Dynamic Seat assignment with the social distancing

Dis· count

April 3, 2023

Abstract

Social distancing has been broadly recognized and practiced as a non-pharmaceutical way to contain the spread of infectious diseases. In this study, we consider the dynamic seat assignment problem with social distancing, where groups arrive dynamically.

During a pandemic, the government may issue minimum physical distance requirements between people, which must be respected in the seating assignment. This problem becomes further complicated by the existence of groups of guests who will be seated together. To address this challenge, we provide an optimal seat assignment solution with given rows of seats and demands of groups.

We also develop a scenario-based method to obtain a seat assignment with stochastic demands of groups. In business, where groups arrive dynamically, we provide both stochastic and non-stochastic methods to address this challenge.

Our results provide valuable insights for policymakers and venue managers regarding seat utilization rates and offer a guideline for policies related to social distancing measures. Overall, our proposed approach provides a practical tool for venues to implement social distancing measures while optimizing seat assignments and ensuring the safety of groups.

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

1 Introduction

Reducing the spread of the virus while minimizing the economic impact is a significant challenge for governments worldwide. During the epidemic, many companies and countries have implemented restrictive measures, including social distancing, as it is believed to be the most effective non-pharmaceutical treatment to reduce the health effects of Covid-19. Additional measures such as hand washing and the use of masks have also been strongly recommended or enforced by companies and governments.

While social distancing is a recognized and practiced method for containing the spread of infectious diseases, operational guidance for its implementation is still lacking, particularly in social distance measures that involve operational details. An example of such a measure is ensuring social distancing in seating plans.

In this paper, we address the dynamic seating assignment problem with a given set of seats in the context of a pandemic. The government may issue minimum physical distance requirements between people, which must be implemented in the seating plan. The problem becomes further complicated by the existence of groups of guests who can sit together. To address this challenge, we develop a mechanism for seat planning, which includes a model to assess the riskiness of a seating plan and a solution approach to balance seat utilization rates and the associated risk of infection. Our proposed algorithm has the potential to help companies and governments optimize seat assignments while maintaining social distance measures and ensuring the safety of groups.

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 distance 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 distance [1, 6]. Recent studies have explored the effects of social distance on health and economics, mainly in the context of aircraft [7, 10, 11]. Our study provides a new perspective to help the government adopt a mechanism for setting seat assignments to protect people in the post-pandemic era.

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 consider the stochastic demand situation where the demands of different group types are random. By using two-stage stochastic programming and Benders decomposition methods, we obtain the optimal linear solution.

Third, to address the dynamic scenario problem, we first obtain a feasible seating plan using scenario-based stochastic planning. We then make a decision for each incoming group based on a nested policy, either accepting or rejecting the group. Our results demonstrate a significant improvement over a first-come first-served baseline strategy and provide guidance on how to develop attendance policies.

This study focuses on dynamic seat assignment with social distancing, where a one-seat distance must be maintained between different groups. Our goal is to obtain the final seating plan that satisfies social distancing constraints and implement the seat assignment when groups arrive. Overall, our proposed approach provides a comprehensive solution for dynamic seat assignment with social distancing, taking into account both deterministic and stochastic demand scenarios.

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 model and analyze its properties. Section 5 demonstrates the dynamic form and its property. Section 6 gives the results. The conclusions are shown in Section 7.

2 Literature Review

2.1 Dynamic seat assignment

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, 8]. This approach is also used in cinemas and concert venues to optimize the seating arrangements for the audience, with the goal of providing the best possible experience for each audience member while maximizing ticket sales and revenue for the venue.

However, implementing dynamic seat assignment with social distance in cinemas and concert venues presents several challenges. The primary challenge is to balance the need for safety with the desire to maximize revenue for the venue. This requires careful consideration of various factors, such as the layout of the venue, the number of available seats, and the preferences of the audience. To implement dynamic seat assignment with social distance, venues may need to adjust their seating plans to accommodate the recommended distance between audience members. This may involve reducing the overall number of seats or reconfiguring the seating arrangements.

Implementing dynamic seat assignment with social distance can be done manually by the staff or through automated systems that use algorithms to optimize the seat assignments based on various factors, such as ticket sales, seat availability, and customer preferences. However, the implementation of social distance measures poses unique challenges that require careful planning and consideration of various factors to balance safety with revenue generation.

Despite these challenges, dynamic seat assignment with social distance is a promising approach to help ensure the safety of audience members in cinemas, concert venues, and other public spaces during the COVID-19 pandemic.

2.2 Seat assignemnt with social distance

Almost all existing work focuses on the static version of seat assignment with social distance.

2.3 Scenario generation

It is challenging to consider all the possible realizations; thus, it is practicable to use discrete distributions with a finite number of scenarios to approximate the random demands. This procedure is often called scenario generation.

Some papers consider obtaining a set of scenarios that realistically represents the distributions of the random parameters but is not too large. [5] [3] [9]

Another process to reduce the calculation is called scenario reduction. It tries to approximate the original scenario set with a smaller subset that retains essential features.

3 Problem Description

In this section, we consider the dynamic seat assignment problem with social distance.

3.1 Dynamic Seat Assignment Problem with Social Distance

Generally speaking, there are two ways to purchase tickets for concerts or movies: no seat selection when booking and seat selection when booking. We consider the following dynamic demand situations, seat assignment after booking period and seat assignment when booking.

For reservations without seat selection, the seat assignment will not be made immediately. Instead, the decision-maker must either accept or reject each request during the making-reservation stage. After the reservation deadline, the seller will inform the customers of the seat layout information before admission. For example, in singing concert venues with many seats and high ticket demand, organizers usually do not determine the seats when booking and then inform customers of the seat information after the overall seat layout is determined.

In contrast, for seat assignment when booking, the specific procedure will be changed to meet the requirements of social distancing. The seat assignment will be arranged before groups book their tickets, and the groups will only need to choose seats of the corresponding size when booking. For example, in movie theaters or small concerts with relatively few seats, the attendance rate is usually low enough to allow free selection of seats directly online. Early seat planning can satisfy the requirement of social distancing and save costs without changing seat allocation. The seat assignment could remain for one day because the same film genre will attract the same feature of different group types.

Our study mainly focuses on the latter situation where customers come dynamically, and the seat assignment needs to be made immediately without knowing the number and composition of future customers. In Section 6, we also consider the situation where the seat assignment can be made after the booking period.

Consider a set of groups, each consisting of no more than m people, to be assigned to a set of seats. These groups are denoted by a demand vector, $\mathbf{d} = (d_1, \ldots, d_m)$. Each element, $d_i, 1 \leq i \leq m$, indicates the number of group type i containing i people. For illustration, we consider the layout as N rows, each row with S_j seats, $j = 1, \ldots, N$.

We use a vector $\mathbf{S} = (S_1^r, S_2^r, \ldots, S_N^r)$ to record the remaining capacity of rows, in every period the group can decide which row to sit. Let $V_t(\mathbf{S})$ denote the maximal expected value to go at period t with capacity of rows. Let $\mathbf{w} = (s_1 + 1, \ldots, s_m + 1)$ be the number of seats occupied by each group type. There are T periods and the arrival probability of the group type i in each period is p_i.

