$. This disparity stems from the system's awareness of the small probability associated with $\boldsymbol{n} = 7$ ($p_7 = 0$), in contrast to the significantly higher probability of $\boldsymbol{n} = 10$ ($p_{10} = 0.73$). Consequently, the efficacy of the online policy at $\boldsymbol{n} = 7$ becomes inconsequential, as long as it performs admirably at $\boldsymbol{n} = 10$.

In addition, we identify certain corresponding patterns between implemented online policies and offline policies. In general, when the system is not congested, such as at $P = 0.2$, the implemented online policy, characterized by a small $\boldsymbol{n}$ value, such as $\boldsymbol{n} = 8$, closely resembles the corresponding

Table 6. Performance of online policies under different realizations of $\boldsymbol{n}$.

$\boldsymbol{n} \sim 7 + binomial(3, P)$		Optimal offline policy with known $\boldsymbol{n}$															
		$\boldsymbol{n}$=7		$\bar{g}_7$	$\boldsymbol{n}$=8		$\bar{g}_8$	$\boldsymbol{n}$=9		$\bar{g}_9$	$\boldsymbol{n}$=10		$\bar{g}_{10}$			
		[1, 3, 6, 8, 11, 13, 16]		2.63	[1, 3, 5, 8, 10, 12, 14, 17]		8.95	[1, 3, 5, 7, 9, 11, 15, 17]		32.76	[1, 3, 5, 7, 9, 11, 13, 15, 17, 18]		105.73			
		Implemented online policy under the realizations of $\boldsymbol{n}$															
P	$E[\boldsymbol{n}]$	Optimal online policy	p_7	$p_7\bar{g}_7$	CR_7	p_8	$p_8\bar{g}_8$	CR_8	p_9	$p_9\bar{g}_9$	CR_9	p_{10}	$p_{10}\bar{g}_{10}$	CR_{10}	CR		
0.1	7.3	[1, 3, 7, 9, 12, 16, 14, 5, 18, 11]	0.73	1.92	158%	0.24	2.15	105%	0.03	0.98	145%	0.00	0.00	123%	158%		
0.2	7.6	[1, 3, 7, 9, 12, 16, 14, 5, 18, 11]	0.51	1.34	158%	0.38	3.40	105%	0.1	3.28	145%	0.01	1.06	123%	158%		
0.3	7.9	[1, 3, 7, 9, 12, 16, 14, 5, 18, 11]	0.34	0.89	158%	0.44	3.94	105%	0.19	6.22	145%	0.03	3.17	123%	158%		
0.4	8.2	[1, 3, 7, 9, 13, 15, 5, 17, 11, 18]	0.22	0.58	293%	0.43	3.85	170%	0.29	9.50	100%	0.06	6.34	100%	293%		
0.5	8.5	[1, 3, 7, 9, 13, 15, 5, 17, 11, 18]	0.12	0.32	293%	0.38	3.40	170%	0.38	12.45	100%	0.12	12.69	100%	293%		
0.6	8.8	[1, 3, 7, 9, 13, 15, 5, 17, 11, 18]	0.06	0.16	293%	0.29	2.60	170%	0.43	14.09	100%	0.22	23.26	100%	293%		
0.7	9.1	[1, 3, 5, 9, 11, 15, 17, 7, 13, 18]	0.03	0.08	327%	0.19	1.70	168%	0.44	14.42	100%	0.34	35.95	100%	327%		
0.8	9.4	[1, 3, 5, 9, 11, 15, 17, 7, 13, 18]	0.01	0.03	327%	0.1	0.90	168%	0.38	12.45	100%	0.51	53.92	100%	327%		
0.9	9.7	[1, 3, 5, 9,11, 15, 17, 7, 13, 18]	0.00	0.00	327%	0.03	0.27	168%	0.24	7.86	100%	0.73	77.18	100%	327%		

