Dynamic Seat Assignment With Social Distancing

February 1, 2024

Abstract

This study addresses the dynamic seat assignment problem with social distancing, which arises when groups arrive at a venue and need to be seated together while respecting minimum physical distance requirements. To tackle this challenge, we develop a scenario-based stochastic programming for generating seat planning and propose a seat assignment policy for accepting or denying arriving groups. We found that our approach performs better than the traditional bid-price policy and booking-limit policy, even well compared with the offline optimal solution. The results provide insights for policymakers and venue managers on seat utilization rates and offer a practical tool for implementing social distancing measures while optimizing seat assignments and ensuring group safety.

Keywords: Social Distancing, Scenario-based Stochastic Programming, Seat Assignment, Dynamic Arrival.

1 Introduction

Governments worldwide have been faced with the challenge of reducing the spread of Covid-19 while minimizing the economic impact. Social distancing has been widely implemented as the most effective non-pharmaceutical treatment to reduce the health effects of the virus. This website records a timeline of Covid-19 and the relevant epidemic prevention measures [18]. For instance, in March 2020, the Hong Kong government implemented restrictive measures such as banning indoor and outdoor gatherings of more than four people, requiring restaurants to operate at half capacity. As the epidemic worsened, the government tightened measures by limiting public gatherings to two people per group in July 2020. As the epidemic subsided, the Hong Kong government gradually relaxed social distancing restrictions, allowing public group gatherings of up to four people in September 2020. In October 2020, pubs were allowed to serve up to four people per table, and restaurants could serve up to six people per table. Specifically, the Hong Kong government also implemented different measures in different venues [14]. For example, the catering businesses will have different social distancing requirements depending on their mode of operation for dine-in services. They can operate at 50%, 75%, or 100% of their normal seating capacity at any one time, with a maximum of 2, 2, or 4 people per table, respectively. Bars and pubs may open with a maximum of 6 persons per table and a total number of patrons capped at 75% of their capacity.

The restrictions on the number of persons allowed in premises such as cinemas, performance venues, museums, event premises, and religious premises will remain at 85% of their capacity.

The measures announced by the Hong Kong government mainly focus on limiting the number of people in each group and the seat occupancy rate. However, implementing these policies in operations can be challenging, especially for venues with fixed seating layouts. In our study, we will focus on addressing this challenge in commercial premises, such as cinemas and music concert venues. We aim to provide a practical tool for venues to optimize seat assignments while ensuring the safety of groups by proposing a seat assignment policy that takes into account social distancing requirements and the given seating layout. We strive to enable venues to implement social distancing measures effectively by offering a solution that provides specific seating arrangements.

To avoid confusion regarding the terms 'seat planning' and 'seat assignment', it is important to clarify the distinction between them. In our context, seat planning involves determining the arrangement of seats within a venue or space based on the size or layout of the room. This includes deciding on seats partition with social distancing. On the other hand, seat assignment involves the specific allocation of seats to individual attendees or groups based on seat availability. This task is typically performed when the seller needs to make a decision each time there is a request.

This paper focuses on addressing the dynamic seating assignment problem with a given set of seats in the context of social distancing. The government issues a maximum number of people allowed in each group which must be implemented in the seat planning. The problem becomes further complicated by the existence of groups of guests who must sit together. To address this challenge, we have developed a scenario-based stochastic programming for seat planning. Our goal is to obtain the seat planning that satisfies social distancing constraints and implement the seat assignment when groups arrive based on this seat planning. The proposed algorithm has the potential to help sellers optimize seat assignments while maintaining social distancing measures. Overall, our study offers a comprehensive solution for dynamic seat assignment with social distancing in the context of a pandemic.

Our main contributions in this paper are summarized as follows. First, this study presents the first attempt to consider the arrangement of seat assignments with social distancing under dynamic arrivals. While many studies in the literature highlight the importance of social distancing in controlling the spread of the virus, they often focus too much on the model and do not provide much insight into the operational significance behind social distancing [1,11]. Recent studies have explored the effects of social distancing on health and economics, mainly in the context of aircraft [13, 27, 28]. Our study provides a new perspective to help the government adopt a mechanism for setting seat assignments to protect people during pandemic.

Second, we establish a deterministic model to analyze the effects of social distancing when the demand is known. Due to the medium size of the problem, we can solve the IP model directly. We then develop the scenario-based stochastic programming by considering the stochastic demands of different group types. By using Benders decomposition methods, we can obtain the optimal linear solution.

Third, to address the problem in the dynamic situation, we first obtain a feasible seat planning from scenario-based stochastic programming. We then make a decision for each incoming group based on our

seat assignment policy, either accepting or rejecting the group. Our results demonstrate a significant improvement over the traditional control policies and provide the insights on the implementation of social distancing.

The rest of this paper is structured as follows. The following section reviews relevant literature. We describe the motivating problem in Section 3. In Section 4, we establish the stochastic model,analyze its properties and give the seat planning. Section 5 demonstrates the dynamic seat assignment policy. Section 6 gives the results and the insights. The conclusions are shown in Section 7.

2 Literature Review

The present study is closely connected to the following research areas – seat planning with social distancing and dynamic seat assignment. The subsequent sections review literature pertaining to each perspective and highlight significant differences between the present study and previous research.

2.1 Seat Planning with Social Distancing

Since the outbreak of covid-19, social distancing is a well-recognized and practiced method for containing the spread of infectious diseases [25]. An example of operational guidance is ensuring social distancing in seat plannings.

Social distancing in seat planning has attacted considerable attention from the research area. The applications include the allocation of seats on airplanes [13], classroom layout planning [4], seat planning in long-distancing trains [16]. The social distancing can be implemented in various forms, such as fixed distances or seat lengths. Fischetti et al. [11] consider how to plant positions with social distancing in restaurants and beach umbrellas. Different venues may require different forms of social distancing; for instance, on an airplane, the distancing between seats and the aisle must be considered [27], while in a classroom, maximizing social distancing between students is a priority [4].

These researchs focus on the static version of the problem. This typically involves creating an IP model with social distancing constraints([4, 13, 16]), which is then solved either heuristically or directly. The seat allocation of the static form is useful for fixed people, for example, the students in one class. But it is not be practical for the dynamic arrivals in commercial events.

The recent pandemic has shed light on the benefits of group reservations, as they have been shown to increase revenue without increasing the risk of infection [24]. In our specific setting, we require that groups be accepted on an all-or-none basis, meaning that members of the same family or group must be seated together. However, the group seat reservation policy poses a significant challenge when it comes to determining the seat assignment policy.

This group seat reservation policy has various applications in industries such as hotels [23], working spaces [11], public transport [9], sports arenas [21], and large-scale events [22]. This policy has significant impacts on passenger satisfaction and revenue, with the study [31] showing that passenger groups increase revenue by filling seats that would otherwise be empty. Traditional works [6, 9]in transportation focus on maximizing capacity utilization or reducing total capacity needed for passenger rail, typically modeling these problems as knapsack or binpacking problems.

Some related literature mentioned the seat planning under pandemic for groups are represented below. Fischetti et al. [11] proposed a seating planning for known groups of customers in amphitheaters. Haque and Hamid [16] considers grouping passengers with the same origin-destination pair of travel and assigning seats in long-distance passenger trains. Salari et al. [27] performed group seat assignment in airplanes during the pandemic and found that increasing passenger groups can yield greater social distancing than single passengers. Haque and Hamid [17] aim to optimize seating assignments on trains by minimizing the risk of virus spread while maximizing revenue. The specific number of groups in their

models is known in advance. But in our study, we only know the arrival probabilities of different groups.

This paper [3] discusses strategies for filling a theater by considering the social distancing and group arrivals, which is similar to ours. However, unlike our project, it only focuses on a specific location layout and it is still based on a static situation by giving the proportion of different groups.

2.2 Dynamic Seat Assignment

Our model in its static form can be viewed as a specific instance of the multiple knapsack problem [26], where we aim to assign a subset of groups to some distinct rows. In our dynamic form, the decision to accept or reject groups is made at each stage as they arrive. The related problem can be dynamic knapsack problem [20], where there is one knapsack.

Dynamic seat assignment is a process of assigning seats to passengers on a transportation vehicle, such as an airplane, train, or bus, in a way that maximizes the efficiency and convenience of the seating arrangements [2, 15, 32].

Our problem is closely related to the network revenue management (RM) problem [30], which is typically formulated as a dynamic programming (DP) problem. However, for large-scale problems, the exponential growth of the state space and decision set makes the DP approach computationally intractable. To address this challenge, we propose using scenario-based programming [5, 10, 19] to determine the seat planning. In this approach, the aggregated supply can be considered as a protection level for each group type. Notably, in our model, the supply of larger groups can also be utilized by smaller groups. This is because our approach focuses on group arrival rather than individual unit, which sets it apart from traditional partitioned and nested approaches [7, 29].

Traditional revenue management focuses on decision-making issues, namely accepting or rejecting a request [12].However, our paper not only addresses decision-making, but also emphasizes the significance of assignment, particularly in the context of seat assignment. This sets it apart from traditional revenue management methods and makes the problem more challenging.

Similarly, the assign-to-seat approach introduced by Zhu et al. [32] also highlights the importance of seat assignment in revenue management. This approach addresses the challenge of selling high-speed train tickets in China, where each request must be assigned to a single seat for the entire journey and takes into account seat reuse. This further emphasizes the significance of seat assignment and sets it apart from traditional revenue management methods.

3 Problem Description

In this section, to incorporate the social distancing into seat planning, we first give the description of the seat planning problem with social distancing. Then we introduce the dynamic seat assignment problem with social distancing.

3.1 Seat Planning Problem with Social Distancing

We consider a layout comprising N rows, with each row containing L_j^0 seats, where $j \in \mathcal{N} \coloneqq \{1, 2, \ldots, N\}$. The seating arrangement is used to accommodate various groups, where each group consists of no more than M individuals. There are M distinct group types, denoted by group type i, where each group type consists of i people. The set of all group types is denoted by $\mathcal{M} \coloneqq \{1, 2, \ldots, M\}$. The demand for each group type is represented by a demand vector $\mathbf{d} = (d_1, d_2, \ldots, d_M)^\intercal$, where d_i represents the number of group type i.

In order to comply with the social distancing requirements, individuals from the same group must sit together, while maintaining a distance from other groups. Let δ denote the social distancing, which could entail leaving one or more empty seats. Specifically, each group must ensure the empty seat(s) with the adjacent group(s).

To model the social distancing requirements into the seat planning process, we add the parameter, δ, to the original group sizes, resulting in the new size of group type i being denoted as $n_i = i + \delta$, where $i \in \mathcal{M}$. Accordingly, the length of each row is also adjusted to accommodate the adjusted group sizes. Consequently, $L_j = L_j^0 + \delta$ represents the length of row j, where L_j^0 indicates the number of seats in row j. By incorporating the additional seat(s) and designating certain seat(s) for social distancing, we can integrate social distancing measures into the seat planning problem.

Let x_{ij} represent the number of group type i planned in row j. The deterministic seat planning problem is formulated below, with the objective of maximizing the number of people accommodated.

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{M} \sum_{j=1}^{N} (n_i - \delta) x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{N} x_{ij} \le d_i, \quad i \in \mathcal{M}, \\
& \sum_{i=1}^{M} n_i x_{ij} \le L_j, j \in \mathcal{N}, \\
& x_{ij} \in \mathbb{Z}_+, \quad i \in \mathcal{M}, j \in \mathcal{N}.
\end{aligned} \tag{1}
$$

This seat planning problem can be regarded as a special case of the multiple knapsack problem. In this context, we define X as the aggregate solution, where $X = (\sum_{j=1}^{N} x_{1j}, \ldots, \sum_{j=1}^{N} x_{Mj})^T$. Each element of X, $\sum_{j=1}^{N} x_{ij}$, represents the available supply for group type i.

In other words, X captures the number each group type that can be allocated to the seat layout by summing up the supplies across all rows. By considering the monotone ratio between the original group sizes and the adjusted group sizes, we can determine the upper bound of supply corresponding to the

optimal solution of the LP relaxation of Problem (1), as demonstrated in Proposition 2.

Although the problem size is small and the optimal solution can be easily obtained using a solver, it is still important to analyze the problem further to gain additional insights and understanding. We introduce the term pattern to refer to the seat planning arrangement for a single row. A specific pattern can be represented by a vector $\boldsymbol{h} = (h_1, \ldots, h_M)$, where h_i represents the number of group type i in the row for $i = 1, \ldots, M$. This vector $\boldsymbol{h}$ must satisfy the condition $\sum_{i=1}^{M} h_i n_i \leq L$ and belong to the set of non-negative integer values, denoted as $\boldsymbol{h} \in \mathbb{Z}_+^M$. Then a seat planning with N rows can be represented by $\boldsymbol{H} = \{\boldsymbol{h}_1; \ldots; \boldsymbol{h}_N\}$, where H_{ji} represents the number of group type i in pattern j.