The dynamic programming formulation for this problem is

$$V_t(\mathbf{S}) = \sum_{i \in m} p_i \max\{[V_{t-1}(\mathbf{S} - \mathbf{w} \circ \mathbf{e}_i) + s_i], V_{t-1}(\mathbf{S})\}, \mathbf{S} \geq \mathbf{0}, V_{T+1}(\mathbf{S}) = 0,$$

where $\mathbf{e}_i$ is the unit vector whose i-th element is 1. $\circ$ is the element-wise product. Initially, we have $\mathbf{S}_T = (S_1, S_2, \ldots, S_N)$.

As we can see, this dynamic programming falls into the curse of dimensionality due to many seating plan combinations. Even worse, DP only records the remaining seats of each row, cannot reach the goal of assigning to seats at each period. To avoid this complexity, we develop an approach that aims directly at the final seating plans, then make the policy to assign groups.

3.2 Seat Planning with Social Distance

In accordance with epidemic prevention requirements, customers from the same group are allowed to sit together, while different groups must maintain social distancing. Specifically, each group must leave one seat empty to maintain social distancing from adjacent groups. Additionally, different rows do not affect each other, meaning that a person from one group can sit directly behind a person from another group.

To achieve the social distancing requirements in the seat planning process, we add one to the original size of each group to create the new size of the group. We also add one dummy seat to each row. Let $s_i = i + 1$ denote the new size of group type i, and let $L_j = S_j + 1$ denote the length of row j, where S_j represents the number of seats in row j.

Then we can illustrate the seat planning for one row below.

Figure 1: Problem Conversion

On the left side of the diagram, the blue squares represent the empty seats required for social distancing, while the orange squares represent the seats occupied by groups. On the right side, we have added one dummy seat at the end of each row. The orange squares surrounded by the red line represent the seats taken by groups in this row, which includes two groups of 1, one group of 2, and one group of 3.

By incorporating these additional seats and designating certain seats for social distancing, we can integrate social distancing measures into the seat planning problem.

To obtain the final seat planning firstly, we develop the scenario-based stochastic programming.

4 Scenario-based Stochastic Programming

Firstly, we give the formulation of the scenario-based stochastic programming. Then we develop the benders decomposition to obtain the optimal linear solution. Finally, we obtain a feasible seat planning.

4.1 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 $\omega \in \Omega, \Omega$ corre-

sponds to a particular realization of the demand vector, $\mathbf{D}_\omega = (d_{1\omega}, d_{2\omega}, \ldots, d_{m,\omega})$. Let p_ω denote the probability of any scenario ω, which we assume to be positive. To maximize the expected value of people over all the scenarios, we propose a scenario-based stochastic programming.

Consider the decision makers who give the seat assignment based on the scenarios then assign the groups to seats according to the realized true demand.

The seat assignment can be denoted by decision variables $\mathbf{x} \in \mathbb{Z}_+^{m\times N}$. Let $x_{i,j}$ stand for the number of group type i in row j. The supply for group type i can be represented by $\sum_{j=1}^{N} x_{ij}$. Regarding the nature of the obtained information, we assume that there are $S = |\Omega|$ possible scenarios. There is a scenario-dependent decision variable, $\mathbf{y}$, to be chosen. It includes two vectors of decisions, $\mathbf{y}^+ \in \mathbb{Z}_+^{m\times S}$ and $\mathbf{y}^- \in \mathbb{Z}_+^{m\times S}$. Each component of $\mathbf{y}^+$, $y_{i\omega}^+$, represents the number of surplus seats for group type i. Similarly, $y_{i\omega}^-$ represents the number of inadequate seats for group type i. Considering that the group can take the seats assigned to the larger group type, we assume that the surplus group type i can be occupied by smaller group type $j < i$ in the descending order of the group size. That is, for any ω, $i \le m-1$, $y_{i\omega}^+ = \left(\sum_{j=1}^{N} x_{ij} - d_{i\omega} + y_{i+1,\omega}^+\right)^+$ and $y_{i\omega}^- = \left(d_{i\omega} - \sum_{j=1}^{N} x_{ij} - y_{i+1,\omega}^+\right)^+$, where $(x)^+$ equals x if $x > 0$, 0 otherwise. Specially, for the largest group type m, we have $y_{m\omega}^+ = (\sum_{j=1}^{N} x_{ij} - d_{i\omega})^+$, $y_{m\omega}^- = (d_{i\omega} - \sum_{j=1}^{N} x_{ij})^+$.

Then we have the deterministic equivalent form of the scenario-based stochastic programming:

$$
\begin{aligned}
\max \quad & E_\omega \left[\sum_{i=1}^{m-1} (s_i - 1)\Big(\sum_{j=1}^{N} x_{ij} + y_{i+1,\omega}^+ - y_{i\omega}^+\Big) + (s_m - 1)\Big(\sum_{j=1}^{N} x_{mj} - y_{m\omega}^+\Big) \right] \\
\text{s.t.} \quad & \sum_{j=1}^{N} x_{ij} - y_{i\omega}^+ + y_{i+1,\omega}^+ + y_{i\omega}^- = d_{i\omega}, \quad i = 1, \ldots, m-1, \omega \in \Omega \\
& \sum_{j=1}^{N} x_{ij} - y_{i\omega}^+ + y_{i\omega}^- = d_{i\omega}, \quad i = m, \omega \in \Omega \\
& \sum_{i=1}^{m} s_i x_{ij} \le L_j, j = 1, \ldots, N \\
& y_{i\omega}^+, y_{i\omega}^- \in \mathbb{Z}_+, \quad i \in I, \omega \in \Omega \\
& x_{ij} \in \mathbb{Z}_+, \quad i = 1, \ldots, m, j = 1, \ldots, N.
\end{aligned} \tag{1}
$$

The objective function contains two parts, the number of the largest group type that can be accommodated is $\sum_{j=1}^{N} x_{mj} - y_{m\omega}^+$. The number of group type i that can be accommodated is $\sum_{j=1}^{N} x_{ij} + y_{i+1,\omega}^+ - y_{i\omega}^+$.

Reformulate the objective function,

$$
\begin{aligned}
&E_\omega \left[\sum_{i=1}^{m-1} (s_i - 1)(\sum_{j=1}^{N} x_{ij} + y^+_{i+1,\omega} - y^+_{i\omega}) + (s_m - 1)(\sum_{j=1}^{N} x_{mj} - y^+_{m\omega}) \right] \\
=&\sum_{j=1}^{N}\sum_{i=1}^{m}(s_i - 1)x_{ij} - \sum_{\omega=1}^{S} p_\omega \left(\sum_{i=1}^{m}(s_i - 1)y^+_{i\omega} - \sum_{i=1}^{m-1}(s_i - 1)y^+_{i+1,\omega} \right) \\
=&\sum_{j=1}^{N}\sum_{i=1}^{m}(s_i - 1)x_{ij} - \sum_{\omega=1}^{S} p_\omega \left((s_1 - 1)y^+_{1\omega} + \sum_{i=2}^{m}(s_i - s_{i-1})y^+_{i\omega} \right)
\end{aligned}
$$

Let $\mathbf{s} = (s_1, \ldots, s_m)$, $\mathbf{L} = (L_1, \ldots, L_N)$ where s_i is the size of seats taken by group type i and L_j is the length of row j as we defined above. Then the row length constraint can be expressed as $\mathbf{sx} \leq \mathbf{L}$.

The linear constraints associated with scenarios can be written in a matrix form as

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

where $\mathbf{1}$ is the 1-vector of size N, $\mathbf{V} = [\mathbf{W},\ \mathbf{I}]$.

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

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

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

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, let us write this problem in the form of the two-stage stochastic programming.