offline policy. When the system is crowded, as observed at $P = 0.8$, the implemented online policy, characterized by a larger $\boldsymbol{n}$ value, such as $\boldsymbol{n} = 10$, aligns precisely with the offline policy. To elucidate this phenomenon, we analyze it from three key perspectives: the expected number of customers $E[\boldsymbol{n}]$, the distribution p_i, and the weighted offline cost $p_i\bar{g}_i$. With regard to $E[\boldsymbol{n}]$, we observe that in most cases, excluding instances where $P = 0.1$ and 0.4, if $i^* \approx E[\boldsymbol{n}]$, then $i^* = \arg\min_i CR_i$. These exceptions indicate that the expectation may not contain sufficient information pertaining to the distribution of $\boldsymbol{n}$. Consequently, we delve into the pattern in terms of p_i and find that in most cases, except for cases with $P = 0.1$, 0.2, and 0.4, if $i^* = \arg\max_i p_i$, then $i^* = \arg\min_i CR_i$. Nevertheless, three exceptions persist. We further relate the offline cost with the distribution of $\boldsymbol{n}$, that is, $p_i\bar{g}_i$. It can be observed that if $i^* = \arg\max_i p_i\bar{g}_i$, then $i^* = \arg\min_i CR_i$, thus providing the most accurate interpretation save for a sole exception at $P = 0.3$.

Overall, for an optimal online policy, we can always find an implemented online policy that is close to or even identical to the corresponding optimal offline policy. The point at which this correspondence appears is closely related to the expected number of customers, the customer number with the highest probability, and the maximum weighted offline cost. This corresponding pattern motivates us to leverage multiple offline optimal policies to approximate an online policy.

6.6 Comparison with Heuristics

We further compare the optimal online nonsequential policies with four heuristics, as illustrated in Table 7. The first heuristic we examine is myopic scheduling (Zacharias and Pinedo, 2014). In this approach, customers call one by one, and once a customer calls, the system schedules a local optimal appointment for them immediately, regardless of possible future arrivals. The second heuristic is the pairwise interchange (Chen and Robinson, 2014). The search procedure for a nonsequential policy starts with the FCFS policy and then checks whether a pairwise interchange can reduce the cost. In each iteration, we keep the first interchange that leads to an improvement, move to a new sequence, and start afresh until no further improvement can be made. The third and fourth heuristics are the greedy algorithm and the online FCFS policy discussed above.

Of the four heuristics evaluated, the pairwise interchange and the greedy algorithm perform best, each showcasing distinct strengths. The pairwise interchange heuristic is particularly well-suited for congested systems, aligning with the observations made in the study conducted by Chen and Robinson (2014) on healthcare systems characterized by a high workload. Conversely, when the system is not crowded, the greedy algorithm demonstrates greater effectiveness. Thus,

Table 7. Comparison between heuristics and the optimum.

	Heuristic 1: Myopic versus OPT					Heuristic 2: Interchange versus OPT					Heuristic 3: Greedy versus OPT					Heuristic 4: FCFS versus OPT				
	0.3	0.4	0.5	0.6	0.7	0.3	0.4	0.5	0.6	0.7	0.3	0.4	0.5	0.6	0.7	0.3	0.4	0.5	0.6	0.7
[7, 11]	53.9%	60.4%	62.4%	61.8%	63.5%	28.6%	13.0%	16.4%	4.4%	5.1%	0.0%	0.0%	0.2%	0.3%	1.8%	47.6%	38.2%	34.4%	17.2%	11.6%
[7, 12]	57.6%	58.7%	58.3%	56.9%	53.4%	18.5%	12.9%	12.9%	2.0%	3.0%	0.1%	0.0%	5.1%	6.5%	8.3%	51.7%	37.2%	24.8%	27.3%	15.5%
[7, 13]	56.8%	55.2%	51.5%	44.6%	39.3%	10.5%	2.2%	2.5%	3.4%	0.3%	0.2%	5.5%	6.7%	7.8%	8.7%	60.0%	49.5%	28.7%	18.4%	10.1%
[7, 14]	53.6%	49.9%	41.4%	35.4%	29.0%	2.6%	2.6%	1.8%	0.3%	0.1%	1.2%	6.3%	6.5%	7.4%	7.8%	79.8%	55.8%	34.9%	20.2%	13.7%

the pairwise interchange and the greedy algorithm can complement each other adeptly, and combining these two heuristics results in a well-performing heuristic that is suitable for general service systems.