Let $|\boldsymbol{h}|$ indicate the number of people that can be assigned according to pattern $\boldsymbol{h}$, i.e., $|\boldsymbol{h}| = \sum_{i=1}^{M} i h_i$. We also introduce the concept of loss, which is the number of unoccupied seats. Mathematically, the loss is defined as $L - \delta - |\boldsymbol{h}|$, where L denotes the length of the row. The loss provides a measure of the number of seats which cannot be taken due to the implementation of social distancing constraints. By examining the losses associated with different patterns, we can assess the effectiveness of various seat planning configurations with respect to accommodating the desired number of individuals while adhering to social distancing requirements.

Definition 1. *Given the length of a row, denoted as L, and the maximum size of a group allowed, denoted as M, we can define certain characteristics of a pattern $\boldsymbol{h} = (h_1, \ldots, h_M)$. We refer to a pattern $\boldsymbol{h}$ as a full pattern if it satisfies the condition $\sum_{i=1}^{M} n_i h_i = L$. In other words, a full pattern is one in which the sum of the product of the number of occurrences h_i and the size n_i of each group in the pattern is equal to the length of the row L. This ensures that the pattern fully occupies the available row seats. Furthermore, we define a pattern $\boldsymbol{h}$ as a largest pattern if it has a size $|\boldsymbol{h}|$ that is greater than or equal to the size $|\boldsymbol{h}'|$ of any other feasible pattern $\boldsymbol{h}'$. In other words, a largest pattern is one that either has the maximum size or is equal in size to other patterns, ensuring that it can accommodate the most number of people within the given row length.*

In many cases, the optimal solution for the seat planning problem tends to involve rows with either full patterns or the largest patterns. Distinguishing these patterns from other configurations can provide valuable insights into effective seat planning strategies that prioritize accommodating as many people as possible while adhering to social distancing guidelines.

When there is high demand for seats, it is advantageous to prioritize the largest patterns. These patterns allow for the accommodation of the largest number of individuals due to social distancing requirements. On the other hand, in scenarios with moderate demand, adopting the full pattern becomes more feasible. The full pattern maximizes seating capacity by utilizing all available seats, except those empty seats needed for social distancing measures. By considering both the largest and full patterns, we can optimize seat planning configurations to efficiently accommodate a significant number of individuals while maintaining adherence to social distancing guidelines.

Proposition 1. *Given the parameters of a row, including its length L, the social distancing requirement δ, and the maximum size of a group allowed M, for one possible largest pattern $\boldsymbol{h}$, the maximum number of people that can be accommodated is given by $|\boldsymbol{h}| = qM + \max\{r - \delta, 0\}$, where $q = \lfloor \frac{L}{M+\delta} \rfloor$,*

$r \equiv L \bmod (M+\delta)$. *The corresponding loss of the largest pattern equals* $q\delta - \delta + \min\{r, \delta\}$, *represents the amount of empty seats due to the social distancing requirement.*

The largest pattern $\boldsymbol{h}$ is unique and full when $r = 0$, indicating that only one pattern can accommodate the maximal number of people. On the other hand, if $r > \delta$, the largest pattern $\boldsymbol{h}$ is full, as it utilizes the available space up to the social distancing requirement.

Example 1. *Consider the given values:* $\delta = 1$, $L = 21$, *and* $M = 4$. *In this case, we have* $n_i = i + 1$ *for* $i = 1, 2, 3, 4$. *The loss of the largest pattern can be calculated as* $\lfloor \frac{21}{5} \rfloor - 1 + 1 = 4$. *The largest patterns are the following:* $(1, 0, 1, 3)$, $(0, 1, 2, 2)$, $(0, 0, 0, 4)$, $(0, 0, 4, 1)$, *and* $(0, 2, 0, 3)$.

Through this example, we observe that the largest pattern does not exclusively consist of large groups but can also include smaller groups. This highlights the importance of considering the various group sizes when using the largest pattern. Another observation relates to the relationship between the largest patterns and full patterns. It is apparent that a full pattern may not necessarily be the largest pattern. For instance, consider the pattern $(1, 1, 4, 0)$, which is a full pattern as it utilizes all available seats. However, its loss value is 6, indicating that it is not the largest pattern. Conversely, a largest pattern may also not necessarily be a full pattern. Take the pattern $(0, 0, 0, 4)$ as an example. It is a largest pattern as it can accommodate the maximum number of individuals. However, it does not satisfy the requirement of fully utilizing all available seats since $4 \times 5 \neq 21$.

Although the optimal solution to the seat planning problem is complex, the LP relaxation of problem (1) has a nice property.

Proposition 2. *In the LP relaxation of problem* (1), *there exists an index* v *such that the optimal solutions satisfy the following conditions:*

- *For* $i = 1, \ldots, v-1$, $x_{ij}^* = 0$ *for all rows, indicating that no group type* i *are assigned to any rows before index* v.
- *For* $i = v+1, \ldots, M$, *the optimal solution assigns* $\sum_j x_{ij}^* = d_i$ *group type* i *to meet the demand for group type* i.
- *For* $i = v$, *the optimal solution assigns* $\sum_j x_{ij}^* = \frac{L - \sum_{i=v+1}^{M} d_i n_i}{n_v}$ *group type* v *to the rows. This quantity is determined by the available supply, which is calculated as the remaining seats after accommodating the demands for group types* $v+1$ *to* M, *divided by the size of group type* v, *denoted as* n_v.

Hence, the corresponding supply values can be summarized as follows: $X_v = \frac{L - \sum_{i=v+1}^{M} d_i n_i}{n_v}$, $X_i = d_i$ *for* $i = v+1, \ldots, M$, *and* $X_i = 0$ *for* $i = 1, \ldots, v-1$. *These supply values represent the allocation of seats to each group type.*

3.2 Dynamic Seat Assignment with Social Distancing

In a more realistic scenario, groups arrive sequentially over time, and the seller must promptly make group assignments upon each arrival while maintaining the required spacing between groups. When a

group is accepted, the seller must also determine which seats should be assigned to that group. It is essential to note that each group must be either accepted in its entirety or rejected entirely; partial acceptance is not permitted. Once the seats are confirmed and assigned to a group, they cannot be changed or reassigned to other groups.

To model this problem, we adopt a discrete-time framework. Time is divided into T periods, indexed forward from 1 to T. We assume that in each period, at most one group arrives and the probability of an arrival for a group of size i is denoted as p_i, where i belongs to the set $\mathcal{M}$. The probabilities satisfy the constraint $\sum_{i=1}^{M} p_i \leq 1$, indicating that the total probability of any group arriving in a single period does not exceed one. We introduce the probability $p_0 = 1 - \sum_{i=1}^{M} p_i$ to represent the probability of no arrival in a given period t. To simplify the analysis, we assume that the arrivals of different group types are independent and the arrival probabilities remain constant over time. This assumption can be extended to consider dependent arrival probabilities over time if necessary.

The state of remaining capacity in each row is represented by a vector $\mathbf{L} = (l_1, l_2, \ldots, l_N)$, where l_j denotes the number of remaining seats in row j. Upon the arrival of a group type i in period t, the seller needs to make a decision denoted by $u_{i,j}^t$, where $u_{i,j}^t = 1$ indicates acceptance of group type i in row j during period t, while $u_{i,j}^t = 0$ signifies rejection of that group type in row j at that period. The feasible decision set is defined as

$$U^t(\mathbf{L}) = \{u_{i,j}^t \in \{0,1\}, \forall i \in \mathcal{M}, \forall j \in \mathcal{N} | \sum_{j=1}^{N} u_{i,j}^t \leq 1, \forall i \in \mathcal{M}; n_i u_{i,j}^t \mathbf{e}_j \leq \mathbf{L}, \forall i \in \mathcal{M}, \forall j \in \mathcal{N}\}.$$

Here, $\mathbf{e}_j$ represents an N-dimensional unit column vector with the j-th element being 1, i.e., $\mathbf{e}_j = (\underbrace{0, \cdots, 0}_{j-1}, 1, \underbrace{0, \cdots, 0}_{n-j})$. In other words, the decision set $U(\mathbf{L})$ consists of all possible combinations of acceptance and rejection decisions for each group type in each row, subject to the constraints that at most one group of each type can be accepted in any row, and the number of seats occupied by each accepted group must not exceed the remaining capacity of the row.

Let $V^t(\mathbf{L})$ denote the maximum expected revenue earned by the best decisions regarding group seat assignments in period t, given remaining capacity $\mathbf{L}$. Then, the dynamic programming formula for this problem can be expressed as:

$$V^t(\mathbf{L}) = \max_{u_{i,j}^t \in U^t(\mathbf{L})} \left\{ \sum_{i=1}^{M} p_i (\sum_{j=1}^{N} i u_{i,j}^t + V^{t+1}(\mathbf{L} - \sum_{j=1}^{N} n_i u_{i,j}^t \mathbf{e}_j)) + p_0 V^{t+1}(\mathbf{L}) \right\} \tag{2}$$

with the boundary conditions $V^{T+1}(\mathbf{L}) = 0, \forall \mathbf{L}$ which implies that the revenue at the last period is 0 under any capacity.

At the beginning of period t, we have the current remaining capacity vector denoted as $\mathbf{L} = (L_1, L_2, \ldots, L_N)$. Our objective is to make group assignments that maximize the total expected revenue during the horizon from period 1 to T which is represented by $V^1(\mathbf{L})$.

Solving the dynamic programming problem described in equation (2) can be challenging due to the curse of dimensionality, which arises when the problem involves a large number of variables or states.

To mitigate this complexity, we aim to develop a heuristic method for assigning arriving groups. In our approach, we begin by generating a seat planning that consists of the largest or full patterns, as outlined in section 4. This initial seat planning acts as a foundation for our heuristic method. In section 3.2, building upon the generated seat planning, we further develop a dynamic seat assignment policy which guides the allocation of seats to the incoming groups sequentially.

4 Seat Planning Composed of Full or Largest Patterns

In this section, we develop the scenario-based stochastic programming (SSP) to obtain the seat planning with available capacity. Due to the well-structured nature of SSP, we implement Benders decomposition to solve it efficiently. However, in some cases, solving the integer programming with Benders decomposition remains still computationally prohibitive. Thus, we can consider the LP relaxation first, then obtain a feasible seat planning by deterministic model. Based on that, we construct a seat planning composed of full or largest patterns to fully utilize all seats.

4.1 Scenario-based Stochastic Programming (SSP) Formulation

Now suppose the demand of groups is stochastic, the stochastic information can be obtained from scenarios through historical data. Use ω to index the different scenarios, each scenario $\omega \in \Omega$. Regarding the nature of the obtained information, we assume that there are $|\Omega|$ possible scenarios. A particular realization of the demand vector can be represented as $\mathbf{d}_\omega = (d_{1\omega}, d_{2\omega}, \ldots, d_{M,\omega})^\intercal$. Let p_ω denote the probability of any scenario ω, which we assume to be positive. To maximize the expected number of people accommodated over all the scenarios, we propose a scenario-based stochastic programming to obtain a seat planning.

The seat planning can be represented by decision variables $\mathbf{x} \in \mathbb{Z}_+^{M\times N}$. Here, x_{ij} represents the number of group type i assigned to row j in the seat planning. As mentioned earlier, we calculate the supply for group type i as the sum of x_{ij} over all rows j, denoted as $\sum_{j=1}^{N} x_{ij}$. However, considering the variability across different scenarios, it is necessary to model the potential excess or shortage of supply. To capture this characteristic, we introduce a scenario-dependent decision variable, denoted as $\mathbf{y}$. It includes two vectors of decisions, $\mathbf{y}^+ \in \mathbb{Z}_+^{M\times|\Omega|}$ and $\mathbf{y}^- \in \mathbb{Z}_+^{M\times|\Omega|}$. Each component of $\mathbf{y}^+$, denoted as $y^+_{i\omega}$, represents the excess supply for group type i for each scenario ω. On the other hand, $y^-_{i\omega}$ represents the shortage of supply for group type i for each scenario ω.

Taking into account the possibility of groups occupying seats planned for larger group types when the corresponding supply is insufficient, we make the assumption that surplus seats for group type i can be occupied by smaller group types $j < i$ in descending order of group size. This means that if there are excess supply available after assigning groups of type i to rows, we can provide the supply to groups of type $j < i$ in a hierarchical manner based on their sizes. That is, for any ω, $i \le M-1$,

$$y^+_{i\omega} = \left(\sum_{j=1}^{N} x_{ij} - d_{i\omega} + y^+_{i+1,\omega}\right)^+, \quad y^-_{i\omega} = \left(d_{i\omega} - \sum_{j=1}^{N} x_{ij} - y^+_{i+1,\omega}\right)^+,$$

where $(x)^+$ equals x if $x > 0$, 0 otherwise. Specially, for the largest group type M, we have $y^+_{M\omega} = (\sum_{j=1}^N x_{Mj} - d_{M\omega})^+$, $y^-_{M\omega} = (d_{M\omega} - \sum_{j=1}^N x_{Mj})^+$. Based on the above mentioned considerations, the total supply of group type i under scenario ω can be expressed as $\sum_{j=1}^N x_{ij} + y^+_{i+1,\omega} - y^+_{i\omega}, i = 1, \ldots, M-1$. For the special case of group type M, the total supply under scenario ω is $\sum_{j=1}^N x_{Mj} - y^+_{M\omega}$.