Let $\mathbf{c}'\mathbf{x} = \sum_{j=1}^{N}\sum_{i=1}^{m} ix_{ij}$, $\mathbf{f}'\mathbf{y}_\omega = -\sum_{i=1}^{m} y^+_{i\omega}$. Then the formulation (1) can be expressed as below,

$$
\begin{aligned}
\max \quad & c'\mathbf{x} + z(\mathbf{x}) \\
\text{s.t.} \quad & \mathbf{sx} \leq \mathbf{L} \\
& \mathbf{x} \in \mathbb{Z}_+^{m \times N},
\end{aligned}
\tag{2}
$$

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

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

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

$$
\begin{aligned}
z_\omega(\mathbf{x}) := \max \quad & \mathbf{f}'\mathbf{y}_\omega \\
\text{s.t.} \quad & \mathbf{x1} + \mathbf{V}\mathbf{y}_\omega = \mathbf{d}_\omega \\
& \mathbf{y}_\omega \geq 0.
\end{aligned} \tag{3}
$$

Here E is the expectation with respect to the scenario set. Problem (3) stands for the second-stage problem and $z_\omega(\mathbf{x})$ is the optimal value of problem (3), together with the convention $z_\omega(\mathbf{x}) = \infty$ if the problem is infeasible.

It is difficult to solve the above problem directly, we can relax problem (2) to stochastic linear programming firstly. In section 4.2, we obtain an optimal linear solution by decomposition approach and generate a near-optimal seat assignment.

4.2 Solve The Scenario-based Two-stage Problem

At first, we generate a closed-form solution to the second-stage problem in section 4.2.1. Then we obtain the solution to the linear relaxation of problem (2) by the delayed constraint generation. Finally, we obtain a near-optimal seat assignment from the linear solution.

4.2.1 Solve The Second Stage Problem

Consider a $\mathbf{x}$ such that $\mathbf{sx} \leq \mathbf{L}$ and $\mathbf{x} \geq 0$ and suppose that this represents our seat assignment for the first stage decisions. Once $\mathbf{x}$ is fixed, the optimal second stage decisions $\mathbf{y}_\omega$ can be determined by solving problem (3) for each ω.

To solve this problem, we should only consider that $\mathbf{x}$ for which $z_\omega(\mathbf{x})$ are all finite. Notice that the feasible region of the dual of problem (3) does not depend on $\mathbf{x}$. We can form its dual problem, which is

$$
\begin{aligned}
\min \quad & \alpha'_\omega(\mathbf{d}_\omega - \mathbf{x1}) \\
\text{s.t.} \quad & \alpha'_\omega \mathbf{V} \geq \mathbf{f}'
\end{aligned} \tag{4}
$$

Let $P = \{\alpha | \alpha' V \geq \mathbf{f}'\}$. We assume that P is nonempty and has at least one extreme point. Then, either the dual problem (4) has an optimal solution and $z_\omega(\mathbf{x})$ is finite, or the primal problem (3) is infeasible and $z_\omega(\mathbf{x}) = \infty$.

Let $\mathcal{O}$ be the set of all extreme points of P and $\mathcal{F}$ be the set of all extreme rays of P. Then $z_\omega > -\infty$ if and only if $(\alpha^k)'(\mathbf{d}_\omega - \mathbf{x1}) \geq 0, \alpha^k \in \mathcal{F}$, which stands for the feasibility cut.

Lemma 1. *The feasible region of problem (4), P, is bounded. In addition, all the extreme points of P are integral.*

(Proof of lemma 1)**.** *Notice that $V = [W,\ I]$, W is a totally unimodular matrix. Then, we have $\alpha' W \geq -\bar{s}, \alpha' I \geq 0$. Thus, the feasible region is bounded. Further more, $\bar{s}_i = s_i - s_{i-1}, s_0 = 1$ are integral, so the extreme points are all integral.* □

Because the feasible region is bounded, then feasibility cuts are not needed. Let z_ω be the lower bound of $z_\omega(x)$ such that $(\alpha^k)'(\mathbf{d}_\omega - \mathbf{x1}) \geq z_\omega, \alpha^k \in \mathcal{O}$, which is the optimality cut.

Corollary 1. *Only the optimality cuts, $\alpha'(\mathbf{d}_\omega - \mathbf{x1}) \geq z_\omega$, will be included in the decomposition approach.*

Corollary 2. *When $s_i = i+1$, $f' = [-\mathbf{1},\ \mathbf{0}], V = [W,\ I]$, we have $\alpha' W \geq -1, \alpha' I \geq 0$. Thus, it is easy to find that the feasible region is bounded, i.e., P does not contain any extreme rays. Furthermore, let $\alpha_0 = 0$, then we have $0 \leq \alpha_i \leq \alpha_{i-1} + 1$, $i = 1, \ldots, m$.*

Corollary 3. *The optimal value of the problem (3), $z_\omega(x)$, is finite and will be attained at extreme points of the set P. Thus, we have $z_\omega(x) = \min_{\alpha^j \in \mathcal{O}} (\alpha^k)'(\mathbf{d}_\omega - \mathbf{x1})$.*

When we are given x^*, the demand that can be satisfied by the assignment is $\mathbf{x}^*\mathbf{1} = \mathbf{d}_0 = (d_{1,0}, \ldots, d_{m,0})$. Then plug them in the subproblem (3), we can obtain the value of $y_{i\omega}$ recursively:

$$\begin{aligned} y^-_{m\omega} &= (d_{m\omega} - d_{m0})^+ \\ y^+_{m\omega} &= (d_{m0} - d_{m\omega})^+ \\ y^-_{i\omega} &= \left(d_{i\omega} - d_{i0} - y^+_{i+1,\omega}\right)^+, i = 1, \ldots, m-1 \\ y^+_{i\omega} &= \left(d_{i0} - d_{i\omega} + y^+_{i+1,\omega}\right)^+, i = 1, \ldots, m-1 \end{aligned} \tag{5}$$

The optimal value for scenario ω can be obtained by $f' y_\omega$, then we need to find the dual optimal solution.

Theorem 1. *The optimal solutions to problem (4) are given by*

$$\begin{aligned} &\alpha_{i\omega} = 0, i = 1, \ldots, m \quad \text{if } y^-_{i\omega} > 0 \\ &\alpha_{i\omega} = \alpha_{i-1,\omega} + 1, i = 1, \ldots, m \quad \text{if } y^+_{i\omega} > 0 \end{aligned} \tag{6}$$

For some i, when $y^+_{i\omega} = 0$ and $y^-_{i\omega} = 0$, $\left(d_{i0} - d_{i\omega} + y^+_{i+1,\omega}\right) = 0$, $d_{i\omega} - d_{i0} = y^+_{i+1,\omega} \geq 0$. If $y^+_{i+1,\omega} > 0$, $\alpha_{i\omega} = 0$; if $y^+_{i+1,\omega} = 0$, $0 \leq \alpha_{i\omega} \leq \alpha_{i-1,\omega} + 1$.

(Proof of Theorem 1). *According to the complementary relaxation property, when $d_{i\omega} > d_{i0} \Rightarrow y^-_{i\omega} > 0$, then $\alpha_{i\omega} = 0$ for all i; when $d_{i\omega} < d_{i0} \Rightarrow y^+_{i\omega} > 0$, then $\alpha_{i\omega} = \alpha_{i-1,\omega} + 1, i = 1, \ldots, m$.*

When $d_{i\omega} = d_{i0}$, we can find that $\alpha_{i\omega} = \alpha_{i-1,\omega} + 1$ will minimize the objective function.

