Assign-to-Seat: Dynamic Capacity Control for Selling Train Tickets

Feng Zhu
Institute for Data, Systems, and Society, Massachusetts Institute of Technology, MA, USA, fengzhu@mit.edu

Shaoxuan Liu
Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai, China, liusx@sjtu.edu.cn

Rowan Wang
Lee Kong Chian School of Business, Singapore Management University, Singapore, rowanwang@smu.edu.sg

Zizhuo Wang
School of Data Science, The Chinese University of Hong Kong, Shenzhen, Shenzhen, China, wangzizhuo@cuhk.edu.cn

We consider a revenue management problem arising from the sales of high-speed train tickets in China. Compared to traditional network revenue management problems, the new feature of our problem is that each request, if accepted, needs to be instantly assigned to a unique seat throughout the whole journey, and later adjustment is not allowed. Therefore, when making decisions, the seller needs to track not only the total capacity available but also the status of each seat.

We build a modified network revenue management model for this problem. We first study a static problem where all requests are given. Despite that the problem is NP-Hard in general, we identify conditions of the capacity matrix for polynomial-time solvability, based on which we propose a booking limit control policy which achieves asymptotic optimality. We then introduce a bid-price control policy based on a novel maximal sequence idea, which allows for non-linearity in bid-prices and thus results in a more accurate approximation of the value function than the traditional bid-price control policy. Finally, combining the dynamic view of the maximal sequence and the static solution of the primal problem, we propose a *re-solving a dynamic primal* (RDP) policy which achieves uniformly bounded revenue loss under a mild assumption. Numerical experiments using both synthetic and real data show the advantage of our proposed policies in improving capacity allocation efficiency.

Key words: revenue management, capacity control, dynamic programming, bid-price control, re-solving

1. Introduction

Revenue management (RM) has played increasingly important roles in many industries in the past few decades. Born in the 1980s from the airline industry, revenue management has transformed various industries (e.g., airline, hotel, and car rental) by revolutionizing the ways sales decisions are made (Cross 1997). This trend of transformation continues if not accelerates in the last few years. With the rapid growth of information technology and the arrival of the big data era, every firm is hoping to use data to inform their sales decisions in order to provide better service to their customers and to grow their revenue.

In this work, we study a revenue management problem that arises from the high-speed railway industry in China. The high-speed railway industry is a fast growing industry in China and the high-speed trains have become a main means of transportation in China, bringing great convenience to the Chinese people. For example, the Beijing-Shanghai high speed train, since it started operating in 2011, has carried more than 825 million passengers until 2018, achieving a revenue of $4.4 billion US dollars in the year of 2018 alone (Global Times 2018). Across China, the high speed train has taken more than 2 billion passengers in 2018, and the annual passenger number still increases at a rate of over 15% (Global Times 2019).

Despite such high demand and fast growth, the high-speed railway industry in China is still struggling to make a profit and to pay off its debt because of the high construction cost and operations cost (Murayama 2017). With such a massive size of passengers, a first thought to make more profits might be to use some advanced dynamic pricing strategy similar to what the airline industry does. However, at least until this point, the train ticket prices are still fully regulated by the Chinese government, which means that the fares between two cities are fixed, regardless of the time of purchase. Nevertheless, the train company can decide whether to close out sales for an itinerary if it is beneficial to do so. Indeed, in many cases, it is difficult to buy train tickets for a short itinerary even though there are seats available as the train company may reserve those seats for longer itineraries. Thus whether to open up an itinerary for sale can be controlled by the train company, resulting in a capacity control problem. Furthermore, the fares in general are not additive in segments. Longer (shorter) itineraries in general have a lower (higher) per-mile price. Taking the Beijing-Shenzhen train G79 for example, it has 7 stops en route, including Wuhan as one stop. The fares for Beijing-Wuhan and Wuhan-Shenzhen segments are 520.5 RMB and 538 RMB respectively, while the fare from Beijing to Shenzhen is 936.5 RMB (all fares are as of 2019/07/01), which is about 12% less compared to the sum of the fares of the two segments. Therefore, selling a seat to two passengers for sub-journeys could make a higher revenue than selling it to one passenger for the whole-journey. In the meantime, reserving seats for shorter journeys may have the risk of not being able to find a matching demand, resulting in empty seats on the train.

In addition, when accepting a passenger's request for an itinerary, according to the current regulation, the train company must assign a seat to that passenger (with a seat number), which means that there must be an available seat throughout the entire journey in order to accept a request. In the airline industry, a trip with multiple segments typically consists of different flights. Passengers need to take off from the previous flight and get on the next one. Therefore, it does not matter whether a passenger sits on the same seat throughout the journey or not (moreover, connecting flights may even use different aircrafts). However, for travelling by train, this is not the case. Although there may be several intermediate stops, passengers stay in the same train

throughout the whole journey. For the Chinese high-speed train system (and perhaps many other train systems), each passenger is assigned to a single seat for his/her entire trip, and the seat number is provided to the passenger at the time of ticket booking. In other words, it is not allowed to serve a passenger by using a combination of multiple seats throughout the journey. We call such restriction the *assign-to-seat* restriction. Also, once a seat assignment is made, it cannot be changed later. All these restrictions make the corresponding capacity allocation problem challenging and distinct from those studied in existing literature.

In this work, we shed light on the above described problem and propose some practically useful capacity allocation policies. In particular, we investigate the following questions:

1. How to model such a capacity allocation problem with the unique restrictions? How to efficiently track the system dynamics, especially for trains with a large number of seats?
2. What are the connections and differences between this problem and the classical network revenue management problem?
3. Are there any simple and easy-to-implement capacity allocation policies? Will the policies perform well under different scenarios?

To answer these questions, we construct a modified network revenue management model. We consider a discrete time model in which passengers arrive sequentially, each requesting a certain itinerary with a certain probability. The price for each itinerary is fixed, and the train company needs to decide in each time period whether to accept a request of a certain itinerary or not and if so, which seat to assign the request to, given the availability status of the remaining seats on the train at that moment. Note that the latter question is not faced in the traditional network revenue management problem, in which only the total capacity matters (as in the airline setting) or the assignment decision only needs to be made at a later point (as in the hotel setting).

To analyze the problem, we start by considering a static problem in which all passengers requests are given in advance. The static problem is useful for both understanding the structure of the problem and instructing the allocation rule in the dynamic setting. We show that the static problem is NP-Hard for a general capacity matrix (i.e., we allow the capacity matrix to reflect any availability status on the train, with certain seats taken on certain segments) regardless of the structures of the prices (e.g., even when prices have a linear structure). Nevertheless, if the initial capacity matrix satisfies certain structure (which we later call the non-overlapping and sharing endpoint structure, or the NSE structure), then there exists a polynomial-time algorithm to solve the static allocation problem. Our analysis connects the problem to traditional network RM problems and gives rise to simplified formulations of the original problem, which lead to approximation algorithms that can provide a near optimal solution efficiently.

To analyze the dynamic model, we first propose a booking limit control policy, which uses the solution to the static problem to set acceptance limits for each type of requests. The booking limit control policy achieves asymptotically optimal performance. However, it may not perform well in non-asymptotic regimes because of the strong restriction in this policy which could lead to inefficient use of the capacity. We then study the bid-price control policies. In our setting, it is important to consider how bid-prices can be generated to instruct which seat to allocate a request, which turns out to be closely related with how we track the dynamics of the capacity matrix. We propose two ways to generate bid-prices, one based on the dual formulation of the static model and the other based on a more delicate formulation of the problem by considering the longest consecutive available segments (which we call the *maximal sequence*) of the seats. Both bid-prices can be generated from an approximate dynamic programming (ADP) point of view, while the latter one allows for *non-linear* bid-prices and thus leads to more accurate approximation of the value function. Furthermore, we propose a policy called "re-solving a dynamic primal" (RDP) which re-solves the primal problem under the maximal sequence formulation in each time period and uses the solution to instruct the allocation in that period. We prove that RDP policy achieves a uniformly bounded revenue loss. To validate the performance of our proposed policies, we conduct numerical experiments using both synthetic data as well as real data from the China Railway High-speed (CRH). From the numerical experiments, we show that the two policies based on maximal sequence achieve great performance, with the potential of increasing the revenue by 5%-8% compared to a naive policy under high demand intensity regime.

To summarize, our paper is the first to study the dynamic capacity allocation problem in the railway setting with the *assign-to-seat* requirement, which is distinct from previous revenue management problems in several aspects. Our analysis sheds light on the structure of such problems and we propose practically useful seat allocation policies. Our results can be served as a first step to understand such problems and ultimately help increase the revenue and utility of train seats.

Before we proceed, we introduce the notation that will be used throughout our discussion. We use $\mathbb{N} = \{0, 1, 2, ...\}$ to denote the set of all natural numbers and $[N]$ to denote the set of all positive integers that are no larger than N, i.e., $[N] = \{1, ..., N\}$. Also, we use $\mathbf{e}_i$ to denote a row vector of zeros with 1 at the ith entry, and $\mathbf{e}_{ij}$ to denote a row vector of zeros with 1s starting from the ith entry to the jth entry. The dimension of $\mathbf{e}_i$ or $\mathbf{e}_{ij}$ will be clear from the context. Moreover, for any set S, we use $|S|$ to denote the cardinality of S, i.e., the number of elements in S. Finally, for two vectors or matrices $\mathbf{a}$ and $\mathbf{b}$, we use $\mathbf{a} \leqslant \mathbf{b}$ to denote componentwise inequalities.

The rest of the paper is organized as follows. In the remaining of this section, we review related literature. In Section 2, we introduce the model of our problem. In Section 3, we study the static model. In Section 4, we study the dynamic model and propose several policies. We perform numerical experiments in Section 5. Section 6 concludes the paper.

Literature Review

On the high level, our work is closely related to two streams of literature, network revenue management and interval scheduling. In the following, we review these two literature and discuss their relation with our work.

Network Revenue Management Broadly speaking, our work is a special case of the quantity-based *network revenue management* (network RM) problem, which has been studied extensively in the literature since Williamson (1992). Though the problem can be fully characterized by a dynamic programming (DP) formulation, a key challenge is that the number of states grows exponentially with the size of the problem, making it impractical to solve directly. In the literature, there have been various attempts to circumvent the difficulty, e.g., by obtaining booking limit or bid-price controls from some static formulation, or by approximating the value function with some simple structures. In the following, we review some of these approaches.

A seminal work in the literature on booking limit control is Gallego and van Ryzin (1997), in which a static model is studied and a make-to-stock policy and a make-to-order policy are proposed. In a general sense, booking limit control owns the property of *asymptotic optimality*. However, it lacks flexibility when facing stochastic demand and may not perform well for practical-sized problems. Talluri and van Ryzin (1998) are among the first to propose bid-price control policies. Since then abundant literature begins to focus on deriving delicate bid-prices and tighter bounds on the value functions. Bertsimas and Popescu (2003) consider a certainty equivalence control policy, which directly uses the optimal value of the static model as an approximation of the initial value function. Adelman (2007) proposes a framework for obtaining an *upper bound* on the value functions via an approximate dynamic programming (ADP) approach. The main idea is to consider the DP as a linear program (LP) and substitute value functions with approximation structures, and then to solve the LP with significantly less variables. In Adelman (2007), an affine structure is considered where bid-prices are naturally embedded. Much work follows this line, see, e.g., Zhang and Adelman (2009), Zhang (2011), Tong and Topaloglu (2013), Kunnumkal and Talluri (2015). There is also another stream of works that use Lagrangian relaxation to approximate the value functions, see Topaloglu (2009), Kunnumkal and Topaloglu (2010), Tong and Topaloglu (2013), Kunnumkal and Talluri (2015), etc.

A useful technique that is usually applied in practice is *re-solving*. The idea is that the performance of a static policy could be potentially improved by incorporating updated state information periodically. The impact of re-solving has been extensively studied in the literature. Cooper (2002) gives a counterexample where re-solving could reduce the collected revenue. Jasin and Kumar (2013) point out that re-solving does not help for a wide range of deterministic policies based on

LP asymptotically. Nevertheless, in various settings, re-solving has been shown to perform well both in theory and in practice (see, e.g., Maglaras and Meissner 2006, Secomandi 2008, Chen and Homem-de Mello 2010). We note that in recent years, researchers begin to explore new forms of policies. For example, policies based on *probabilistic allocation* have attracted attention, and elegant theoretical properties have been established, see, e.g., Reiman and Wang (2008), Jasin and Kumar (2012), Bumpensanti and Wang (2018).

We now explain the relation between network RM problems and our problem. First, our problem is related to the *Railway Revenue Management* problems (see, e.g., Ciancimino et al. 1999, Armstrong and Meissner 2010), a special case of general network RM problems. Commonly, in railway RM problems, each product is a consecutive combination of legs, and our problem also inherits this property. However, we consider the *assign-to-seat* restriction which is missing in existing literature. This difference entails one to keep track of not only the total remaining capacity of each leg, but also the detailed occupation status of each seat. As we show in this paper, such difference indeed brings different analysis and unique results for the corresponding revenue management problem. Second, our problem is also related to the literature on *Hotel Revenue Management* for multiple stays, see, e.g., Goldman et al. (2002), Liu et al. (2008), Nadarajah et al. (2015), Aydin and Birbil (2018). Similar to the difference between our work and the traditional railway RM problems, these works do not consider the *assign-to-seat* restriction, which is the key ingredient in our problem. In hotel industry, a hotel does not need to assign the room number instantly after a customer books a room. Usually the room number is assigned when the customer checks into the hotel. As we will show in our discussion, in such situation, even if each customer needs to be assigned a room eventually, the problem still reduces to a traditional network RM problem, which means that hotels only need to consider the constraints for total room capacity upon booking.

In recent years, there has been an increasing amount of work on revenue management for reusable resources, see, e.g., Levi and Radovanović (2010), Chen et al. (2017), Gong et al. (2019), Feng et al. (2019), Rusmevichientong et al. (2020), Lei and Jasin (2020). Our work shares some similarity with this line of work in the sense that, in our model, the consumption of resources (seats) is in the form of an interval. However, our model and solution approach are different from those considered in this line of literature. First, in our model, the resource is not ordered by time; thus, there is no direct relation between the order of arrival with the requested resource. On the contrary, in many RM problems with reusable resources, there is an important notion of advance reservation time, which affects the type of resources needed at a particular time, see, e.g., Chen et al. (2017). Second, in our model, the decision is on whether to accept and the assignment of a request, whereas in many reusable RM models, the decision is on the assortment (Feng et al. 2019, Gong et al. 2019, Rusmevichientong et al. 2020) or pricing (Owen and Simchi-Levi 2018, Lei and Jasin 2020). Finally,

and most importantly, we consider the assign-to-seat requirement, that is, we need to assign a request to a resource upon arrival. However, most RM literature on reusable resource focus on total capacity (Levi and Radovanović 2010, Chen et al. 2017, Gong et al. 2019). In terms of results, we propose dynamic allocation policies with asymptotic optimal performance, while many of the works in the reusable RM literature obtain a constant competitive ratio policy (Feng et al. 2019, Gong et al. 2019, Rusmevichientong et al. 2020). Therefore, although our work is related to the reusable RM literature, the model, analysis and results are quite different.

Last but not least, we note that there is also a line of research focusing on *online packing* problems, which are closely related to network RM problems. In such problems, each arriving request consists of a combination of items and there is a capacity constraint on the total amount of each item. Dynamic algorithms have been proposed for this problem under various settings, see, e.g., Devanur and Hayes (2009), Molinaro and Ravi (2013), Agrawal et al. (2014), Kesselheim et al. (2014), Freund and Banerjee (2019). However, in our problem, each request consists of *consecutive seats*, providing an additional structure. In addition, there is an *assign-to-seat* restriction in our problem. These differences make our problem different from typical packing problems.

Interval Scheduling From a technical point of view, our work is also closely related to the works on *interval scheduling*, which is a special type of *scheduling* problems. Interval scheduling problems are formulated in terms of *machines* and *jobs*. Each job is represented by an *interval* describing the time when it needs to be carried out. Machines are to execute jobs, and their types may be either identical or nonidentical, depending on problem structures. The goal is to schedule the jobs on the machines to minimize the total costs or maximize the total rewards. For a comprehensive review of the literature on interval scheduling, we refer the readers to Schmidt (2000), Kolen et al. (2007), Kovalyov et al. (2007), etc. In the following, we only review those works that are closely related to our paper.

Arkin and Silverberg (1987) is among the earliest to consider scheduling jobs with fixed starting and ending times and different weights. Under the assumption of *identical* machines, they propose an algorithm that runs polynomially in the number of jobs. Brucker and Nordmann (1994) extend identical machines to non-identical ones but consider jobs with equal weights. They call such problem *k-track assignment problem*, and show that checking whether there exists a valid schedule for given jobs and machines is generally NP-Hard. They also propose a dynamic programming method to solve the problem. Kolen and Kroon (1993) investigate the computational complexity of a broader range of k-track assignment problems when the weights are arbitrary and the types of machines are given. Many other works follow this line of research and consider maximizing the total weight of scheduled jobs in different settings, e.g., considering identical machines (Carlisle

and Lloyd 1995, Bouzina and Emmons 1996), allowing processing multiple jobs on one machine, (Faigle et al. 1999, Angelelli et al. 2014), and adding constraints on overall processing time (Eliiyi and Azizoglu 2006).

A special case of the interval scheduling problem is called the job interval scheduling problem (JISP) in which jobs are packaged into several groups such that the schedule admits at most one job in each group. For this NP-Hard problem, Spieksma (1999) formalizes the problem with equal weights using integer optimization. By considering the linear optimization relaxation, he obtained a 1/2-approximation algorithm. Chuzhoy et al. (2006) propose a randomized algorithm that improves the ratio to $1 - 1/e - \epsilon$ for arbitrary positive number ϵ. Many other works have also focused on obtaining approximation algorithms with theoretical guarantees, see, e.g., Bar-Noy et al. (2001a,b), Bhatia et al. (2007).

Now we point out the connection and difference between our study and those in the literature. In the static setting with all the requests known, our problem is a special case of the weighted JISP. Therefore, some theoretical properties or algorithms of JISP could be applied to the static setting of our problem. However, different from the common literature, we study the effect of aggregation of different jobs in solving the problem. In particular, in our problem, there are often a large number of identical jobs with identical weights, and we study when an aggregation formulation can be helpful. Such aggregation idea is unique in our analysis and thus the algorithm we propose and its corresponding complexity results are also different from those in the literature. In the dynamic setting, our problem are connected to the *online interval scheduling* problems, see, e.g., Lipton and Tomkins (1994), Seiden (1998), Erlebach and Spieksma (2003), Im and Wang (2011), Shalom et al. (2014). Typically, in such problems, competitive ratios are analyzed where jobs arrive in an adversarial order and the number of identical machines are limited, while the type of jobs varies. The main distinction of our study is that we focus on the situation where there are a small or fixed number of job/machine types but a large number of machines, and the jobs arrive in a stochastic online fashion. Moreover, we allow the initial seat status to be arbitrary, which means, translated into online interval scheduling settings, the machines can be non-identical or occupied during some intervals at the beginning. Because of these differences, we tackle the dynamic problem with approaches and analysis that are completely different from those in the online interval scheduling literature, and we obtain asymptotically optimal policies for the dynamic problem.

2. Model

We consider the problem of a passenger railway service company selling train tickets for a particular route with N homogeneous seats. The route consists of $M+1$ stops, or equivalently M legs. Here, a leg represents the trip between two adjacent stops. The first leg is denoted by leg 1, and the

last leg is denoted by leg M. In the following of the paper, we will only use the term *leg* in our discussion, unless otherwise stated. The seller could sell itineraries containing any leg i to any leg $j \geqslant i$, which we denote as $i \to j$, at a price of p_{ij}. In our model, we assume that the ticket prices p_{ij}s are fixed, which is common in many regulated industries. At the time a passenger purchases a train ticket, he/she must be assigned to a fixed seat throughout the entire trip (in practice, with a given seat number). In other words, assigning a passenger to different seats on different legs of his/her itinerary is not allowed.

We adopt a discrete time model, in which the time is discretized to $1, ..., T$, where 1 is the start of the selling horizon and T is the end of the selling horizon[1]. In each period t, a passenger with request $i \to j$ comes with probability λ_{ij}^t. By making time periods sufficiently small, we assume that $\sum_{1 \leqslant i \leqslant j \leqslant M} \lambda_{ij}^t \leqslant 1$ for all t. The probability that no customer arrives in period t is $\lambda_0^t = 1 - \sum_{1 \leqslant i \leqslant j \leqslant M} \lambda_{ij}^t$. For any $1 \leqslant t_1 \leqslant t_2 \leqslant T$, we denote $d_{ij}^{[t_1,t_2]}$ as the number of coming request $i \to j$ in $[t_1, t_2]$. We also denote $\lambda_{ij}^{[t_1,t_2]} = \mathbb{E}\left[d_{ij}^{[t_1,t_2]}\right]$ as the expected number of request $i \to j$ in $[t_1, t_2]$.

At the beginning of each period, the seller decides which itineraries to offer for the current period, based on the remaining number of seats available on different legs. At the same time, the seller also needs to decide the seat number assignment for each itinerary to offer. These decisions determine whether an arriving passenger can find his/her requested itinerary available for purchase and if so, the seat number assigned to him/her. We call such a decision rule a *policy* for the seller and denote it by π subsequently. More precisely, let $C^t \in \{0,1\}^{N \times M}$ be the matrix such that $C_{k\ell}^t$ denotes whether leg ℓ in seat k is available at the beginning of period t. We call C^t the *capacity matrix* which characterizes the state of seats at the beginning of period t. For the ease of notation, we define $C_{k0}^t = C_{k(M+1)}^t = 0$ for all k and t. Then a policy π can be interpreted as a mapping from time period t and the capacity matrix C^t to a set of binary variables $u_{k,ij}^t$s where $u_{k,ij}^t \in \{0,1\}$ represents whether a request $i \to j$ arriving in period t will be accepted and assigned to seat k. In our model, seat assignment is determined at the time of purchase and cannot be changed at a later time, and any rejected request is lost. In addition, we do not allow overbooking. The objective of the seller is to determine a policy that leads to the highest expected revenue from ticket sales throughout the entire selling horizon.

[1] We remark that in practice, the selling horizons for different itineraries could end at different times. For example, China Railway High-speed (CRH) stops selling the ticket for a particular train 10 minutes before its departure time. In our paper, for simplicity, we ignore such difference and assume that the selling horizons of all itineraries end at the same time.

3. The Static Problem

In this section, we consider the static problem in which all requests from passengers are known in advance. Analyzing the static problem is common in the literature of network revenue management. It has two purposes. First, the static problem can be used as a useful offline benchmark to measure the performance of any dynamic policy. Second, since the static problem is equivalent to collecting all requests and carrying out the final decision at the end of the booking horizon, it could often lead to efficient policies. In this section, we study the structure and important properties of the static problem, from which we obtain an efficient booking limit control policy that achieves asymptotic optimality.