Moreover, although both myopic scheduling and greedy algorithms yield locally optimal decisions in the immediate context, their overall performance diverges considerably. This observation implies that a policy that appears optimal for the first few customers may not be optimal for the system.

7 Conclusions and Future Research

We study a fundamental online nonsequential scheduling problem with a random number of requests. Furthermore, our model incorporates customer cancellations, late arrivals, and no-shows, thus providing a comprehensive representation of real-world scenarios. To tackle this problem, we first investigate the online FCFS scheduling problem, which serves as a benchmark and provides an upper bound for optimal nonsequential scheduling. Subsequently, we introduce an innovative approach by integrating a dynamic assignment model and a shortest path model to develop an efficient B&B algorithm. This algorithm incorporates several elimination rules and introduces a lower bound to expedite the search process.

Through numerical studies, we assess the performance of our algorithms across different scenarios. The B&B algorithm is efficient enough to be applied to the daily appointment operations of a service system, such as 24 slots of length $t = 10$ min in a 4-hour morning or afternoon session, or 24 slots of length $t = 20$ min in an 8-hour day. In addition, our greedy algorithm serves as an excellent alternative for tackling larger-scale problems, as it can swiftly identify a suitable policy compared to the B&B algorithm.

Our computational study not only demonstrates the commendable performance of the proposed nonsequential policy but also several managerial insights. First, we point out situations where the adoption of nonsequential scheduling is more appropriate, namely in systems with highly fluctuating number of customers or in systems that are not congested. Second, we unveil the distinctive impacts of lateness, cancellations, and no-shows on the system. Patient lateness exerts a substantial increase in system costs, whereas cancellations and no-shows bring a certain degree of cost reduction for the given day. Through the competitive analysis, we observe a correspondence between an implemented online policy and an offline policy, suggesting that we can employ several optimal offline policies to approximate the online policy. Regarding the performance of heuristics, the pairwise interchange is more suitable for congested systems, while the greedy algorithm is more effective in relatively idle systems. The combination of these two heuristics is a well-performing heuristic suitable for general service systems. In addition, the result that the greedy algorithm outperforms myopic scheduling shows that a policy that is optimal for the first few customers is not necessarily optimal for the entire system.

Our work has limitations and future research will be valuable in several ways. First, evaluating overbooking in an online setting is valuable, where customers may have the same appointment time. Second, it would be interesting to study online appointing problems involving heterogeneous customers, such as different service time distributions. Note that in such a case, it is necessary to employ a *dynamic* policy that updates beliefs regarding future arrivals when scheduling real-time requests. Third, providing multiple choices to customers (Liu et al., 2019) has emerged as a new trend in online appointment problems, presenting an interesting direction for future research.

Acknowledgments

The authors sincerely thank the Senior Editor and two anonymous reviewers for their constructive comments that have substantially improved the quality of this work.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The work is partially supported by Hong Kong RGC CRF (Grant No. C6103-20G).

ORCID iDs

Zhixin Liu https://orcid.org/0000-0001-6146-8932
Xiangtong Qi https://orcid.org/0000-0003-4573-0157

Supplemental Material

Supplemental material for this article is available online (doi: 10.1177/10591478231224926).

Notes

1. Union Hospital, https://www.union.org/new/tc_chi/services/full/tko_timetable.pdf, accessed 24 October 2023.
2. MaxHealth in Texas, https://www.maxhealthmed.com/ accessed 24 October 2023.
3. HA Go (APP), https://www2.ha.org.hk/hago/, accessed 24 October 2023.

References

Ahmadi-Javid A, Jalali Z and Klassen KJ (2017) Outpatient appointment systems in healthcare: A review of optimization studies. *European Journal of Operational Research* 258(1): 3–34.