Let $\Delta d = d_\omega - d_0$, then the elements in Δd will be a negative integer, positive integer and zero. Only the negative element will affect the objective function. The larger the value of α associated with a negative integer is, the smaller the objective function will be. Thus, let $\alpha_{i\omega} = \alpha_{i-1,\omega} + 1$ when $d_{i\omega} = d_{i0}$ can obtain the minimized objective function. □

We can use the forward method, calculating from $\alpha_{1\omega}$ to $\alpha_{m\omega}$, to obtain the value of α_ω instead of solving linear programming.

4.2.2 Delayed Constraint Generation

Benders decomposition works with only a subset of those exponentially many constraints and adds more constraints iteratively until the optimal solution of Benders Master Problem(BMP) is attained. This procedure is known as delayed constraint generation.

Use the characterization of $z_\omega(x)$ in the problem (2) and take into account the optimality cut, we can conclude the BMP will have the form:

$$\begin{aligned} \max \quad & c'x + \sum_{\omega\in\Omega} p_\omega z_\omega \\ \text{s.t.} \quad & \sum_{i=1}^{m} s_i x_{ij} \le L_j, j = 1,\dots,N \\ & (\alpha^k)'(\mathbf{d}_\omega - \mathbf{x}\mathbf{1}) \ge z_\omega, \alpha^k \in \mathcal{O}, \forall\omega \\ & \mathbf{x} \ge 0 \end{aligned} \tag{7}$$

When substituting $\mathcal{O}$ with its subset, $\mathcal{O}^t$, the problem (7) becomes the Restricted Benders Master Problem(RBMP).

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

Lemma 2. *RBMP is always bounded with at least any one optimality cut for each scenario.*

(Proof of lemma 2)**.** *Suppose we have one extreme point α^ω for each scenario. Then we have the following problem.*

$$\begin{aligned} \max \quad & c'x + \sum_{\omega\in\Omega} p_\omega z_\omega \\ s.t. \quad & \sum_{i=1}^{m} s_i x_{ij} \le L_j, j = 1,\dots,N \\ & (\alpha^\omega)'\mathbf{d}_\omega \ge (\alpha^\omega)'\mathbf{x}\mathbf{1} + z_\omega, \forall\omega \\ & \mathbf{x} \ge 0 \end{aligned} \tag{8}$$

Problem (8) reaches its maximum when $(\alpha^\omega)'\mathbf{d}_\omega = (\alpha^\omega)'\mathbf{x}\mathbf{1} + z_\omega, \forall\omega$. Substitute z_ω with these equations, we have

$$\begin{aligned} \max \quad & c'x - \sum_{\omega} p_\omega(\alpha^\omega)'\mathbf{x}\mathbf{1} + \sum_{\omega} p_\omega(\alpha^\omega)'\mathbf{d}_\omega \\ s.t. \quad & \sum_{i=1}^{m} s_i x_{ij} \le L_j, j = 1,\dots,N \\ & \mathbf{x} \ge 0 \end{aligned} \tag{9}$$

Notice that $\mathbf{x}$ is bounded by $\mathbf{L}$, then the problem (8) is bounded. Adding more optimality cuts will not make the optimal value larger. Thus, RBMP is bounded. □

Given the initial $\mathcal{O}^t$, we can have the solution $\mathbf{x}_0$ and $\mathbf{z}^0 = (z_1^0,\dots,z_S^0)$. Then $c'\mathbf{x}_0 + \sum_{\omega\in\Omega} p_\omega z_\omega^0$ is an upper bound of problem (7).

When $\mathbf{x}_0$ is given, the optimal solution, α_ω^1, to problem (4) can be obtained according to Theorem 1. $z_\omega^{(0)} = \alpha_\omega^1(d_\omega - \mathbf{x}_0\mathbf{1})$ and $(\mathbf{x}_0, \mathbf{z}_\omega^{(0)})$ is a feasible solution to problem (7) because it satisfies all the constraints. Thus, $c'\mathbf{x}_0 + \sum_{\omega\in\Omega} p_\omega \mathbf{z}_\omega^{(0)}$ is a lower bound of problem (7).

If for every scenario, the optimal value of the corresponding problem (4) is larger than or equal to z_ω^0, all contraints are satisfied, we have an optimal solution, (x_0, z_ω^0), to the BMP. Otherwise, add one new constraint, $(\alpha_\omega^1)'(\mathbf{d}_\omega - \mathbf{x}\mathbf{1}) \ge z_\omega$, to RBMP.

The steps of the algorithm are described as below,

Algorithm 1 The benders decomposition algorithm

Step 1. Solve LP (8) with all $\alpha_\omega^0 = \mathbf{0}$ for each scenario. Then, obtain the solution $(\mathbf{x}_0, \mathbf{z}^0)$.

Step 2. Set the upper bound $UB = c'\mathbf{x}_0 + \sum_{\omega\in\Omega} p_\omega z_\omega^0$.

Step 3. For x_0, we can obtain α_ω^1 and $z_\omega^{(0)}$ for each scenario, set the lower bound $LB = c'x_0 + \sum_{\omega\in\Omega} p_\omega z_\omega^{(0)}$

Step 4. For each ω, if $(\alpha_\omega^1)'(\mathbf{d}_\omega - \mathbf{x}_0\mathbf{1}) < z_\omega^0$, add one new constraint, $(\alpha_\omega^1)'(\mathbf{d}_\omega - \mathbf{x}\mathbf{1}) \geq z_\omega$, to RBMP.

Step 5. Solve the updated RBMP, obtain a new solution (x_1, z^1) and update UB.

Step 6. Repeat step 3 until $UB - LB < \epsilon$.(In our case, UB converges.)

Remark 1. *From the Lemma 2, we can set $\alpha_\omega^0 = \mathbf{0}$ initially in Step 1.*

Remark 2. *Notice that only contraints are added in each iteration, thus LB and UB are both monotone. Then we can use $UB - LB < \epsilon$ to terminate the algorithm in Step 6.*

After the algorithm terminates, we obtain the optimal $\mathbf{x}^*$. The demand that can be satisfied by the arrangement is $\mathbf{x}^*\mathbf{1} = d_0 = (d_{1,0}, \ldots, d_{m,0})$. Then we can obtain the value of $y_{i\omega}$ from equation (5).

We show the results of Benders and IP in the section 6.1.

4.3 Obtain The Feasible Seat Planning

The decomposition method only gives a fractional solution and the stochastic model does not provide an appropriate seat planning when the number of people in scenario demands is way smaller than the number of the seats. Thus, we change the linear solution from the decomposition method to obtain a feasible seat planning. Before that, we will discuss the deterministic model that can help achieve the goal.