In the static problem, the number of request $i \to j$ is known to the seller and we denote it by $d_{ij} \in \mathbb{N}$. The seller needs to decide how many each type of request to accept in order to maximize the revenue. Let $x_{k,ij}$ denote whether seat k is used to serve one request $i \to j$. The static allocation problem can be formulated as the following integer program (IP):

$$\begin{aligned}
\text{maximize} \quad & \sum_{1\leqslant i\leqslant j\leqslant M} p_{ij} \sum_{k=1}^{N} x_{k,ij} && && (1)\\
\text{subject to} \quad & \sum_{k=1}^{N} x_{k,ij} \leqslant d_{ij}, && \forall 1 \leqslant i \leqslant j \leqslant M,\\
& \sum_{(i,j):i\leqslant \ell\leqslant j} x_{k,ij} \leqslant C_{k\ell}, && \forall k \in [N], \ell \in [M],\\
& x_{k,ij} \in \{0,1\}, && \forall k \in [N], 1 \leqslant i \leqslant j \leqslant M.
\end{aligned}$$

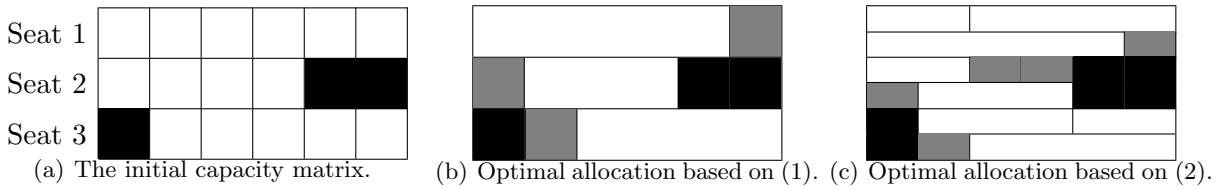
Note that in (1) we adopt a formulation with a general capacity matrix C where $C_{k\ell}$ represents whether seat k is available on leg l. This formulation is useful since we want to use such a static model to instruct allocation during the selling horizon (when certain seats have already been taken). If all seats are available (this is the case at the beginning of the sales horizon), then one can set all $C_{k\ell}$ equal to 1. Later we will show that the profile of $C_{k\ell}$ actually affects the complexity of solving this problem.

Since (1) is an integer program, it is natural to consider its linear program (LP) relaxation given as below.

$$\begin{aligned}
\text{maximize} \quad & \sum_{1\leqslant i\leqslant j\leqslant M} p_{ij} \sum_{k=1}^{N} x_{k,ij} && && (2)\\
\text{subject to} \quad & \sum_{k=1}^{N} x_{k,ij} \leqslant d_{ij}, && \forall 1 \leqslant i \leqslant j \leqslant M,\\
& \sum_{(i,j):i\leqslant \ell\leqslant j} x_{k,ij} \leqslant C_{k\ell}, && \forall k \in [N], \ell \in [M],\\
& x_{k,ij} \geqslant 0, && \forall k \in [N], 1 \leqslant i \leqslant j \leqslant M.
\end{aligned}$$

Here we ignore the $x_{k,ij} \leqslant 1$ constraints because the second group of constraints in (2) has already ensured that $x_{k,ij} \leqslant 1$. The first question we would like to answer is whether there is a gap between the IP formulation (1) and its LP relaxation (2). The following example shows that in general there could be a gap between (1) and (2), even when the prices p_{ij}s satisfy some nice properties.

EXAMPLE 1. Consider the capacity matrix shown in Figure 1(a), with 3 seats and 6 legs. We assume that the passengers' requests are as follows: $d_{ij} = 1$ if $(i,j) \in \{(1,2),(1,5),(2,4),(3,6),(5,6)\}$ and $d_{ij} = 0$ otherwise. Moreover, we set $p_{ij} = \sum_{i \leqslant t \leqslant j} v_t$, where $v_1 = v_2 = v_3 = \frac{1}{2}v_4 = v_5 = \frac{1}{2}v_6 = 1$. In this setting, the optimal value of IP (1) is 16 (the optimal allocation is shown in Figure 1(b)) and the optimal value of LP (2) is 16.5 (the optimal allocation is shown in Figure 1(c)).



(a) The initial capacity matrix. (b) Optimal allocation based on (1). (c) Optimal allocation based on (2).

Figure 1 Illustrations of Example 1. The black, white, and grey blocks represent seat-leg pairs that are already occupied and not available for allocation, allocated to demand requests, and unused in the allocation, respectively. Note that in (c), fraction of requests can be accepted and the seats are used to serve fractional requests.

Note that in this example, prices are *linear*. That is, we could endow each leg with a *unit price*, and the price of a ticket is the sum of the unit prices of its occupied legs. This example shows that even when prices are linear, there might be an integrality gap between (1) and (2). □

Example 1 shows that in general one cannot count on solving the LP relaxation (2), to obtain an optimal integer solution to the static problem (1). In fact, as we show below, this problem is NP-Hard in general.

THEOREM 1. *Problem (1) is NP-Hard.*

The proof of Theorem 1 is given in EC.1. In the proof, we establish a polynomial-time reduction from a known NP-Hard problem — Fixed Job Scheduling Problem — to our problem (1), given an arbitrary set of prices. In fact, our proof shows that problem (1) is NP-Hard even if the prices satisfy any predetermined structures (e.g., the prices are additive in each leg or the prices are the same for different itineraries). Despite the hardness of solving the general problem, the question still remains that whether we can efficiently solve (1) under certain conditions. As we will show, the structure of the capacity matrix C plays a key role in determining the solvability of the problem. If the capacity matrix C satisfies certain properties, then we can prove that the problem becomes polynomial-time solvable, which in turn implies (1) = (2).

To present our results, we introduce a few definitions that characterize the structure of the capacity matrices. In the following, we use $C_k \in \{0,1\}^{1\times M}$ to denote the kth row of the capacity matrix C. We start with the following definitions.

Definition 1 (Maximal Sequence). Given a capacity matrix $C \in \{0,1\}^{N\times M}$, we call $[u,v]$ a *maximal sequence of seat* k in C, denoted by $[u,v] \sim C_k$, if and only if (we define $C_{k0} = C_{k(M+1)} = 0$ for all k)

$$C_{ku} = C_{k(u+1)} = \cdots = C_{kv} = 1,\ C_{k(u-1)} = C_{k(v+1)} = 0.$$

Furthermore, we call $[u,v]$ a *maximal sequence in* C, denoted by $[u,v] \sim C$, if there exists a $k \in [N]$ such that $[u,v]$ is a maximal sequence of seat k. We also define $\mathcal{M}_{uv}(C)$ as the set of seats that contains $[u,v]$ as a maximal sequence, i.e., $\mathcal{M}_{uv}(C) = \{k \in [N] | [u,v] \sim C_k\}$. □

Definition 2 (NSE and Strongly NSE). Given a capacity matrix $C \in \{0,1\}^{N\times M}$, we call C to have the *NSE* (nonoverlapping or sharing endpoints) property, or simply, C is *NSE* if for *any* two maximal sequences $[u_1,v_1]$ and $[u_2,v_2]$ in C, one of the following holds:

1. (sharing at least one endpoint) $u_1 = u_2$ or $v_1 = v_2$;
2. (non-overlapping) $u_1 > v_2$ or $v_1 < u_2$.

Furthermore, we call C to be *strongly NSE* if one of the following holds:

1. (sharing at least one endpoint) $u_1 = u_2$ or $v_1 = v_2$;
2. (strongly non-overlapping) $u_1 > v_2 + 1$ or $v_1 < u_2 - 1$. □

Next, we introduce a formulation that is only based on the aggregated capacity, i.e., without the assign-to-seat restriction. Let $c_\ell = \sum_{k=1}^{N} C_{k\ell}$ be the total number of seats with leg ℓ available. We further define $c_0 = c_{M+1} = 0$. Let x_{ij} denote the number of request $i \to j$ that is accepted. We write the aggregated allocation problem as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{1\leqslant i\leqslant j\leqslant M} p_{ij}x_{ij} && && (3)\\ \text{subject to} \quad & 0 \leqslant x_{ij} \leqslant d_{ij}, && \forall 1 \leqslant i \leqslant j \leqslant M, \\ & \sum_{(i,j):i\leqslant\ell\leqslant j} x_{ij} \leqslant c_\ell, && \forall \ell \in [M]. \end{aligned}$$

Problem (3) has some nice properties. First, both the number of variables and constraints are $O(M^2)$ and do not depend on N. More importantly, it always has an integral optimal solution. To see this, we note that the constraints of (3) are totally unimodular (see, e.g., Ciancimino et al. 1999). Moreover, Problem (3) can be viewed as a network flow problem which can be solved in strongly polynomial time (see, e.g., Arkin and Silverberg 1987). Because of the two reasons, it is enticing to ask whether (3) could provide guidance for solving (1) under some circumstances. Theorem 2 presents the answer to this question, which is related to the NSE property of the capacity matrix.

THEOREM 2. *For any fixed positive prices $\{p_{ij}\}$, (1) and (3) have the same optimal value for any nonnegative integers $\{d_{ij}\}$ if and only if C is strongly NSE. Furthermore, if C is NSE, then (1) can be solved in polynomial time.*

The detailed proof of Theorem 2 can be found in EC.1. Here we describe the main ideas and steps in the proof. The proof of Theorem 2 consists of four steps. In the first step, we show that any strongly NSE matrix can be decomposed into non-overlapping groups in polynomial time, with the maximal sequences in each group having a specific structure. In the second step, we show that when the maximal sequences in a capacity matrix satisfy the abovementioned structure, we can solve the aggregate optimization problem (3) and recover a seat assignment from the solution of (3) in polynomial time. The assignment is constructed by assigning the accepted requests in a particular order. Then in the third step, we show the reverse direction. That is, if C is not strongly NSE, then there always exists a set of demands $\{d_{ij}\}$ such that there is no way to assign the solution of (3). In the last step, we show that given a capacity matrix C that is NSE, we can add a dummy leg between each pair of consecutive legs, converting the problem to one with a strongly NSE capacity matrix. Then we show that we can find an optimal solution in polynomial time.

Theorem 2 gives rise to the following corollary, which will be useful in our subsequent discussion.

COROLLARY 1. *If C is strongly NSE, then for any nonnegative real numbers $\{d_{ij}\}$, (2) and (3) have the same optimal value.*

By Theorem 2, for strongly NSE capacity matrices C, from the static view, we can achieve the same revenue with or without the assign-to-seat restriction, regardless of the demand. For other classes of capacity matrices, the property does not hold. From the dynamic view, within the special class of strongly NSE matrices, if we are allowed to assign seats for the requests *after* the final period T, then as long as the accepted requests satisfy the total capacity constraint, we are always able to find a feasible assignment. In practice, the initial capacity matrix C at the beginning of the selling horizon is completely unoccupied, and thus C is strongly NSE. Therefore, the difference between the revenue achieved by any dynamic policy with and without the assign-to-seat restriction can also be viewed as the value of delayed assignment.

We also give some remarks on the difference between our problem and the problem in hotel revenue management. In hotel revenue management with multiple-day stays, there also exists an "assign-to-seat" restriction: Customers need to remain in the same room during their stays. Nevertheless, different from our problem, the hotel could assign room numbers while customers check in. When the hotel is planning for a certain periods of time, the availabilities of each room is consecutive in time and ends on an identical day, forming a strongly NSE matrix. Our results indicate that as long as the requests satisfy the total capacity constraints for each day, the hotel

could always find a feasible assignment. Therefore, although there is an "assign-to-seat" equivalent restriction in hotel industry, the possibility of delayed assignment makes the problem less difficult. In contrast, in our problem, the assignment needs to be done at the time of request, which makes solving the problem much more challenging.

Now we discuss some related results on the hardness of the problem in the literature. Note that most relevant results are in the *scheduling* literature, and we discuss them after transferring their settings into ours. When C is completely unoccupied, i.e., when $C_{k\ell} = 1$ for all k and ℓ, Arkin and Silverberg (1987) show that (1) and (3) have the same optimal value and give an $O(D^2 \log D)$ algorithm to obtain an integral optimal solution of (1), where D is the total number of requests. Later, Brucker and Nordmann (1994) establish the equivalence between (1) and (3) under the condition that p_{ij}s are all equal and all maximal sequences have the same starting or ending leg.

A closer result to ours is by Kolen and Kroon (1993). They discuss the computational complexity of a problem more general than ours. If translated into our setting, their problem is equivalent to maximizing the total revenue when the "topological structure" of maximal sequences are given, but the amount of maximal sequences, requests, and the price of tickets have no restrictions (identical tickets may have different prices). They show that the problem is polynomial-time solvable if C is NSE. In addition, under a definition of problem class, they show that the NSE property is also necessary for polynomial-time solvability. Here we point out two main differences between their results and ours. First and foremost, our analysis pinpoints the condition for feasibility of aggregation on legs, thus identifying the relation between our problem and traditional network RM problems, while their analysis does not touch upon such a relation. Such aggregation idea has not appeared in literature, and is crucial to deal with the situation where there is a large demand for the same type of tickets, and paves the way for our further discussion on equivalent formulations and approximation algorithms. Second, their algorithm is rather complex, where the main idea is to construct many dummy requests and reduce them to a *Maximum Fixed Job Scheduling* problem where all the seats have exactly the same endpoints, while our algorithm is based on solving a network flow problem and directly carrying out the assignment. Therefore, our result improves over theirs both in terms of the insight of the problem and the practicality of the algorithm.

4. The Dynamic Problem

Having analyzed the static problem, we now focus on the dynamic allocation problem. We start from a booking limit control policy and analyze its properties. Then we focus on a bid-price control policy with nonlinear bid-prices and flexible design. Finally, building on re-solving a delicate LP enlightened from former discussions, we arrive at a policy that achieves a uniformly bounded loss. There are two reasons for focusing on these three types of policies. First, these three types of

control policies are classical ones in the network revenue management literature and have been well studied (see, e.g., Gallego and van Ryzin 1997, Talluri and van Ryzin 1998, and Reiman and Wang 2008). Thus it is of interest to study whether the results in previous literature still hold in our problem, and if so, how they can overcome the "assign-to-seat" challenge. Second, these policies are implementable in practice. In the discussion of these policies, we will also show the connections of these approaches to the approximate dynamic programming approach.

To evaluate the performance of any policy, we use the notion of *asymptotic loss*, which is a common criterion in the revenue management literature (see, e.g., Gallego and van Ryzin 1997, Talluri and van Ryzin 1998). In our case, given any initial capacity matrix C, arrival rate $\{\lambda_{ij}^t\}_{i\leqslant j}$, we define a sequence of problems indexed by $\theta\in\mathbb{N}$ with $C(\theta)\in\{0,1\}^{\theta N\times M}$ and $\{\lambda_{ij}^t(\theta)\}_{i\leqslant j}^{\theta T}$ satisfying

$$C(\theta)_{(k+(s-1)N)\ell}=C_{k\ell},\quad \forall s\in[\theta],k\in[N],\ell\in[M],\quad \text{and}\quad \lambda_{ij}^t(\theta)=\lambda_{ij}^{\lceil t/\theta\rceil},\quad \forall t\in[\theta T],i\leqslant j.$$

That is, we scale the number of seats (with the same profile) and the arrival rates proportionally.

For any policy π, we are interested in the gap between the expected revenue collected under policy π in problem θ, denoted by $V_\theta^\pi(C)$ and the maximal expected revenue $V_{\theta T}(C(\theta))$ (a formal definition of $V_t(\cdot)$ is given in Section 4.2, and here we assume such a maximum exists). We call the gap the *loss* of policy π, denoted by

$$L_\theta^\pi(C)\triangleq V_{\theta T}(C(\theta))-V_\theta^\pi(C).$$

4.1. Booking Limit Control Policy

Booking limit control policy is a simple yet popular method in network revenue management. In this policy, we replace the real demand by the expected one and solve the corresponding static problem using the expected demand. Then for every type of requests, we only allocate a fixed amount according to the static solution and reject all other exceeding requests. In our case, solving the static problem in general could be challenging (because of the NP-Hardness of the problem, see Theorem 1), thus we must consider some approximation algorithms instead.

In standard approximation algorithms, the rounding errors are typically multiplicative (approximation ratio) rather than additive[2]. However, a general approximation ratio is not enough for our problem since we aim at near-optimal asymptotic performance. By considering aggregation of requests and carrying out a different analysis based on the NSE structure we identified before, we can solve a condensed formulation (4) such that a simple rounding incurs only a bounded *additive loss*.

[2] As an example, in Bhatia et al. (2007), they obtain a $1-1/e$ approximation guarantee for a similar problem to ours. If translated into our setting, their problem is to maximize the revenue under general capacity matrices and prices, but under $d_{ij}\in\{0,1\}$.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i\leqslant j}\left(p_{ij}\sum_{u:u\leqslant i}\xi_{uij}\right), && && (4)\\
\text{subject to} \quad & \sum_{u:u\leqslant i}\xi_{uij}\leqslant d_{ij}, && \forall 1\leqslant i\leqslant j\leqslant M,\\
& \sum_{(i,j):u\leqslant i\leqslant \ell\leqslant j}\xi_{uij}\leqslant m_{u\ell}, && \forall 1\leqslant u\leqslant \ell\leqslant M,\\
& \xi_{uij}\geqslant 0, && \forall 1\leqslant u\leqslant i\leqslant j\leqslant M,
\end{aligned}
$$

In (4), we classify all maximal sequences into M groups based on their starting legs. Here, ξ_{uij} represents the number of requests of type $i\to j$ that are allocated to group u, i.e., $\xi_{uij}=\sum_k x_{k,ij}\mathbb{1}\{\exists v:[u,v]\sim C_k, u\leqslant i\leqslant j\leqslant v\}$ and $m_{u\ell}=\sum_{v\geqslant\ell}|\mathcal{M}_{uv}(C)|$ is the number of maximal sequences that start with u and include ℓ in C. We first have the following equivalence between (2) and (4), the proof of which can be found in EC.1.

Lemma 1. *For any nonnegative numbers $\{d_{ij}\}$, (2) and (4) have the same optimal value.*

We should note that in (4), both the number of variables and constraints are $O(M^3)$. Thus, solving them are much more efficient compared to directly solving (1) or (2), especially when $M\ll N$, which is the common case in the context of high-speed train (for example, the train from Beijing to Shanghai usually has less than 10 legs, while it usually has more than 1000 seats in total). To be more precise, we list the number of variables and constraints in Table 1 for comparison.

Problem	Number of Variables	Number of Constraints
(1), (2)	$O(NM^2)$	$O(NM+M^2)$
(3)	$O(M^2)$	$O(M^2)$
(4)	$O(M^3)$	$O(M^2)$

Table 1 Comparison between different formulations

Next, we describe the static booking limit control (SBLC) policy in Algorithm 1, and establish the asymptotic optimality of SBLC in Theorem 3.

Theorem 3. *For any initial capacity matrix $C\in\{0,1\}^{N\times M}$, any set of prices $\{p_{ij}\}_{i\leqslant j}$ and arrival rates $\{\lambda_{ij}^t\}_{i\leqslant j}$, we have $L_\theta^{SBLC}(C)=O(\sqrt{\theta})$.*

The proof of Theorem 3 can be found in EC.2. Note that the constants in the bound only depend on $\{p_{ij}\}_{i\leqslant j}$ and $\{\lambda_{ij}^t\}_{i\leqslant j}$, but do not depend on C. Theorem 3 shows that although simple, the static booking limit control policy is asymptotically optimal, with an asymptotic loss of order $\sqrt{\theta}$. Note that the order of the loss is the same as that in the traditional network RM problem (Gallego

Algorithm 1: Static Booking Limit Control (SBLC)

Solve (4) with $\{d_{ij}\} = \{\lfloor \lambda_{ij}^{[1,T]} \rfloor\}$ and obtain an optimal solution $\{\xi_{uij}^*\}$;
Initialize $\{x_{k,ij}\} = \{0\}$;
for $u = 1, ..., M$ **do**
  for $j = M, ..., u$ **do**
    for $i = u, ..., j$ **do**
      Let $\mathcal{K}_{uij}$ be a set of $\lfloor \xi_{uij}^* \rfloor$ seats with leg u to j unoccupied. Set $x_{kij} = 1, \forall k \in \mathcal{K}_{uij}$.
    end
  end
end
for *any request of type* $i \to j$ **do**
  if $\exists k : x_{k,ij} = 1$ **then** Allocate the request to seat k and set $x_{k,ij} = 0$;
  else Reject the request;
end

and van Ryzin 1997). Therefore, the classical results about the booking limit control policy in the network RM setting can be extended to our setting.

Despite the simplicity and asymptotic optimal performance of the SBLC, for small- or medium-sized problems, it is quite restrictive on the allowable allocation and thus the performance could be suboptimal. We will illustrate this in the numerical experiments in Section 5. In the following sections, we consider two other control policies from a maximal sequence perspective, which enjoy more flexibility in allocation and better performance in theory/practice.

4.2. Bid-Price Control Policies

In the network RM literature, another popular class of policy is the *bid-price control policy*. In this policy, at each time period, each resource is associated with a *bid-price* which captures a fair value of the resource at that time, and the allocation is made based on the bid-prices. Bid-price control policies are flexible (one can use different ways to calculate the bid-prices), and are also easy to implement in practice (after the bid-prices are calculated, the allocation is often very simple). In this section, we investigate bid-price control policies for our problem. We propose two different ways to dynamically compute bid-prices, one based on the dual formulation of the static model, the other based on a more delicate formulation of the problem by considering the longest consecutive available segments, i.e., *maximal sequence*, of the seats. We will show how our new bid-prices are related with (approximate) dynamic programming approaches and can be applied to make dynamic capacity allocation decisions.

4.2.1. Bid-Price based on the Static Model In this section, we propose a bid-price control policy based on the static model (2). We first consider the dual problem of (2) with d_{ij} replaced

by the expected total demands $\lambda_{ij} = \sum_t \lambda_{ij}^t$. Let z_{ij} be the dual variables for the first constraint and $\beta_{k\ell}$ be the dual variables for the second constraint. The dual problem of (2) can be written as follows:

$$\begin{aligned} \text{minimize}_{z,\beta} \quad & \sum_{1\leqslant i\leqslant j\leqslant M} \lambda_{ij} z_{ij} + \sum_{k=1}^{N}\sum_{\ell=1}^{M} C_{k\ell}\beta_{k\ell} && (5)\\ \text{subject to} \quad & z_{ij} + \sum_{\ell=i}^{j} \beta_{k\ell} \geqslant p_{ij}, && \forall 1\leqslant i\leqslant j\leqslant M, k\in[N],\\ & z_{ij} \geqslant 0, && \forall 1\leqslant i\leqslant j\leqslant M,\\ & \beta_{k\ell} \geqslant 0, && \forall k\in[N], \ell\in[M]. \end{aligned}$$

Here $\beta_{k\ell}$ can be interpreted as the static bid-price for the kth seat on the ℓth leg. When a request $i \to j$ arrives, we can calculate $p_{ij} - \sum_{\ell=i}^{j}\beta_{k\ell}$ for all k and choose $\text{argmax}_k\{p_{ij} - \sum_{\ell=i}^{j}\beta_{k\ell}\}$ as the seat to allocate the request. Moreover, in practice, one can re-solve (5) to obtain time-dependent bid-prices $\{\beta_{k\ell}^t\}$, and use $\beta_{k\ell}^t$ instead of $\beta_{k\ell}$. The bid-price control policy based on the static model (BPC-S) is formally stated in Algorithm 2.