Then we have the formulation of SSP:

$$\max \quad E_\omega \left[(n_M - \delta)(\sum_{j=1}^N x_{Mj} - y^+_{M\omega}) + \sum_{i=1}^{M-1} (n_i - \delta)(\sum_{j=1}^N x_{ij} + y^+_{i+1,\omega} - y^+_{i\omega}) \right] \tag{3}$$

$$\text{s.t.} \quad \sum_{j=1}^N x_{ij} - y^+_{i\omega} + y^+_{i+1,\omega} + y^-_{i\omega} = d_{i\omega}, \quad i = 1, \ldots, M-1, \omega \in \Omega \tag{4}$$

$$\sum_{j=1}^N x_{ij} - y^+_{i\omega} + y^-_{i\omega} = d_{i\omega}, \quad i = M, \omega \in \Omega \tag{5}$$

$$\sum_{i=1}^M n_i x_{ij} \le L_j, j \in \mathcal{N} \tag{6}$$

$$y^+_{i\omega}, y^-_{i\omega} \in \mathbb{Z}_+, \quad i \in \mathcal{M}, \omega \in \Omega$$

$$x_{ij} \in \mathbb{Z}_+, \quad i \in \mathcal{M}, j \in \mathcal{N}.$$

The objective function consists of two parts. The first part represents the number of people in the largest group type that can be accommodated, given by $(n_M - \delta)(\sum_{j=1}^N x_{Mj} - y^+_{M\omega})$. The second part represents the number of people in group type i, excluding M, that can be accommodated, given by $(n_i - \delta)(\sum_{j=1}^N x_{ij} + y^+_{i+1,\omega} - y^+_{i\omega}), i = 1, \ldots, M-1$. The overall objective function is subject to an expectation operator denoted by E_ω, which represents the expectation with respect to the scenario set. This implies that the objective function is evaluated by considering the average values of the decision variables and constraints over the different scenarios.

By reformulating the objective function, we have

$$\begin{aligned}
&E_\omega \left[\sum_{i=1}^{M-1} (n_i - \delta)(\sum_{j=1}^N x_{ij} + y^+_{i+1,\omega} - y^+_{i\omega}) + (n_M - \delta)(\sum_{j=1}^N x_{Mj} - y^+_{M\omega}) \right] \\
=&\sum_{j=1}^N \sum_{i=1}^M (n_i - \delta) x_{ij} - \sum_{\omega=1}^{|\Omega|} p_\omega \left(\sum_{i=1}^M (n_i - \delta) y^+_{i\omega} - \sum_{i=1}^{M-1} (n_i - \delta) y^+_{i+1,\omega} \right) \\
=&\sum_{j=1}^N \sum_{i=1}^M i \cdot x_{ij} - \sum_{\omega=1}^{|\Omega|} p_\omega \sum_{i=1}^M y^+_{i\omega}
\end{aligned}$$

In the optimal solution, at most one of $y^+_{i\omega}$ and $y^-_{i\omega}$ can be positive for any i, ω. Suppose there exist i_0 and ω_0 such that $y^+_{i_0\omega_0}$ and $y^-_{i_0\omega_0}$ are positive. Substracting $\min\{y^+_{i_0,\omega_0}, y^-_{i_0,\omega_0}\}$ from these two values will still satisfy constraints (4) and (5) but increase the objective value when p_{ω_0} is positive. Thus, in the optimal solution, at most one of $y^+_{i\omega}$ and $y^-_{i\omega}$ can be positive.

Considering the analysis provided earlier, we find it advantageous to obtain a seat planning that only consists of full or largest patterns. However, the seat planning associated with the optimal solution

obtained by solver to SSP may not consist of the largest or full patterns. We can convert the optimal solution to another optimal solution which is composed of the largest or full patterns.

Proposition 3. *There exists an optimal solution to the stochastic programming problem such that the patterns associated with this optimal solution are composed of the full or largest patterns under any given scenarios.*

Given a specific pattern, we can convert it into a largest or full pattern while ensuring that the original group type requirements are met. When multiple full patterns are possible, our objective is to generate the pattern with minimal loss. Mathematically, for any pattern $\boldsymbol{h} = (h_1, \ldots, h_N)$, we seek to find a pattern $\boldsymbol{h'} = (h_1{}', \ldots, h_N{}')$ that maximizes $|\boldsymbol{h'}|$ while satisfying the following constraints: $h_1{}' \geq h_1$, $h_1{}' + h_2{}' \geq h_1 + h_2, \cdots, h_1{}' + \ldots + h_N{}' \geq h_1 + \ldots + h_N$. In other words, we want to find a pattern $\boldsymbol{h'}$ where each element h_i' is greater than or equal to the corresponding element h_i in $\boldsymbol{h}$, and the cumulative sums of the elements in $\boldsymbol{h'}$ are greater than or equal to the cumulative sums of the elements in $\boldsymbol{h}$. By finding such a pattern $\boldsymbol{h'}$, we can ensure that the converted pattern meets or exceeds the requirements of the original group types. Among the possible full patterns that satisfy these constraints, we prioritize the one with the smallest loss.

Now, we demonstrate the specific allocation scheme. Let $\beta = L_j - \sum_i n_i x_{ij}$. If row j is not the largest or full, then $\beta > 0$. We aim to allocate the remaining unoccupied seats in row j in a way that maximizes the number of planned groups that become the largest in size. Find the smallest group type in the pattern denoted as k. If $k = M$, it means that this row corresponds to a largest pattern. If $k \neq M$, we reduce the number of group type k by one and increase the number of group type $\min\{(k + \beta), M\}$ by one, the number of unoccupied seats will be reduced correspondingly.

We continue this procedure until either all the planned groups become the largest or $\beta = 0$. If $\beta = 0$, it indicates that the pattern is full. In this case, we have assigned all the unoccupied seats to the existing groups without incurring any additional loss. Therefore, this full pattern has the minimal loss while satisfying the groups requirement. However, if all the planned groups become the largest and $\beta \neq 0$, we can repeatly follow the steps outlined below to obtain the largest pattern:

- If $\beta \geq n_M$, we can assign n_M seats to a new group type M.
- If $n_1 \leq \beta < n_M$, we can assign β seats to a new group type $\beta - n_1 + 1$.
- If $0 < \beta < n_1$, it means that the current pattern is already the largest possible pattern because all the planned groups in the pattern are the largest.

By following these steps and always prioritizing the largest group type for seat planning, we can achieve either the largest pattern or a full pattern with minimal loss. This approach guarantees efficient seat allocation, maximizing the utilization of available seats while still accommodating the original groups' requirements.

To construct the largest or full pattern for each row, we can employ the following algorithm. Since patterns are independent of each other, we can process them row by row within a given seat planning.

This enables us to optimize the seat planning by maximizing the utilization of available seats and effectively accommodating the arriving groups.

Algorithm 1: Construct The Largest or Full Pattern

```
while β > 0 do
    k ← min_i{h_i ≠ 0};                    /* Find the smallest group type in the pattern */
    if k ≠ M then
        h_k ← h_k − 1;
        h_{min{k+β,M}} ← h_{min{k+β,M}} + 1;
        β ← β − max{1, M − k};   /* Change the current group type to a group type as
         large as possible */
    else
        if β ≥ n_M then
            q ← ⌊β / n_M⌋;
            β ← β − q n_M;
            h_M ← h_M + q;          /* Assign seats to as many the largest group type as
             possible */
        else
            if n_1 ≤ β < n_M then
                h_{β−n_1+1} ← h_{β−n_1+1} + 1;
                β ← 0;
            else
                h is the largest;
                β ← 0;
            end
        end
    end
end
```

Let $\mathbf{n} = (n_1, \ldots, n_M)$ represent the vector of seat sizes for each group type, where n_i denotes the size of seats taken by group type i. Let $\mathbf{L} = (L_1, \ldots, L_N)$ represent the vector of row lengths, where L_j denotes the length of row j as defined previously. The constraint (6) can be expressed as $\mathbf{nx} \leq \mathbf{L}$. This constraint ensures that the total size of seats occupied by each group type, represented by $\mathbf{nx}$, does not exceed the available row lengths $\mathbf{L}$. We can use the product $\mathbf{x1}$ to indicate the supply of group types, where $\mathbf{1}$ is a column vector of size N with all elements equal to 1.

The linear constraints associated with scenarios, denoted by constraints (4) and (5), can be expressed in matrix form as:

$$\mathbf{x1} + \mathbf{V}\mathbf{y}_\omega = \mathbf{d}_\omega, \omega \in \Omega,$$

where $\mathbf{V} = [\mathbf{W},\ \mathbf{I}]$.

$$\mathbf{W} = \begin{bmatrix} -1 & 1 & 0 & \ldots & \ldots & 0 \\ 0 & -1 & 1 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & -1 & 1 & 0 \\ 0 & \ldots & \ldots & 0 & -1 & 1 \\ 0 & \ldots & \ldots & \ldots & 0 & -1 \end{bmatrix}_{M \times M}$$

and $\mathbf{I}$ is the identity matrix with the dimension of M. For each scenario $\omega \in \Omega$,

$$\mathbf{y}_\omega = \begin{bmatrix} \mathbf{y}_\omega^+ \\ \mathbf{y}_\omega^- \end{bmatrix}, \mathbf{y}_\omega^+ = \begin{bmatrix} y_{1\omega}^+ & y_{2\omega}^+ & \cdots & y_{M\omega}^+ \end{bmatrix}^\intercal, \mathbf{y}_\omega^- = \begin{bmatrix} y_{1\omega}^- & y_{2\omega}^- & \cdots & y_{M\omega}^- \end{bmatrix}^\intercal.$$

As we can find, this deterministic equivalent form is a large-scale problem even if the number of possible scenarios Ω is moderate. However, the structured constraints allow us to simplify the problem by applying Benders decomposition approach. Before using this approach, we could reformulate this problem as the following form. Let $\mathbf{c}^\intercal\mathbf{x} = \sum_{j=1}^{N}\sum_{i=1}^{M} i \cdot x_{ij}$, $\mathbf{f}^\intercal\mathbf{y}_\omega = -\sum_{i=1}^{M} y_{i\omega}^+$. Then the SSP formulation can be expressed as below,

$$\begin{aligned} \max \quad & \mathbf{c}^\intercal\mathbf{x} + z(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{n}\mathbf{x} \leq \mathbf{L} \\ & \mathbf{x} \in \mathbb{Z}_+^{M\times N}, \end{aligned} \tag{7}$$

where $z(\mathbf{x})$ is defined as

$$z(\mathbf{x}) := E(z_\omega(\mathbf{x})) = \sum_{\omega\in\Omega} p_\omega z_\omega(\mathbf{x}),$$

and for each scenario $\omega \in \Omega$,

$$\begin{aligned} z_\omega(\mathbf{x}) := \max \quad & \mathbf{f}^\intercal\mathbf{y}_\omega \\ \text{s.t.} \quad & \mathbf{V}\mathbf{y}_\omega = \mathbf{d}_\omega - \mathbf{x}\mathbf{1} \\ & \mathbf{y}_\omega \geq 0. \end{aligned} \tag{8}$$

We can solve problem (7) quickly if we can efficiently solve problem (8). Next, we will mention how to solve problem (8).

4.2 Solve SSP by Benders Decomposition

We reformulate problem (7) into a master problem and a subproblem (8). The iterative process of solving the master problem and subproblem is known as Benders decomposition. The solution obtained from the master problem provides inputs for the subproblem, and the subproblem solutions help update the master problem by adding constraints, iteratively improving the overall solution until convergence is achieved. Firstly, we generate a closed-form solution to problem (8), then we obtain the solution to the LP relaxation of problem (7) by the constraint generation.

4.2.1 Solve The Subproblem

Notice that the feasible region of the dual of problem (8) remains unaffected by $\mathbf{x}$. This observation provides insight into the properties of this problem. Let $\boldsymbol{\alpha}$ denote the vector of dual variables. For each ω, we can form its dual problem, which is

$$\begin{aligned} \min \quad & \boldsymbol{\alpha}_\omega^\intercal(\mathbf{d}_\omega - \mathbf{x}\mathbf{1}) \\ \text{s.t.} \quad & \boldsymbol{\alpha}_\omega^\intercal \mathbf{V} \geq \mathbf{f}^\intercal \end{aligned} \tag{9}$$

Lemma 1. *Let* $\mathbb{P} = \{\boldsymbol{\alpha} \in \mathbb{R}^M | \boldsymbol{\alpha}^\intercal \mathbf{V} \geq \mathbf{f}^\intercal\}$. *The feasible region of problem* (9), $\mathbb{P}$, *is nonempty and bounded. Furthermore, all the extreme points of* $\mathbb{P}$ *are integral.*

Therefore, the optimal value of the problem (8), $z_\omega(\mathbf{x})$, is finite and can be achieved at extreme points of the set $\mathbb{P}$. Let $\mathcal{O}$ be the set of all extreme points of $\mathbb{P}$. That is, we have $z_\omega(\mathbf{x}) = \min_{\boldsymbol{\alpha}_\omega \in \mathcal{O}} \boldsymbol{\alpha}_\omega^\intercal(\mathbf{d}_\omega - \mathbf{x}\mathbf{1})$.