When $|\Omega| = 1$ in problem (1), the stochastic programming will be

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{m}\sum_{j=1}^{N}(s_i - 1)x_{ij} - \sum_{i=1}^{m} y_i^+ \\
\text{s.t.} \quad & \sum_{j=1}^{N} x_{ij} - y_i^+ + y_{i+1}^+ + y_i^- = d_i, \quad i = 1, \ldots, m-1, \\
& \sum_{j=1}^{N} x_{ij} - y_i^+ + y_i^- = d_i, \quad i = m, \\
& \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\
& y_i^+, y_i^- \in \mathbb{Z}_+, \quad i = 1, \ldots, m \\
& x_{ij} \in \mathbb{Z}_+, \quad i = 1, \ldots, m, j = 1, \ldots, N.
\end{aligned}
\tag{10}
$$

To maximize the objective function, we can take $y_i^+ = 0$. Notice that $y_i^- \geq 0$, thus the constraints $\sum_{j=1}^{N} x_{ij} + y_i^- = d_i, i = 1, \ldots, m$ can be rewritten as $\sum_{j=1}^{N} x_{ij} \leq d_i, i = 1, \ldots, m$, then we have

$$\begin{aligned} \max \quad & \sum_{i=1}^{m}\sum_{j=1}^{N}(s_i - 1)x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{N} x_{ij} + y_i^- \leq d_i, \quad i = 1, \ldots, m, \\ & \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\ & y_i^+, y_i^- \in \mathbb{Z}_+, \quad i = 1, \ldots, m \\ & x_{ij} \in \mathbb{Z}_+, \quad i = 1, \ldots, m, j = 1, \ldots, N. \end{aligned} \tag{11}$$

Problem (11) represents the deterministic model. Demand, $d_i, i = 1, \ldots, m$ is known in advance, our goal is to accommodate as many as people possible in the fixed rows.

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 this MKP. 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 [4] becomes $u_{\text{MKP}} = \sum_{j=1}^{b-1} p_j + \left(L - \sum_{j=1}^{b-1} w_j\right) \frac{p_b}{w_b}$.

Let $\sum_{j=1}^{N} x_{ij}$ indicate the supply for group type i. Denote by $(\sum_{j=1}^{N} x_{1j}, \ldots, \sum_{j=1}^{N} x_{mj})$ the integrated solution to the linear relaxation of MKP. Suppose item b is in type h, then the integrated solution is $(0, \ldots, x, d_{h+1}, \ldots, d_m)$, where $x = (L - \sum_{i=h+1}^{m} d_i s_i)/s_h$. That is, we will place as large groups as possible when the capacity allows.

Suppose we obtain the optimal linear solution x_{ij}^* from the stochastic model, set the supply $\mathbf{s}^0 = \sum_j x_{ij}^*$ as the upper bound of demand in problem (11). We can get a feasible integer solution by solving this problem, denote by $\mathbf{s}^1$ the corresponding supply. As we mentioned above, this solution does not utilize the empty seats when the scenario demands are smaller than supply. Thus, we should set the supply $\mathbf{s}^1$ as the lower bound of demand, then re-solve a seat assignment problem. We substitute the constraint $\sum_{j=1}^{N} x_{ij} \leq d_i, i = 1, \ldots, m$ with the new constraint $\sum_{j=1}^{N} x_{ij} \geq s_i^1, i = 1, \ldots, m$ in problem (11), s_i^1 represents the number of group type i we must allocate seats.

$$\{\max \sum_{j=1}^{N}\sum_{i=1}^{m}(s_i - 1)x_{ij} : \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N; \sum_{j=1}^{N} x_{ij} \geq s_i^1, i = 1, \ldots, m; x_{ij} \in Z^+\} \tag{12}$$

The optimal solution to this problem with the lower bound will give a better seat assignment. The numerical results show that this seat assignment has good performances under any stochastic demands, and also shows good results when dealing with the dynamic demands.

Algorithm 2 Feasible seat planning algorithm

Step 1. Obtain the solution, $\mathbf{x}^*$, from stochatic linear programming by benders decomposition.

Step 2. Aggregate the solution to the supply, $s_i^0 = \sum_j x_{ij}^*$.

Step 3. Obtain the optimal solution, $\mathbf{x}^1$, from problem (11) by setting the supply $\mathbf{s}^0$ as the upper bound.

Step 4. Aggregate the solution to the supply, $s_i^1 = \sum_j x_{ij}^1$.

Step 5. Obtain the optimal solution, $\mathbf{x}^2$, from problem (12) by setting the supply $\mathbf{s}^1$ as the lower bound.

Step 6. Aggregate the solution to the supply, $s_i^2 = \sum_j x_{ij}^2$, which is the feasible seat planning.

Remark 3. *Step 3 can give a feasible integer supply. In Step 5, problem (11) with this supply as the lower bound can always give an integer solution. Thus, we can obtain the near-optimal seat assignment by solving stochastic programming once and deterministic programming twice.*

5 Dynamic Seat Assignment(DSA)

In this section, we will present the methods to assign seats with stochastic information. We can estimate the arrival rate from the historical data, $p_i = \frac{N_i}{N_0}, i = 1, \ldots, J$, where N_0 is the number of total groups, N_i is the number of group type i.

Suppose there are T independent periods; at most, one group will arrive in each period. There are J different group types. Let $\mathbf{y}$ be a discrete random variable indicating the number of people in the group. Let $\mathbf{p}$ be the vector probability, where $p(y = j) = p_j$, $j = 1, \ldots, J$ and $\sum_j p_j = 1$.

The number of scenarios is too large to enumerate all possible cases. Thus, we choose to sample some sequences from the multinomial distribution.

5.1 Assign-to-Seat Rules

Once we obtain a feasible seat planning, we must follow some basic rules to assign seats.

- When the supply of one arriving group is enough, we will accept the group directly.
- When the supply of one arriving group is 0, the demand can be satisfied by only one larger-size supply.
- When one group is accepted to occupy the larger-size seats, the rest empty seat(s) can be reserved for future demand.

We can assign the seats to the corresponding-size group. But when a group comes while the corresponding supply is 0, should we give this group to the larger-size seats? Now we demonstrate the nested policy for this problem.

Suppose we accept a group of i to take over j-size seats. In that case, the expected served people is $i + (j - i - 1)P(D_{j-i-1} \geq x_{j-i-1} + 1)$, where $i < j$, $P(D_i \geq x_i)$ is the probability of that the expected demand of group type i in the following periods is no less than x_i, the remaining supply of group type i.

When a group of i occupies j-size seats, $(j - i - 1)$ seats can be provided for one group of $j - i - 1$ with one seat of social distancing. Thus, the term, $P(D_{j-i-1} \geq x_{j-i-1} + 1)$, indicates the probability that the demand of group type $(j - i - 1)$ in the future is no less than its current remaining supply plus 1. If $j - i - 1 = 0$, then this term equals 0.

Similarly, when the expected demand of a group of j in the future is no less than its remaining supply currently, we would reject a group of i, the expected served people is $jP(D_j \geq x_j)$.

Let $d(i, j)$ be the difference of expected served people between acceptance and rejection on group i occupying j-size seats. Then $d(i, j) = i + (j - i - 1)P(D_{j-i-1} \geq x_{j-i-1} + 1) - jP(D_j \geq x_j), j > i$.

One intuitive decision is to choose the largest difference. We can obtain $d(i, j) = jP(D_j \leq x_j - 1) - (j - i - 1)P(D_{j-i-1} \leq x_{j-i-1}) - 1$ after reformulating. Let $F_j(x; T)$ be the cumulative distribution function of the number of arrival groups D_j in T periods. Then $F_j(x; T_r) = P(D_j \leq x)$, and D_j follows a binomial distribution $B(T_r, p_j)$, where T_r is the numebr of remaining periods.

Thus, $d(i, j) = jF_j(x_j - 1; T) - (j - i - 1)F_{j-i-1}(x_{j-i-1}; T) - 1$. For all $j > i$, find the largest $d(i, j)$, denoted as $d(i, j^*)$. If $d(i, j^*) > 0$, we will place the group i in j^*-size seats. Otherwise, reject the group.