Algorithm 2: Bid-Price Control with Static Model (BPC-S)

for $t = 1, ..., T$ **do**
  Observe a request of type $i^t \to j^t$;
  if $\{k|\mathbf{e}_{ij} \leqslant C_k^t\} = \emptyset$ **then** Reject the request.;
  else
    Solve (5) with $C = C^t, \lambda_{ij} = \lambda_{ij}^{[t,T]}$ and obtain an optimal solution $\beta^t = \{\beta_{k\ell}^t\}$.
    Set $k_t = \arg\max_{k\in[N]}\{p_{ij} - \sum_{\ell=i}^{j}\beta_{k\ell}^t | \mathbf{e}_{ij} \leqslant C_k^t\}$. Break ties arbitrarily;
    if $p_{i^t j^t} - \sum_{\ell=i}^{j}\beta_{k_t\ell}^t \geqslant 0$ **then**
      Allocate the request to seat k_t;
      Let $C^{t+1} \leftarrow C^t - \mathbf{e}_{k_t}^\top \mathbf{e}_{ij}$;
      else Reject the request, let $C^{t+1} = C^t$;
    end
  end
end

In the network RM literature, it has been shown that the bid-prices can also be derived from an approximate dynamic programming (ADP) point of view, see, e.g., Adelman (2007), Topaloglu (2009). In our setting, the relation still holds, and we explain it in the following.

To obtain the connection, we first establish a dynamic programming (DP) formulation for the dynamic allocation problem as follows. Recall that $V_t(C)$ is the maximal expected value to go at

the beginning of period t and when the capacity matrix is C. Then the recursive formula for $V_t(C)$ can be written as follows:

$$V_t(C) = \max_{u^t \in \mathcal{D}^t(C)} \left\{ \sum_{i \leqslant j} \lambda_{ij}^t \left(\sum_{k=1}^{N} p_{ij} u_{k,ij}^t + V_{t+1}(C - \sum_{k=1}^{N} u_{k,ij}^t \mathbf{e}_k^\top \mathbf{e}_{ij}) \right) + \lambda_0^t V_{t+1}(C) \right\}, \quad \forall t \in [T], C \geqslant 0, \tag{6}$$

$$V_{T+1}(C) = 0, \quad \forall C \geqslant 0,$$

where $\mathcal{D}^t(C) = \{u_{k,ij}^t \in \{0,1\}, \forall i,j,k | \sum_{k=1}^N u_{k,ij}^t \leq 1, \quad \forall i,j, \text{ and } u_{k,ij} \mathbf{e}_k^\top \mathbf{e}_{ij} \leqslant C, \quad \forall i,j,k\}$.

We can also rewrite (6) in a more compact form as follows,

$$\begin{aligned} V_t(C) &= \mathbb{E}_{(i,j)\sim\lambda^t} \left[\max_{\substack{k \in [N]: \\ C_k \geqslant \mathbf{e}_{ij}}} \left\{ V_{t+1}(C - \mathbf{e}_k^\top \mathbf{e}_{ij}) + p_{ij}, V_{t+1}(C) \right\} \right], && \forall t \in [T], C \geqslant 0, \\ V_{T+1}(C) &= 0, && \forall C \geqslant 0, \end{aligned} \tag{7}$$

where $(i,j) \sim \lambda^t$ stands for the probability distribution of the incoming request in time period t. For notational brevity, we assume that there is a probability λ_0^t such that $(i,j) = (0,0)$, in which case $\{k \in [N] | C_k \geq \mathbf{e}_{ij}\} = \emptyset$. Following the ADP method in Adelman (2007), we approximate $V_t(C)$ as

$$V_t(C) \approx \theta_t + \sum_{k=1}^{N} \sum_{\ell=1}^{M} C_{k\ell} \beta_{k\ell} \triangleq \tilde{V}_t(C), \tag{8}$$

where $\beta_{k\ell}$ can be interpreted as the *static bid price* for the kth seat on the ℓth leg. To compute the tightest approximation for $V_t(C)$, one can solve the following problem:

$$\begin{aligned} \text{minimize} \quad & \tilde{V}_t(C) && && (9) \\ \text{subject to} \quad & \tilde{V}_t(\bar{C}) \geqslant \mathbb{E}_{(i,j)\sim\lambda^t} \left[\max_{\substack{k \in [N]: \\ \bar{C}_k \geqslant \mathbf{e}_{ij}}} \left\{ \tilde{V}_{t+1}(\bar{C} - \mathbf{e}_k^\top \mathbf{e}_{ij}) + p_{ij}, \tilde{V}_{t+1}(\bar{C}) \right\} \right], && \forall t \in [T], C \geqslant \bar{C} \geqslant 0, \\ & \tilde{V}_{T+1}(\bar{C}) \geqslant 0, && \forall \bar{C} \geqslant 0. \end{aligned}$$

Plugging (8) into (9), setting $t = 0$ and $\theta_0 = 0$, we recover exactly (5). Also, (9) gives an upper bound on $V_t(C)$, which we denote as $\hat{V}_t(C)$. Thus, the bid-prices solved from (5) can also be viewed as the coefficients in the ADP approach.

4.2.2. Bid-Price based on the Maximal Sequence Formulation In this section, we consider an alternative DP formulation based on the idea of *maximal sequence*, and derive a different group of bid-prices and ultimately a different and stronger control policy. Before that, we introduce several additional notations. Let $\mathcal{A}$ denote the set of all $M \times M$ upper triangular matrices. For any capacity matrix C, we construct a projection $f : C \to \mathcal{A}$ as follows,

$$f(C)_{uv} = |\mathcal{M}_{uv}(C)|, \forall 1 \leqslant u \leqslant v \leqslant M, \tag{10}$$

i.e., the (u,v)th entry of $f(C)$ calculates the total number of $[u,v]\sim C$. Let $\mathcal{A}_{ij}$ be the *action set* defined as follows,

$$\mathcal{A}_{ij}=\left\{R\in\mathbb{R}^{M\times M}\;\middle|\;\exists u,v:u\leqslant i\leqslant j\leqslant v,\text{ s.t. }R_{u(i-1)}=-\mathbb{1}\{u<i\},R_{(j+1)v}=-\mathbb{1}\{j<v\},R_{uv}=1\right\}.$$

We now explain the definition of $\mathcal{A}_{ij}$. For any request $i\to j$, if we accept it, then we can only assign it to a seat with legs i to j unoccupied. If we assign the request to seat k with $[u,v]\sim C_k(u\leqslant i\leqslant j\leqslant v)$, then $[u,v]$ is split into $[u,i-1]\sim C_k$ and $[j+1,v]\sim C_k$. In other words, we consume one unit of $[u,v]\sim C_k$ while creating one unit of $[u,i-1]\sim C_k$ and $[j+1,v]\sim C_k$. That is why $R_{uv}=1$ but $R_{u(i-1)}=-\mathbb{1}\{u<i\}$ and $R_{(j+1)v}=-\mathbb{1}\{j<v\}$. We then track the dynamics of $\mathcal{M}(C)$ as in (11).

$$\begin{aligned}
&\mathcal{M}_{uv}(C)\leftarrow\mathcal{M}_{uv}(C)\setminus\{k\}, && f(C)_{uv}\leftarrow f(C)_{uv}-1,\\
&\mathcal{M}_{u(i-1)}(C)\leftarrow\mathcal{M}_{u(i-1)}(C)\cup\{k\}, && f(C)_{u(i-1)}\leftarrow f(C)_{u(i-1)}+1,\quad \text{if } u<i,\\
&\mathcal{M}_{(j+1)v}(C)\leftarrow\mathcal{M}_{(j+1)v}(C)\cup\{k\}, && f(C)_{(j+1)v}\leftarrow f(C)_{(j+1)v}+1,\quad \text{if } v>j.
\end{aligned}\tag{11}$$

Now we are ready to present the alternative DP formulation. For any $A\in\mathcal{A}$, consider the following DP,

$$\begin{aligned}
&V_t^\dagger(A)=\mathbb{E}_{(i,j)\sim\lambda^t}\left[\max_{\substack{R\in\mathcal{A}_{ij}\\ R\leqslant A}}\left\{V_{t+1}^\dagger(A-R)+p_{ij},V_{t+1}^\dagger(A)\right\}\right], && \forall t\in[T],A\geqslant 0,\\
&V_{T+1}^\dagger(A)=0, && \forall A\geqslant 0.
\end{aligned}\tag{12}$$

Here the meaning of the notation $(i,j)\sim\lambda^t$ is the same as in (7). The equivalence between (7) and (12) is established in Proposition 1.

PROPOSITION 1. *If $V_t(\cdot)$ is defined by (7), and $V_t^\dagger(\cdot)$ is defined by (12), then we have*

$$V_t(C)=V_t^\dagger(f(C)),\quad\forall C\geqslant 0,t\in[T],\tag{13}$$

where f is defined in (10).

Proposition 1 claims that aggregation based on *maximal sequence* is sufficient for tracking the dynamics of the state, or in other words, (12) is a valid dynamic formulation for the original problem. Though we need $\{\mathcal{M}_{uv}(C)\}$ to characterize the overall information, the value function is only concerned with the number of each type of maximal sequence. Note that an advantage of formulation (12) compared with formulation (7) is that the state space is greatly reduced. Particularly, the dimension of the state space of (7) is NM, while the dimension of the state space of (12) is only $\binom{M+1}{2}$. As we mentioned earlier, M is much less than N in our context. Later, we will show that indeed formulation (12) could lead to policies that are both computationally more efficient and also of higher quality.

Even though (12) has a smaller state space, directly solving it is still computationally prohibitive. Now we again adopt the ADP approach. We approximate $V_t^\dagger(A)$ as

$$V_t^\dagger(A) \approx \theta_t^\dagger + \sum_{u \leqslant v} A_{uv}\beta_{uv}^\dagger \triangleq \tilde{V}_t^\dagger(A). \tag{14}$$

Here, we can view $\beta_{uv}^\dagger$ as the approximated value for each maximal sequence $[u,v] \sim C$, or in our context, the value of having an additional empty (unoccupied) seat from leg u to leg v. To compute the tightest approximation at A, one can solve:

$$\begin{aligned} \text{minimize} \quad & \tilde{V}_t^\dagger(A) && && (15)\\ \text{subject to} \quad & \tilde{V}_t^\dagger(\bar{A}) \geqslant \mathbb{E}_{(i,j)\sim\lambda^t}\left[\max_{\substack{R\in\mathcal{A}_{ij}\\ R\leqslant\bar{A}}}\left\{\tilde{V}_{t+1}^\dagger(\bar{A}-R)+p_{ij}, \tilde{V}_{t+1}^\dagger(\bar{A})\right\}\right], && \forall t\in[T], \bar{A}\geqslant 0,\\ & \tilde{V}_{T+1}^\dagger(\bar{A}) \geqslant 0, && \forall \bar{A}\geqslant 0. \end{aligned}$$

Plugging (14) into (15), setting $t=0$ and $\theta_0^\dagger = 0$, and organizing terms yield

$$\begin{aligned} \text{minimize}_{\beta^\dagger, z^\dagger} \quad & \sum_{i\leqslant j}\lambda_{ij}z_{ij}^\dagger + \sum_{u\leqslant v}A_{uv}\beta_{uv}^\dagger && && (16)\\ \text{subject to} \quad & z_{ij}^\dagger + \beta_{uv}^\dagger \geqslant p_{ij} + \beta_{u(i-1)}^\dagger + \beta_{(j+1)v}^\dagger, && \forall u\leqslant i\leqslant j\leqslant v,\\ & z_{ij}^\dagger \geqslant 0, \beta_{ij}^\dagger \geqslant 0, && \forall i\leqslant j,\\ & \beta_{ij}^\dagger = 0, && \forall i > j. \end{aligned}$$

Note that (15) gives an upper bound on $V_t^\dagger(A)$, which we denote as $\hat{V}_t^\dagger(A)$. Once we obtain a group of bid-prices $\{\beta_{uv}^\dagger\}$ from (16), we can then make decisions based on them. Moreover, in practice, we can re-solve (16) to obtain time-dependent bid-prices $\{\beta_{uv}^{\dagger t}\}$. We formally state the algorithm in Algorithm 3.

Now that we have introduced two types of bid-prices, then is there any relation between them? The following theorem shows that there is advantage in the maximal sequence approach, where bid-prices are allowed to be non-linear, in terms of approximating the value function.

THEOREM 4. *For any $t\in[T], C\in\{0,1\}^{N\times M}$ and any group of bid prices $\{\beta_{k\ell}^t\}$ in (5), there exists a group of bid prices $\{\beta_{uv}^{\dagger t}\}$ in (16), such that*

$$\beta_{uv}^{\dagger t} = \min_{k\in[N]}\left\{\sum_{\ell: u\leqslant\ell\leqslant v}\beta_{k\ell}^t\right\}, \quad \forall u\leqslant v. \tag{17}$$

The proof of Theorem 4 is relegated to EC.2, where we establish that by maintaining $z^\dagger = z$, (17) is both feasible and optimal. Thus, it shows that when using the bid-prices at an identical state to estimate the value functions, BPC-M could lead to a lower approximation of the value function

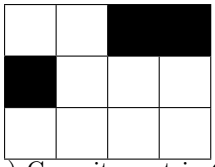
Algorithm 3: Bid-Price Control with Maximal Sequence (BPC-M)

for $t=1,...,T$ **do**
  Observe a request of type $i^t \to j^t$;
  if $\{(u,v)|u \leqslant i \leqslant j \leqslant v, \mathcal{M}_{uv}(C^t) \neq \emptyset\} = \emptyset$ **then** Reject the request;
  else
    Solve (16) with $A = f(C^t), \lambda_{ij} = \lambda_{ij}^{[t,T]}$ and obtain an optimal solution $\{\beta_{uv}^{\dagger t}\}$;
    Set $(u^t, v^t) = \arg\max_{(u,v)}\{p_{i^t j^t} + \beta_{u(i^t-1)}^{\dagger t} + \beta_{(j^t+1)v}^{\dagger t} - \beta_{uv}^{\dagger t} | \mathcal{M}_{uv}(C) \neq \emptyset\}$. Break ties arbitrarily;
    if $p_{i^t j^t} + \beta_{u^t(i^t-1)}^{\dagger t} + \beta_{(j^t+1)v^t}^{\dagger t} - \beta_{u^t v^t}^{\dagger t} \geqslant 0$ **then**
      Allocate the request to seat $k \in \mathcal{M}_{u^t v^t}(C^t)$;
      Update $\{\mathcal{M}_{uv}(C^t)\}$ according to (11);
      else Reject the request;
    end
  end
end

than BPC-S under *any* (other) capacity matrix C'. Remember that both bid-price approaches give an upper bound on the value function at any state, thus it further implies that BPC-M gives a more accurate approximation for the value function than the static approach (BPC-S).

To explain further, the reason that BPC-M can give a more accurate approximation is because it allows for *non-linear* bid-prices. Note that for any fixed t, $\beta_{k\ell}^t$ only depends on k and ℓ, thus the value of consecutive seats must follow a linear structure with respective to legs, while BPC-M relaxes such a restriction, allowing a more flexible structure for the values. In the following, we give an example where BPC-M provides more information for allocation than BPC-S does. A more concrete example where BPC-M rejects a particular request which BPC-S accepts is given in EC.2.

Example 2. Consider the capacity matrix in Figure 2(a) in period 1. We construct the prices as $p_{ij} = 10(j-i+1)$, and $\{\lambda_{ij}^{1+}\}$ as $\lambda_{12}^{1+} = \lambda_{22}^{1+} = \lambda_{24}^{1+} = \lambda_{44}^{1+} = 0, \lambda_{11}^{1+} = \lambda_{13}^{1+} = \lambda_{23}^{1+} = \lambda_{33}^{1+} = \lambda_{34}^{1+} = 1, \lambda_{14}^{1+} = 2$. A group of bid-prices $\{\beta_{k\ell}^1\}$ obtained by solving (5) are presented in Figure 2(b). The bid-prices obtained by solving (16) are as follows: $\beta_{11}^{\dagger 1} = \beta_{12}^{\dagger 1} = \beta_{22}^{\dagger 1} = \beta_{44}^{\dagger 1} = 0, \beta_{23}^{\dagger 1} = \beta_{24}^{\dagger 1} = \beta_{33}^{\dagger 1} = \beta_{34}^{\dagger 1} = 20, \beta_{13}^{\dagger 1} = \beta_{14}^{\dagger 1} = 40$. Note that here $\beta_{uv}^{\dagger 1} = \min_{k\in[N]} \sum_{\ell=u}^{v} \beta_{k\ell}^1$ always hold.



(a) Capacity matrix C

0	0	40	0
20	0	20	0
20	0	20	0

(b) Bid-prices computed from (5), i.e., $\{\beta_{k\ell}\}$

Figure 2 Illustrations of Example 2

Now assume that in period 1, a request of type $2 \to 3$ comes. If we use $\{\beta^1_{k\ell}\}$, then we will *accept* the request and allocate it to one of Seat 2 and Seat 3, but we cannot determine which seat to allocate. Meanwhile, if we apply $\{\beta^{\dagger 1}_{uv}\}$, we will allocate the request to Seat 2. In fact, for Seat 2, we have $\beta^{\dagger 1}_{24} - \beta^{\dagger 1}_{44} = 20 = p_{23}$; for Seat 3, we have $\beta^{\dagger 1}_{14} - \beta^{\dagger 1}_{11} - \beta^{\dagger}_{44} = 40 > p_{23}$. □

4.3. Re-solving a Dynamic Primal: Uniformly Bounded Loss

So far, we have proposed two types of policies, both of which have their advantages as well as drawbacks. SBLC makes good use of the NSE structure and achieves asymptotic optimality, but it stems from a purely *static* view. BPC-M better captures the dynamic change of the state space and has more flexibility, but bid-prices are established on "dual" that might lose some information contained in the primal problem. In online decision-making literature, re-solving a primal problem is a simple yet powerful idea that sometimes leads to strong theoretical results (see, e.g., Jasin and Kumar 2012, Bumpensanti and Wang 2018, Vera and Banerjee 2019). In this section, we propose a policy that is based on re-solving a *dynamic primal* — the dual of (16) — to combine the advantages of the two afore-mentioned policies. Furthermore, the policy can be proven to achieve uniformly bounded loss asymptotically. All of the proofs can be founded in EC.2.

We first write down the dual of (16) as (18).

$$
\begin{aligned}
\max_{\gamma} \quad & \sum_{i \leqslant j} \left(p_{ij} \sum_{\substack{(u,v):\\ u \leqslant i \leqslant j \leqslant v}} \gamma_{uijv} \right), && && (18)\\
\text{s.t.} \quad & \sum_{\substack{(u,v):\\ u \leqslant i \leqslant j \leqslant v}} \gamma_{uijv} + \gamma_{0ij0} = d_{ij}, && && \forall i \leqslant j,\\
& \sum_{\substack{(i,j):\\ u \leqslant i \leqslant j \leqslant v}} \gamma_{uijv} \leqslant \sum_{\substack{(k,\ell):\\ v+1 \leqslant k \leqslant \ell}} \gamma_{u(v+1)k\ell} + \sum_{\substack{(k,\ell):\\ \ell \leqslant k \leqslant u-1}} \gamma_{\ell k(u-1)v} + A_{uv}, && && \forall u \leqslant v,\\
& \gamma_{uijv} \geqslant 0, \quad \forall u \leqslant i \leqslant j \leqslant v, && \gamma_{0ij0} \geqslant 0, \quad \forall i \leqslant j.
\end{aligned}
$$

We call (18) the *dynamic primal* model. It is totally based on the maximal sequence formulation. To explain, the variable γ_{uijv} can be interpreted as the count of times when request $i \to j$ is allocated into a $[u, v] \sim C$, and we explicitly use γ_{0ij0} to denote the number of request $i \to j$ we reject.

For any sample path ω, we adopt the following notations:

- $A^t_{uv} = f(C^t)_{uv}$: The number of maximal sequence $[u, v]$ at time t, prior to our decision. $A^{[T+1]}_{uv}$ means the number of remaining $[u, v]$ after the booking horizon ends.
- $\gamma^{[t_1,t_2]}_{uijv}$: The number of request $i \to j$ put into $[u, v]$ in $[t_1, t_2]$ by our algorithm (soon to show).

Note that $[t_1, t_2] = [t_1, t_2 + 1)$. In the following analysis, we will drop ω for simplicity. We also let $d^{[t,t)} = \gamma^{[t,t)} = 0$ for all $t \geqslant 1$.

To facilitate our demonstration, we introduce the following LP. Denote $OPT(A,d,\hat{\gamma})$ as the problem instance/objective value of (19),

$$\begin{aligned}
\max_{\gamma} \quad & \sum_{i\leqslant j}\left(p_{ij}\sum_{\substack{(u,v):\\ u\leqslant i\leqslant j\leqslant v}}\gamma_{uijv}\right), && && (19)\\
\text{s.t.} \quad & \sum_{\substack{(u,v):\\ u\leqslant i\leqslant j\leqslant v}}\gamma_{uijv}+\gamma_{0ij0}=d_{ij}, && \forall i\leqslant j,\\
& \sum_{\substack{(i,j):\\ u\leqslant i\leqslant j\leqslant v}}\gamma_{uijv}\leqslant\sum_{\substack{(k,\ell):\\ v+1\leqslant k\leqslant \ell}}\gamma_{u(v+1)k\ell}+\sum_{\substack{(k,\ell):\\ \ell\leqslant k\leqslant u-1}}\gamma_{\ell k(u-1)v}+A_{uv}, && \forall u\leqslant v,\\
& \gamma_{uijv}\geqslant\hat{\gamma}_{uijv},\quad \forall u\leqslant i\leqslant j\leqslant v, \qquad \gamma_{0ij0}\geqslant\hat{\gamma}_{0ij0},\quad \forall i\leqslant j,
\end{aligned}$$

where $A=\{A_{uv}\}$, $d=\{d_{ij}\}$, and $\hat{\gamma}=\{\hat{\gamma}_{uijv}\}$. It is easy to see that (18) is the same as $OPT(A,d,0)$.

In the following, we introduce the Re-solving a Dynamic Primal (RDP) policy. In the RDP policy, each time, we solve (18) with additional constraints and carry out the decision. The detailed policy is shown in Algorithm 4.