Alternatively, $z_\omega(\mathbf{x})$ is the largest number z_ω such that $\boldsymbol{\alpha}_\omega^\intercal(\mathbf{d}_\omega - \mathbf{x}\mathbf{1}) \geq z_w, \forall \boldsymbol{\alpha}_\omega \in \mathcal{O}$. We use this characterization of $z_w(\mathbf{x})$ in problem (7) and conclude that problem (7) can thus be put in the form by setting z_w as the variable:

$$\begin{aligned} \max \quad & \mathbf{c}^\intercal \mathbf{x} + \sum_{\omega \in \Omega} p_\omega z_\omega \\ \text{s.t.} \quad & \mathbf{n}\mathbf{x} \leq \mathbf{L} \\ & \boldsymbol{\alpha}_\omega^\intercal(\mathbf{d}_\omega - \mathbf{x}\mathbf{1}) \geq z_\omega, \forall \boldsymbol{\alpha}_\omega \in \mathcal{O}, \forall \omega \\ & \mathbf{x} \in \mathbb{Z}_+ \end{aligned} \tag{10}$$

Before applying Benders decomposition to solve problem (10), it is important to address the efficient computation of the optimal solution to problem (9). When we are given $\mathbf{x}^*$, the demand that can be satisfied by the seat planning is $\mathbf{x}^*\mathbf{1} = \mathbf{d}_0 = (d_{1,0}, \ldots, d_{M,0})^\intercal$. By plugging them in the subproblem (8), we can obtain the value of $y_{i,\omega}$ recursively:

$$\begin{aligned} y_{M\omega}^- &= (d_{M\omega} - d_{M0})^+ \\ y_{M\omega}^+ &= (d_{M0} - d_{M\omega})^+ \\ y_{i\omega}^- &= \left(d_{i\omega} - d_{i0} - y_{i+1,\omega}^+\right)^+, i = 1, \ldots, M-1 \\ y_{i\omega}^+ &= \left(d_{i0} - d_{i\omega} + y_{i+1,\omega}^+\right)^+, i = 1, \ldots, M-1 \end{aligned} \tag{11}$$

The optimal solutions to problem (9) can be obtained according to the value of $\mathbf{y}_\omega$.

Proposition 4. *The optimal solutions to problem* (9) *are given by*

$$\begin{aligned} \alpha_i = 0 \quad & \text{if } y_{i\omega}^- > 0, i = 1, \ldots, M \text{ or } y_{i\omega}^- = y_{i\omega}^+ = 0, y_{i+1,\omega}^+ > 0, i = 1, \ldots, M-1 \\ \alpha_i = \alpha_{i-1} + 1 \quad & \text{if } y_{i\omega}^+ > 0, i = 1, \ldots, M \\ 0 \leq \alpha_i \leq \alpha_{i-1} + 1 \quad & \text{if } y_{i\omega}^- = y_{i\omega}^+ = 0, i = M \text{ or } y_{i\omega}^- = y_{i\omega}^+ = 0, y_{i+1,\omega}^+ = 0, i = 1, \ldots, M-1 \end{aligned} \tag{12}$$

Instead of solving this linear programming directly, we can compute the values of α_ω by performing a forward calculation from $\alpha_{1\omega}$ to $\alpha_{M\omega}$.

4.2.2 Constraint Generation

Due to the computational infeasibility of solving problem (10) with an exponentially large number of constraints, it is a common practice to use a subset, denoted as $\mathcal{O}^t$, to replace $\mathcal{O}$ in problem (10).

This results in a modified problem known as the Restricted Benders Master Problem(RBMP). To find the optimal solution to problem (10), we employ the technique of constraint generation. It involves iteratively solving the RBMP and incrementally adding more constraints until the optimal solution to problem (10) is obtained.

We can conclude that the RBMP will have the form:

$$\begin{aligned} \max \quad & \mathbf{c}^\intercal \mathbf{x} + \sum_{\omega \in \Omega} p_\omega z_\omega \\ \text{s.t.} \quad & \mathbf{n}\mathbf{x} \leq \mathbf{L} \\ & \boldsymbol{\alpha}_\omega^\intercal (\mathbf{d}_\omega - \mathbf{x}\mathbf{1}) \geq z_\omega, \boldsymbol{\alpha}_\omega \in \mathcal{O}^t, \forall \omega \\ & \mathbf{x} \in \mathbb{Z}_+ \end{aligned} \tag{13}$$

To determine the initial $\mathcal{O}^t$, we have the following proposition.

Proposition 5. *RBMP is always bounded with at least any one feasible constraint for each scenario.*

Given the initial $\mathcal{O}^t$, we can have the solution $\mathbf{x}^*$ and $\mathbf{z}^* = (z_1^*, \ldots, z_{|\Omega|}^*)$. Then $c^\intercal \mathbf{x}^* + \sum_{\omega \in \Omega} p_\omega z_\omega^*$ is an upper bound of problem (13). When $\mathbf{x}^*$ is given, the optimal solution, $\tilde{\boldsymbol{\alpha}}_\omega$, to problem (9) can be obtained according to Proposition 4. Let $\tilde{z}_\omega = \tilde{\boldsymbol{\alpha}}_\omega (d_\omega - \mathbf{x}^* \mathbf{1})$, then $(\mathbf{x}^*, \tilde{\mathbf{z}})$ is a feasible solution to problem (13) because it satisfies all the constraints. Thus, $\mathbf{c}^\intercal \mathbf{x}^* + \sum_{\omega \in \Omega} p_\omega \tilde{z}_\omega$ is a lower bound of problem (10).

If for every scenario ω, the optimal value of the corresponding problem (9) is larger than or equal to z_ω^*, which means all contraints are satisfied, then we have an optimal solution, $(\mathbf{x}^*, \mathbf{z}^*)$, to problem (10). However, if there exists at least one scenario ω for which the optimal value of problem (9) is less than z_ω^*, indicating that the constraints are not fully satisfied, we need to add a new constraint $(\tilde{\boldsymbol{\alpha}}_\omega)^\intercal (\mathbf{d}_\omega - \mathbf{x}\mathbf{1}) \geq z_\omega$ to RBMP.

Algorithm 2: Benders Decomposition

Input: Initial problem (13) with $\boldsymbol{\alpha}_\omega = 0, \forall \omega$, $LB = 0$, $UB = \infty$, ϵ.
Output: $\mathbf{x}^*$

```
while UB − LB > ϵ do
    Obtain (x*, z*) from problem (13);
    UB ← c^⊺ x* + Σ_{ω∈Ω} p_ω z*_ω;
    for ω = 1, ..., |Ω| do
        Obtain α̃_ω from Proposition 4;
        z̃_ω = (α̃_ω)^⊺ (d_ω − x*1);
        if z̃_ω < z*_ω then
            Add one new constraint, (α̃_ω)^⊺ (d_ω − x1) ≥ z_ω, to problem (13);
        end
    end
    LB ← c^⊺ x* + Σ_{ω∈Ω} p_ω z̃_ω;
end
```

From Proposition 5, we can set $\boldsymbol{\alpha}_\omega = \mathbf{0}$ initially. Notice that only contraints are added in each iteration, thus UB is decreasing monotone over iterations. Then we can use $UB - LB < \epsilon$ to terminate the algorithm.

However, solving problem (13) even with the simplified constraints directly can be computationally

challenging in some cases, so practically we first obtain the optimal solution to the LP relaxation of problem (7). Then, we generate an integral seat planning from this solution.

4.3 Obtain The Seat Planning Composed of Full or Largest Patterns

We may obtain a fractional optimal solution when we solve the LP relaxation of problem (7). This solution represents the optimal allocations of groups to seats but may involve fractional values, indicating partial assignments. Based on the fractional solution obtained, we use the deterministic model to generate a feasible seat planning. The objective of this model is to allocate groups to seats in a way that satisfies the supply requirements for each group without exceeding the corresponding supply values obtained from the fractional solution. To accommodate more groups and optimize seat utilization, we aim to construct a seat planning composed of full or largest patterns based on the feasible seat planning obtained in the last step.

Let the optimal solution to the LP relaxation of problem (13) be $\mathbf{x}^*$. Aggregate $\mathbf{x}^*$ to the number of each group type, $X_i^* = \sum_j x_{ij}^*, i \in \mathbf{M}$. Replace the vector $\mathbf{d}$ with X^* in the deterministic model, we have the following problem,

$$\left\{ \max \sum_{j=1}^{N} \sum_{i=1}^{M} (n_i - \delta) x_{ij} : \sum_{i=1}^{M} n_i x_{ij} \leq L_j, j \in \mathcal{N}; \sum_{j=1}^{N} x_{ij} \leq X_i^*, i \in \mathcal{M}; x_{ij} \in Z_+ \right\} \tag{14}$$

Then solve the resulting problem (14) to obtain the optimal solution, $\tilde{\mathbf{x}}$, which represents a feasible seat planning. We can construct the largest or full patterns by Algorithm 1.

Algorithm 3: Seat Planning Construction

1 Obtain the optimal solution, $\mathbf{x}^*$, from the LP relaxation of problem (13);
2 Aggregate $\mathbf{x}^*$ to the number of each group type, $\tilde{X}_i = \sum_j x_{ij}^*, i \in \mathbf{M}$;
3 Obtain the optimal solution, $\tilde{\mathbf{x}}$, and the corresponding pattern, $\boldsymbol{H}$, from problem (14) with $\tilde{X}$;
4 Construct the full or largest patterns by Algorithm 1 with $\tilde{\mathbf{x}}$ and $\boldsymbol{H}$;

5 Dynamic Seat Assignment Policy

In this section, we discuss how to assign the arriving groups in the dynamic situation. Our policy involves making decisions regarding seat allocations for each arriving group based on a seat planning which can be obtained from Section 4. Within each period, the seat assignment process involves two main steps. First, we determine the appropriate group type used to accommodated the arriving group. The second step is to choose a specific row according to the group type and make the final decision by evaluating stochastic programming. For the whole periods, we regenerate the seat planning under certain conditions to optimize computational efficiency.

5.1 Determine The Group Type

One intuitive approach is to utilize stochastic programming to make decisions by comparing the values obtained when accepting or rejecting the currently arriving group. Stochastic programming aids in generating seat planning, and when there are available seats planned for the group, we readily accept and allocate it to the corresponding position according to the seat planning. When there are no suitable supply available for the current group, we need to perform calculations of stochastic programming by considering the placement of the group in each possible row, then compare these values to make a decision. However, it is important to note that integer stochastic programming can be computationally expensive and will be unsolvable in some instances. Additionally, when there is no supply in the seat planning available for the current group, evaluating the values associated with placing the group in each possible row can require a significant amount of computation.

Therefore, in order to mitigate the computational challenges, we utilize the LP relaxation of stochastic programming as an approximation to compare the values when deciding whether to accept or reject a group. However, one challenge arises from the fact that the LP relaxation results in the same objective values for the acceptance group in any possible row. This poses the question of determining which row to place the group in when we accept it. To address this challenge, we developed the group-type control policy which narrows down the row options based on the seat planning.

The group-type control aims to find the group type to assign the arriving group, that helps us narrow down the option of rows for seat assignment. Seat planning serves as a representation of the supply available for each group type. Based on the supply, we can determine whether to accept an incoming group. When a group arrives, if there is sufficient supply available for an arriving group, we will accept the group and choose the group type accordingly. However, if there is no corresponding supply available for the arriving group, we need to decide whether to use a larger group's supply to meet the need of the arriving group. When a group is accepted and assigned to larger-size seats, the remaining empty seat(s) can be reserved for future demand without affecting the rest of the seat planning. To determine whether to use larger seats to accommodate the incoming group, we compare the expected values of accepting the group in the larger seats and rejecting the group based on the current seat planning. Then we identify the possible rows where the incoming group can be assigned based on the group types and seat availability.

Specifically, suppose the supply is $(x_1, \ldots, x_M)$ at period t, the number of remaining periods is $(T-t)$. For the coming group type i, if $x_i > 0$, then accept it, let $x_i = x_i - 1$. If $x_i = 0$, in the following part, we will demonstrate how to decide whether to accept the group to occupy larger-size seats when there is no corresponding supply available. For any $j = i+1, \ldots, M$, we can use one supply of group type j to accept a group type i. In that case, when $j = i+1, \ldots, i+\delta$, the expected number of accepted people is i and the remaining seats beyond the accepted group, which is $j-i$, will be wasted. When $j = i+\delta+1, \ldots, M$, the rest $(j-i-\delta)$ seats can be provided for one group type $j-i-\delta$ with δ seats of social distancing. Let D_j^t be the random variable indicates the number of group type j in t periods. The expected number of accepted people is $i+(j-i-\delta)P(D_{j-i-\delta}^{T-t} \geq x_{j-i-\delta}+1)$, where $P(D_i^{T-t} \geq x_i)$ is the probability that the demand of group type i in $(T-t)$ periods is no less than x_i, the remaining supply

of group type i. Thus, the term, $P(D_{j-i-\delta}^{T-t} \geq x_{j-i-\delta}+1)$, indicates the probability that the demand of group type $(j-i-\delta)$ in $(T-t)$ periods is no less than its current remaining supply plus 1.