The algorithm is shown below:

Algorithm 3 Dynamic seat assignment(DSA) method

Step 1. Obtain the supply, $\mathbf{X}^0 = [x_1, \ldots, x_m]$, from the feasible seat planning algorithm.

Step 2. For the arrival group type i at period T', if $x_i > 0$, accept it. Let $x_i = x_i - 1$. Go to step 4.

Step 3. If $x_i = 0$, find $d(i, j^*)$. If $d(i, j^*) > 0$, accpect group type i. Set $x_{j^*} = x_{j^*} - 1$. Let $x_{j-i-1} = x_{j-i-1} + 1$ when $j - i - 1 > 0$. If $d(i, j^*) \leq 0$, reject group type i.

Step 4. If $T' \leq T$, move to next period, set $T' = T' + 1$, go to step 2. Otherwise, terminate this algorithm.

We show the results of Benders and IP under nested policy in section 6.1.

5.2 Seat assignment during booking period

The dynamic seat assignment method can give the seat planning before the group arrivals. The specific procedures are demonstrated in the above sections. The first step is to obtain the feasible seat planning from Algorithm 2. Then accept or reject group arrivals according to the nested policy in section 5.1.

5.3 Seat assignment after booking period

We have developed a dynamic programming(DP)-based method by relaxing all rows to one row with the same capacity. Assuming that there always exists a seat assignment under the total capacity of seats, we can use DP to make decisions in each period.

$$V_t(S) = \sum_{i \in m} p_i \max\{[V_{t-1}(S - s_i - 1) + s_i], V_{t-1}(S)\}$$

After obtaining the pre-accepted sequence, we still need to check whether this sequence is feasible for the seat assignment. In most cases, the pre-accepted sequence is feasible, which is why we use relaxation. However, in cases where the sequence is not feasible, we must delete groups one by one from the last arrival of the sequence until it becomes feasible. In practice, we begin to reject the arrival until we cannot find a feasible seat assignment.

5.4 Benchmark

5.4.1 FCFS

For seat assignment after booking period, the intuitive but trivial method will be on a first-come-first-served basis. Relax all rows to one row with the total number of seats. For the arrival sequence, find the target arrival when the number of seats taken by the preceding arrivals does not exceed the capacity. Then we obtain a new sub-sequence, including the arrivals from the first to the target and a possible arrival. For the convenience of calculation, we check the feasibility of constructing a seat assignment from the end of the sub-sequence. When it is not feasible for the seat assignment, we should delete the group one by one from this sub-sequence until a feasible seat assignment is found. In reality, we need to check the feasibility one group by one.

5.4.2 Largest Patterns Planning

For seat assignment during booking period, we don't have a good benchmark.

For each row, we choose the patterns from I_1. Accept the group such that the largest pattern can be maintained. When the arrival cannot be assigned in the planning patterns from I_1, we can change the largest pattern to a second largest pattern according to the coming arrival.

Algorithm 4 Method by using the largest patterns

Step 1. Generate the largest pattern by the greedy way for each row.

Step 2. Denote the minimal and maximal size of group in the pattern of row i as $\min_i$ and $\max_i$.

Step 3. For the arrival with the size of a in period t, if there exists i such that $\min_i + \max_i >= a$ and $a > \min_i$, accept this arrival at row i, go to step 5; otherwise, go to step 4.

Step 4. Find a maximal group of seats to accept this arrival, otherwise, reject this arrival.

Step 5. Move to the next period. Repeat step 3.

Step 2: the pattern will have the same loss by these procedures. ($\min_i$ can be 0.) (Can we use the partial information?)

This method can be used without stochastic information. The performance will improve when the total demand can construct the largest patterns for all rows.

6 Results

6.1 Benders Decomposition versus IP

The running times of solving IP directly and using Benders decomposition are shown in Table 1.

Table 1: Running time of Decomposion and IP

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

The parameters in the columns of the table are the number of scenarios, the range of demands, running time of integer programming, 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.

6.2 Near-optimal Solution versus IP Solution under Nested Policy

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

$$p\left(N_1, \ldots, N_J \mid \mathbf{p}\right) = \frac{T!}{N_1!, \ldots, N_J!} \prod_{j=1}^{J} p_j^{N_j}, T = \sum_{j=1}^{J} N_j.$$

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

Use $D(T, J)$ to denote the number of scenarios, which equals the number of different solutions to $x_1 + \ldots + x_J = T, \mathbf{x} \geq 0$. Then, we know the recurrence relation $D(T, J) = \sum_{i=0}^{T} D(i, J-1)$ and boundary condition, $D(i, 1) = 1$. So we have $D(T, 2) = T + 1$, $D(T, 3) = \frac{(T+2)(T+1)}{2}$, $D(T, J) = O(T^{J-1})$.

Then, we will show the near-optimal seat assignment has a close performance with IP when considering nested policy.

Table 2: Results of Decomposion and IP under nested policy

# samples	T	probabilities	# rows	people served by decomposition	people served by IP
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.

We can find that the people served by Benders decomposition and IP under nested policy are close. But obtaining the near-optimal seat assignment will be faster.

6.3 Periods

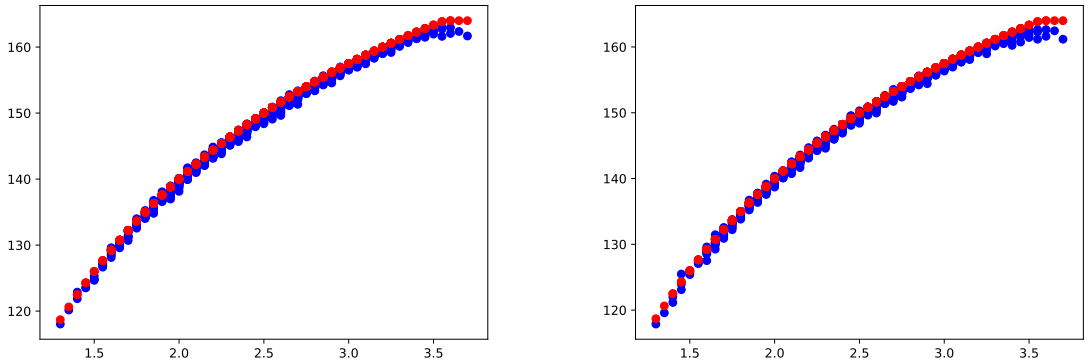
Table 3: Results of Decomposion and IP under nested policy

T	probabilities	# of rows	DSA(%)	Row by Row(%)	DP(%)	FCFS-based(%)
60	[0.45 0.05 0.05 0.45]	10	98.24	97.39	99.49	99.08
60	[0.35 0.05 0.35 0.25]	10	98.12	97.93	99.62	99.25
60	[0.25 0.05 0.65 0.05]	10	98.55	97.91	99.39	99.28
60	[0.35 0.15 0.15 0.35]	10	97.28	98.14	99.29	99.12
60	[0.25 0.15 0.45 0.15]	10	96.30	97.59	99.46	99.09

6.4 Seat Layout

The number of seats are $[17, 18, 19, 20, 21, 21, 22, 23, 24, 25]$. The maximal number of people that can be served is 164.

Random seat layout: $[26, 21, 24, 20, 20, 17, 23, 19, 21, 19]$. The maximal number of people that can be served is 164.



(a) Average of 50 instances for step-size seat layout

(b) Average of 50 instances for random seat layout

Figure 2: The number of people served versus c

6.5 Result of Different Demands

In this subsection, we discuss the effect of the number of periods on the results of our experiments. Specifically, we consider two situations: c = 2.5 and c = 1.9.