Algorithm 4: Re-solving a Dynamic Primal (RDP)

for $t=1,...,T$ **do**
- Observe a request of type $i^t\to j^t$;
- Solve (18) with $A^t=f(C^t)$ and $d=\lambda^{[t,T]}$, plus the following constraints

$$\gamma_{ui^tj^tv}\leqslant A^t_{uv},\quad \forall(u,v): u\leqslant i^t\leqslant j^t\leqslant v \tag{20}$$

  and obtain an optimal solution $\{\gamma^{\mathrm{RDP},t}\}$;
- Set $(u^t,v^t)=\arg\max_{(u,v)}\{\gamma^{\mathrm{RDP},t}_{ui^tj^tv}\}$. Break ties arbitrarily;
- Allocate $i^t\to j^t$ to $[u^t,v^t]$ ($[u^t,v^t]=[0,0]$ means that we reject the request);

end

To explain Algorithm 4, we re-solve the primal problem (18) at each time period using the expected remaining demand. We add (20) to ensure the feasibility of potential assignment. Then we assign the request to the maximal sequence that has the maximum number of assignments in the primal solution for the corresponding itinerary. To analyze the performance of Algorithm 4, we first introduce the following crucial lemma, which shows adding constraints (20) guarantees the feasibility of allocation, but surprisingly does not change the objective value of (18).

Lemma 2. *Optimization problem (18) has the same objective value with (2). Furthermore, (18) always has an optimal solution that satisfies (20).*

In the following, we analyze the asymptotic performance of RDP. We startR with a lemma that shows the connection between different OPTs.

LEMMA 3. *For any $d \geqslant 0$ and $1 \leqslant t_1 \leqslant t_2 \leqslant T+1$,*

$$OPT(A^1, d + d^{[1,t_2)}, \gamma^{[1,t_2)}) = \sum_{i \leqslant j} \left(p_{ij} \sum_{\substack{(u,v): \\ u \leqslant i \leqslant j \leqslant v}} \gamma_{uijv}^{[1,t_1)} \right) + OPT(A^{t_1}, d + d^{[t_1,t_2)}, \gamma^{[t_1,t_2)}).$$

From Lemmas 2 and 3, for any sample path ω, the regret incurred by RDP can be upper bounded by

$$\begin{aligned} & OPT\left(A^1, d^{[1,T]}, 0\right) - OPT\left(A^1, d^{[1,T]}, \gamma^{[1,T]}\right) \\ = & \sum_{t=1}^{T} \left[OPT\left(A^1, d^{[1,T]}, \gamma^{[1,t)}\right) - OPT\left(A^1, d^{[1,T]}, \gamma^{[1,t+1)}\right) \right]. \end{aligned} \tag{21}$$

Next, we give two lemmas that are critical to establish our main results. Lemma 4 shows that the gap between "adjacent" OPTs can be uniformly bounded above. Though Lemma 4 shares some similar idea with the analysis in Freund and Banerjee (2019), our proof is based on the unique structure of the dynamic primal (18).

LEMMA 4. *There is some $l > 0$ only dependent on $\{p_{ij}\}$ such that*

$$0 \leqslant OPT(A^1, d^{[1,T]}, \gamma^{[1,t)}) - OPT(A^1, d^{[1,T]}, \gamma^{[1,t+1)}) \leqslant 2l$$

for any A^1, $d^{[1,T]}$ and $t \in [1,T]$.

Lemma 5, which is a special case of Theorem 2.4 in Mangasarian and Shiau (1987), indicates the sensitivity of optimal solutions when the right-hand side of (18) changes.

LEMMA 5. *There exists some $\delta > 0$ that only depends on the constraint matrix of (18), independent of A^t, $d \geqslant 0$, such that for any optimal solution $\gamma^{*,t}$ of $OPT(A^t, d, 0)$, there exists an optimal solution $\tilde{\gamma}^{*,t}$ of $OPT(A^t, \tilde{d}, 0)$ such that $\|\gamma^{*,t} - \tilde{\gamma}^{*,t}\|_\infty \leqslant \delta \|d - \tilde{d}\|_\infty$.*

Now we are ready to introduce our main theorem in this section: RDP incurs a uniformly bounded loss asymptotically for any given input parameters under a mild assumption.

THEOREM 5. *For any initial capacity matrix $C \in \{0,1\}^{N \times M}$, any set of prices $\{p_{ij}\}_{i \leqslant j}$ and arrival rates $\{\lambda_{ij}^t\}_{i \leqslant j, t \in [T]}$, if*

$$\inf_{\substack{i \leqslant j: \\ \lambda_{ij}^{[1,T]} > 0}} \lambda_{ij}^T > 0, \tag{22}$$

then $L_\theta^{RDP}(C) = O(1)$.

Note that the condition in (22) depends only on the base problem but not the scaling parameter θ. Now we provide some explanation for the assumption (22). It states that, for each possible type of request ($\lambda_{ij}^{[1,T]} > 0$), there must be a positive probability that such request comes in the last period ($\lambda_{ij}^{T} > 0$). This assumption is reasonable in practice, since customers often purchase tickets in the last minutes when the selling horizon approaches to its end. In particular, when the demand process is stationary, (22) is automatically satisfied. Therefore, under such mild condition, the RDP policy achieves a constant regret. As we will show later in our numerical experiments, the RDP policy achieves superior performance especially when the problem size is large.

5. Numerical Experiment

In this section, we conduct numerical experiments to test the performance of the dynamic policies proposed in Section 4 using both synthetic and real data. In particular, we consider the static booking limit control (SBLC) policy proposed in Section 4.1, the bid-price control policy with static model (BPC-S) proposed in Section 4.2.1, the bid-price control policy with maximal sequence (BPC-M) proposed in Section 4.2.2, and the Re-solving a Dynamic Primal policy (RDP) proposed in Section 4.3. We start by describing the settings and some implementation details in our experiments.

5.1. Settings and Implementation Details

Synthetic Data In the numerical tests with synthetic data, our goal is to testify the theoretical results in Section 4 numerically. For this part, we fix $M = 6$ and consider seven groups of parameters with $T = 5N$ and $N \in \{100, 200, 500, 1000, 2000, 5000, 10000\}$. In each parameter setting, the prices of each itinerary are chosen as $p_{ij} = \lfloor 10 \times (j - i + 1)^{4/5} \rfloor$. The choice reflects that the true prices in practice are usually subadditive in the distance between two stops. For arrival probability, we assume that in each time period, there is a probability $\lambda_0 = 0.2$ that no passenger arrives, and we consider two different cases regarding the arrival probability of each type of passengers:

- Case 1: Homogeneous arrival: In this case, $\lambda_{ij}^t \propto 1$, which means all itineraries arrive with equal probability.
- Case 2: Inhomogeneous arrival with shorter itineraries arriving first: In this case, we partition the whole time horizon into M episodes, with the sth episode ($s = 1, ..., M$) being $(\lfloor (s-1)T/M \rfloor, \lfloor sT/M \rfloor]$. In episode s, each request with length s arrives with equal probability $0.5/(M + 1 - s)$, and all other types of requests arrive with equal but lower probability $0.3/(M(M+1)/2 - M + s - 1)$(remember there is a probability $\lambda_0 = 0.2$ that no passenger arrives).

We note that Case 2 is intuitively a more challenging case because if longer requests come first, then there will not be many choices for different assignment, and subsequent shorter requests can be easily fit into the remaining maximal sequences. However, if shorter requests come first, then

different policies may result in different assignments which could more likely lead to inefficient usage of resources.

In all of our tests, we assume that the starting capacity matrix $C = 1^{N\times M}$ and fix the re-solving frequency f as once a time. We run 100 simulated sample paths and record the average revenue rev_π obtained from each policy π. We can then calculate the loss as $L_\pi = \text{rev}_{\text{OP}} - \text{rev}_\pi$.

Real Data In practice, the decision maker may never know what true arrival probabilities are, and thus we have to estimate them using past information. By using real data, we want to answer the following questions.

1. How do different policies perform in a setting close to reality?
2. How do estimation errors influence the performance?

Through a collaboration with the China Railway High-speed (CRH), we have access to the booking information from August to September in 2019 of train G315, which departs daily from Jinan at around 10:00 and arrives at Chongqing at around 22:30. At the beginning of the selling horizon, the train is always completely unoccupied. There are 13 intermediate stations along the route (i.e., $M = 14$ legs). The booking information is confined to second-class carriages, with a total number of approximately 1000 seats. The data consists of the booking time, the itinerary, and the assigned seat of each accepted request (however, the data does not contain any request that is not accepted).

To examine the performance of different policies in real settings, we regard the data in each day as a sample path of sequential requests. Note that for our proposed policies, it is sufficient to estimate $\lambda^{[t,T]}$ rather than λ^t itself. On day d, for any time t_d prior to the end of booking horizon T_d, we take average over the empirical intensities obtained from a set of days $S_d = \{d-1, d-2, d-7, d-14, d-21\}$ for estimating intensities on day d (such a choice reflects the considerations of recent days and same day-of-week in the past). More precisely, we have

$$\lambda_{d,ij}^{[t_d,T_d]} = \frac{1}{|S_d|}\sum_{d'\in S_d}\sum_{r=1}^{n_{d'}} \mathbb{1}\left\{\text{request } r \text{ is } i\to j\right\}\cdot \mathbb{1}\left\{\text{the timing of request } r \geqslant T_{d'} - T_d + t_d\right\}, \tag{23}$$

where on day d' there are $n_{d'}$ requests indexed by r. In our numerical experiments, we simply choose T_d as 22:30 on each day d. The advantage of the estimation procedure described above is that we do not need to explicitly cut the whole booking horizon into discrete time intervals. We note that our estimation is a rough one, and in practice, firms might be able to apply other useful information for more accurate forecasting.

To give insight on how estimation errors affect the performance, we also run the experiments using the real intensities (i.e., assuming we know the arrival patterns of that day in advance). That is, we set

$$\lambda_{d,ij}^{[t_d,T_d]} = \sum_{r=1}^{n_d} \mathbb{1}\left\{\text{request } r \text{ is } i\to j\right\}\cdot \mathbb{1}\left\{\text{the timing of request } r \geqslant t_d\right\} \tag{24}$$

In the numerical experiments with real data, we set the prices $\{p_{ij}\}$ to be the real prices for each itinerary (shown in EC.1). The initial capacity matrix C is chosen as $1^{N\times M}$ where $N \in \{800, 600, 400\}$. The choice of N characterizes different scarcity levels of the resource, with less seats meaning higher resource scarcity. Note that the true capacity of the train is close to 1000, so an N of $\{800, 600, 400\}$ roughly corresponds to a 20%, 40%, and 60% increase in the arrival intensity, respectively. Since the data only contains accepted requests, if we use the estimated intensity directly, then most requests can be accepted, and the difference of different policies will not be clear. Thus, it is reasonable to consider these shrank capacity settings. Our experiment starts from 08/26 (Monday) to 09/26 (Thursday), with a total number of 32 days[3]. We delete the last 4 days in September because they locate near the Chinese National Holidays (10/01 - 10/07), during which the demands exhibited a pattern very different from normal times.

Tie-Breaking Rule BPC and RDP policies may face "ties" when two or more different decisions lead to the same maximal value. We adopt the following tie-breaking rule in our implementation: Suppose in some period t, we are about to accept a request $i \to j$, and there are a set of available seats $\{k_\ell\}$ with maximal sequence $[u_\ell, v_\ell] \sim C_{k_\ell}$ as candidates (Line 6 in Algorithms 2 and 3 and Line 4 in Algorithm 4), where $u_\ell \leqslant i \leqslant j \leqslant v_\ell$. In this case, we select ℓ^* such that $u_{\ell^*} \geqslant u_\ell$ for all $\ell \neq \ell^*$, and if $u_\ell = u_{\ell^*}$ for some ℓ, then $v_\ell \geqslant v_{\ell^*}$. We assign the request to seat with maximal sequence $[u_{\ell^*}, v_{\ell^*}]$.[4] In addition, for RDP policy, we prefer accept to reject. In other words, this rule orders all maximal sequences that contain leg i to leg j as

$$[i,j] \prec \cdots \prec [i,M] \prec [i-1,j] \prec \cdots \prec [i-1,M] \prec \cdots \prec [1,j] \prec \cdots \prec [1,M] \prec [0,0],$$

and then selects the smallest one under this order when there is a tie.

5.2. Results and Interpretation

In addition to testing SBLC, BPC-S, BPC-M and RDP, we consider an oracle policy (OP), which only makes the allocation decision when all requests are known, therefore achieving the optimal value for the static model (1). Note that OP always achieves higher revenue than any other policies, serving as a benchmark for other policies. We also consider a myopic policy (MP), where we always accept a request as long as there are available seats and the allocation is decided by the above-described tie-breaking rule. We present our main findings as follows.

[3] We start from 08/26 because in (24) we need three weeks of data to estimate the intensities, and we select a Monday as the beginning of testing periods.

[4] It is easy to see that assigning a request to the same maximal sequence in different seats will give rise to the same revenue at the end. Therefore, the tie-breaking rule is on the maximal sequence level.

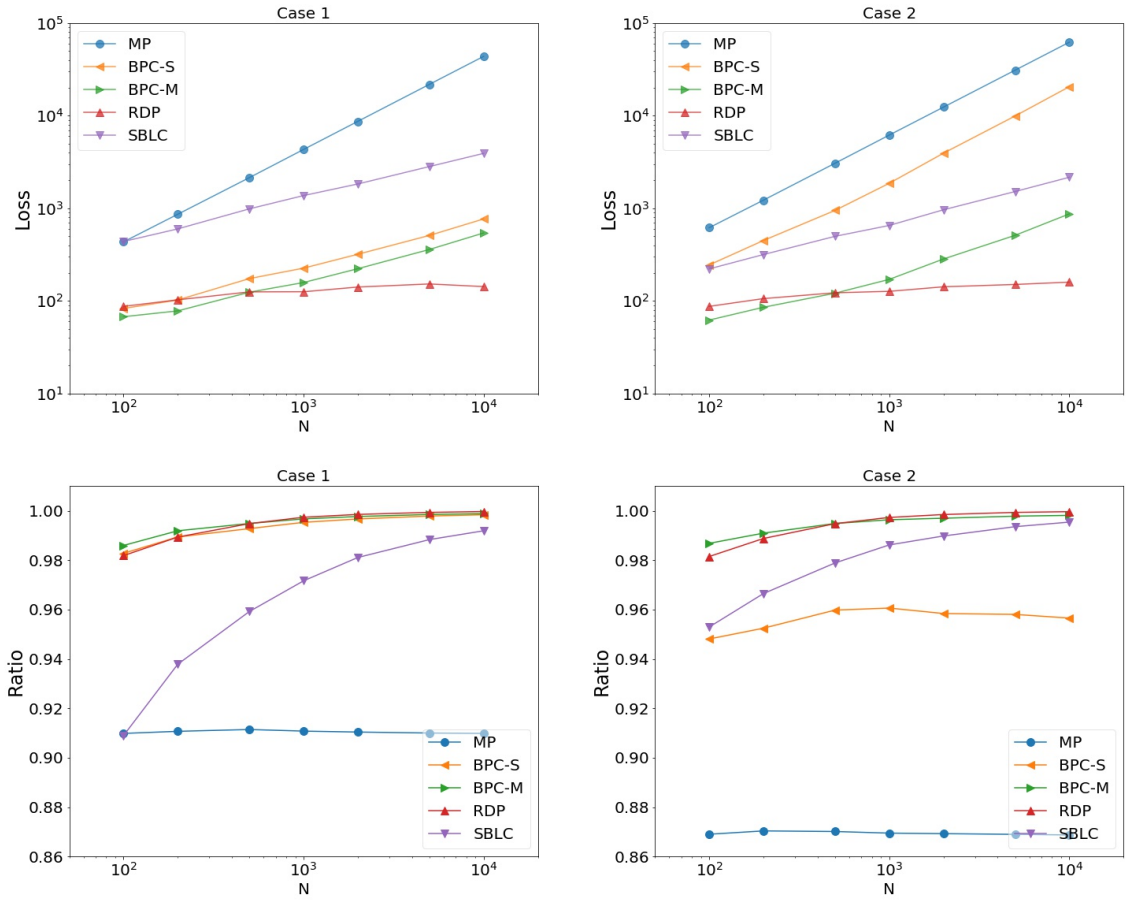


Figure 3 Results of synthetic data

Synthetic Data The results are presented in Figure 3, where we apply a log scale to both the horizontal and vertical axes. We also plot $\text{rev}_\pi/\text{rev}_{\text{OP}}$, i.e., the ratio between the revenue achieved by various policies and the optimal revenue. In both cases, the trend of loss is very clear by observing the slope of lines in different colors. Particularly, MP (myopic policy) performs the worst, incurring a linear loss with respect to time. SBLC has roughly a $\sqrt{T}$ loss, which is consistent to our result in Theorem 3, stating that SBLC is asymptotically optimal. Nevertheless, there exists a relatively large gap between the results obtained under SBLC and that under the optimal policy, for small values of N.

In Case 1, BPC and RDP policies perform much better for both medium- and large-sized problems. However, there exists difference. Lines of BPC policies are nearly parallel to that of SBLC, with smaller intercepts. This implies that BPC policies may still exhibit a $\sqrt{T}$ loss asymptotically. Within BPC policies, BPC-M is consistently better than BPC-S. From Figure 3, $\log L_{\text{BPC-S}} - \log L_{\text{BPC-M}}$ remains a steady positive number, showing that $L_{\text{BPC-M}}$ is approximately a proportion $\alpha < 1$ of $L_{\text{BPC-S}}$. For RDP policy, $\log L_{\text{RDP}}$ exhibits a near horizontal relationship with $\log T$, and

as T increases, the slope tends to zero. This is also consistent with our theoretical result in Section 4.3, stating that the loss of RDP is uniformly bounded.

In Case 2, most of our observations above are still valid, except that BPC policies perform very differently. In this challenging case, BPC-S incurs even linear loss with respective to the time horizon, while BPC-M performs significantly better than BPC-S. The advantage of BPC-M over BPC-S, lying in the smart use of the maximal sequence structure, has been testified.

There is also an interesting finding worth noting that for medium-sized problems ($N \leqslant 500$), BPC-S could perform better than RDP. This is compatible with our results, since the bounded-loss property of RDP is in an asymptotic sense.

Real Data In the numerical tests with real data, we plot $\text{rev}_\pi/\text{rev}_{\text{OP}}$, i.e., the ratio between the revenue achieved by a policy and the optimal revenue, for different policies on all of 32 days in Figure 4. Solid lines are results using estimated parameters in (23) with our proposed policies. Dotted lines are those using real parameters in (24)[5]. The grey bars represent the optimal revenue rev_{OP} that can be achieved in each day (with the scale on the right of the figures). In our experiments, we find that SBLC performs badly even compared to MP, yielding a revenue 10% less than rev_{MP} in many cases. This is not surprising, since SBLC only solves the static problem at the beginning of the time horizon but does not adjust according to the realized demands. Therefore, we will not consider the SBLC policy in our numerical results but rather focus on the remaining 4 policies. We summarize our main findings as follows.

1. Performance Quality: Both BPC and RDP policies perform much better than the myopic policy. Summing over the 32 days, the ratio between the total revenue collected by different policies and the optimal policy for different seat numbers are presented in Table 2. Among the BPC policies, BPC-M performs consistently better than BPC-S, with a revenue increase from 0.5% to 2.2%.

N	MP	BPC-S	BPC-M	RDP
800	96.00%	98.08%	98.66%	97.71%
600	92.70%	97.03%	98.39%	98.05%
400	89.51%	96.07%	98.26%	98.33%

Table 2 Average performance of different policies compared to the optimal

2. Parameter Sensitivity: There is a sharp decrease in demand around 9/15, before and after which the demands appear normal. We see that BPC policies could perform adaptively to such change, while the performance of the RDP is not satisfying. This potentially means that

[5] Since MP does not use past data, its dotted line is the same as the solid one.

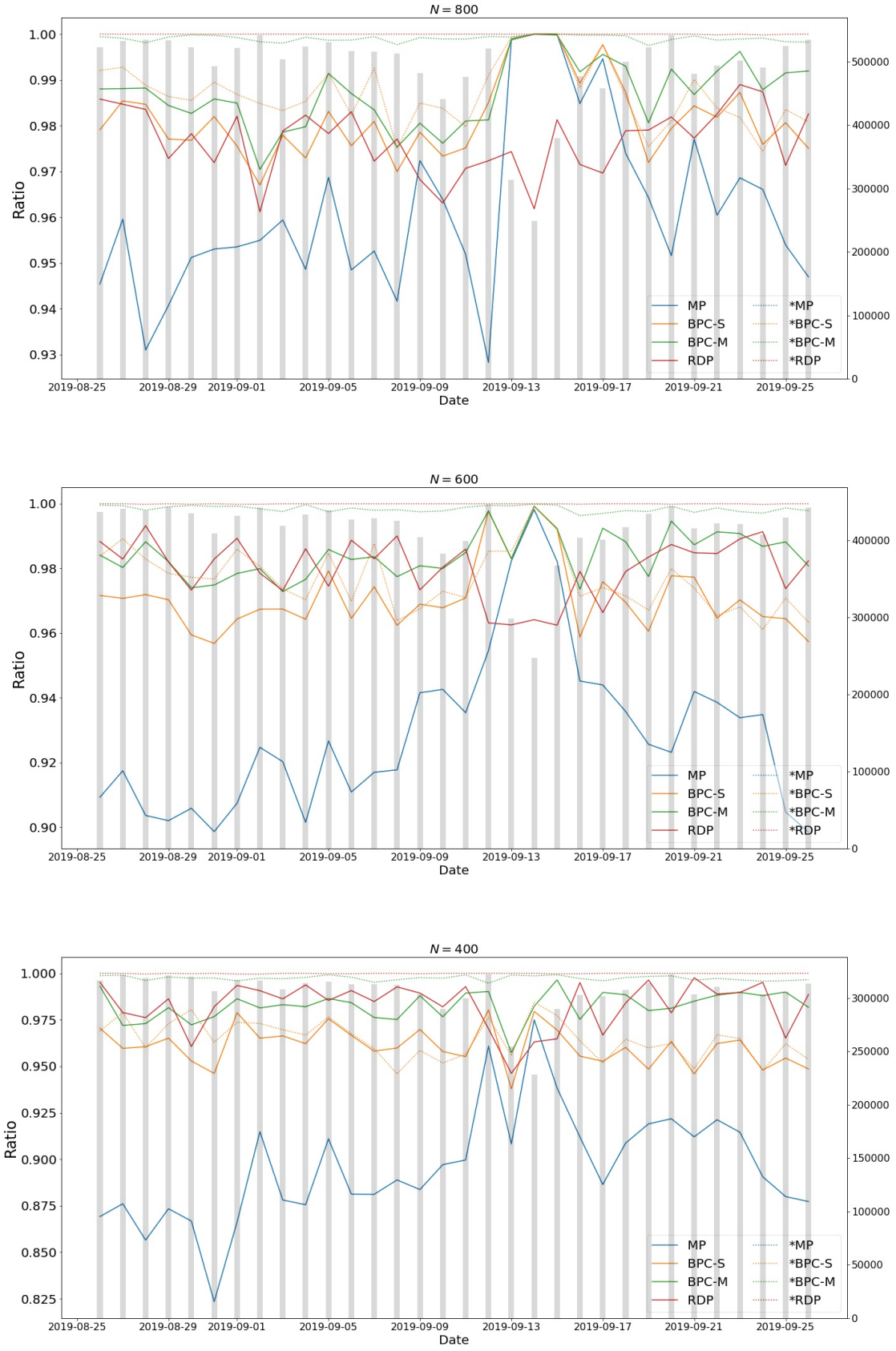


Figure 4 Results of real data

RDP is more sensitive to estimation errors than BPC policies are. Meanwhile, we can see from dotted lines that when we use the real parameters, RDP always performs the best and achieves the optimal. This implies that, RDP should be the preferred choice under accurate parameter values.