Bandi C and Gupta D (2020) Operating room staffing and scheduling. *Manufacturing & Service Operations Management* 22(5): 958–974.

Begen MA and Queyranne M (2011) Appointment scheduling with discrete random durations. *Mathematics of Operations Research* 36(2): 240–257.

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

Cayirli T, Veral E and Rosen H (2006) Designing appointment scheduling systems for ambulatory care services. *Health Care Management Science* 9(1): 47–58.

Chen RR and Robinson LW (2014) Sequencing and scheduling appointments with potential call-in patients. *Production and Operations Management* 23(9): 1522–1538.

Deceuninck M, Fiems D and De Vuyst S (2018) Outpatient scheduling with unpunctual patients and no-shows. *European Journal of Operational Research* 265(1): 195–207.

de Kemp MA, Mandjes M and Olver N (2021) Performance of the smallest-variance-first rule in appointment sequencing. *Operations Research* 69(6): 1909–1935.

Deng Y, Shen S and Denton B (2019) Chance-constrained surgery planning under conditions of limited and ambiguous data. *INFORMS Journal on Computing* 31(3): 559–575.

Denton B and Gupta D (2003) A sequential bounding approach for optimal appointment scheduling. *IIE Transactions* 35(11): 1003–1016.

Denton B, Viapiano J and Vogl A (2007) Optimization of surgery sequencing and scheduling decisions under uncertainty. *Health Care Management Science* 10(1): 13–24.

Diamant A (2021) Dynamic multistage scheduling for patient-centered care plans. *Health Care Management Science* 24(4): 827–844.

Erdogan SA, Gose A and Denton B (2015) Online appointment sequencing and scheduling. *IIE Transactions* 47(11): 1267–1286.

Feldman J, Liu N, Topaloglu H, et al. (2014) Appointment scheduling under patient preference and no-show behavior. *Operations Research* 62(4): 794–811.

Freeman NK, Melouk SH and Mittenthal J (2016) A scenario-based approach for operating theater scheduling under uncertainty. *Manufacturing & Service Operations Management* 18(2): 245–261.

Green LV, Savin S and Wang B (2006) Managing patient service in a diagnostic medical facility. *Operations Research* 54(1): 11–25.

Guda H, Dawande M, Janakiraman G, et al. (2016) Optimal policy for a stochastic scheduling problem with applications to surgical scheduling. *Production and Operations Management* 25(7): 1194–1202.

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

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

Hassin R and Mendel S (2008) Scheduling arrivals to queues: A single-server model with no-shows. *Management Science* 54(3): 565–572.

Izady N (2019) An integrated approach to demand and capacity planning in outpatient clinics. *European Journal of Operational Research* 279(2): 645–656.

Jiang B, Tang J and Yan C (2019) A stochastic programming model for outpatient appointment scheduling considering unpunctuality. *Omega* 82: 70–82.

Jouini O, Benjaafar S, Lu B, et al. (2022) Appointment-driven queueing systems with non-punctual customers. *Queueing Systems* 101(1-2): 1–56.

Kemper B, Klaassen CA and Mandjes M (2014) Optimized appointment scheduling. *European Journal of Operational Research* 239(1): 243–255.

Keyvanshokooh E, Kazemian P, Fattahi M, et al. (2022) Coordinated and priority-based surgical care: An integrated distributionally robust stochastic optimization approach. *Production and Operations Management* 31(4): 1510–1535.

Keyvanshokooh E, Shi C and Van Oyen MP (2021) Online advance scheduling with overtime: A primal-dual approach. *Manufacturing & Service Operations Management* 23(1): 246–266.

Kong Q, Lee C-Y, Teo C-P, et al. (2016) Appointment sequencing: Why the smallest-variance-first rule may not be optimal. *European Journal of Operational Research* 255(3): 809–821.

Kong Q, Li S, Liu N, et al. (2020) Appointment scheduling under time-dependent patient no-show behavior. *Management Science* 66(8): 3480–3500.