When c = 2.5, we set the parameters as follows: T = 10-100, step size = 1, expected number of periods = 60, expected number of demand (people) = 150, number of rows = 10, and number of seats per row = 21.

When c = 1.9, we set the parameters as follows: T = 30-120, step size = 1, expected number of periods = 72, expected number of demand (people) = 137, number of rows = 10, and number of seats per row = 21.

For each c, we give several probabilities in the table. The gap point represents the first period where the number of people without social distancing is larger than that with social distancing and the gap percentage is the corresponding percentage of total seats.

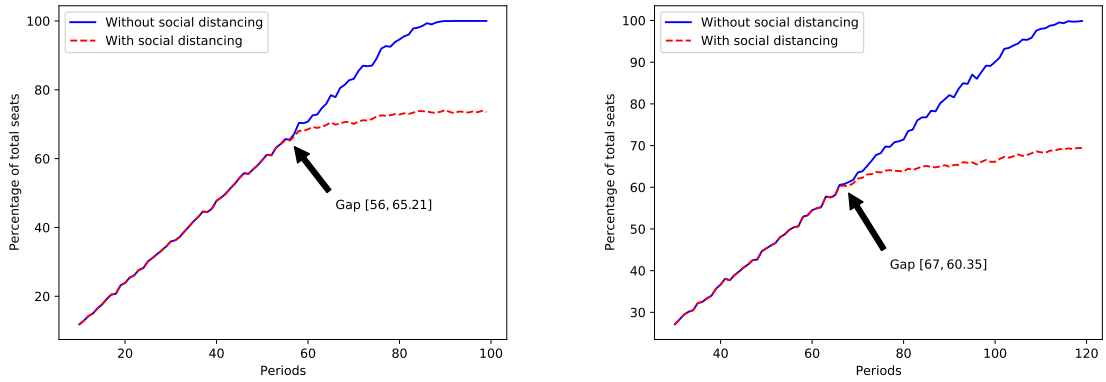
We provide the results of our experiments for people accepted when applying social distance and not applying social distance over different periods. The different probabilities with the same c share the similar pattern of the figure, so we only provide one case to show the detailed figure.

There are three stages, the first stage is when the capacity is sufficient. The measure of social distancing will not cause any effect. The gap is becoming larger as T increases at the second stage. At the third stage, as T continues to increase, the gap will converge when the capacity is limited.

We can estimate the attendance rate from $\frac{c}{c+1} *$ the number of seats. The number of periods will be $E(D)/c$. Thus, the government can make the policy on how much attendance rate can be established.

Table 4: Gap points of different probabilities

c	probabilities	gap point	gap percentage
2.5	[0.25,0.25,0.25,0.25]	56	65.21
2.5	[0.1,0.4,0.4,0.1]	55	65.59
2.5	[0.1, 0.5, 0.2, 0.2]	55	65.45
2.5	[0.2, 0.3, 0.3, 0.2]	54	64.56
2.5	[0.3, 0.2, 0.2, 0.3]	55	65.51
2.5	[0.2, 0.4, 0.1, 0.3]	55	65.41
1.9	[0.4, 0.4, 0.1, 0.1]	67	60.35
1.9	[0.5, 0.2, 0.2, 0.1]	67	58.9
1.9	[0.3, 0.5, 0.2, 0]	68	61.7
1.9	[0.6, 0.1, 0.1, 0.2]	66	58.31



(a) When $c = 2.5$

(b) When $c = 1.9$

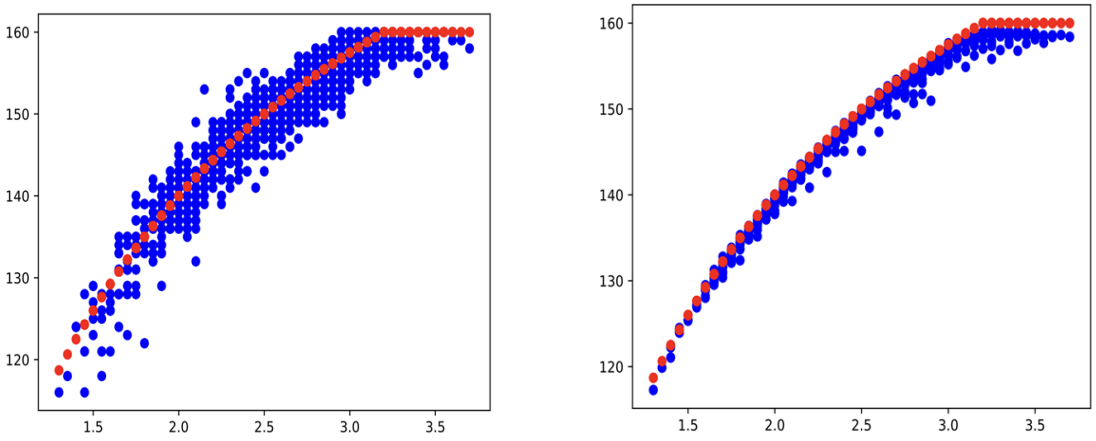
Figure 3: The number of people served versus periods

6.6 Results of Different Arriving People Per Period

In this subsection, we discuss the effect of different probabilities on our experimental results. Assuming that groups of up to four people can sit together, we calculate the expected number of demands as $E(D) = (p_1 * 1 + p_2 * 2 + p_3 * 3 + p_4 * 4)T$, where p_1, p_2, p_3, and p_4 represent the probabilities of groups with one, two, three, and four people, respectively. We also define $c = p_1 * 1 + p_2 * 2 + p_3 * 3 + p_4 * 4$ and choose $T(E(D)/c)$ such that the supply is near the demand. We then compare the number of people served under different values of c.

We sample p_1, p_2, and p_3 from 0.05 to 0.95 with an increment of 0.05. We call each realization (p_1, p_2, p_3, p_4) the probability combination when $p_1 + p_2 + p_3 + p_4 = 1$. The number of all sampled probability combinations is n_p. The number of rows is set at 10, and the number of seats per row is 21 (including one dummy seat). We simulate 200 periods, and the number of people served with respect to the value of c is shown in the figure below. For each probability combination, the blue point represents the average number of people served from 50 instances and the red point represents the expected number of people served.

Suppose we accept D_a people with T arrivals, where $D_a = c * T$, and the sum of D_a and T is equal to the total number of seats. In this case, the estimation of the occupancy rate is $\frac{c}{c+1}$. The number of



(a) One instance for each probability combination

(b) Average of 50 instances for each probability combination

Figure 4: The number of people served versus c

people served is near $\frac{c}{c+1} * 210$ on average (as shown by the red points in the figure).

If the largest pattern is assigned for each row, the occupancy rate is $\frac{16}{21}$. The maximum number of people that can be served is $210 * \frac{16}{21} = 160$, which is the upper bound of the number of people served.