3. Resource Scarcity: When the number of seats is large, e.g., $N = 800$ (i.e., the resource is less scarce), RDP performs even inferior to BPC-S. On the contrary, when the number of seats is relatively small, e.g., $N = 400$ (i.e., the resource is more scarce), RDP performs better than BPC-M in most cases. This means that RDP becomes useful when demands exceed resources to a large extent.

Furthermore, we also examine the influence of the assign-to-seat restriction. We test BPC-A, which corresponds to bid-price policies, and RDP-A, which corresponds to the RDP policy. Both BPC-A and RDP-A decide whether to accept or reject the request at each time period, but do not need to assign a seat until the end of the horizon. According to Theorem 2, as long as the accepted requests satisfy the total capacity constraints, one can always find a feasible assignment in the end. Therefore, we use them to estimate the possible loss due to the need of assigning seat immediately. Note that BPC-A and RDP-A are not valid policies in our problem, and we consider them only for understanding the possible loss due to the assign-to-seat restriction. We describe the two policies in detail in Algorithms 5 and 6.

Algorithm 5: BPC-A Policy

for $t = 1, ..., T$ **do**
  Observe a request of type $i^t \to j^t$;
  Compute the dual prices of (3) with $d = \lambda^{[t,T]}$ as the bid prices for each leg $\{\beta_\ell^t\}$;
  If $p_{i^t j^t} \geqslant \sum_{i_t \leqslant \ell \leqslant j_t} \beta_\ell^t$ and there are enough capacities, we accept the request. Else we reject the request.
end

Algorithm 6: RDP-A Policy

for $t = 1, ..., T$ **do**
  Observe a request of type $i^t \to j^t$;
  Solve (3) with $d = \lambda^{[t,T]}$ and obtain an optimal solution $\{x^{\text{RDP-A},t}\}$;
  If $x_{i^t j^t}^{\text{RDP-A},t} \geqslant d_{i^t j^t}^{[t,T]}/2$ (meaning that accepting the request is more preferred than not) and there are enough capacities, we accept the request. Else we reject the request;
end

We use the same set of data to test the performance of BPC-A and RDP-A and compare them with policies in Section 4. The results are summarized in Table 3.

N	MP	BPC-S	BPC-M	BPC-A	RDP	RDP-A
800	96.00%	98.08%	98.66%	98.50%	97.71%	98.83%
600	92.70%	97.03%	98.39%	98.44%	98.05%	98.77%
400	89.51%	96.07%	98.26%	98.37%	98.33%	98.68%

Table 3 Average performance of different policies compared to the optimal

We find that as to the assign-to-seat restriction, the performance of those policies based on the maximal sequence idea (BPC-M, RDP) are comparable to their counterparts without considering the restriction. In particular, the average performance gap between with and without the restriction is no larger than 1.2%, and decreases to near-zero as the resource scarcity increases. This shows that in a practical setting, the assign-to-seat restriction does not affect the performance much under our proposed policies.

As a brief summary, we can see from real data experiments that BPC and RDP policies are all effective policies. The useful insight for companies is that, BPC-M, preferable to BPC-S, appears robust to estimation errors and performs quite well for medium-sized problems. Nevertheless, when the decision maker has much confidence in forecasting parameters accurately, or the resource scarcity is high, RDP could be the more preferred choice.

Summary To summarize, from the numerical experiments, we show that BPC-S, BPC-M and RDP are all effective policies for the dynamic capacity allocation problems studied in this paper, with superior performance than SBLC or MP. In particular, BPC-M achieves better performance than BPC-S, and RDP enjoys bounded loss asymptotically. Both BPC-M and RDP can obtain close-to-optimal revenue. Moreover, the gap between with and without the assign-to-seat restriction is small. In real-time application, for medium-sized problems with rough parameter estimation, we advocate BPC-M as a preferred policy; for large-sized problems with accurate demand forecasting, we recommend RDP as the preferred policy.

6. Conclusion

In this paper, we study a dynamic allocation problem that arises from the practice of selling train tickets. The problem is different from the traditional ones studied in the network revenue management problems, due to the special feature of real-time seat assignment. Through analyzing first the static and then the dynamic versions of the problem, we propose efficient allocation control policies. Numerical experiments involving real data from the practice are performed to show the performance of our approaches.

There are several directions for future research. From the theoretical side, first for the bid-price control polices, it is interesting to study the asymptotic performance of the proposed bid-price control policies. Particularly, how to optimally break "tie"s remains an issue. Second, for the optimal revenue, the relation between the dynamic programming formulations with and without the assign-to-seat restriction is worth investigating. From the practical side, first, it is useful to consider simpler and easy-to-implement control policies (e.g., open and close certain type of itineraries during certain time intervals) and study their effectiveness. Second, it is valuable to embed more practical features into our model. For example, in practice passengers have the freedom to select their seats (e.g., window or aisle seats). Our analysis and results in this paper can possibly be extended to consider such features.

References

Adelman, D. 2007. Dynamic bid prices in revenue management. *Operations Research* **55**(4) 647–661.

Agrawal, S., Z. Wang, Y. Ye. 2014. A dynamic near-optimal algorithm for online linear programming. *Operations Research* **62**(4) 876–890.

Angelelli, E., N. Bianchessi, C. Filippi. 2014. Optimal interval scheduling with a resource constraint. *Computers & Operations Research* **51** 268–281.

Arkin, E. M., E. B. Silverberg. 1987. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics* **18**(1) 1–8.

Armstrong, A., J Meissner. 2010. Railway revenue management: Overview and models. Working Paper.

Aydin, N., S. I. Birbil. 2018. Decomposition methods for dynamic room allocation in hotel revenue management. *European Journal of Operational Research* **271**(1) 179–192.

Bar-Noy, A., R. Bar-Yehuda, A. Freund, J. Naor, B. Schieber. 2001a. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM* **48**(5) 1069–1090.

Bar-Noy, A., S. Guha, J. Naor, B. Schieber. 2001b. Approximating the throughput of multiple machines in real-time scheduling. *SIAM Journal on Computing* **31**(2) 331–352.

Bertsimas, D., I. Popescu. 2003. Revenue management in a dynamic network environment. *Transportation Science* **37**(3) 257–277.

Bhatia, R., J. Chuzhoy, A. Freund, J. S. Naor. 2007. Algorithmic aspects of bandwidth trading. *ACM Transactions on Algorithms* **3**(1) 10.

Bouzina, K. I., H. Emmons. 1996. Interval scheduling on identical machines. *Journal of Global Optimization* **9**(3-4) 379–393.

Brucker, P., L. Nordmann. 1994. The k-track assignment problem. *Computing* **52**(2) 97–122.

Bumpensanti, P., H. Wang. 2018. A re-solving heuristic with uniformly bounded loss for network revenue management. Working Paper.

Carlisle, M. C., E. L. Lloyd. 1995. On the k-coloring of intervals. *Discrete Applied Mathematics* **59**(3) 225–235.

Chen, L., T. Homem-de Mello. 2010. Re-solving stochastic programming models for airline revenue management. *Annals of Operations Research* **177**(1) 91–114.

Chen, Y., R. Levi, C. Shi. 2017. Revenue management of reusable resources with advanced reservations. *Production and Operations Management* **26**(5) 836–859.

Chuzhoy, J., R. Ostrovsky, Y. Rabani. 2006. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research* **31**(4) 730–738.

Ciancimino, A., G. Inzerillo, S. Lucidi, L. Palagi. 1999. A mathematical programming approach for the solution of the railway yield management problem. *Transportation Science* **33**(2) 168–181.

Cooper, W. L. 2002. Asymptotic behavior of an allocation policy for revenue management. *Operations Research* **50**(4) 720–727.

Cross, R. G. 1997. *Revenue Management: Hard-Core Tactics for Market Domination*. Crown Business.

Devanur, N. R., T. P. Hayes. 2009. The adwords problem: Online keyword matching with budgeted bidders under random permutations. *Proceedings of the 10th ACM Conference on Electronic Commerce*. ACM, 71–78.

Eliiyi, D. T., M. Azizoglu. 2006. Spread time considerations in operational fixed job scheduling. *International Journal of Production Research* **44**(20) 4343–4365.

Erlebach, T., F. C-R. Spieksma. 2003. Interval selection: Applications, algorithms, and lower bounds. *Journal of Algorithms* **46**(1) 27–53.

Faigle, U., W. Kern, W. M. Nawijn. 1999. A greedy on-line algorithm for the k-track assignment problem. *Journal of Algorithms* **31**(1) 196–210.

Feng, Y., R. Niazadeh, A. Saberi. 2019. Linear programming based online policies for real-time assortment of reusable resources. *Available at SSRN 3421227* .

Freund, D., S. Banerjee. 2019. Uniform loss algorithms for online stochastic decision-making with applications to bin packing. *Available at SSRN 3479189* .

Gallego, G., G. van Ryzin. 1997. A multiproduct dynamic pricing problem and its applications to network yield management. *Operations Research* **45**(1) 24–41.

Global Times. 2018. Beijing-shanghai high-speed rail makes $4.4b. `http://www.globaltimes.cn/content/1131205.shtml`.

Global Times. 2019. China's high-speed rail carries record 10 billion passengers. `http://www.globaltimes.cn/content/1149608.shtml`.

Goldman, P., R. Freling, K. Pak, N. Piersma. 2002. Models and techniques for hotel revenue management using a rolling horizon. *Journal of Revenue and Pricing Management* **1**(3) 207–219.

Gong, X., V. Goyal, G. Iyengar, D. Simchi-Levi, R. Udwani, S. Wang. 2019. Online assortment optimization with reusable resources. *Available at SSRN 3334789* .

Im, S., Y. Wang. 2011. Secretary problems: Laminar matroid and interval scheduling. *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1265–1274.

Jasin, S., S. Kumar. 2012. A re-solving heuristic with bounded revenue loss for network revenue management with customer choice. *Mathematics of Operations Research* **37**(2) 313–345.

Jasin, S., S. Kumar. 2013. Analysis of deterministic LP-based booking limit and bid price controls for revenue management. *Operations Research* **61**(6) 1312–1320.

Kesselheim, T., A. Tönnis, K. Radke, B. Vöcking. 2014. Primal beats dual on online packing lps in the random-order model. *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*. ACM, 303–312.

Kolen, A. W-J, L. G. Kroon. 1993. On the computational complexity of (maximum) shift class scheduling. *European Journal of Operational Research* **64**(1) 138–151.

Kolen, A. W-J., J. K. Lenstra, C. H. Papadimitriou, F. C-R. Spieksma. 2007. Interval scheduling: A survey. *Naval Research Logistics* **54**(5) 530–543.

Kovalyov, M. Y., C. T. Ng, T. C. E. Cheng. 2007. Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research* **178**(2) 331–342.

Kunnumkal, S., K. Talluri. 2015. On a piecewise-linear approximation for network revenue management. *Mathematics of Operations Research* **41**(1) 72–91.

Kunnumkal, S., H. Topaloglu. 2010. Computing time-dependent bid prices in network revenue management problems. *Transportation Science* **44**(1) 38–62.

Lei, Y., S. Jasin. 2020. Real-time dynamic pricing for revenue management with reusable resources, advance reservation, and deterministic service time requirements. *Operations Research* **68**(3) 676–685.

Levi, R., A. Radovanović. 2010. Provably near-optimal lp-based policies for revenue management in systems with reusable resources. *Operations Research* **58**(2) 503–507.

Lipton, R. J., A. Tomkins. 1994. Online interval scheduling. *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 302–311.

Liu, S, KK Lai, SY Wang. 2008. Booking models for hotel revenue management considering multiple-day stays. *International Journal of Revenue Management* **2**(1) 78–91.

Maglaras, C., J. Meissner. 2006. Dynamic pricing strategies for multiproduct revenue management problems. *Manufacturing & Service Operations Management* **8**(2) 136–148.

Mangasarian, Olvi L, T-H Shiau. 1987. Lipschitz continuity of solutions of linear inequalities, programs and complementarity problems. *SIAM Journal on Control and Optimization* **25**(3) 583–595.

Molinaro, M., R. Ravi. 2013. The geometry of online packing linear programs. *Mathematics of Operations Research* **39**(1) 46–59.

Murayama, H. 2017. China pays a high price for world's fastest train. `https://asia.nikkei.com/Economy/China-pays-a-high-price-for-world-s-fastest-train`.

Nadarajah, S., Y. F. Lim, Q. Ding. 2015. Dynamic pricing for hotel rooms when customers request multiple-day stays. *Available at SSRN 2639188* .

Owen, Z., D. Simchi-Levi. 2018. Price and assortment optimization for reusable resources. *Available at SSRN 3070625* .

Reiman, M. I., Q. Wang. 2008. An asymptotically optimal policy for a quantity-based network revenue management problem. *Mathematics of Operations Research* **33**(2) 257–282.

Rusmevichientong, P., M. Sumida, H. Topaloglu. 2020. Dynamic assortment optimization for reusable products with random usage durations. *Management Science* .

Schmidt, G. 2000. Scheduling with limited machine availability. *European Journal of Operational Research* **121**(1) 1–15.

Secomandi, N. 2008. An analysis of the control-algorithm re-solving issue in inventory and revenue management. *Manufacturing & Service Operations Management* **10**(3) 468–483.

Seiden, S. S. 1998. Randomized online interval scheduling. *Operations Research Letters* **22**(4-5) 171–177.

Shalom, M., A. Voloshin, P. W. Wong, F. C. Yung, S. Zaks. 2014. Online optimization of busy time on parallel machines. *Theoretical Computer Science* **560** 190–206.

Spieksma, F. C-R. 1999. On the approximability of an interval scheduling problem. *Journal of Scheduling* **2**(5) 215–227.

Talluri, K., G. van Ryzin. 1998. An analysis of bid-price controls for network revenue management. *Management Science* **44**(1) 1577–1593.

Tong, C., H. Topaloglu. 2013. On the approximate linear programming approach for network revenue management problems. *INFORMS Journal on Computing* **26**(1) 121–134.

Topaloglu, H. 2009. Using Lagrangian relaxation to compute capacity-dependent bid prices in network revenue management. *Operations Research* **57**(3) 637–649.

Vera, A., S. Banerjee. 2019. The bayesian prophet: A low-regret framework for online decision making. *ACM SIGMETRICS Performance Evaluation Review* **47**(1) 81–82.

Williamson, E. L. 1992. Airline network seat inventory control: Methodologies and revenue impacts. Ph.D. thesis, Massachusetts Institute of Technology.

Zhang, D. 2011. An improved dynamic programming decomposition approach for network revenue management. *Manufacturing & Service Operations Management* **13**(1) 35–52.

Zhang, D., D. Adelman. 2009. An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science* **43**(3) 381–394.

Online Supplementary Material

EC.1. Proofs for Section 3

Proof of Theorem 1.

We first introduce the *Fixed Job Scheduling Problem* (FJSP) as follows.

Definition EC.1 (Fixed Job Scheduling Problem). We are given D jobs, each of which starts at time s_d and ends at time e_d, and N machines, each of which has availabilities from time a_k to b_k. A *schedule* is an assignment of jobs to machines such that for the jobs d assigned to the same machine k, the corresponding intervals $[s_d, e_d)$ do not overlap and $a_k \leqslant s_d < e_d \leqslant b_k$ for all these jobs. The task of FJSP is to determine whether a *schedule* exists.

In Brucker and Nordmann (1994), the authors showed that FJSP is NP-Hard. In the following, to show Problem (1) is NP-Hard, we construct a polynomial time reduction from FJSP to Problem (1).

Given an instance of the FJSP described above, we first sort all endpoints $\{s_d\}$, $\{e_d\}$ and $\{a_j\}$, $\{b_j\}$ in an increasing manner and delete duplicate values. This can be done in $O((D+N)\log(D+N))$ time. Set M to be the length of such sequence minus 1 (note that $M \leq 2(D+N)$). Then we can rewrite the sequence as $t_0 < \cdots < t_M$ and the sequence partitions $[t_0, t_M)$ into M consecutive intervals. We regard each interval $[t_{\ell-1}, t_\ell)$ as the ℓth leg and each endpoint t_ℓ as the ℓth stop. We regard each job as a request of itinerary $i \to j$ if and only if it occupies legs from i to j. We also represent each machine by a $\{0,1\}^{1\times M}$ vector indicating whether it is occupied in each leg, and thus treat each machine as a seat. Let N be the total number of seats and d_{ij} be the total number of requests of type $i \to j$. Let $C \in \{0,1\}^{N\times M}$ be the capacity matrix consisting of all the seats. Then we have conducted a reduction to our setting in polynomial time.

With such a reduction, it is easy to see that, given any positive prices $\{p_{ij}\}$, there exists a *schedule* in FJSP if and only if the optimal value in the reduced problem is $\sum_{i,j} p_{ij} d_{ij}$. Thus, Problem (1) is NP-Hard. □

Proof of Theorem 2. The proof consists of four steps.

Step 1. Strongly NSE matrices could be decomposed into non-overlapping groups.

In this first step, we prove that, for any strongly NSE matrix C, we can decompose its maximal sequences into W groups such that in the wth group, there is a *dominating* maximal sequence $[u_w, v_w] \sim C$, of which any other $[u, v] \sim C$ in this group satisfy either $u = u_w$ or $v = v_w$. In addition, for each w and the corresponding $[u_w, v_w]$, there exists $\ell_w (u_w \leqslant \ell_w \leqslant v_w)$ such that $C_{k\ell_w} = 1$ if and only if $[u_w, v_w] \sim C_k$. Furthermore, different groups have no intersection in *stop* (not only leg), which means $c_{u_w - 1} = c_{v_w + 1} = 0$.

Figure EC.1 gives all possible types of configurations for a single group. In the first configuration, all maximal sequences start from u_2 and $l_2 = v_2$. In the second configuration, all maximal sequences end at v_3 and $l_3 = u_3$. The last configuration is the most general one where all maximal sequences either start from u_4 or end with v_4. l_4 is the fourth leg such that any seat that has l_4 available must have entire $[u_4, v_4]$ unoccupied.

Now we show the maximal sequences of a strongly NSE matrix can be decomposed into such groups. Let C be a strongly NSE matrix. Let $w = 1$ and $\tau = 0$. Let

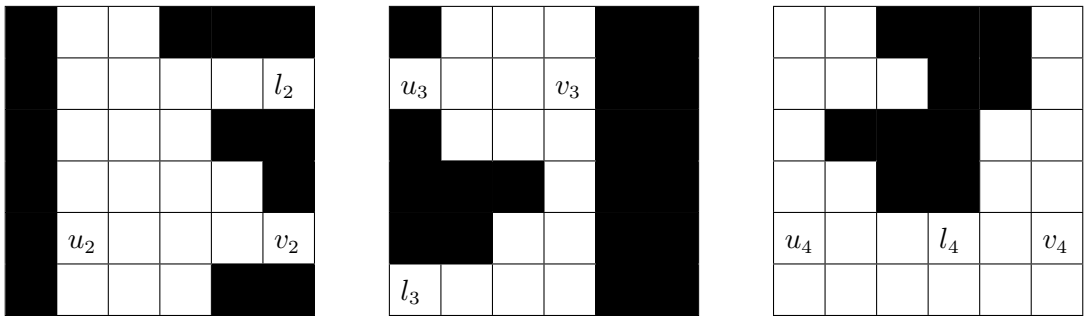


Figure EC.1 Examples of possible structures in a single group. Black boxes indicate occupied seats and white boxes indicate available seats.

$$u_w = \min\{u > \tau | \exists v : [u, v] \sim C\} \text{ and } v_w = \max\{v | [u_w, v] \sim C\}.$$

If u_w does not exist, then we terminate. Otherwise, for any $[u, v] \sim C$ with $u > \tau$, because C is strongly NSE, one of the followings must be true:

1. $u_w \leqslant v_w < u - 1$
2. $u_w = u \leqslant v \leqslant v_w$
3. $u_w \leqslant u \leqslant v = v_w$

We consider the latter two types of maximal sequences. Let

$$V_w = \{v < v_w | [u_w, v] \sim C\} \text{ and } U_w = \{u > u_w | [u, v_w] \sim C\}.$$

Let $v'_w = \max V_w$ and $u'_w = \min U_w$. If $V_w = \emptyset$, we let $v'_w = u_w - 1$. If $U_w = \emptyset$, we let $u'_w = v_w + 1$. Since C is strongly NSE, we must have $v'_w + 1 < u'_w$. Let $\ell_w = v'_w + 1$, then $u_w \leqslant \ell_w \leqslant v_w$ and l_w is available in some seat k only if $[u_w, v_w] \sim C_k$. We represent group w as (u_w, ℓ_w, v_w). Now we let $\tau = v_w + 1$, $w \leftarrow w + 1$, and repeat the procedures above until we terminate. Note that the above procedures consume at most $O(M^3)$ time in total.

Step 2. If C is *strongly NSE*, then any integral optimal solution of (3) could be transformed into an assignment of (1) in polynomial time.

Suppose we have obtained an optimal integral solution $\{x^*_{ij}\}$ of (3) and a characterization of C in Step 2: $\{(u_1, \ell_1, v_1), ..., (u_W, \ell_W, v_W)\}$. In the following, we show that $\{x^*_{ij}\}$ can be turned into a feasible solution $\{x^*_{k,ij}\}$ of (1) with the same objective value in polynomial time. The detailed algorithm is shown in Algorithm 7.

The idea of Algorithm 7 is to assign the requests sequentially according to a particular order of the legs. In Algorithm 7, the seemingly complex operation on $\{\mathcal{M}_{uv}(C)\}$ is to track the dynamics of maximal sequences in C. In the following, we demonstrate that each time we assign requests that include leg ℓ, there are enough seats to assign those requests, or in other words, the procedure in Algorithm 7 is valid. Since different groups are disjoint in stop, there will be no requests that cross two groups. Hence we only need to show within each group, we can assign all requests.

For the wth group, we first assign $\sum_{(i,j):i \leqslant \ell_w \leqslant j} x^*_{ij}$ requests that occupy leg ℓ_w to different seats (Line 2-9 in Algorithm 7). Since $c_{u_w - 1} = c_{v_w + 1} = 0$, any request of type $i \to j$ with $i \leqslant \ell_w \leqslant j$ must satisfy $u_w \leqslant i \leqslant j \leqslant v_w$.