LaGanga LR and Lawrence SR (2012) Appointment overbooking in health care clinics to improve patient service and clinic performance. *Production and Operations Management* 21(5): 874–888.

Liu N, Van De Ven PM and Zhang B (2019) Managing appointment booking under customer choices. *Management Science* 65(9): 4280–4298.

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

Mak H-Y, Rong Y and Zhang J (2014a) Appointment scheduling with limited distributional information. *Management Science* 61(2): 316–334.

Mak H-Y, Rong Y and Zhang J (2014b) Sequencing appointments for service systems using inventory approximations. *Manufacturing & Service Operations Management* 16(2): 251–262.

Mancilla C and Storer R (2012) A sample average approximation approach to stochastic appointment sequencing and scheduling. *IIE Transactions* 44(8): 655–670.

Muthuraman K and Lawley M (2008) A stochastic overbooking model for outpatient clinical scheduling with no-shows. *IIE Transactions* 40(9): 820–837.

Pan X, Geng N and Xie X (2021) Appointment scheduling and real-time sequencing strategies for patient unpunctuality. *European Journal of Operational Research* 295(1): 246–260.

Patrick J, Puterman ML and Queyranne M (2008) Dynamic multipriority patient scheduling for a diagnostic resource. *Operations Research* 56(6): 1507–1525.

Qi J (2017) Mitigating delays and unfairness in appointment systems. *Management Science* 63(2): 566–583.

Samorani M and Ganguly S (2016) Optimal sequencing of unpunctual patients in high-service-level clinics. *Production and Operations Management* 25(2): 330–346.

Sauré A and Puterman ML (2014) The appointment scheduling game. *INFORMS Transactions on Education* 14(2): 73–85.

Stein C, Truong V-A and Wang X (2020) Advance service reservations with heterogeneous customers. *Management Science* 66(7): 2929–2950.

Truong V-A (2015) Optimal advance scheduling. *Management Science* 61(7): 1584–1597.

Wang D, Muthuraman K and Morrice D (2019) Coordinated patient appointment scheduling for a multistation healthcare network. *Operations Research* 67(3): 599–618.

Wang PP (1999) Sequencing and scheduling *n* customers for a stochastic server. *European Journal of Operational Research* 119(3): 729–738.

Wang S, Liu N and Wan G (2020) Managing appointment-based services in the presence of walk-in customers. *Management Science* 66(2): 667–686.

Wang X and Truong V-A (2018) Multi-priority online scheduling with cancellations. *Operations Research* 66(1): 104–122.

Weiss EN (1990) Models for determining estimated start times and case orderings in hospital operating rooms. *IIE Transactions* 22(2): 143–150.

Xie X, Fan Z and Zhong X (2021) Appointment capacity planning with overbooking for outpatient clinics with patient no-shows. *IEEE Transactions on Automation Science and Engineering* 19(2): 864–883.

Youn S, Geismar HN and Pinedo M (2022) Planning and scheduling in healthcare for better care coordination: Current understanding, trending topics, and future opportunities. *Production and Operations Management* 31(12): 4407–4423.

Zacharias C, Liu N and Begen MA (2022) Dynamic interday and intraday scheduling. *Operations Research*. DOI: 10.1287/opre.2022.2342.

Zacharias C and Pinedo M (2014) Appointment scheduling with no-shows and overbooking. *Production and Operations Management* 23(5): 788–801.

Zacharias C and Yunes T (2020) Multimodularity in the stochastic appointment scheduling problem with discrete arrival epochs. *Management Science* 66(2): 744–763.

Zeng B, Turkcan A, Lin J, et al. (2010) Clinic scheduling models with overbooking for patients with heterogeneous no-show probabilities. *Annals of Operations Research* 178(1): 121–144.

Zhalechian M, Keyvanshokooh E, Shi C, et al. (2022) Online resource allocation with personalized learning. *Operations Research* 70(4): 2138–2161.

How to cite this article

Zhu Y, Liu Z and Qi X (2024) Nonsequential Appointment Scheduling With a Random Number of Requests. *Production and Operations Management* 33(1): 184–204.