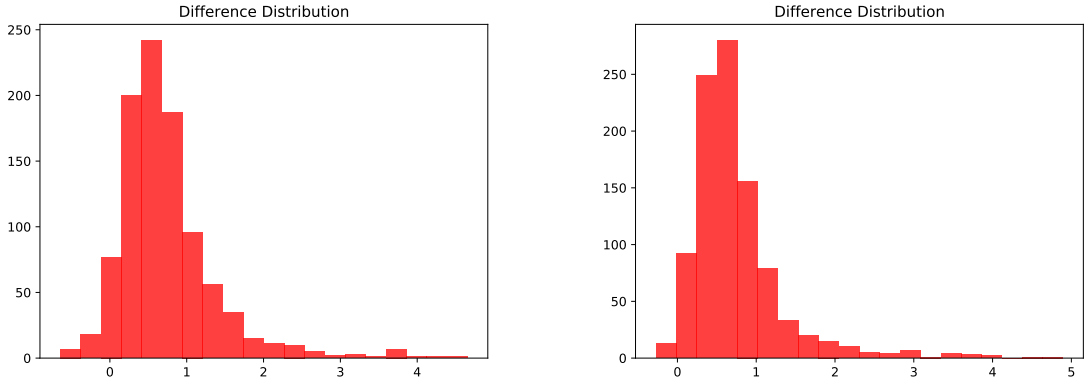
We also observe that some blue points are far from the red points in the figure. For example, when the probability is $[0.05, 0.05, 0.85, 0.05](c = 2.9)$, the demands can be $[4, 1, 45, 2]$ or $[2, 2, 47, 1]$. It is not possible to construct all full patterns for every row with these demands, violating our assumption and resulting in a large gap between the blue and red points in this case.

We can give the absolute difference between the blue point and red point for each probability combination as below.

Table 5: Difference Distribution

# of instances	abs_diff ≥ 1	abs_diff ≥ 2	abs_diff ≥ 3	abs_diff ≥ 4
20	32.92 %	5.13 %	1.74%	0.51 %
50	22.46 %	4.31 %	1.54 %	0.31 %
100	20.00 %	4.21 %	1.54 %	0.31 %

The results show that we can estimate attendance rate based on c for most probability combinations.



(a) Average of 50 instances for each probability combination (b) Average of 100 instances for each probability combination

Figure 5: The difference distribution

7 Extension

7.1 Online booking

The groups can select the seats arbitrarily but with some constraints.

When the customers book the tickets, they will be asked how many seats they are going to book at first. Then we give the possible row numbers for their selection. Finally we give the seat number for their choice.

The second step is based on the other choices of reserved groups, we need to check which rows in the seat assignment include the corresponding-size seats.

In third step, we need to check whether the chosen number destroys the pattern of the row. Use subset sum problem to check every position in the row.

7.2 How to give a balanced seat assignment

Notice we only give the solution of how to assign seats for each row, but the order is not fixed.

In order to obtain a balanced seat assignment, we use a greedy way to place the seats.

Sort each row by the number of people. Then place the smallest one in row 1, place the largest one in row 2, the second smallest one in row 3 and so on.

For each row, sort the groups in an ascending/descending order. In a similar way.

7.3 Property

Although the solver can solve this problem easily, the analyses on the property of the solution to this problem can help us generate a method for the dynamic situation.

At first, we consider the types of pattern, which refers to the seat assignment for each row. For each pattern k, we use α_k, β_k to indicate the number of groups and the left seat, respectively. Denote

by $l(k) = \alpha_k + \beta_k - 1$ the loss for pattern k. The loss represents the number of people lost compared to the situation without social distancing.

Let I_1 be the set of patterns with the minimal loss. Then we call the patterns from I_1 are largest. Similarly, the patterns from I_2 are the second largest, so forth and so on. The patterns with zero left seat are called full patterns. We use the descending form $(t_1, t_2, \ldots, t_k)$ to represent a pattern, where t_i is the new group size.

Example 1. *The length of one row is $S = 21$ and the new size of groups be $L = [2, 3, 4, 5]$. Then these patterns, $(5, 5, 5, 5, 1), (5, 4, 4, 4, 4), (5, 5, 5, 3, 3)$, belong to I_1. The demand is $[10, 12, 9, 8]_d$.*

To represent a pattern with a fixed length of form, we can use a $(m+1)-$dimensional vector with m group types. The aggregated form can be expressed as $[n_0, n_1, \ldots, n_m]$, where n_i is the number of i-th group type, $i = 1, \ldots, m$. n_0 is the number of left seat, its value can only be $0, 1$ because two or more left seats will be assigned to groups. Thus the pattern, $[1, 0, 0, 0, 4]$, is not full because there is one left seat.

Suppose u is the size of the largest group allowed, all possible seats can be assigned are the consecutive integers from 2 to u, i.e., $[2, 3, \ldots, u]$. Then we can use the following greedy way to generate the largest pattern. Select the maximal group size,u, as many as possible and the left space is assigned to the group with the corresponding size. Let $S = u \cdot q + r$, where q is the number of times u selected. When $r > 0$, there are $d[0][u - r][q + 1]$ largest patterns with the same loss of q. When $r = 0$, there is only one possible largest pattern.

Use dynamic programming to solve. $d[k][i][j]$ indicates the number of assignment of using i capacity to allocate j units, k is the number of capacity allocated on the last unit. In our case, $u - r$ is the capacity need to be allocated, $q + 1$ is the number of units which corresponds to the groups. Notice that we only consider the number of combinations, so we fix the allocation in ascending order, which means the allocation in current unit should be no less than the last unit.

The number of largest patterns equals the number of different schemes that allocate $u - r$ on $q + 1$ units,i.e., $d[0][u - r][q + 1]$.

The recurrence relation is $d[k][i][j] = \sum_{t=k}^{i-k} d[t][i - k][j - 1]$. When $i < k$, $d[k][i][j] = 0$; when $i \geq k$, $d[k][i][1] = 1$.

Lemma 3. *The seat assignment made up of the largest patterns is optimal.*

This lemma holds because we cannot find a better solution occupying more seats.

When the demand is so large that the largest patterns can be generated in all rows, an optimal seat assignment can be obtained.

Proposition 1. *Let $k^* = \arg\max_{k \in I_1} \min_i \{\lfloor \frac{d_i}{b_i^k} \rfloor\}$. When $N \leq \max_{k \in I_1} \min_i \{\lfloor \frac{d_i}{b_i^k} \rfloor\}$, the optimal seat assignment can be constructed by repeating pattern k^* N times. N is the number of rows, d_i is the number of i-th group type, $i = 1, 2, \ldots, m$, b_i^k is the number of group type i placed in pattern k.*

Use the above example 1 to explain. Take $(5, 5, 5, 5), (5, 4, 4, 4, 4), (5, 5, 4, 4, 3)$ as the alternative patterns, denoted by pattern $1, 2, 3$ respectively. When $k = 1, 2, 3$, $\min_i \{\lfloor \frac{d_i}{b_i^k} \rfloor\}$ will be $2, 3, 5$, thus

$k^* = 3$. So when $N \leq 5$, we can select the pattern $(5, 5, 4, 4, 3)$ five times to construct the optimal seat assignment.

Proposition 2. *We can construct a seat assignment in the following way. Every time we can select one pattern from I_1, then minus the corresponding number of group types from demand and update demand. Repeat this procedure until we cannot generate a largest pattern. If the number of generated patterns is no less than the number of rows, this assignment is optimal.*

8 Conclusion

We mainly focus on how to provide a way to ...

In our study, we stressed....

Our main results show that ...

Moreover, our analysis provides managerial guidance on how to place the seats under the background of 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] Michael S Casey and Suvrajeet Sen. The scenario generation algorithm for multistage stochastic linear programming. *Mathematics of Operations Research*, 30(3):615–631, 2005.

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

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

Proof

(Theorem 1). □

(Lemma 1). □

(Lemma 2). □

(Theorem 2). □