Algorithm 7: Assignment Algorithm

for $w = 1, ..., W$ **do**
  /* Carry out Assignment in the wth group. */
  for $(i,j) : i \leqslant \ell_w \leqslant j$ **do**
    /* Assign the $\sum_{(i,j):i\leqslant\ell_w\leqslant j} x^*_{ij}$ requests that occupy leg ℓ_w. */
    Retrieve $\mathcal{K}_{ij}$ as a subset of $\mathcal{M}_{u_w v_w}(C)$, such that $|\mathcal{K}_{ij}| = x^*_{ij}$;
    Assign x^*_{ij} requests of type $i \to j$ to seats in $\mathcal{K}_{ij}$. Let $x^*_{k,ij} = 1, \forall k \in \mathcal{K}_{ij}$;
    $\mathcal{M}_{u_w v_w}(C) \leftarrow \mathcal{M}_{u_w v_w}(C) \setminus \mathcal{K}_{ij}$;
    if $u_w < i$ **then** $\mathcal{M}_{u_w(i-1)}(C) \leftarrow \mathcal{M}_{u_w(i-1)}(C) \cup \mathcal{K}_{ij}$;
    if $v_w > j$ **then** $\mathcal{M}_{(j+1)v_w}(C) \leftarrow \mathcal{M}_{(j+1)v_w}(C) \cup \mathcal{K}_{ij}$;
  end
  for $\ell = \ell_w + 1, ..., v_w$ **do**
    /* Assign the $\sum_{j\geqslant\ell} x^*_{\ell j}$ requests that start with leg ℓ. */
    for $j : j \geqslant \ell$ **do**
      for $z = 1, ..., x^*_{\ell j}$ **do**
        Retrieve an element k from $\cup_{\ell':\ell'\leqslant\ell} \mathcal{M}_{\ell' v_w}(C)$. Suppose $[u, v_w] \sim C_k$, then $u \leqslant \ell$;
        Assign a request $\ell \to j$ to seat k. Let $x^*_{k,\ell j} = 1$;
        $\mathcal{M}_{u v_w}(C) \leftarrow \mathcal{M}_{u v_w}(C) \setminus \{k\}$;
        if $u < \ell$ **then** $\mathcal{M}_{u(\ell-1)}(C) \leftarrow \mathcal{M}_{u(\ell-1)}(C) \cup \{k\}$;
        if $v_w > j$ **then** $\mathcal{M}_{(j+1)v_w}(C) \leftarrow \mathcal{M}_{(j+1)v_w}(C) \cup \{k\}$;
      end
    end
  end
  for $\ell = \ell_w - 1, ..., u_w$ **do**
    /* Assign the $\sum_{i\leqslant\ell} x^*_{i\ell}$ requests that end with leg ℓ. */
    for $i : i \leqslant \ell$ **do**
      for $z = 1, ..., x^*_{i\ell}$ **do**
        Retrieve an element k from $\cup_{\ell':\ell'\geqslant\ell} \mathcal{M}_{u_w \ell'}(C)$. Suppose $[u_w, v] \sim C_k$, then $v \geqslant \ell$;
        Assign a request $i \to \ell$ to seat k. Let $x^*_{k,i\ell} = 1$;
        $\mathcal{M}_{u_w v}(C) \leftarrow \mathcal{M}_{u_w v}(C) \setminus \{k\}$;
        if $u_w < i$ **then** $\mathcal{M}_{u_w(i-1)}(C) \leftarrow \mathcal{M}_{u_w(i-1)}(C) \cup \{k\}$;
        if $v_w > l$ **then** $\mathcal{M}_{(l+1)v_w}(C) \leftarrow \mathcal{M}_{(l+1)v_w}(C) \cup \{k\}$;
      end
    end
  end
end

Since $\sum_{(i,j):i\leqslant\ell_w\leqslant j} x^*_{ij} \leqslant c_{\ell_w}$, and by the construction in Step 1 in the proof, any seat with leg l_w unoccupied must contain $[u_w, v_w] \sim C$, thus this step is valid.

For $\ell > \ell_w$ and any seat k, if the ℓth leg is not occupied before the corresponding iteration, then $C_{k\ell} = \cdots = C_{kv_w} = 1$ and $C_{k(v_w+1)} = c_{v_w+1} = 0$. Any request of type $\ell \to j$ must satisfy $j \leqslant v_w$ and thus could be assigned to seat k. Therefore, we only need to verify that each time when we move to leg ℓ (Line 9-18 in Algorithm 7), the number of requests that start with leg ℓ is no more than the number of seats that still have leg ℓ unoccupied at that iteration. Note that the number of requests that start with ℓ is $\sum_{j:j\geqslant\ell} x^*_{\ell j}$, and the number of seats that still have leg ℓ unoccupied is $c_\ell - \sum_{(i,j):i<\ell\leqslant j} x^*_{ij}$, and we have $c_\ell \geqslant \sum_{(i,j):i\leqslant\ell\leqslant j} x^*_{ij} = \sum_{j:j\geqslant\ell} x^*_{\ell j} + \sum_{(i,j):i<\ell\leqslant j} x^*_{ij}$. Therefore, all requests that start with leg ℓ can be assigned.

For $\ell < \ell_w$ and any seat k, if the ℓth leg is not occupied before the corresponding iteration, then $C_{k\ell} = \cdots = C_{ku_w} = 1$ and $C_{k(u_w-1)} = c_{u_w-1} = 0$. Any request of type $i \to \ell$ must satisfy $i \geqslant u_w$ and thus could be assigned to seat k. Therefore, we only need to verify that each time when we move to leg ℓ (Line 21-29 in Algorithm 7), the number of requests that end with leg ℓ is no more than the number of seats that still have ℓ unoccupied ar that iteration. Note that the number of requests that end with ℓ is $\sum_{i:i\leqslant\ell} x^*_{i\ell}$, and the number of seats that still have leg ℓ unoccupied is $c_\ell - \sum_{(i,j):i\leqslant\ell<j} x^*_{ij}$, and we have $c_\ell \geqslant \sum_{(i,j):i\leqslant\ell\leqslant j} x^*_{ij} = \sum_{i:i\leqslant\ell} x^*_{i\ell} + \sum_{(i,j):i\leqslant\ell<j} x^*_{ij}$. Therefore, all requests that end with leg ℓ can be assigned.

Hence, when C is strongly NSE, for any integral optimal solution of (3) we can construct a feasible solution of (1) with the same objective value. Meanwhile, it is easy to see that the optimal value of (3) offers an upper bound to that of (1). Thus, when C is strongly NSE, (1) and (3) have the same optimal value, and (1) can be solved in polynomial time.

Step 3. For any fixed positive $\{p_{ij}\}$, if (1) and (3) have the same optimal value for any nonnegative integers $\{d_{ij}\}$, then C must be strongly NSE.

Suppose the number of $[u,v] \sim C$ is $m_{uv} = |\mathcal{M}_{uv}(C)|$. We construct $\{d_{ij}\}_{i\leqslant j}$ as follows.

$$d_{ij} = \begin{cases} m_{ij}+1, \text{ if } (i,j) = (u_1,v_2) \text{ or } (u_2,v_1) \\ m_{ij}-1, \text{ if } (i,j) = (u_1,v_1) \text{ or } (u_2,v_2) \\ m_{ij}, \qquad\qquad \text{otherwise.} \end{cases}$$

It is easy to verify that $x^*_{ij} = d_{ij}$ is an integral optimal solution of (3), which means that, without the assign-to-seat restriction, we can accept all the requests, and the optimal objective value of (3) is $\sum_{i\leqslant j} p_{ij} d_{ij}$.

Now we show that if C is not strongly NSE, then the optimal value of (1) is different from $\sum_{i\leqslant j} p_{ij} d_{ij}$. To show this, suppose C is not strongly NSE, then by definition, there must exist $[u_1,v_1] \sim C$ and $[u_2,v_2] \sim C$ such that one of the following holds (the three cases are illustrated in Figure EC.2:

1. $u_1 \leqslant v_1 = u_2 - 1 \leqslant v_2 - 1$
2. $u_1 < u_2 \leqslant v_1 < v_2$
3. $u_1 < u_2 \leqslant v_2 < v_1$

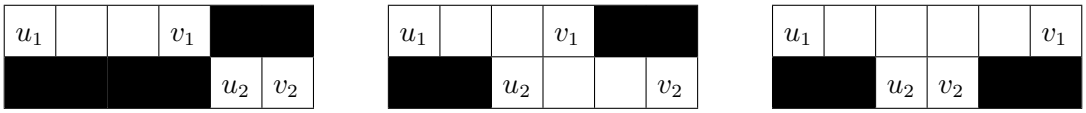


Figure EC.2 Illustrations of the three cases when C is not strongly NSE.

Now we consider each of the three cases. For the first and second cases, consider

$$\sum_k \sum_{(i,j):i\leqslant u_1<v_2\leqslant j} x_{k,ij},$$

which is the total possible number of accepted requests that occupy leg u_1 to v_2. Such requests can only be assigned to seats with leg u_1 to v_2 unoccupied, and each seat could only be assigned at most one request of these types. Therefore,

$$\sum_k \sum_{\substack{(i,j):\\ i\leqslant u_1<v_2\leqslant j}} x_{k,ij} \leqslant \sum_{\substack{(i,j):\\ i\leqslant u_1<v_2\leqslant j}} m_{ij} = \sum_{\substack{(i,j):\\ i\leqslant u_1<v_2\leqslant j}} d_{ij} - 1,$$

where the last inequality follows from $d_{ij} = m_{ij} + 1$ when $(i,j) = (u_1, v_2)$. Therefore, we could not accept all the requests, and thus (1) has strictly smaller optimal value than that of (3).

For the third case, we consider

$$\sum_k \sum_{\substack{(i,j):\\ i\leqslant u_1<v_2\leqslant j\\ \text{or}\\ i\leqslant u_2<v_1\leqslant j}} x_{k,ij},$$

which is the total possible number of accepted requests that occupy leg u_1 to v_2 or leg u_2 to v_1. Such requests can only be assigned to seats with leg u_1 to v_2 or leg u_2 to v_1 unoccupied, and since $[u_1, v_2] \cap [u_2, v_1] \neq \emptyset$, each seat could only be assigned at most one request of these types. Therefore,

$$\sum_k \sum_{\substack{(i,j):\\ i\leqslant u_1<v_2\leqslant j\\ \text{or}\\ i\leqslant u_2<v_1\leqslant j}} x_{k,ij} \leqslant \sum_{\substack{(i,j):\\ i\leqslant u_1<v_2\leqslant j\\ \text{or}\\ i\leqslant u_2<v_1\leqslant j}} m_{ij} = \sum_{\substack{(i,j):\\ i\leqslant u_1<v_2\leqslant j\\ \text{or}\\ i\leqslant u_2<v_1\leqslant j}} d_{ij} - 1,$$

where the last inequality follows from $d_{ij} = m_{ij} + 1$ when $(i,j) = (u_1, v_2)$ or (u_2, v_1), and $d_{ij} = m_{ij} - 1$ when $(i,j) = (u_1, v_1)$. Therefore, we could not accept all the requests, and thus again, (1) has strictly smaller objective value than that of (3).

Therefore, we showed that if C is not strongly NSE, then for any positive prices $\{p_{ij}\}$, there exist a set of nonnegative integers $\{d_{ij}\}$ such that (1) and (3) have different optimal objective values.

Step 4. If C is NSE, then problem (1) can be solved in polynomial time.

Given a capacity matrix C that is NSE, we construct a new capacity matrix $\tilde{C}$ together with new requests and prices. The idea of the construction is to add a *dummy* leg between each pair of neighboring legs in the original problem and to keep the maximal sequence structure. More precisely, we construct $\tilde{C} \in \{0,1\}^{N\times(2M-1)}$, where the number of seats is N but the number of legs is $2M-1$. For any $[u,v] \sim C_k$, we let $\tilde{C}_{k\ell} = 1$ for $2u - 1 \leq \ell \leq 2v - 1$. For the rest of $\tilde{C}_{k\ell}$, we let them be zero. Then for any request of type $i \to j$, we construct a new request of type $\tilde{i} \to \tilde{j}$ with $\tilde{i} = 2i - 1$ and $\tilde{j} = 2j - 1$. This means that in $\tilde{C}$, there are only requests that both start and end with an odd leg.

We claim that if C is NSE, then $\tilde{C}$ must be strongly NSE. To prove the claim, for any two disjoint $[\tilde{u}_1, \tilde{v}_1] \sim \tilde{C}$ and $[\tilde{u}_2, \tilde{v}_2] \sim \tilde{C}$, we have all of $\tilde{u}_1$, $\tilde{u}_2$, $\tilde{v}_1$, and $\tilde{v}_2$ are odd numbers, thus if $\tilde{u}_2 > \tilde{v}_1$ ($\tilde{u}_1 > \tilde{v}_2$, resp.), then it must be that $\tilde{u}_2 > \tilde{v}_1 + 1$ ($\tilde{u}_1 > \tilde{v}_2 + 1$, resp.). Also, a request of type $i \to j$ can be assigned to a $[u,v] \sim C_k$ if and only if $\tilde{i} \to \tilde{j}$ can be assigned to a $[\tilde{u}, \tilde{v}] \sim \tilde{C}_k$.

For all $(\tilde{i},\tilde{j}):1\leqslant\tilde{i}\leqslant\tilde{j}\leqslant 2M-1$, if both $\tilde{i}$ and $\tilde{j}$ are odd numbers, then we let

$$\tilde{p}_{\tilde{i}\tilde{j}}=p_{(\frac{\tilde{i}+1}{2})(\frac{\tilde{j}+1}{2})},\tilde{d}_{\tilde{i}\tilde{j}}=d_{(\frac{\tilde{i}+1}{2})(\frac{\tilde{j}+1}{2})},\tilde{\mathcal{M}}_{\tilde{i}\tilde{j}}(\tilde{C})=\mathcal{M}_{(\frac{\tilde{i}+1}{2})(\frac{\tilde{j}+1}{2})}(C).$$

Otherwise, we let $\tilde{p}_{\tilde{i}\tilde{j}}=1$, $\tilde{d}_{\tilde{i}\tilde{j}}=0$ and $\tilde{\mathcal{M}}_{\tilde{i}\tilde{j}}(C)=\emptyset$. To construct an assignment, we first solve

$$\begin{aligned}
\text{maximize}\quad & \sum_{\tilde{i},\tilde{j}}\tilde{p}_{\tilde{i}\tilde{j}}\tilde{x}_{\tilde{i}\tilde{j}} & \\
\text{subject to}\quad & 0\leqslant\tilde{x}_{\tilde{i}\tilde{j}}\leqslant\tilde{d}_{\tilde{i}\tilde{j}}, & \forall(\tilde{i},\tilde{j}):1\leqslant\tilde{i}\leqslant\tilde{j}\leqslant 2M-1,\\
& \sum_{(\tilde{i},\tilde{j}):\tilde{i}\leqslant\tilde{\ell}\leqslant\tilde{j}}\tilde{x}_{\tilde{i}\tilde{j}}\leqslant\sum_{k}\tilde{C}_{k\tilde{\ell}}=\tilde{c}_{\tilde{\ell}}, & \forall\tilde{\ell}\in[2M-1],
\end{aligned}$$

and obtain an integral optimal solution $\{\tilde{x}^*_{\tilde{i}\tilde{j}}\}$. We then can obtain an assignment $\{\tilde{x}^*_{k,\tilde{i}\tilde{j}}\}$ using Step 1 and Algorithm 7 with $\{\tilde{x}^*_{\tilde{i}\tilde{j}}\}$ and $\{\tilde{\mathcal{M}}_{\tilde{u}\tilde{v}}(\tilde{C})\}$. During the process, we update $x^*_{k,ij}$ and $\tilde{x}^*_{k,\tilde{i}\tilde{j}}$ simultaneously so that $x^*_{k,(\frac{\tilde{i}+1}{2})(\frac{\tilde{j}+1}{2})}=\tilde{x}^*_{k,\tilde{i}\tilde{j}}$. As a result, we get an integral optimal solution of (1) in polynomial time.

Combining the four steps, we have shown that if C is NSE, then (1) can be solved in polynomial time. Furthermore, for any fixed positive prices $\{p_{ij}\}$, (1) and (3) have the same optimal value for any nonnegative integers $\{d_{ij}\}$ if and only if C is strongly NSE. Thus Theorem 2 is proved.

□

Proof of Corollary 1. We will only prove the case when $\{d_{ij}\}$ are all rational, since other cases can be obtained by approximating $\{d_{ij}\}$ through rational numbers. In the rational case, we let θ be a positive integer such that $\theta d_{ij}\in\mathbb{Z}$ for all $1\leqslant i\leqslant j\leqslant M$. Now we copy the capacity matrix C with θ times, denoted as $C(\theta)$, and consider allocating θd_{ij} requests of type $i\to j$ ($\forall 1\leqslant i\leqslant j\leqslant M$) into it. Since C is strongly NSE, $C(\theta)$ is also strongly NSE. By Theorem 2, (1) and (3) have the same optimal value if we replace C with $C(\theta)$ and $\{d_{ij}\}$ with $\{\theta d_{ij}\}$. Then dividing the optimal solutions both by θ yields the corollary.

□

EC.2. Proofs for Section 4

Proof of Lemma 1. First we show that the optimal objective value of (2) is no larger than that of (4). For any feasible solution $\{x^*_{k,ij}\}_{i\leqslant j}$ of (2), let

$$\xi^*_{uij}=\sum_k x^*_{k,ij}\mathbb{1}\{\exists v:[u,v]\sim C_k,u\leqslant i\leqslant j\leqslant v\}.$$

Then for any given $i\leqslant j$, we have

$$\begin{aligned}
\sum_{u:u\leqslant i}\xi^*_{uij}&=\sum_{u:u\leqslant i}\sum_k x^*_{k,ij}\mathbb{1}\{\exists v:[u,v]\sim C_k,u\leqslant i\leqslant j\leqslant v\}\\
&=\sum_k x^*_{k,ij}\sum_{u:u\leqslant i}\mathbb{1}\{\exists v:[u,v]\sim C_k,u\leqslant i\leqslant j\leqslant v\}\\
&=\sum_k x^*_{k,ij}\cdot 1\\
&=\sum_k x^*_{k,ij}\leqslant d_{ij}.
\end{aligned}$$

The third equality holds because when $x^*_{k,ij} > 0$, there exists exactly one $[u,v] \sim C_k$ such that $u \leqslant i \leqslant j \leqslant v$. Also, for given $u \leqslant \ell$,

$$\begin{aligned}
\sum_{(i,j):u\leqslant i\leqslant \ell\leqslant j} \xi^*_{uij} &= \sum_{(i,j):u\leqslant i\leqslant \ell\leqslant j} \sum_k x^*_{k,ij} \mathbb{1}\{\exists v : [u,v] \sim C_k, u \leqslant i \leqslant j \leqslant v\} \\
&= \sum_k \sum_{(i,j):u\leqslant i\leqslant \ell\leqslant j} x^*_{k,ij} \mathbb{1}\{\exists v : [u,v] \sim C_k, u \leqslant i \leqslant j \leqslant v\} \\
&\leqslant \sum_k \sum_{(i,j):u\leqslant i\leqslant \ell\leqslant j} x^*_{k,ij} \mathbb{1}\{\exists v : [u,v] \sim C_k, u \leqslant \ell \leqslant v\} \\
&\leqslant \sum_k \mathbb{1}\{\exists v : [u,v] \sim C_k, u \leqslant \ell \leqslant v\} \\
&= m_{u\ell}.
\end{aligned}$$

Thus, $\{\xi^*_{uij}\}_{u\leqslant i\leqslant j}$ is also a feasible solution of (4). Moreover, $\{\xi^*_{uij}\}_{u\leqslant i\leqslant j}$ and $\{x^*_{k,ij}\}_{i\leqslant j}$ reach the same objective value.

Next, we show that the optimal objective value of (4) is no larger than that of (2). Let $\{\xi^*_{uij}\}_{u\leqslant i\leqslant j}$ be an optimal solution of the LP relaxation of (4). We additionally define $\xi^*_{uij} = 0$ if $i < u$ and $m_{u\ell} = 0$ if $u > \ell$. For any $u \in [M]$, consider Problem (EC.1),

$$\begin{aligned}
\text{maximize} \quad & \sum_{i\leqslant j} \left(p_{ij} \xi_{uij}\right), && && \text{(EC.1)} \\
\text{subject to} \quad & \sum_{(i,j):i\leqslant \ell\leqslant j} \xi_{uij} \leqslant m_{u\ell}, && \forall \ell \in [M], \\
& 0 \leqslant \xi_{uij} \leqslant \xi^*_{uij}, && \forall i \leqslant j,
\end{aligned}$$

and it is easy to see that $\{\xi^*_{uij}\}_{i\leqslant j}$ is an optimal solution to (EC.1). Now we consider a new capacity matrix $C[u]$, which consists of all maximal sequences in C that start with leg u:

$$C[u]_{k\ell} = \begin{cases} 1, & \text{if } \exists v : [u,v] \sim C_k, u \leqslant \ell \leqslant v, \\ 0, & \text{otherwise.} \end{cases}$$

Then $C[u]$ is strongly NSE, and we have

$$m_{u\ell} = \sum_k C[u]_{k\ell} \text{ and } C = \sum_u C[u].$$

Thus, from Corollary 1, (EC.1) has the same objective value with (EC.2),

$$\begin{aligned}
\text{maximize} \quad & \sum_{i,j} \left(p_{ij} \sum_k x_{k,ij}[u] \right), && && \text{(EC.2)} \\
\text{subject to} \quad & \sum_k x_{k,ij}[u] \leqslant \xi^*_{uij}, && \forall i \leqslant j, \\
& \sum_{(i,j):i\leqslant \ell\leqslant j} x_{k,ij}[u] \leqslant C[u]_{k\ell}, && \forall k \in [N], \ell \in [M], \\
& 0 \leqslant x_{k,ij}[u] \leqslant 1, && \forall k \in [N], i \leqslant j.
\end{aligned}$$

Let $\{x^*_{k,ij}[u]\}$ be an optimal solution of (EC.2). Then $\sum_k x^*_{k,ij}[u] = \xi^*_{uij}$ for any $i \leqslant j$. Now we define $x^*_{k,ij}$ as

$$x^*_{k,ij} = \sum_{u=1}^{M} x^*_{k,ij}[u]$$

and verify that $\{x^*_{k,ij}\}_{i\leqslant j}$ is feasible for (2). In fact, for fixed $i\leqslant j$,

$$\sum_k x^*_{k,ij}=\sum_k\sum_u x^*_{k,ij}[u]=\sum_u\sum_k x^*_{k,ij}[u]=\sum_u \xi^*_{uij}=\sum_{u:u\leqslant i}\xi^*_{uij}\leqslant d_{ij},$$

Also, for fixed $k\in[N]$ and $\ell\in[M]$, we have

$$\sum_{(i,j):i\leqslant\ell\leqslant j} x^*_{k,ij}=\sum_{(i,j):i\leqslant\ell\leqslant j}\sum_u x^*_{k,ij}[u]=\sum_u\sum_{(i,j):i\leqslant\ell\leqslant j} x^*_{k,ij}[u]\leqslant\sum_u C[u]_{k\ell}=C_{k\ell}.$$

Thus, $\{x^*_{k,ij}\}_{i\leqslant j}$ and $\{\xi^*_{uij}\}_{u\leqslant i\leqslant j}$ reach the same objective value.