Similarly, when we retain the supply of group type j by rejecting a group of i, the expected number of accepted people is $jP(D_j^{T-t} \geq x_j)$. The term, $P(D_j^{T-t} \geq x_j)$, indicates the probability that the demand of group type j in $(T-t)$ periods is no less than its current remaining supply.

Let $d^t(i,j)$ be the difference of expected number of accepted people between acceptance and rejection on group i occupying $(j+\delta)$-size seats at period t. Then we have

$$d^t(i,j) = \begin{cases} i + (j-i-\delta)P(D_{j-i-\delta}^{T-t} \geq x_{j-i-\delta}+1) - jP(D_j^{T-t} \geq x_j), & \text{if } j = i+\delta+1,\ldots,M \\ i - jP(D_j^{T-t} \geq x_j), & \text{if } j = i+1,\ldots,i+\delta. \end{cases}$$

One intuitive decision is to choose j with the largest difference. For all $j = i+1,\ldots,M$, find the largest $d^t(i,j)$, denoted as $d^t(i,j^*)$. If $d^t(i,j^*) > 0$, we will plan to assign the group type i in $(j^*+\delta)$-size seats. Otherwise, reject the group.

Group-type control policy can only tell us which group type's seats are planned to provide for the smaller group based on the current planning, we still need to further compare the values of the stochastic programming problem when accepting or rejecting a group on the specific row.

5.2 Decision on Assigning The Group to A Specific Row

To make the final decision, first, we determine a specific row by the tie-breaking rule, then we assign the group based on the values of relaxed stochastic programming. To determine the appropriate row for seat assignment, we can apply a tie-breaking rule among the possible options obtained by the group-type control. This rule helps us decide on a particular row when there are multiple choices available.

Break Tie for Determining A Specific Row

A tie occurs when there are serveral rows to accommodate the group. As mentioned above, let β represents the remaining capacity in a row after considering the seat allocation for other groups. When the supply is sufficient for the particular group type, we assign the group to the row with the smallest β value. That allows us to fill in the row according to the full pattern. When the supply is insufficient for the group type and we plan to assign the group to seats designated for larger groups, we follow a similar approach. In this case, we assign the group to a row that contains the larger group and has the largest β value. That helps to reconstruct the pattern with smaller β value. When there are multiple rows with the same β, we can choose randomly. By considering the row with the β value in both scenarios, we prioritize filling seats and aim to minimize the fragmentation of available seating capacity. This approach helps optimize the utilization of seats and leads to better capacity management.

As an example to illustrate group-type control and the tie-breaking rule, consider a scenario where we have four rows with available seats as follows: row 1 has 3 seats, row 2 has 4 seats, row 3 has 5 seats and row 4 has 6 seats. The corresponding patterns for each row are $(0,1,0,0)$, $(0,0,1,0)$, $(0,0,0,1)$ and $(0,0,0,1)$, respectively. There are $M = 4$ groups, the social distance is $\delta = 1$. Now, a group of one

arrives, and the group-type control indicates the possible rows where the group can be assigned. We assume this group can be assigned to the seats of the largest group according to the group-type control, then we have two choices: row 3 or row 4. To determine which row to select, we can apply the breaking tie rule. The β value of the rows will be used as the criterion, we would choose row 4 because its β value is larger. Because when we assign it in row 4, there will be two seats reserved for future group of one, but when we assign it in row 3, there will be one seat remaining unused.

In the above example, the group of one can be assigned to any row with the available seats. The group-type control can help us find the larger group type that can be used to place the arriving group while maximizing the expected values. Maybe there are multiple rows containing the larger group type. Then we can choose the row containing the larger group type according to the breaking tie rule. Finally, we use the stochastic programming to calculate the VoA and VoR. Comparing these values allows us to make the decision on whether to accept or reject the group.

Decision on Assigning The Group

Then, we compare the values of the relaxed stochastic programming when accepting the group at the chosen row versus rejecting it. This evaluation allows us to assess the potential revenues and make the final decision. Simultaneously, after this calculation, we can generate a new seat planning according to Algorithm 3. For the situation where the supply is enough in the first step, we can skip the final step because we already accept the group. Specifically, after we plan to assign the arriving group in a specific row, we determine whether to place the arriving group in the row based on the values of the stochastic programming problem. For the objective values of the relaxed stochastic programming, we consider the potential revenues that could result from accepting the current arrival, i.e., the Value of Acceptance (VoA), as well as the potential outcomes that could result from rejecting it, i.e., the Value of Rejection (VoR).

Let us consider the set of scenarios, denoted as Ω^t, at period t. The VoA is the value of relaxed stochastic programming by considering the scenario set Ω^t when we accept the arriving group at period t. On the other hand, the VoR is calculated when we reject the arriving arrival. Suppose the available supply is $\mathbf{L}^t = (L_1^t, \ldots, L_N^t)$ before making the decision at period t. We calculate the relaxed stochastic programming with $\mathbf{L}^t = (L_1^t, \ldots, L_N^t)$ when we reject group type i as the VoR. If we plan to accept group type i in row j, we need to subtract the size of group type i from L_j^t. Let $L_j^t = L_j^t - n_i$, then we calculate the relaxed stochastic programming with $\mathbf{L}^t = (L_1^t, \ldots, L_N^t)$ when we accept group type i in row j as the VoA.

In each period, we can calculate the relaxed stochastic programming values only twice: once for the acceptance option (VoA) and once for the rejection option (VoR). By comparing the values of VoA and VoR, we can determine whether to accept or reject the group arrival. The decision will be based on selecting the option with the higher expected value, i.e., if the VoA is larger than the VoR, we accept the arrival; if the VoA is less than the VoR, we will reject the incoming group.

By combining the group-type control strategy with the evaluation of relaxed stochastic programming values, we obtain a comprehensive decision-making process within a single period. This integrated

approach enables us to make informed decisions regarding the acceptance or rejection of incoming groups, as well as determine the appropriate row for the assignment when acceptance is made. By considering both computation time savings and potential revenues, we can optimize the overall performance of the seat assignment process.

5.3 Regenerate The Seat Planning

To optimize computational efficiency, it is not necessary to regenerate the seat planning for every period. Instead, we can employ a more streamlined approach. Considering that largest group type can meet the needs of all smaller group types, thus, if the supply for the largest group type diminishes from one to zero, it becomes necessary to regenerate the seat planning. This avoids rejecting the largest group due to infrequent regenerations. Another situation that requires seat planning regeneration is when we determine whether to assign the arriving group to a larger group. In such case, we can obtain the corresponding seat planning after solving the relaxed stochastic programmings. By regenerating the seat planning in such situations, we ensure that we have an accurate supply and can give the allocation of seats based on the group-type control and the comparisons of VoA and VoR.

The decision-making algorithm is shown below.

Algorithm 4: Dynamic Seat Assignment

for $t = 1, \ldots, T$ **do**
  Observe group type i;
  if $X_i > 0$ **then**
    Find row k such that $H_{ki} > 0$ according to tie-breaking rule;
    Accept group type i in row k, $L_k \leftarrow L_k - n_i$;
    $H_{ki} \leftarrow H_{ki} - 1$, $X_i \leftarrow X_i - 1$; `/* Accept group type` i `when the supply is sufficient */`
    if $i = M$ *and* $X_M = 0$ **then**
      Regenerate $\boldsymbol{H}$ from Algorithm 3;
      Update the corresponding $\boldsymbol{X}$; `/* Regenerate the seat planning when the supply of the largest group type is 0 */`
    end
  else
    Calculate $d^t(i, j^*)$;
    if $d^t(i, j^*) > 0$ **then**
      Find row k such that $H_{kj^*} > 0$ according to tie-breaking rule;
      Calculate the VoA under scenario Ω_A^t and the VoR under scenario Ω_R^t;
      if $VoA > VoR$ **then**
        Accept group type i, $L_k \leftarrow L_k - n_i$;
        Regenerate $\boldsymbol{H}$ from Algorithm 3;
        Update the corresponding $\boldsymbol{X}$;
      else
        Reject group type i;
        Regenerate $\boldsymbol{H}$ from Algorithm 3;
        Update the corresponding $\boldsymbol{X}$;
      end
    else
      Reject group type i;
    end
  end
end

6 Results

We carried out several experiments, including analyzing the performances of different policies, evaluating the impact of implementing social distancing.

6.1 Performances of Different Policies

In this section, we compare the performance of five dynamic seat assignment policies to the optimal value, which can be obtained by solving the deterministic model after observing all arrivals. The policies under examination are the stochastic planning policy, DP Base-heuristic, bid-price control, booking limit control and FCFS policy.

Bid-price Control

Bid-price control is a classical approach discussed extensively in the literature on network revenue management. It involves setting bid prices for different group types, which determine the eligibility of groups to take the seats. Bid-prices refer to the opportunity costs of taking one seat. As usual, we estimate the bid price of a seat by the shadow price of the capacity constraint corresponding to some row. In this section, we will demonstrate the implementation of the bid-price control policy.

The dual of LP relaxation of problem (1) is:

$$\begin{aligned} \min \quad & \sum_{i=1}^{M} d_i z_i + \sum_{j=1}^{N} L_j \beta_j \\ \text{s.t.} \quad & z_i + \beta_j n_i \geq (n_i - \delta), \quad i \in \mathcal{M}, j \in \mathcal{N} \\ & z_i \geq 0, i \in \mathcal{M}, \beta_j \geq 0, j \in \mathcal{N}. \end{aligned} \tag{15}$$

In (15), β_j can be interpreted as the bid-price for a seat in row j. A request is only accepted if the revenue it generates is above the sum of the bid prices of the seats it uses. Thus, if its revenue is more than its opportunity costs, i.e., $i - \beta_j n_i \geq 0$, we will accept the group type i. And choose $j^* = \arg\max_j \{i - \beta_j n_i\}$ as the row to allocate that group.

Lemma 2. *The optimal solution to problem (15) is given by* $z_1, \ldots, z_v = 0$, $z_i = \frac{\delta(n_i - n_v)}{n_v}$ *for* $i = v+1, \ldots, M$ *and* $\beta_j = \frac{n_v - \delta}{n_v}$ *for all* j.

The bid-price decision can be expressed as $i - \beta_j n_i = i - \frac{n_v - \delta}{n_v} n_i = \frac{\delta(i-v)}{n_v}$. When $i < v$, $i - \beta_j n_i < 0$. When $i \geq v$, $i - \beta_j n_i \geq 0$. This means that group type i with i greater than or equal to v will be accepted if the capacity allows. However, it should be noted that β_j does not vary with j, which means the bid-price control cannot determine the specific row to assign the group to. In practice, groups are often assigned arbitrarily based on availability when the capacity allows, which can result in a large number of empty seats.

The bid-price control policy based on the static model is stated below.

Algorithm 5: Bid-price Control Algorithm

for $t = 1, \ldots, T$ **do**
  Observe group type i;
  Solve the LP relaxation of problem (1) with $d_i^t = (T-t) \cdot p_i$ and $\mathbf{L}^t$;
  Obtain v such that the aggregate optimal solution is $xe_v + \sum_{i=v+1}^{M} d_i e_i$;
  if $i \geq v$ *and* $\max_j L_j \geq n_i$ **then**
    Accept the group and assign the group to row k such that $L_k \geq n_i$;
    $\mathbf{L}^{t+1} \leftarrow \mathbf{L}^t - n_i \mathbf{e}_k$;
  else
    Reject the group;
    $\mathbf{L}^{t+1} \leftarrow \mathbf{L}^t$;
  end
end

Booking Limit Control

The booking limit control policy involves setting a maximum number of reservations that can be accepted for each group type. By controlling the booking limits, revenue managers can effectively manage demand and allocate inventory to maximize revenue.

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. When we solve the linear relaxation of problem (1), the aggregate optimal solution is the limits for each group type. Interestingly, the bid-price control policy is found to be equivalent to the booking limit control policy.

When we solve problem (1) directly, we can develop the booking limit control policy.

Algorithm 6: Booking limit Control Algorithm

for $t = 1, \ldots, T$ **do**
  Observe group type i;
  Solve problem (1) with $d_i^t = (T-t) \cdot p_i$ and $\mathbf{L}^t$;
  Obtain the optimal solution, x_{ij}^* and the aggregate optimal solution, $\mathbf{X}$;
  if $X_i > 0$ **then**
    Accept the group and assign the group to row k such that $x_{ik} > 0$;
    $\mathbf{L}^{t+1} \leftarrow \mathbf{L}^t - n_i \mathbf{e}_k$;
  else
    Reject the group;
    $\mathbf{L}^{t+1} \leftarrow \mathbf{L}^t$;
  end
end

Dynamic Programming Base-heuristic

To simplify the complexity of the original dynamic programming problem, we can consider a simplified version by relaxing all rows to a single row with the same total capacity, denoted as $L = \sum_{j=1}^{N} L_j$. With this simplification, we can make decisions for each group arrival based on the relaxed dynamic programming. By relaxing the rows to a single row, we aggregate the capacities of all individual rows into a single capacity value. This allows us to treat the seat assignment problem as a one-dimensional problem, reducing the computational complexity. Using the relaxed dynamic programming approach, we can determine the seat assignment decisions for each group arrival based on the simplified problem.