□

Proof of Theorem 3. Let $v_\theta(C;\{d_{ij}\})$ denote the optimal objective value of (EC.3)

$$\begin{aligned}
\text{maximize}\quad & \sum_{i\leqslant j} p_{ij}\sum_{k=1}^{\theta N} x_{k,ij} && && \text{(EC.3)}\\
\text{s.t.}\quad & \sum_{k=1}^{\theta N} x_{k,ij}\leqslant d_{ij}, && \forall i\leqslant j,\\
& \sum_{(i,j):i\leqslant\ell\leqslant j} x_{k,ij}\leqslant C(\theta)_{k\ell}, && \forall k\in[\theta N],\ell\in[M],\\
& 0\leqslant x_{k,ij}\leqslant 1, && \forall k,i\leqslant j.
\end{aligned}$$

Then we have

$$V_{\theta T}(C(\theta))\leqslant\mathbb{E}_{\{d_{ij}\}}[v_\theta(C;\{d_{ij}\})]\leqslant v_\theta(C;\{\mathbb{E}[d_{ij}]\})=v_\theta(C;\{\theta\lambda_{ij}\}),\tag{EC.4}$$

where the first inequality holds because the revenue collected from any sample path under any policy must be no larger than that under an oracle who knows the true demand; the second inequality follows because $v_\theta(C;\{d_{ij}\})$ is concave in $\{d_{ij}\}$; and the third equality is by definition. Also, $v_\theta(C;\{\theta\lambda_{ij}\})\geqslant\theta v_1(C;\{\lambda_{ij}\})>0$ which is because we can multiply the optimal solution of $v_1(C;\{\lambda_{ij}\})$ by θ and obtain a feasible solution to (EC.3).

Let $\{x^\theta_{k,ij}\}$ be an integral optimal solution of (EC.3) obtained from Algorithm 1, with $d_{ij}=\lfloor\theta\lambda_{ij}\rfloor$. Let $d^\theta_{ij}=\sum_{k=1}^{\theta N}x^\theta_{k,ij}$. From Lemma 1, we have

$$\sum_{i\leqslant j}p_{ij}d^\theta_{ij}\geqslant v_\theta(C;\{\lfloor\theta\lambda_{ij}\rfloor\})-\sum_{i\leqslant j}p_{ij}i.\tag{EC.5}$$

Also,

$$v_\theta(C;\{\theta\lambda_{ij}\})-v_\theta(C;\{\lfloor\theta\lambda_{ij}\rfloor\})\leqslant v_\theta(C;\{\lceil\theta\lambda_{ij}\rceil\})-v_\theta(C;\{\lfloor\theta\lambda_{ij}\rfloor\})\leqslant\sum_{i\leqslant j}p_{ij}.\tag{EC.6}$$

Here the last inequality is because for any integral optimal solution of $v_\theta(C;\{\lceil\theta\lambda_{i,j}\rceil\})$ and any request of type $i\to j$, we could reject at most one such request, and the new solution is still feasible for $v_\theta(C;\{\lfloor\theta\lambda_{i,j}\rfloor\})$.

Now we bound $L^{\mathrm{SBLC}}_\theta(C)$, we have

$$V^{SBLC}_\theta(C)=\mathbb{E}_{\{d_{ij}\}}\left[\sum_{i\leqslant j}p_{ij}\min\{d^\theta_{ij},d_{ij}\}\right].\tag{EC.7}$$

Therefore,

$$\begin{aligned}
&V_{\theta T}(C(\theta)) - V_\theta^{SBLC}(C)\\
\leqslant\ & v_\theta(C;\{\theta\lambda_{ij}\}) - V_\theta^{SBLC}(C)\\
=\ & v_\theta(C;\{\theta\lambda_{ij}\}) - v_\theta(C;\{\lfloor\theta\lambda_{ij}\rfloor\}) + v_\theta(C;\{\lfloor\theta\lambda_{ij}\rfloor\}) - V_\theta^{SBLC}(C)\\
\leqslant\ & \sum_{i\leqslant j} p_{ij} + \mathbb{E}_{\{d_{ij}\}}\left[\sum_{i\leqslant j} p_{ij}(d_{ij}^\theta + i - \min\{d_{ij}^\theta, d_{ij}\})\right]\\
\leqslant\ & \sum_{i\leqslant j} p_{ij}(i+1) + \mathbb{E}_{\{d_{ij}\}}\left[\sum_{i\leqslant j}\frac{1}{2}p_{ij}(d_{ij}^\theta - d_{ij} + |d_{ij}^\theta - d_{ij}|)\right]\\
\leqslant\ & \sum_{i\leqslant j} p_{ij}(i+1) + \frac{1}{2}\sum_{i\leqslant j} p_{ij}\left(d_{ij}^\theta - \mathbb{E}[d_{ij}] + |d_{ij}^\theta - \mathbb{E}[d_{ij}]| + \sqrt{\mathrm{Var}[d_{ij}]}\right)\\
\leqslant\ & \sum_{i\leqslant j} p_{ij}(i+1) + \frac{1}{2}\sum_{i\leqslant j} p_{ij}\sqrt{\mathrm{Var}[d_{ij}]}\\
\leqslant\ & \sum_{i\leqslant j} p_{ij}(i+1) + \frac{1}{2}\sum_{i\leqslant j} p_{ij}\sqrt{\theta\lambda_{ij}},
\end{aligned}$$

where the third to last inequality is because for any two random variables A and B, $\mathbb{E}(A) - \mathbb{E}(B) \leqslant \mathbb{E}(|A - B|) \leqslant \sqrt{\mathbb{E}(A-B)^2}$, the second to last inequality is because $d_{ij}^\theta \leqslant \mathbb{E}[d_{ij}]$ and the last inequality is because of the independent arrival in each period. Thus the theorem holds.

□

Proof of Theorem 4. We first testify that $\{\beta_{uv}^{\dagger t}\}$ defined in (17) together with $\{z_{ij}^\dagger\} = \{z_{ij}\}$ in (16) satisfy the constraints in (16). The nonnegativity is quite simple. For fixed $u_0 \leqslant v_0$, we let $k_0 \in [N]$ be an integer such that

$$\sum_{\ell=u_0}^{v_0} \beta_{k_0\ell}^t = \min_{k'\in[N]}\{\sum_{\ell=u_0}^{v_0}\beta_{k'\ell}^t\}$$

Then

$$\begin{aligned}
&\beta_{u_0v_0}^{\dagger t} - \beta_{u_0(i-1)}^{\dagger t} - \beta_{(j+1)v_0}^{\dagger t}\\
=\ & \sum_{\ell=u_0}^{v_0}\beta_{k_0\ell}^t - \min_{k'\in[N]}\{\sum_{\ell=u_0}^{i-1}\beta_{k'\ell}^t\} - \min_{k'\in[N]}\{\sum_{\ell=j+1}^{v_0}\beta_{k'\ell}^t\}\\
\geqslant\ & \sum_{\ell=u_0}^{v_0}\beta_{k_0\ell}^t - \sum_{\ell=u_0}^{i-1}\beta_{k_0\ell}^t - \sum_{\ell=j+1}^{v_0}\beta_{k_0\ell}^t\\
=\ & \sum_{\ell=i}^{j}\beta_{k_0\ell}^t \geqslant p_{ij} - z_{ij}^\dagger,
\end{aligned}$$

Thus, (17) is feasible. Moreover, from (17) we can infer that $\hat{V}_t(C) \geqslant \hat{V}_t^\dagger(f(C))$.

We second point out that $\{z_{ij}^\dagger\}$ together with $\{\beta_{uv}^{\dagger t}\}$ form an optimal solution of (16). To prove this, we prove a stronger result: $\hat{V}_t(C) \leqslant \hat{V}_t^\dagger(f(C))$. This is done by constructing a feasible solution of (5) from an optimal solution of (16). Let $\{z_{ij}^\dagger\}$ and $\{\beta_{uv}^{\dagger t}\}$ be an optimal solution of (16) such that $c = \#\{(u,v) : u \leqslant v, \beta_{uv}^{\dagger t} < \beta_{(u+1)v)}^{\dagger t}\}$ achieves its minimum. We will show that $c = 0$. Otherwise, let $(u_0, v_0) \in \arg\min_v\{(u,v) : u \leqslant v, \beta_{uv}^{\dagger t} < \beta_{(u+1)v}^{\dagger t}\}$. Then for any $u_0 + 1 \leqslant i \leqslant j \leqslant v_0$, we have

$$\begin{aligned}
z_{ij}^\dagger + \beta_{(u_0+1)v_0}^{\dagger t} &\geqslant p_{ij} + \beta_{(u_0+1)(i-1)}^{\dagger t} + \beta_{(j+1)v_0}^{\dagger t},\\
z_{ij}^\dagger + \beta_{u_0v_0}^{\dagger t} &\geqslant p_{ij} + \beta_{u_0(i-1)}^{\dagger t} + \beta_{(j+1)v_0}^{\dagger t} \geqslant p_{ij} + \beta_{(u_0+1)(i-1)}^{\dagger t} + \beta_{(j+1)v_0}^{\dagger t}.
\end{aligned}$$

Now we decrease $\beta^{\dagger t}_{(u_0+1)v_0}$ to $\beta^{\dagger t}_{u_0v_0}$, then the constraints are still not violated, but the objective value will not increase. This is a contradiction.

We let $\{z_{ij}\}=\{z^\dagger_{ij}\}$ and construct $\{\beta^t_{k\ell}\}$ as follows,

$$\beta^t_{k\ell}=\begin{cases}\beta^{\dagger t}_{\ell v}-\beta^{\dagger t}_{(\ell+1)v}\geqslant 0, \text{ if } u\leqslant\ell\leqslant v, [u,v]\sim C_k,\\ +\infty, \qquad\qquad \text{otherwise.}\end{cases}$$

Then for any $k\in[N]$ and $i\leqslant j$, if there is some $\ell\in[i,j]$ such that $C_{k\ell}=0$, then we must have

$$z_{ij}+\sum_{\ell:i\leqslant\ell\leqslant j}\beta^t_{k\ell}\geqslant p_{ij}.$$

Else, there must exist a $[u,v]\sim C_k$ such that $u\leqslant i\leqslant j\leqslant v$, we have

$$z_{ij}+\sum_{\ell:i\leqslant\ell\leqslant j}\beta^t_{k\ell}=z^\dagger_{ij}+\beta^{\dagger t}_{iv}-\beta^{\dagger t}_{(j+1)v}\geqslant p_{ij}.$$

Also,

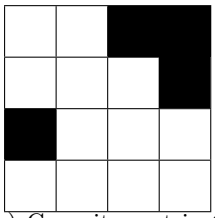
$$\begin{aligned}&\sum_{i\leqslant j}\lambda_{ij}z_{ij}+\sum_{k\in[N]}\sum_{\ell\in[M]}C_{k\ell}\beta^t_{k\ell}\\ =&\sum_{i\leqslant j}\lambda_{ij}z^\dagger_{ij}+\sum_{k\in[N]}\sum_{\substack{(u,v):\\ [u,v]\sim C_k}}\sum_{\ell:u\leqslant\ell\leqslant v}\beta^t_{k\ell}\\ =&\sum_{i\leqslant j}\lambda_{ij}z^\dagger_{ij}+\sum_{k\in[N]}\sum_{\substack{(u,v):\\ [u,v]\sim C_k}}\beta^{\dagger t}_{uv}\\ =&\sum_{i\leqslant j}\lambda_{ij}z^\dagger_{ij}+\sum_{u\leqslant v}f(C)_{uv}\beta^{\dagger t}_{uv}=\hat{V}^{\dagger t}(f(C)).\end{aligned}$$

Thus we've completed the proof.

□

An Example for Non-linearity of BPC-M.

EXAMPLE EC.1. Consider the capacity matrix in Figure EC.3(a) in period 1. We construct the prices as $p_{ij}=10(j-i+1)$, and $\{\lambda^{1+}_{ij}\}$ as $\lambda^{1+}_{12}=\lambda^{1+}_{22}=\lambda^{1+}_{24}=0, \lambda^{1+}_{11}=\lambda^{1+}_{13}=\lambda^{1+}_{23}=\lambda^{1+}_{33}=\lambda^{1+}_{34}=1, \lambda^{1+}_{14}=2$. A group of bid-prices $\{\beta^1_{k\ell}\}$ obtained by solving (5) are presented in Figure EC.3(b). The bid-prices obtained by solving (16) are as follows: $\beta^{\dagger 1}_{11}=\beta^{\dagger 1}_{12}=\beta^{\dagger 1}_{22}=\beta^{\dagger 1}_{44}=0, \beta^{\dagger 1}_{33}=10, \beta^{\dagger 1}_{13}=\beta^{\dagger 1}_{23}=\beta^{\dagger 1}_{24}=\beta^{\dagger 1}_{34}=20, \beta^{\dagger 1}_{14}=40$. Note that here $\beta^{\dagger 1}_{uv}=\min_{k\in[N]}\sum_{\ell=u}^{v}\beta^1_{k\ell}$ always hold.



(a) Capacity matrix C

0	0	40	0
0	10	10	20
20	0	20	0
0	10	10	20

(b) Bid-prices computed from (5), i.e., $\{\beta_{k\ell}\}$

Figure EC.3 Illustrations of Example EC.1

Now assume that in period 1 a request of type $3\to 3$ comes. If we use $\{\beta^1_{k\ell}\}$, then we will *accept* the request and allocate it to one of Seat 2 and Seat 4 because $\beta^1_{23}=\beta^1_{43}=10$. Meanwhile, if we apply $\{\beta^{\dagger 1}_{uv}\}$, we will *reject*

the request. In fact, for Seat 2, we have $\beta^{\dagger 1}_{13}-\beta^{\dagger 1}_{12}=20>p_{33}$; for Seat 3, we have $\beta^{\dagger 1}_{24}-\beta^{\dagger 1}_{22}-\beta^{\dagger}_{44}=20>p_{33}$; for Seat 4, we have $\beta^{\dagger 1}_{14}-\beta^{\dagger 1}_{12}-\beta^{\dagger}_{44}=40>p_{33}$.

□

Proof of Lemma 2. From Theorem 4, (2) $\geqslant$ (18). Therefore, we only need to prove that (2) $\leqslant$ (18) + (20), which in turn implies our results.

Step 1. In our first step, we show that the optimal objective value of (1), the IP version of (2), is no more than that of (EC.8), the IP version of (18)+(20).

$$
\begin{aligned}
\max_{\gamma} \quad & \sum_{i\leqslant j}\left(p_{ij}\sum_{\substack{(u,v):\\ u\leqslant i\leqslant j\leqslant v}}\gamma_{uijv}\right), && && \text{(EC.8)}\\
\text{s.t.} \quad & \sum_{\substack{(u,v):\\ u\leqslant i\leqslant j\leqslant v}}\gamma_{uijv}\leqslant d_{ij}, && \forall i\leqslant j,\\
& \sum_{\substack{(i,j):\\ u\leqslant i\leqslant j\leqslant v}}\gamma_{uijv}\leqslant \sum_{\substack{(k,\ell):\\ v+1\leqslant k\leqslant \ell}}\gamma_{u(v+1)k\ell}+\sum_{\substack{(k,\ell):\\ \ell\leqslant k\leqslant u-1}}\gamma_{\ell k(u-1)v}+A_{uv}, && \forall u\leqslant v,\\
& \gamma_{ui^tj^tv}\leqslant A_{uv}, && \forall u\leqslant i^t\leqslant j^t\leqslant v,\\
& \gamma_{uijv}\in\mathbb{N}, && \forall u\leqslant i\leqslant j\leqslant v.
\end{aligned}
$$

For any optimal solution $\{x^*_{k,ij}\}$ of (1), we impose an index on all the allocated requests. Note that we put the smallest indexes on all $i^t\to j^t$. Now we consider to take out all requests and assign the requests again, but in a *sequential* manner, i.e., we allocate the requests one by one according to the index. During this procedure, we will arrange a tuple $(\cdot,\cdot,\cdot,\cdot)$ to each ticket and track the dynamics.

Suppose we have allocated $t-1$ requests and we are allocating the tth request $i\to j$ into seat k, then seat k must own a unique *maximal sequence* $[u,v]$ such that $u\leqslant i\leqslant j\leqslant v$. Then the tuple given to the tth seat is $g(t)=(u,i,j,v)$. In fact, the middle two numbers indicate the type of this request $(i\to j)$, and the other two numbers represent the "environment" *when it is allocated* (*maximal sequence* $[u,v]$). When we've finished allocating the tth ticket $i\to j$ into $[u,v]\sim C_k$, we track the dynamics as in (11).

Since we first allocate all $i_t\to j_t$ into C, we must have

$$\gamma_{ui^tj^tv}\leqslant f(C)_{uv},\quad \forall u\leqslant i^t\leqslant j^t\leqslant v.$$

For any given $u\leqslant i\leqslant j\leqslant v$, denote γ_{uijv} as the total number of requests that are arranged with the tuple (u,i,j,v), i.e.,

$$\gamma^*_{uijv}=\sum_{t=1}^{d}\mathbb{1}\left\{g(t)=(u,i,j,v)\right\}.$$

Now let's examine the procedures stated above and explore some necessary conditions for $\{\gamma^*_{uijv}\}$. First, the total number of allocated request $i\to j$ is clearly

$$\sum_{\substack{(u,v):\\ u\leqslant i\leqslant j\leqslant v}}\gamma^*_{uijv}=\sum_k x^*_{k,ij},$$

which should be no larger than d_{ij}.

Second, during the above process, the number of times that $[u,v]$ splits is

$$\sum_{\substack{(i,j):\\ u\leqslant i\leqslant j\leqslant v}} \gamma^*_{uijv}.$$

The number of times that $[u,v]$ initially exists or is generated by other *maximal sequence*s is

$$\left(\sum_{\substack{(k,\ell):\\ v+1\leqslant k\leqslant \ell}} \gamma^*_{u(v+1)k\ell}\right)+\left(\sum_{\substack{(k,\ell):\\ \ell\leqslant k\leqslant u-1}} \gamma^*_{\ell k(u-1)v}\right)+f(C)_{uv}.$$

Thus $\{\gamma^*_{uijv}\}$ must satisfy

$$\begin{aligned}
&\sum_{\substack{(u,v):\\ u\leqslant i\leqslant j\leqslant v}} \gamma^*_{uijv}\leqslant d_{ij}, && \forall i\leqslant j,\\
&\sum_{\substack{(i,j):\\ u\leqslant i\leqslant j\leqslant v}} \gamma^*_{uijv}\leqslant \left(\sum_{\substack{(k,l):\\ v+1\leqslant k\leqslant l}} \gamma^*_{u(v+1)kl}\right)+\left(\sum_{\substack{(k,l):\\ l\leqslant k\leqslant u-1}} \gamma^*_{lk(u-1)v}\right)+f(C)_{uv}, && \forall u\leqslant v, \qquad \text{(EC.9)}\\
&\gamma_{ui^tj^tv}\leqslant f(C)_{uv}, && \forall u\leqslant i^t\leqslant j^t\leqslant v,\\
&\gamma^*_{uijv}\in\mathbb{N}, && \forall u\leqslant i\leqslant j\leqslant v,
\end{aligned}$$

which is exactly the constraints of (EC.8). Thus the optimal value of (EC.8) is at least as large as the optimal value of (1).

Step 2. Now we prove the relationship between (2) and (18)+(20). We will prove that the optimal objective value of (2) is no more than that of (18)+(20). Since the coefficients of (2) are all rational, there exists an optimal solution $\{x^*_{k,ij}\}$ such that all the variables are rational. Let θ_1 be a positive integer such that $\theta_1 x^*_{k,ij}\in\mathbb{N}$. We consider to *copy* the capacity matrix C by θ_1 times as $C(\theta_1)$. Now for given $k\in[N]$, $\{\theta_1 x^*_{k,ij}\}_{i\leqslant j}$ satisfy

$$\sum_{\substack{(i,j):\\ i\leqslant \ell\leqslant j}} \theta_1 x^*_{k,ij}\leqslant \theta_1 C_{k\ell}=\sum_{s=0}^{\theta_1-1} C_{(k+sN)\ell}, \quad \forall k\in[N],\ell\in[M],$$

which is the constraint at an aggregation level. Note that for any k, the matrix formed by $\{C_{k+sN}\}_s$ is a strongly NSE matrix. Therefore, by Theorem 2, there exists $\{\tilde{x}_{k,ij}\}$ such that

$$\begin{aligned}
&\sum_{s=0}^{\theta_1-1} \tilde{x}_{k+sN,ij}=\theta_1 x^*_{k,ij}, && \forall k\in[N], i\leqslant j,\\
&\sum_{\substack{(i,j):\\ i\leqslant \ell\leqslant j}} \tilde{x}_{k,ij}\leqslant C(\theta_1)_{k\ell}, && \forall k\in[\theta_1 N],\ell\in[M],\\
&\tilde{x}_{k,ij}\in\{0,1\}, && \forall k\in[\theta_1 N], i\leqslant j.
\end{aligned}$$

Thus we have $\sum_{k=1}^{\theta_1 N}\tilde{x}_{k,ij}=\sum_{k=1}^{N}\theta_1 x^*_{k,ij}\leqslant\theta_1 d_{ij}$. From Step 1, there exist $\{\tilde{\gamma}_{uijv}\}$ such that

$$\begin{aligned}
&\sum_{\substack{(u,v):\\ u\leqslant i\leqslant j\leqslant v}} \tilde{\gamma}_{uijv}=\sum_{k=1}^{\theta_1 N}\tilde{x}_{k,ij}\leqslant\theta_1 d_{ij}, && \forall 1\leqslant i\leqslant j\leqslant M,\\
&\sum_{\substack{(i,j):\\ u\leqslant i\leqslant j\leqslant v}} \tilde{\gamma}_{uijv}\leqslant \sum_{\substack{(k,l):\\ v+1\leqslant k\leqslant l}} \tilde{\gamma}_{u(v+1)kl}+\sum_{\substack{(k,l):\\ l\leqslant k\leqslant u-1}} \tilde{\gamma}_{lk(u-1)v}+\theta_1 f(C)_{uv}, && \forall u\leqslant v,\\
&\tilde{\gamma}_{ui^tj^tv}\leqslant\theta_1 f(C)_{uv}, && \forall u\leqslant i^t\leqslant j^t\leqslant v,\\
&\tilde{\gamma}_{uijv}\in\mathbb{N}, && \forall u\leqslant i\leqslant j\leqslant v.
\end{aligned}$$

Set $\gamma^*_{uijv} = \frac{1}{\theta_1}\tilde{\gamma}_{uijv}$ completes our Step 2. Thus, the proof is completed.

□

Proof of Lemma 3. The proof is based on the following equalities. For any $t \geqslant 1$, we have

$$\begin{aligned} d_{ij}^{[1,t)} &= \sum_{\substack{(u,v):\\ u\leqslant i\leqslant j\leqslant v}} \gamma_{uijv}^{[1,t)} + \gamma_{0ij0}^{[1,t)}, \quad \forall i \leqslant j, \\ A_{uv}^{t} - A_{uv}^{1} &= \sum_{\substack{(k,\ell):\\ v+1\leqslant k\leqslant \ell}} \gamma_{u(v+1)k\ell}^{[1,t)} + \sum_{\substack{(k,\ell):\\ \ell\leqslant k\leqslant u-1}} \gamma_{\ell k(u-1)v}^{[1,t)} - \sum_{\substack{(i,j):\\ u\leqslant i\leqslant j\leqslant v}} \gamma_{uijv}^{[1,t)}, \quad \forall u \leqslant v. \end{aligned}$$

Then for any optimal solution γ^* of $OPT(A^{t_1}, d + d^{[t_1,t_2)}, \gamma^{[t_1,t_2)})$, $\gamma^* + \gamma^{[1,t_1)}$ is a feasible solution of $OPT(A^1, d + d^{[1,t_2)}, \gamma^{[1,t_2)})$. Also, for any optimal solution γ^* of $OPT(A^1, d + d^{[1,t_2)}, \gamma^{[1,t_2)})$, $\gamma^* - \gamma^{[1,t_1)}$ is a feasible solution of $OPT(A^{t_1}, d + d^{[t_1,t_2)}, \gamma^{[t_1,t_2)})$. The proof is completed.

□

Proof of Lemma 4. The left-hand side is trivial, since $\gamma^{[1,t+1)} \geqslant \gamma^{[1,t)}$ leads to a smaller feasible region of $OPT(A^1, d^{[1,T]}, \gamma^{[1,t+1)})$ than $OPT(A^1, d^{[1,T]}, \gamma^{[1,t)})$.

Now we consider the right-hand side. We first prove that $OPT(A^1, d^{[1,T]}, \gamma^{[1,t)})$ is equivalent to solving (EC.10),

$$\begin{aligned} \max_{\gamma} \quad & \sum_{i\leqslant j}\left(p_{ij}\sum_{\substack{(u,v):\\ u\leqslant i\leqslant j\leqslant v}}\gamma_{uijv}\right) - \sum_{u\leqslant v}\beta_{uv}\left(\sum_{\substack{(i,j):\\ u\leqslant i\leqslant j\leqslant v}}\gamma_{uijv} - \sum_{\substack{(k,\ell):\\ v+1\leqslant k\leqslant \ell}}\gamma_{u(v+1)k\ell} - \sum_{\substack{(k,\ell):\\ \ell\leqslant k\leqslant u-1}}\gamma_{\ell k(u-1)v} - A_{uv}^1\right)^+, \\ \text{s.t.} \quad & \sum_{\substack{(u,v):\\ u\leqslant i\leqslant j\leqslant v}}\gamma_{uijv} + \gamma_{0ij0} = d_{ij}^{[1,T]}, \quad \forall i \leqslant j, \\ & \gamma_{uijv} \geqslant \gamma_{uijv}^{[1,t)}, \quad \forall u \leqslant i \leqslant j \leqslant v, \\ & \gamma_{0ij0} \geqslant \gamma_{0ij0}^{[1,t)}, \quad \forall i \leqslant j. \end{aligned} \tag{EC.10}$$

with fixed and appropriately chosen β such that

$$\beta_{uv} > \beta_{u(i-1)} + p_{ij} + \beta_{(j+1)v}, \quad \forall u \leqslant i \leqslant j \leqslant v.$$

We denote (EC.10) as $\widetilde{OPT}(A^1, d^{[1,T]}, \gamma^{[1,t)})$. We only need to prove that the optimal solution of $\widetilde{OPT}(A^1, d^{[1,T]}, \gamma^{[1,t)})$ must satisfy the additional constraints of $OPT(A^1, d^{[1,T]}, \gamma^{[1,t)})$. Otherwise, we can select some $\gamma_{u'i'j'v'}$ such that

$$\begin{aligned} & \gamma_{u'i'j'v'} > \gamma_{u'i'j'v'}^{[1,t)}, \\ & \sum_{\substack{(i,j):\\ u'\leqslant i\leqslant j\leqslant v'}}\gamma_{u'ijv'} - \sum_{\substack{(k,\ell):\\ v'<k\leqslant \ell}}\gamma_{u(v'+1)k\ell} - \sum_{\substack{(k,\ell):\\ \ell\leqslant k<u'}}\gamma_{\ell k(u'-1)v} - A_{u'v'}^1 > 0 \end{aligned}$$

Let $f(\gamma; A^1)$ denote the objective function in $\widetilde{OPT}(A^1, d^{[1,T]}, \gamma^{[1,t)})$, then f is uniformly Lipschitz with γ under the L_1 norm by some positive constant

$$l = \max_{u\leqslant i\leqslant j\leqslant v}\left\{\beta_{u(i-1)} + p_{ij} + \beta_{(j+1)v}\right\},$$

regardless of what A^1 is. Note that l is dependent only on $\{p_{ij}\}$. Then $\partial f/\partial\gamma_{u'i'j'v'} \leqslant \beta_{u'(i'-1)} + p_{i'j'} + \beta_{(j'+1)v'} - \beta_{u'v'} < 0$, indicating that we can decrease $\gamma_{u'i'j'v'}$ and increase $\gamma_{0i'j'0}$ a bit such that f strictly increases while the constraints of $\widetilde{OPT}(A^1, d, \gamma^{[1,t)})$ are still satisfied. Thus, OPT and $\widetilde{OPT}$ have the same objective value.

Now let's construct a feasible solution $\gamma^{\text{fea},t+1}$ of $\widetilde{OPT}(A^1, d^{[1,T]}, \gamma^{[1,t+1)})$ from the optimal solution $\gamma^{*,t}$ of $\widetilde{OPT}(A^1, d^{[1,T]}, \gamma^{[1,t)})$ such that $\|\gamma^{\text{fea},t+1} - \gamma^{*,t}\|_1 \leqslant 2$. Notice that $\gamma^{[1,t)}$ and $\gamma^{[1,t+1)}$ differs at exactly one component with

$$\gamma_{uijv}^{[1,t+1)} = \begin{cases} \gamma_{uijv}^{[1,t)} + 1, & \text{if } (u,i,j,v) = (u^t, i^t, j^t, v^t) \\ \gamma_{uijv}^{[1,t)}, & \text{otherwise.} \end{cases}$$

We let

$$\gamma_{uijv}^{\text{fea},t+1} = \begin{cases} \gamma_{uijv}^{*,t} + \max\{\gamma_{uijv}^{[1,t+1)} - \gamma_{uijv}^{*,t}, 0\}, & \text{if } (u,i,j,v) = (u^t, i^t, j^t, v^t) \\ \gamma_{uijv}^{*,t} - (\gamma_{uijv}^{*,t} - \gamma_{uijv}^{[1,t)})\epsilon, & \text{if } (i,j) = (i^t, j^t), (u,v) \neq (u^t, v^t), \\ \gamma_{uijv}^{*,t}, & \text{otherwise,} \end{cases}$$

where

$$\epsilon = \frac{\max\{\gamma_{u^t i^t j^t v^t}^{[1,t+1)} - \gamma_{u^t i^t j^t v^t}^{*,t}, 0\}}{\sum_{(u,v)\neq(u^t,v^t)}(\gamma_{ui^t j^t v}^{*,t} - \gamma_{ui^t j^t v}^{[1,t)})} = \max\left\{0, \frac{\gamma_{u^t i^t j^t v^t}^{[1,t+1)} - \gamma_{u^t i^t j^t v^t}^{*,t}}{d_{i^t j^t}^{[1,T]} - \gamma_{u^t i^t j^t v^t}^{*,t} - \sum_{(u,v)\neq(u^t,v^t)} \gamma_{ui^t j^t v}^{[1,t+1)}}\right\} \leqslant 1.$$

Here ϵ is chosen to maintain the equality constraints in (EC.10). $\epsilon \leqslant 1$ ensures that $\gamma_{uijv}^{\text{fea},t+1} \geqslant \gamma_{uijv}^{[1,t)} = \gamma_{uijv}^{[1,t+1)}$ for all $(u,i,j,v) \neq (u^t, i^t, j^t, v^t)$. Therefore,

$$\begin{aligned} &OPT(A^1, d^{[1,T]}, \gamma^{[1,t)}) - OPT(A^1, d^{[1,T]}, \gamma^{[1,t+1)}) \\ &= \widetilde{OPT}(A^1, d^{[1,T]}, \gamma^{[1,t)}) - \widetilde{OPT}(A^1, d^{[1,T]}, \gamma^{[1,t+1)}) \\ &\leqslant l\|\gamma^{\text{fea},t+1} - \gamma^{*,t}\|_1 \leqslant 2l. \end{aligned}$$

□

Proof of Theorem 5. As a preliminary step, we demonstrate that

$$\inf_{\substack{t\in[\theta T], i\leqslant j: \\ \lambda_{ij}^t > 0}} \frac{\lambda_{ij}^{[t,T]}}{\theta T - t + 1}$$

is lower bounded by some positive constant

$$\lambda_{\min} \triangleq \inf_{\substack{t\in[T], i\leqslant j: \\ \lambda_{ij}^t > 0}} \frac{\lambda_{ij}^T}{T},$$

irrelevant to θ. In fact, when $t \leqslant (\theta-1)T$,

$$\inf_{\substack{i\leqslant j: \\ \lambda_{ij}^t > 0}} \frac{\lambda_{ij}^{[t,\theta T]}(\theta)}{\theta T - t + 1} \geqslant \inf_{\substack{i\leqslant j: \\ \lambda_{ij}^{[1,T]} > 0}} \frac{\theta\lambda_{ij}^T}{\theta T} \geqslant \lambda_{\min}.$$

When $(\theta-1)T < t \leqslant \theta T$,

$$\inf_{\substack{i\leqslant j: \\ \lambda_{ij}^t > 0}} \frac{\lambda_{ij}^{[t,\theta T]}(\theta)}{\theta T - t + 1} \geqslant \inf_{\substack{i\leqslant j: \\ \lambda_{ij}^{[1,T]} > 0}} \frac{(\theta T - t + 1)\lambda_{ij}^T}{\theta T - t + 1} \geqslant T\lambda_{\min}.$$

Therefore, we have

$$\inf_{\substack{t\in[\theta T],i\leqslant j:\\ \lambda_{ij}^t>0}} \frac{\lambda_{ij}^{[t,T]}}{\theta T-t+1} \geqslant \lambda_{\min}>0.$$

Now we proceed to the formal proof. Let $\theta > \frac{2}{T}\lfloor \frac{(1+\delta)(M+1)^2}{2\lambda_{\min}} \rfloor$ be any scaling parameter. First, the loss can be upper bounded by

$$\begin{aligned}
&\mathbb{E}\left[OPT(A^1, d^{[1,\theta T]}, \gamma^{[1,1)})\right] - \mathbb{E}\left[OPT(A^1, d^{[1,\theta T]}, \gamma^{[1,\theta T]})\right] \\
=&\sum_{t=1}^{T} \mathbb{E}\left[OPT(A^1, d^{[1,\theta T]}, \gamma^{[1,t)}) - OPT(A^1, d^{[1,\theta T]}, \gamma^{[1,t+1)})\right] \\
=&\sum_{t=1}^{T} \mathbb{E}\left[OPT(A^t, d^{[t,\theta T]}, 0) - OPT(A^t, d^{[t,\theta T]}, \gamma^t)\right] \\
\leqslant& 2l\sum_{t=1}^{T} \mathbb{P}\left(OPT(A^t, d^{[t,\theta T]}, 0) > OPT(A^t, d^{[t,\theta T]}, \gamma^t)\right)
\end{aligned}$$

For given t and $OPT(A^t, d, 0)$, we note that in RDP policy $d=\lambda^{[t,\theta T]}$, while in the sample path hindsight optimum $d=d^{[t,\theta T]}$. From Lemma 5, we can find $\gamma^{*,t}$ such that it is δ-insensitive with the solution $\gamma^{\mathrm{RDP},t}$ obtained in RDP policy. If $\gamma^{*,t}_{u^t i^t j^t v^t} \geqslant 1$, then $\gamma^{*,t}$ is still feasible for $OPT(A^t, d^{[t,\theta T]}, \gamma^t)$, so the two OPTs must equal.

Based on the maximum choice of (u^t, v^t) in RDP, we have $\gamma^{\mathrm{RDP},t}_{u^t i^t j^t v^t} \geqslant (\sum_{(u,v)} \gamma^{\mathrm{RDP},t}_{u i^t j^t v})/(\sum_{(u,v)} 1) \geqslant \lambda^{[t,\theta T]}_{i^t j^t}/((M+1)^2/4)$. Therefore, the loss can be further bounded as follows.

$$\begin{aligned}
&\mathbb{E}\left[OPT(A^1, d^{[1,\theta T]}, \gamma^{[1,1)})\right] - \left[OPT(A^1, d^{[1,\theta T]}, \gamma^{[1,\theta T]})\right] \\
\leqslant& 2l\mathbb{P}\left(\bigcup_{t=1}^{\theta T}\left\{\gamma^{*,t}_{u^t i^t j^t v^t} < 1\right\}\right) \\
\leqslant& 2l\sum_{t=1}^{\theta T}\mathbb{P}\left(\gamma^{*,t}_{u^t i^t j^t v^t} < \gamma^{\mathrm{RDP},t}_{u^t i^t j^t v^t} + 1 - \frac{\lambda^{[t,\theta T]}_{i^t j^t}}{(M+1)^2/4}\right) \\
\leqslant& 2l\sum_{t=1}^{\theta T}\mathbb{P}\left(\max_{i'\leqslant j'}\left|\mathbb{E}[d^{[t,\theta T]}_{i'j'}] - d^{[t,\theta T]}_{i'j'}\right| > \frac{\mathbb{E}[d_{i^t j^t}[t,\theta T]] - (M+1)^2/4}{\delta(M+1)^2/4}\right)
\end{aligned}$$

Let $T_0 = \lceil \frac{(M+1)^2}{2\lambda_{\min}} \rceil < \frac{\theta T}{2}$. For any arrival process,

$$\mathbb{E}[d^{[t,\theta T]}_{ij}] = \lambda^{[t,\theta T]}_{ij} \geqslant \lambda_{\min}(\theta T - t + 1)$$

if $\mathbb{P}\left((i^t, j^t) = (i,j)\right) = \lambda^t_{ij} > 0$. Then

$$\begin{aligned}
&\mathbb{E}\left[OPT(A^1, d^{[1,\theta T]}, \gamma^{[1,1)})\right] - \mathbb{E}\left[OPT(A^1, d^{[1,\theta T]}, \gamma^{[1,\theta T]})\right] \\
\leqslant& 2l\sum_{t=1}^{\theta T-T_0}\sum_{i'\leqslant j'}\mathbb{P}\left(\left|\mathbb{E}[d^{[t,\theta T]}_{i'j'}] - d^{[t,\theta T]}_{i'j'}\right| > \frac{\mathbb{E}[d_{i^t j^t}[t,\theta T]] - (M+1)^2/4}{\delta(M+1)^2/4}\right) + 2l\sum_{t=\theta T-T_0+1}^{T} 1 \\
\leqslant& 2l\sum_{t=1}^{\theta T-T_0}\frac{(M+1)^2}{4} 2\exp\left(-2\frac{(\lambda_{\min}(\theta T-t+1) - (M+1)^2/4)^2}{(\delta(M+1)^2/4)^2(\theta T-t+1)}\right) + 2l\sum_{t=\theta T-T_0+1}^{T} 1 \\
\leqslant& 2lT_0 + 2l\sum_{t=T_0+1}^{\theta T}\frac{(M+1)^2}{2}\exp\left(-\frac{8\lambda^2_{\min} t}{\delta^2(M+1)^4}\right) \\
=& \lambda^{-2}_{\min} O(1).
\end{aligned}$$

We note that for any stationary arrival process, the $\lambda_{\min}^{-2}$ above can be improved to $\lambda_{\min}^{-1}$. Let $T_0 = \lceil \frac{(1+\delta)(M+1)^2}{2\lambda_{\min}} \rceil < \frac{\theta T}{2}$, then we have

$$
\begin{aligned}
&\mathbb{E}\left[OPT(A^1, d^{[1,\theta T]}, \gamma^{[1,1)})\right] - \mathbb{E}\left[OPT(A^1, d^{[1,\theta T]}, \gamma^{[1,\theta T]})\right] \\
&\leqslant 2lT_0 + 2l \sum_{t=1}^{\theta T - T_0} \sum_{i' \leqslant j'} \sum_{i \leqslant j} \mathbb{P}\left(\left| \mathbb{E}[d_{i'j'}^{[t,\theta T]}] - d_{i'j'}^{[t,\theta T]} \right| > \frac{\mathbb{E}[d_{i^t j^t}^{[t,\theta T]}] - (M+1)^2/4}{\delta (M+1)^2/4} \middle| (i^t, j^t) = (i,j) \right) \mathbb{P}\left((i^t, j^t) = (i,j)\right) \\
&= 2lT_0 + 2l \sum_{i \leqslant j} \mathbb{P}\left((i^t, j^t) = (i,j)\right) \sum_{t=1}^{\theta T - T_0} \sum_{i' \leqslant j'} \mathbb{P}\left(\left| \mathbb{E}[d_{i'j'}^{(t,\theta T]}] - d_{i'j'}^{(t,\theta T]} \right| > \frac{\mathbb{E}[d_{ij}^{[t,\theta T]}] - (1+\delta)(M+1)^2/4}{\delta (M+1)^2/4} \right) \\
&\leqslant 2lT_0 + 2l \sum_{i \leqslant j} \mathbb{P}\left((i^t, j^t) = (i,j)\right) \sum_{t=1}^{\theta T - T_0} \frac{(M+1)^2}{4} 2 \exp\left(-2 \frac{(\lambda_{ij}^1(\theta T - t + 1) - (1+\delta)(M+1)^2/4)^2}{(\delta (M+1)^2/4)^2 (\theta T - t)} \right) \\
&\leqslant 2lT_0 + \sum_{i \leqslant j} \lambda_{ij}^1 \sum_{t=T_0+1}^{\theta T} \frac{(M+1)^2}{2} \exp\left(-\frac{8(\lambda_{ij}^1)^2 t}{\delta^2 (M+1)^4} \right) \\
&= 2lT_0 + \sum_{i \leqslant j : \lambda_{ij}^1 > 0} (\lambda_{ij}^1)^{-1} O(1) = \lambda_{\min}^{-1} O(1).
\end{aligned}
$$

□

EC.3. Implementation Details for Numerical Experiments in Section 5

Computation of BPC Policies In BPC-S, directly solving (5) may be computationally intensive, since the size of the problem is proportional to N. We adopt the idea of *maximal sequence* to simplify the computation of (5). Observe first that when $C_{k\ell} = 0$, we can set β_{kl} to $+\infty$. Second, same type of maximal sequence on different seats should have the same bid-prices. That is to say, for fixed (u, v), we consider the bid-prices on any seat $k \in \mathcal{M}_{uv}(C)$ and let $\beta_{k\ell} \leftarrow \frac{1}{|\mathcal{M}_{uv}(C)|} \sum_{k' \in \mathcal{M}_{uv}(C)} \beta_{k'\ell}$. These two operations will not violate the constraints nor increase the objective value. Thus, we instead solve the following program:

$$
\begin{aligned}
\text{minimize}_{z,\beta} \quad & \sum_{1 \leqslant i \leqslant j \leqslant M} \lambda_{ij} z_{ij} + \sum_{u \leqslant v} \sum_{\ell : u \leqslant \ell \leqslant v} \beta_{uv,\ell} f(C)_{uv} && && \text{(EC.11)} \\
\text{subject to} \quad & z_{ij} + \sum_{\ell=i}^{j} \beta_{uv,\ell} \geqslant p_{ij}, && \forall u \leqslant i \leqslant j \leqslant v, \\
& z_{ij} \geqslant 0, \beta_{uv,\ell} \geqslant 0, && \forall 1 \leqslant i \leqslant j \leqslant M, 1 \leqslant u \leqslant \ell \leqslant v \leqslant M,
\end{aligned}
$$

where $\beta_{uv,\ell}$ can be interpreted as the bid price of leg ℓ if it is unoccupied in a $[u, v] \sim C$. After we've computed $\{\beta_{uv,\ell}\}$, we can then dynamically allocate the requests on maximal sequences without assigning the seat number explicitly.

In our experiments, we use `cvxpy` with GUROBI 9.0.2 to obtain bid-prices.

Ticket Prices of G315 We list the ticket prices (measured in RMB) of G315 and the corresponding distance between each pair of stops (presented in the brackets, measured in kilometers) in Table EC.1.

Start \ End	Qufu	Tengzhou	Xuzhou	Shangqiu	Zhengzhou	Luohe	Zhumadian	Xinyang	Hankou	Jingzhou	Yichang	Enshi	Fuling	Chongqing
Jinan	59.5	84	134.5	210.5	298	368.5	394.5	443.5	526	602	634	696.5	769	796
	(132)	(188)	(289)	(459)	(649)	(804)	(868)	(986)	(1189)	(1393)	(1481)	(1695)	(1942)	(2034)
Qufu		24	69.5	150	237.5	308	334	383	465.5	541.5	573.5	636	708.5	735.5
		(56)	(157)	(327)	(517)	(672)	(736)	(854)	(1057)	(1261)	(1349)	(1563)	(1810)	(1902)
Tengzhou			44	124	212	282	308	357	440	516	548	610	683	710
			(101)	(271)	(461)	(616)	(680)	(798)	(1001)	(1205)	(1293)	(1507)	(1754)	(1846)
Xuzhou				78	165.5	236	262	311	393.5	469.5	501.5	564	636.5	663.5
				(170)	(360)	(515)	(579)	(697)	(900)	(1104)	(1192)	(1406)	(1653)	(1745)
Shangqiu					87	158.5	188	240.5	323.5	399.5	431.5	494	566.5	593.5
					(190)	(345)	(409)	(527)	(730)	(934)	(1022)	(1236)	(1483)	(1575)
Zhengzhou						70.5	100.5	154.5	245	321	353	415.5	488	515
						(155)	(219)	(337)	(540)	(744)	(832)	(1046)	(1293)	(1385)
Luohe							28.5	83.5	174.5	250.5	282.5	345	417.5	444.5
							(64)	(182)	(385)	(589)	(677)	(891)	(1138)	(1230)
Zhumadian								53.5	145	221	257	315.5	388	415
								(118)	(321)	(525)	(613)	(827)	(1074)	(1166)
Xinyang									91	167	199	261.5	334	361
									(203)	(407)	(495)	(709)	(956)	(1048)
Hankou										76	108	170.5	243	270
										(204)	(292)	(506)	(753)	(845)
Jingzhou											33	95.5	168	195
											(88)	(302)	(549)	(641)
Yichang												62.5	135	162
												(214)	(461)	(553)
Enshi													72	99.5
													(247)	(339)
Fuling														27
														(92)

Table EC.1 Ticket prices of G315