Let u denote the decision, where $u^t = 1$ if we accept a request in period t, $u^t = 0$ otherwise. Similar to the DP in section 3.2, the DP with one row can be expressed as:

$$V^t(l) = \max_{u^t \in \{0,1\}} \left\{ \sum_i p_i [V^{t+1}(l - n_i u^t) + i u^t] + p_0 V^{t+1}(l) \right\}$$

with the boundary conditions $V^{T+1}(l) = 0, \forall l \geq 0$, $V^t(0) = 0, \forall t$.

After accepting one group, assign it in some row arbitrarily when the capacity of the row allows.

Algorithm 7: Dynamic Programming Base-heuristic Algorithm

Calculate $V^t(l)$, $\forall t = 2, \ldots, T; \forall l = 1, \ldots, L$;
$l^1 \leftarrow L$;
for $t = 1, \ldots, T$ **do**
    Observe group type i;
    if $V^{t+1}(l^t) \leq V^{t+1}(l^t - n_i) + i$ **then**
        Accept the group and assign the group to an arbitrary row k such that $L_k \geq n_i$;
        $L_k^{t+1} \leftarrow L_k^t - n_i$, $l^{t+1} \leftarrow l^t - n_i$;
    else
        Reject the group;
        $L_k^{t+1} \leftarrow L_k^t$, $l^{t+1} \leftarrow l^t$;
    end
end

First Come First Served (FCFS) Policy

For dynamic seat assignment for each group arrival, the intuitive but trivial method will be on a first-come-first-served basis. Each accepted request will be assigned seats row by row. If the capacity of a row is insufficient to accommodate a request, we will allocate it to the next available row. If a subsequent request can fit exactly into the remaining capacity of a partially filled row, we will assign it to that row immediately. Then continue to process requests in this manner until all rows cannot accommodate any groups.

Algorithm 8: FCFS Policy Algorithm

for $t = 1, \ldots, T$ **do**
  Observe group type i;
  if $\exists k$ *such that* $L_k \geq n_i$ **then**
    Accept the group and assign the group to row k;
    $L_k \leftarrow L_k - n_i$;
  else
    Reject the group;
  end
end

Numerical Results

The seat layout consists of 10 rows, each with 21 seats (including one dummy seat as the social distance), and the group size can range up to 4 people. We conducted experiments over 60 to 100 periods to demonstrate the policies' performance under varying demand levels. We selected three probabilities to ensure that the expected number of people for each period is consistent. The table below displays the average of 200 instances for each number.

An arrival sequence during T periods can be expressed as $y_1, y_2, \ldots, y_T$. Let $N_i = \sum_t I(y_t = i)$, i.e., the count number of times group type i arrives during T periods. Then the scenarios, $(N_1, \ldots, N_M)$, follow a multinomial distribution,

$$p\left(N_1, \ldots, N_M \mid \mathbf{p}\right) = \frac{T!}{N_1!, \ldots, N_M!} \prod_{i=1}^{M} p_i^{N_i}, T = \sum_{i=1}^{M} N_i.$$

It is clear that the number of different sequences is M^T. The number of different scenarios is $O(T^{M-1})$ which can be obtained by the following DP. The number of scenarios is too large to enumerate all possible cases. Thus, we choose to sample some sequences from the multinomial distribution.

We can find that the stochastic planning policy are better than DP Base-heuristic and bid-price policy consistently, and FCFS policy works worst. As we mentioned previously, DP Base-heuristic and bid-price policy can only make the decision to accept or deny, cannot decide which row to assign the group to. FCFS accepts groups in sequential order until the capacity cannot accommodate more.

For the first three policies, their performance tends to initially drop and then increase as the number of periods increases. When the number of periods is small, the demand for capacity is relatively low, and the policies can achieve relatively optimal performance. However, as the number of periods increases, the policies may struggle to always obtain a perfect allocation plan, leading to a decrease in performance. Nevertheless, when the number of periods continue to become larger, these policies tend to accept larger groups, and as a result, narrow the gap with the optimal value, leading to an increase in performance.

Table 1: Performances of Different Policies

T	probabilities	DSA (%)	DP1 (%)	Bid-price (%)	Booking (%)	FCFS (%)
60	[0.25, 0.25, 0.25, 0.25]	99.12	98.42	98.38	96.74	98.17
70		98.34	96.87	96.24	97.18	94.75
80		98.61	95.69	96.02	98.00	93.18
90		99.10	96.05	96.41	98.31	92.48
100		99.58	95.09	96.88	98.70	92.54
60	[0.25, 0.35, 0.05, 0.35]	98.94	98.26	98.25	96.74	98.62
70		98.05	96.62	96.06	96.90	93.96
80		98.37	96.01	95.89	97.75	92.88
90		99.01	96.77	96.62	98.42	92.46
100		99.23	97.04	97.14	98.67	92.00
60	[0.15, 0.25, 0.55, 0.05]	99.14	98.72	98.74	96.61	98.07
70		99.30	96.38	96.90	97.88	96.25
80		99.59	97.75	97.87	98.55	95.81
90		99.53	98.45	98.69	98.81	95.50
100		99.47	98.62	98.94	98.90	95.25

6.2 Impact of Implementing Social Distancing in DSA

In this section, our focus is to analyze the influence of social distancing on the number of accepted individuals. Intuitively, when demand is small, we will accept all arrivals, thus there is no difference whether we implement the social distancing. What is interesting for us is when the difference occurs. Our primary objective is to determine the first time period at which, on average, the number of people accepted without social distancing is not less than the number accepted with social distancing plus one. This critical point, referred to as the gap point, is of interest to us. Additionally, we will examine the corresponding occupancy rate at this gap point. It should be noted that the difference at a specific time period may vary depending on the total number of periods considered. Therefore, when evaluating the difference at a particular time period, we assume that there are a total of such periods under consideration.

It is evident that as the demand increases, the effect of social distancing becomes more pronounced. We aim to determine the specific time period where the absence of social distancing results in a higher number of accepted individuals compared to when social distancing measures are in place. Additionally, we will calculate the corresponding occupancy rate during this period.

By analyzing and comparing the data, we can gain insights into the relation between demand, social distancing, the number of accepted individuals, and occupancy rates. This information is valuable for understanding the impact of social distancing policies on overall capacity utilization and making informed decisions regarding resource allocation and operational strategies.

6.2.1 Estimation of Gap Point

Based on our findings, we observed that the seat allocation derived from the optimal solution consistently satisfies the formation of either the largest pattern or the full pattern, regardless of different probability combinations. However, certain counterexamples arise when the requirements associated with specific probability combinations are unable to form a full pattern, resulting in gaps in the seating

arrangement. The occurrence of these counterexamples is closely tied to the seat layout itself. The ratio of the number of largest patterns to the number of full patterns in the final seat allocation is influenced by the expected number of people in each period. We can leverage the expected number of people in each period to estimate the gap point when utilizing the DSA. This approach allows us to approximate the period at which the number of people accepted without social distancing surpasses the number accepted with social distancing, based on the performance characteristics observed in the DSA.

To find such a first period, we aim to find the maximum period such that we could assign all the groups during these periods into the seats, i.e., for each group type i, we have $\sum_j x_{ij} \geq d_i$, where x_{ij} is the number of group type i in row j. Meanwhile, we have the capacity constraint $\sum_i n_i x_{ij} \leq L_j$, thus, $\sum_i n_i d_i \leq \sum_i n_i \sum_j x_{ij} \leq \sum_j L_j$. Notice that $E(d_i) = p_i T$, we have $\sum_i n_i p_i T \leq \sum_j L_j$ by taking the expectation. Let $L = \sum_j L_j$, representing the total number of seats, $\gamma = \sum_i i p_i$, representing the average number of people who arrive in each period, we can obtain $T \leq \frac{L}{\gamma+\delta}$, then the upper bound of the expected maximum period is $T' = \frac{L}{\gamma+\delta}$.

Assuming that we accept all incoming groups within T' periods, filling all the available seats, the corresponding occupancy rate at this period can be calculated as $\frac{\gamma T'}{(\gamma+\delta)T'-N\delta} = \frac{\gamma}{\gamma+\delta}\frac{L}{L-N\delta}$. However, it is important to note that the actual maximum period will be smaller than T' because it is impossible to accept groups to fill all seats exactly. To estimate the gap point when applying DSA, we can use $y_1 = c_1\frac{L}{\gamma+\delta}$, where c_1 is a discount rate compared to the ideal assumption. Similarly, we can estimate the corresponding occupancy rate as $y_2 = c_2\frac{\gamma}{\gamma+\delta}\frac{L}{L-N\delta}$, where c_2 is a discount rate for the occupancy rate compared to the ideal scenario.

We consider the scenario where the number of group types is limited to 4. In this case, the average number of people per period, denoted as γ, can be expressed as $\gamma = p_1 \cdot 1 + p_2 \cdot 2 + p_3 \cdot 3 + p_4 \cdot 4$, where p_1, p_2, p_3, and p_4 represent the probabilities of groups with one, two, three, and four people, respectively. We assume that p_4 always has a positive value. Additionally, the social distancing requirement is set to one seat.

To analyze the relation between the increment of γ and the gap point, we define each combination (p_1, p_2, p_3, p_4) satisfying $p_1 + p_2 + p_3 + p_4 = 1$ as a probability combination. We conducted an analysis using a sample of 200 probability combinations. The figure below illustrates the gap point as a function of the increment of γ, along with the corresponding estimations. For each probability combination, we considered 100 instances and plotted the gap point as blue points. Additionally, the occupancy rate at the gap point is represented by red points.

To provide estimations, we utilize the equations $y_1 = \frac{c_1 L}{\gamma+\delta}$(blue line in the figure) and $y_2 = c_2\frac{\gamma}{\gamma+\delta}\frac{L}{L-N\delta}$(orange line in the figure), which are fitted to the data. These equations capture the relation between the gap point and the increment of γ, allowing us to approximate the values. By examining the relation between the gap point and the increment of γ, we can find that γ can be used to estimate gap point.

The estimation of c_1 and c_2 can be affected by different seat layouts. To investigate this impact, we conduct several experiments using different seat layouts, specifically with the number of rows $\times$ the number of seats configurations set as 10×16, 10×21, 10×26 and 10×31. Similarly, we perform an

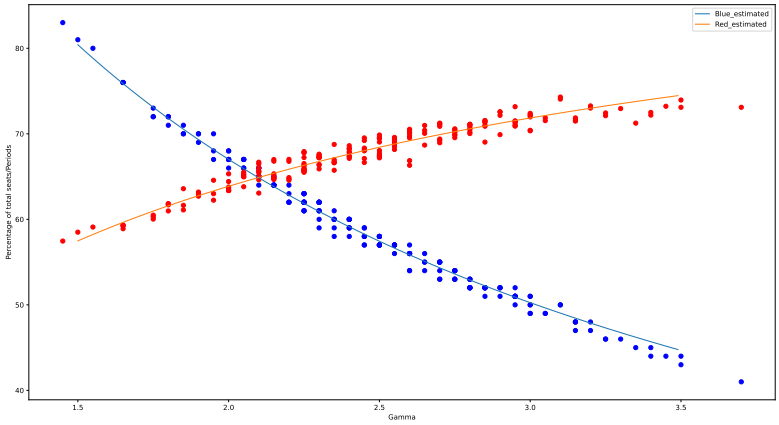


Figure 1: Gap points and their estimation under 200 probabilities

analysis using a sample of 100 probability combinations, each with a mean equal to γ. The values of γ range from 1.5 to 3.4. We employed an Ordinary Least Squares (OLS) model to fit the data and derive the parameter values. The goodness of fit is assessed using the R-square values, which are found to be 1.000 for all models, indicating a perfect fit between the data and the models.

The results of the estimation of c_1 and c_2 are presented in the table below:

Table 2: Estimation of c_1 and c_2

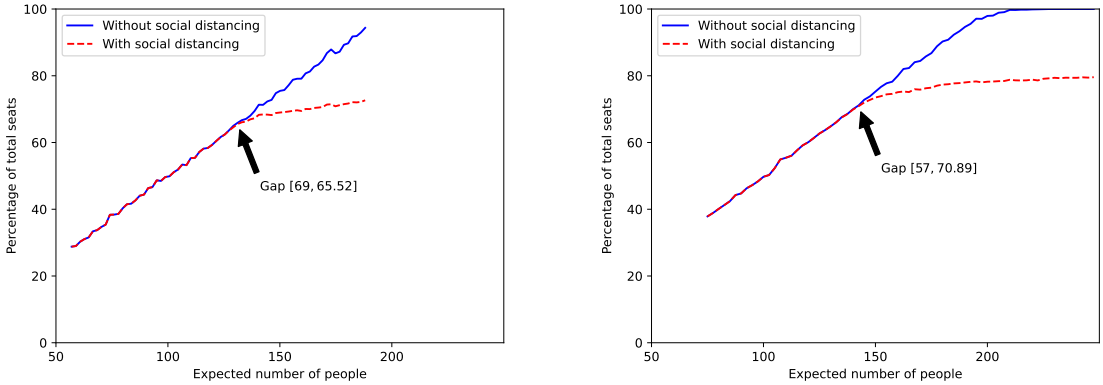
Seat layout(# of rows × # of seats)	Estimation of c_1	Estimation of c_2
10 × 11	0.909 ± 0.013	89.89 ± 1.436
10 × 16	0.948 ± 0.008	94.69 ± 0.802
10 × 21	0.955 ± 0.004	95.44 ± 0.571
10 × 26	0.966 ± 0.004	96.23 ± 0.386
10 × 31	0.965 ± 0.003	96.67 ± 0.434
10 × 36	0.968 ± 0.003	97.04 ± 0.289

6.2.2 Impact of Social Distancing as Demands Increase

Now, we consider impact of social distance as demands increase by changing T. Specifically, we consider two situations: $\gamma = 1.9$ and $\gamma = 2.5$. We set the parameters as follows: T varies from 30 to 120, the step size is 1. The seat layout consists 10 rows and the number of seats per row is 21. The social distancing is 1 seat.

The figure below displays the outcomes of groups who were accepted under two different conditions: with social distancing measures and without social distancing measures. For the former case, we employ DSA to obtain the results. In this case, we consider the constraints of social distancing and optimize the seat allocation accordingly. For the latter case, we adopt a different approach. We simply accept all incoming groups as long as the capacity allows, without considering the constraints of groups needing to sit together. This means that we prioritize filling the available seats without enforcing any specific seating arrangements or social distancing requirements. We conduct an analysis using a sample of 100 probability combinations associated with the same γ. The occupancy rate at different demands is calculated as the

mean of these 100 samples. The figures depicting the results are presented below.



(a) When $\gamma = 1.9$ (b) When $\gamma = 2.5$

Figure 2: The occupancy rate over the number of arriving people

The analysis consists of three stages. In the first stage, when the capacity is sufficient, the outcome remains unaffected by the implementation of social distancing measures. In the second stage, as the value of T increases, the difference between the outcomes with and without social distancing measures becomes more pronounced. Finally, in the third stage, as T continues to increase, both scenarios reach their maximum capacity acceptance. At this point, the gap between the outcomes with and without social distancing measures begins to converge. For the social distancing situation, according to Proposition 1, when the largest pattern is assigned to each row, the resulting occupancy rate is $\frac{16}{20} = 80\%$, which is the upper bound of occupancy rate.

The below table presents the percentage differences for different demand levels (130, 150, 170, 190, 210).

Table 3: Gap points and percentage differences under different demands of different gammas

γ	gap point	percentage differences under different demands				
		130	150	170	190	210
1.9	[69, 65.52]	0.25	5.82	12.82	20.38	24.56
2.1	[64, 67.74]	0.05	4.11	11.51	18.77	21.87
2.3	[61, 69.79]	0	2.29	10.21	17.36	21.16
2.5	[57, 70.89]	0	1.45	9.30	15.78	19.80
2.7	[53, 71.28]	0	1.38	7.39	14.91	19.14

7 Conclusion

Since the outbreak of the pandemic, social distancing has been widely recognized as a crucial measure for containing the spread of the virus. It has been implemented in seating areas to ensure safety. While static seating arrangements can be addressed through integer programming by defining specific social distancing constraints, dealing with the dynamic situations is challenging.

Our paper focuses on the problem of dynamic seat assignment with social distancing in the context

of a pandemic. To tackle this problem, we propose a scenario-based stochastic programming approach to obtain a seat planning that adheres to social distancing constraints. We utilize the benders decomposition method to solve this model efficiently, leveraging its well-structured property. However, solving the integer programming formulation directly can be computationally prohibitive in some cases. Therefore, in practice, we consider the linear programming relaxation of the problem and devise an approach to obtain the seat planning, which consists of full or largest patterns. In our approach, seat planning can be seen as the supply for each group type. We assign groups to seats when the supply is sufficient. However, when the supply is insufficient, we employ a stochastic planning policy to make decisions on whether to accept or reject group requests.

We conducted several experiments to investigate various aspects of our approach. These experiments included comparing the running time of the benders decomposition method and integer programming, analyzing different policies for dynamic seat assignment, and evaluating the impact of implementing social distancing. The results of our experiments demonstrated that the benders decomposition method efficiently solves our model. In terms of dynamic seat assignment policies, we considered the classical bid-price control, booking limit control in revenue management, dynamic programming-based heuristics, and the first-come-first-served policy. Comparatively, our proposed policy exhibited superior performance.

Building upon our policies, we further evaluated the impact of implementing social distancing. By defining the gap point as the period at which the difference between applying and not applying social distancing becomes evident, we established a relationship between the gap point and the expected number of people in each period. We observed that as the expected number of people in each period increased, the gap point occurred earlier, resulting in a higher occupancy rate at the gap point.

Overall, our study highlights the importance of considering the operational significance behind social distancing and provides a new perspective for the government to adopt mechanisms for setting seat assignments to protect people during the pandemic.

References

[1] Michael Barry, Claudio Gambella, Fabio Lorenzi, John Sheehan, and Joern Ploennigs. Optimal seat allocation under social distancing constraints. *arXiv preprint arXiv:2105.05017*, 2021.

[2] Matthew E Berge and Craig A Hopperstad. Demand driven dispatch: A method for dynamic aircraft capacity assignment, models and algorithms. *Operations research*, 41(1):153–168, 1993.

[3] Danny Blom, Rudi Pendavingh, and Frits Spieksma. Filling a theater during the covid-19 pandemic. *INFORMS Journal on Applied Analytics*, 52(6):473–484, 2022.

[4] Juliano Cavalcante Bortolete, Luis Felipe Bueno, Renan Butkeraites, Antônio Augusto Chaves, Gustavo Collaço, Marcos Magueta, FJR Pelogia, LL Salles Neto, TS Santos, TS Silva, et al. A support tool for planning classrooms considering social distancing between students. *Computational and Applied Mathematics*, 41:1–23, 2022.

[5] Michael S Casey and Suvrajeet Sen. The scenario generation algorithm for multistage stochastic linear programming. *Mathematics of Operations Research*, 30(3):615–631, 2005.

[6] Tommy Clausen, Allan Nordlunde Hjorth, Morten Nielsen, and David Pisinger. The off-line group seat reservation problem. *European journal of operational research*, 207(3):1244–1253, 2010.

[7] Renwick E Curry. Optimal airline seat allocation with fare classes nested by origins and destinations. *Transportation science*, 24(3):193–204, 1990.

[8] George B Dantzig. Discrete-variable extremum problems. *Operations research*, 5(2):266–288, 1957.

[9] Igor Deplano, Danial Yazdani, and Trung Thanh Nguyen. The offline group seat reservation knapsack problem with profit on seats. *IEEE Access*, 7:152358–152367, 2019.

[10] Yonghan Feng and Sarah M Ryan. Scenario construction and reduction applied to stochastic power generation expansion planning. *Computers & Operations Research*, 40(1):9–23, 2013.

[11] Martina Fischetti, Matteo Fischetti, and Jakob Stoustrup. Safe distancing in the time of covid-19. *European Journal of Operational Research*, 2021.

[12] Guillermo Gallego and Garrett Van Ryzin. A multiproduct dynamic pricing problem and its applications to network yield management. *Operations research*, 45(1):24–41, 1997.

[13] Elaheh Ghorbani, Hamid Molavian, and Fred Barez. A model for optimizing the health and economic impacts of covid-19 under social distancing measures; a study for the number of passengers and their seating arrangements in aircrafts. *arXiv preprint arXiv:2010.10993*, 2020.

[14] GovHK. Government relaxes certain social distancing measures. https://www.info.gov.hk/gia/general/202209/30/P2022093000818.htm, 2022.

[15] Younes Hamdouch, HW Ho, Agachai Sumalee, and Guodong Wang. Schedule-based transit assignment model with vehicle capacity and seat availability. *Transportation Research Part B: Methodological*, 45(10):1805–1830, 2011.

[16] Md Tabish Haque and Faiz Hamid. An optimization model to assign seats in long distance trains to minimize sars-cov-2 diffusion. *Transportation Research Part A: Policy and Practice*, 162:104–120, 2022.

[17] Md Tabish Haque and Faiz Hamid. Social distancing and revenue management—a post-pandemic adaptation for railways. *Omega*, 114:102737, 2023.

[18] Healthcare. Covid-19 timeline. https://www.otandp.com/covid-19-timeline, 2023.

[19] Réne Henrion and Werner Römisch. Problem-based optimal scenario generation and reduction in stochastic programming. *Mathematical Programming*, pages 1–23, 2018.

[20] Anton J Kleywegt and Jason D Papastavrou. The dynamic and stochastic knapsack problem. *Operations research*, 46(1):17–35, 1998.

[21] Sungil Kwag, Woo Jin Lee, and Young Dae Ko. Optimal seat allocation strategy for e-sports gaming center. *International Transactions in Operational Research*, 29(2):783–804, 2022.

[22] Rhyd Lewis and Fiona Carroll. Creating seating plans: a practical application. *Journal of the Operational Research Society*, 67(11):1353–1362, 2016.

[23] Yihua Li, Bruce Wang, and Luz A Caudillo-Fuentes. Modeling a hotel room assignment problem. *Journal of Revenue and Pricing Management*, 12:120–127, 2013.

[24] Jane F Moore, Arthur Carvalho, Gerard A Davis, Yousif Abulhassan, and Fadel M Megahed. Seat assignments with physical distancing in single-destination public transit settings. *Ieee Access*, 9:42985–42993, 2021.

[25] Imad A Moosa. The effectiveness of social distancing in containing covid-19. *Applied Economics*, 52(58):6292–6305, 2020.

[26] David Pisinger. An exact algorithm for large multiple knapsack problems. *European Journal of Operational Research*, 114(3):528–541, 1999.

[27] Mostafa Salari, R John Milne, Camelia Delcea, and Liviu-Adrian Cotfas. Social distancing in airplane seat assignments for passenger groups. *Transportmetrica B: Transport Dynamics*, 10(1):1070–1098, 2022.

[28] Mostafa Salari, R John Milne, Camelia Delcea, Lina Kattan, and Liviu-Adrian Cotfas. Social distancing in airplane seat assignments. *Journal of Air Transport Management*, 89:101915, 2020.

[29] Garrett Van Ryzin and Gustavo Vulcano. Simulation-based optimization of virtual nesting controls for network revenue management. *Operations research*, 56(4):865–880, 2008.

[30] Elizabeth Louise Williamson. *Airline network seat inventory control: Methodologies and revenue impacts*. PhD thesis, Massachusetts Institute of Technology, 1992.

[31] Benson B Yuen. Group revenue management: Redefining the business process—part i. *Journal of Revenue and Pricing Management*, 1:267–274, 2002.

[32] Feng Zhu, Shaoxuan Liu, Rowan Wang, and Zizhuo Wang. Assign-to-seat: Dynamic capacity control for selling high-speed train tickets. *Manufacturing & Service Operations Management*, 2023.

Proof

(Proof of Proposition 1). *We can utilize a greedy approach to construct a pattern, denoted as* $\boldsymbol{h}_g$*, by following the steps outlined below. This approach aims to generate a pattern that maximizes the number of people accommodated within the given constraints.*

- *Begin by selecting the maximum group size, denoted as* n_M*, as many times as possible to fill up the available seats in the row.*
- *Allocate the remaining seats(if possible) in the row to the group with the corresponding size.*

Let $L = n_M \cdot q + r$*, where* q *represents the number of times* n_M *is selected (the quotient), and* r *represents the remainder, indicating the number of remaining seats. It holds that* $0 \leq r < n_M$*.*

The number of people accommodated in the pattern $\boldsymbol{h}_g$ *is given by* $|\boldsymbol{h}_g| = qM + \max\{r - \delta, 0\}$*. To establish the optimality of* $|\boldsymbol{h}_g|$ *as the largest possible number of people accommodated given the constraints of* L*,* δ*, and* M*, we can employ a proof by contradiction.*

Assuming the existence of a pattern $\boldsymbol{h}$ *such that* $|\boldsymbol{h}| > |\boldsymbol{h}_g|$*, we can derive the following inequalities:*

$$
\begin{aligned}
& \sum_i (n_i - \delta) h_i > qM + \max\{r - \delta, 0\} \\
\Rightarrow \; & L \geq \sum_i n_i h_i > \sum_i \delta h_i + qM + \max\{r - \delta, 0\} \\
\Rightarrow \; & q(M + \delta) + r > \sum_i \delta h_i + qM + \max\{r - \delta, 0\} \\
\Rightarrow \; & q\delta + r > \sum_i \delta h_i + \max\{r - \delta, 0\}
\end{aligned}
$$

Breaking down the above inequality into two cases:

(i) When $r > \delta$*, the inequality becomes* $q + 1 > \sum_i h_i$*. It should be noted that* h_i *represents the number of group type* i *in the pattern. Since* $\sum_i h_i \leq q$*, the maximum number of people that can be accommodated is* $qM < qM + r - \delta$*.*

(ii) When $r \leq \delta$*, we have the inequality* $q\delta + \delta \geq q\delta + r > \sum_i \delta h_i$*. Similarly, we obtain* $q + 1 > \sum_i h_i$*. Thus, the maximum number of people that can be accommodated is* qM*, which is not greater than* $|\boldsymbol{h}_g|$*.*

Therefore, $\boldsymbol{h}$ *cannot exist. All largest patterns can accommodate the same maximum number of people and have the same loss. Hence, the maximum number of people that can be accommodated in the largest pattern is* $qM + \max\{r - \delta, 0\}$*. Correspondingly, the loss of the largest pattern* $|\boldsymbol{h}_g|$ *is* $q\delta - \delta + \min\{r, \delta\}$*.* □

(Proof of Proposition 2). *Treat the groups as the items, the rows as the knapsacks. There are* M *types of items, the total number of which is* $K = \sum_i d_i$*, each item* k *has a profit* p_k *and weight* w_k*.*

Then this Integer Programming is a special case of the Multiple Knapsack Problem(MKP). Consider the solution to the linear relaxation of (1)*. Sort these items according to profit-to-weight ratios* $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq$

$\ldots \geq \frac{p_K}{w_K}$. *Let the break item* b *be given by* $b = \min\{j : \sum_{k=1}^{j} w_k \geq L\}$, *where* $L = \sum_{j=1}^{N} L_j$ *is the total size of all knapsacks. Then the Dantzig upper bound [8] becomes* $u_{\text{MKP}} = \sum_{j=1}^{b-1} p_j + \left(L - \sum_{j=1}^{b-1} w_j\right) \frac{p_b}{w_b}$. *The corresponding optimal solution is to accept the whole items from* 1 *to* $b-1$ *and fractional* $(L - \sum_{j=1}^{b-1} w_j)$ *item* b. *Suppose the item* b *belong to type* v, *then for* $i < v$, $x_{ij}^* = 0$; *for* $i > v$, $x_{ij}^* = d_i$; *for* $i = v$, $\sum_j x_{ij}^* = (L - \sum_{i=v+1}^{M} d_i n_i)/n_v$. □

(Proof of Lemma 1). *Note that* $\mathbf{f}^\intercal = [-\mathbf{1},\ \mathbf{0}]$ *and* $V = [W,\ I]$. *Based on this, we can derive the following inequalities:* $\boldsymbol{\alpha}^\intercal W \geq -\mathbf{1}$ *and* $\boldsymbol{\alpha}^\intercal I \geq \mathbf{0}$. *These inequalities indicate that the feasible region is nonempty and bounded. Moreover, let* $\alpha_0 = 0$. *From this, we can deduce that* $0 \leq \alpha_i \leq \alpha_{i-1} + 1$ *for* $i \in \mathcal{M}$. *Consequently, all extreme points within the feasible region are integral.* □

(Proof of Proposition 4). *According to the complementary slackness property, we can obtain the following equations*

$$\begin{aligned}
&\alpha_i(d_{i0} - d_{i\omega} - y_{i\omega}^+ + y_{i+1,\omega}^+ + y_{i\omega}^-) = 0, i = 1, \ldots, M-1 \\
&\alpha_i(d_{i0} - d_{i\omega} - y_{i\omega}^+ + y_{i\omega}^-) = 0, i = M \\
&y_{i\omega}^+(\alpha_i - \alpha_{i-1} - 1) = 0, i = 1, \ldots, M \\
&y_{i\omega}^- \alpha_i = 0, i = 1, \ldots, M.
\end{aligned}$$

When $y_{i\omega}^- > 0$, *we have* $\alpha_i = 0$; *when* $y_{i\omega}^+ > 0$, *we have* $\alpha_i = \alpha_{i-1} + 1$. *Let* $\Delta d = d_\omega - d_0$, *then the elements of* Δd *will be a negative integer, positive integer and zero. When* $y_{i\omega}^+ = y_{i\omega}^- = 0$, *if* $i = M$, $\Delta d_M = 0$, *the value of objective function associated with* α_M *is always* 0, *thus we have* $0 \leq \alpha_M \leq \alpha_{M-1} + 1$; *if* $i < M$, *we have* $y_{i+1,\omega}^+ = \Delta d_i \geq 0$. *If* $y_{i+1,\omega}^+ > 0$, *the objective function associated with* α_i *is* $\alpha_i \Delta d_i = \alpha_i y_{i+1,\omega}^+$, *thus to minimize the objective value, we have* $\alpha_i = 0$; *if* $y_{i+1,\omega}^+ = 0$, *we have* $0 \leq \alpha_i \leq \alpha_{i-1} + 1$. □

(Proof of Proposition 5). *Suppose we have one extreme point* $\boldsymbol{\alpha}_\omega^0$ *for each scenario. Then we have the following problem.*

$$\begin{aligned}
\max \quad & \mathbf{c}^\intercal \mathbf{x} + \sum_{\omega \in \Omega} p_\omega z_\omega \\
s.t. \quad & \mathbf{n}\mathbf{x} \leq \mathbf{L} \\
& (\boldsymbol{\alpha}_\omega^0)^\intercal \mathbf{d}_\omega \geq (\boldsymbol{\alpha}_\omega^0)^\intercal \mathbf{x}\mathbf{1} + z_\omega, \forall \omega \\
& \mathbf{x} \in \mathbb{Z}_+
\end{aligned} \tag{16}$$

Problem (16) *reaches its maximum when* $(\boldsymbol{\alpha}_\omega^0)^\intercal \mathbf{d}_\omega = (\boldsymbol{\alpha}_\omega^0)^\intercal \mathbf{x}\mathbf{1} + z_\omega, \forall \omega$. *Substitute* z_ω *with these equations, we have*

$$\begin{aligned}
\max \quad & \mathbf{c}^\intercal \mathbf{x} - \sum_{\omega} p_\omega (\boldsymbol{\alpha}_\omega^0)^\intercal \mathbf{x}\mathbf{1} + \sum_{\omega} p_\omega (\boldsymbol{\alpha}_\omega^0)^\intercal \mathbf{d}_\omega \\
s.t. \quad & \mathbf{n}\mathbf{x} \leq \mathbf{L} \\
& \mathbf{x} \in \mathbb{Z}_+
\end{aligned} \tag{17}$$

Notice that $\mathbf{x}$ *is bounded by* $\mathbf{L}$, *then the problem* (16) *is bounded. Adding more constraints will not make the optimal value larger. Thus, RBMP is bounded.* □

(Proof of Proposition 3). *In any optimal solution where one of the corresponding patterns is not full or largest, we have the flexibility to allocate the remaining unoccupied seats. These seats can be assigned to either a new seat planning or added to an existing seat planning. Importantly, since a group can utilize the seat planning of a larger group, the allocation scheme based on the original optimal solution will not affect the optimality of the solution. For each row, there are three situations to allocate the seats. First, when the rest seats can be allocated to the existing groups, then the corresponding pattern becomes a full pattern. Second, when all the existing groups are the largest groups and the rest seats cannot construct a new group, the pattern becomes the largest. Third, when all the existing groups are the largest groups and the rest seats can construct new groups, the rest seats can be used to construct the largest groups until there is no enough capacity, then the pattern becomes the largest. Finally, we can allocate the seats such that each row in the seat planning becomes either full or largest.* □

(Proof of Lemma 2). *According to the Proposition 2, the aggregate optimal solution to relaxation of problem (1) takes the form* $xe_h + \sum_{i=h+1}^{M} d_i e_i$, *then according to the complementary slackness property, we know that* $z_1, \ldots, z_h = 0$. *This implies that* $\beta_j \geq \frac{n_i - \delta}{n_i}$ *for* $i = 1, \ldots, h$. *Since* $\frac{n_i - \delta}{n_i}$ *increases with* i, *we have* $\beta_j \geq \frac{n_h - \delta}{n_h}$. *Consequently, we obtain* $z_i \geq n_i - \delta - n_i \frac{n_h - \delta}{n_h} = \frac{\delta(n_i - n_h)}{n_h}$ *for* $i = h+1, \ldots, M$.

Given that $\mathbf{d}$ *and* $\mathbf{L}$ *are both no less than zero, the minimum value will be attained when* $\beta_j = \frac{n_h - \delta}{n_h}$ *for all* j, *and* $z_i = \frac{\delta(n_i - n_h)}{n_h}$ *for* $i = h+1, \ldots, M$. □

8 Other Results

8.1 Running times of solving SSP and relaxation of SSP with Benders Decomposition

The running times of solving SSP directly and solving the LP relaxation of SSP with Benders decomposition are shown in Table 4.

Table 4: Running times of solving SSP and relaxation of SSP with Benders Decomposition

# of scenarios	Demands	# of rows	# of groups	# of seats	Running time of IP(s)	Benders (s)
1000	(150, 350)	30	8	(21, 50)	5.1	0.13
5000		30	8		28.73	0.47
10000		30	8		66.81	0.91
50000		30	8		925.17	4.3
1000	(1000, 2000)	200	8	(21, 50)	5.88	0.29
5000		200	8		30.0	0.62
10000		200	8		64.41	1.09
50000		200	8		365.57	4.56
1000	(150, 250)	30	16	(41, 60)	17.15	0.18
5000		30	16		105.2	0.67
10000		30	16		260.88	1.28
50000		30	16		3873.16	6.18

The parameters in the columns of the table are the number of scenarios, the range of demands, running time of SSP, running time of Benders decomposition method, the number of rows, the number of group types and the number of seats for each row, respectively. Take the first experiment as an example, the scenarios of demands are generated from (150, 350) randomly, the number of seats for each row is generated from (21, 50) randomly.

It is evident that the utilization of Benders decomposition can enhance computational efficiency.

8.2 Seat Planning From SSP Relaxation versus Seat Planning From SSP

In this section, we compare the accpected people of seat plannings from SSP relaxation and SSP under group-type control policy.

An arrival sequence can be expressed as $\{y_1, y_2, \ldots, y_T\}$. Let $N_i = \sum_t I(y_t = i)$, i.e., the number of times group type i arrives during T periods. Then the scenarios, $(N_1, \ldots, N_M)$, follow a multinomial distribution,

$$p\left(N_1, \ldots, N_M \mid \mathbf{p}\right) = \frac{T!}{N_1!, \ldots, N_M!} \prod_{i=1}^{M} p_i^{N_i}, T = \sum_{i=1}^{M} N_i.$$

It is clear that the number of different sequences is M^T. The number of different scenarios is $O(T^{M-1})$ which can be obtained by the following DP.

Use $D(T, M)$ to denote the number of scenarios, which equals the number of different solutions to $x_1 + \ldots + x_M = T, \mathbf{x} \geq 0$. Then, we know the recurrence relation $D(T, M) = \sum_{i=0}^{T} D(i, M-1)$ and boundary condition, $D(i, 1) = 1$. So we have $D(T, 2) = T + 1$, $D(T, 3) = \frac{(T+2)(T+1)}{2}, D(T, M) = O(T^{M-1})$. The number of scenarios is too large to enumerate all possible cases. Thus, we choose to sample some sequences from the multinomial distribution.

Then, we will show these two seat plannings have a close performance when considering group-type control policy.

Table 5: Seat planning from SSP relaxation versus seat planning from SSP

# samples	T	probabilities	# rows	people served by SSP relaxation	people served by SSP
1000	45	[0.4,0.4,0.1,0.1]	8	85.30	85.3
1000	50	[0.4,0.4,0.1,0.1]	8	97.32	97.32
1000	55	[0.4,0.4,0.1,0.1]	8	102.40	102.40
1000	60	[0.4,0.4,0.1,0.1]	8	106.70	NA
1000	65	[0.4,0.4,0.1,0.1]	8	108.84	108.84
1000	35	[0.25,0.25,0.25,0.25]	8	87.16	87.08
1000	40	[0.25,0.25,0.25,0.25]	8	101.32	101.24
1000	45	[0.25,0.25,0.25,0.25]	8	110.62	110.52
1000	50	[0.25,0.25,0.25,0.25]	8	115.46	NA
1000	55	[0.25,0.25,0.25,0.25]	8	117.06	117.26
5000	300	[0.25,0.25,0.25,0.25]	30	749.76	749.76
5000	350	[0.25,0.25,0.25,0.25]	30	866.02	866.42
5000	400	[0.25,0.25,0.25,0.25]	30	889.02	889.44
5000	450	[0.25,0.25,0.25,0.25]	30	916.16	916.66

Each entry of people served is the average of 50 instances. IP will spend more than 2 hours in some instances, as 'NA' showed in the table. The number of seats is 20 when the number of rows is 8, the number of seats is 40 when the number of rows is 30.