Seat assignment with the social distancing

Dis· count

March 24, 2023

Abstract

Social distancing, a non-pharmaceutical way to contain the spread of an infectious disease, has been broadly recognized and practiced. In this paper, we consider the seat assignment problem with social distancing when encountering deterministic, stochastic and dynamic demands.

In a pandemic, the government may issue a minimum physical distance between people, which must be respected in the seating assignment. The problem is further complicated by the existence of groups of guests who will be seated together. To achieve such a goal, we provide an optimal seat assignment 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, the group arrives dynamically, we give the method both with and without stochastic information.

With these results, we provide a guideline for policies related to seat utilization rate.

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

1 Introduction

Social distancing, a non-pharmaceutical way to contain the spread of Social distancing, a physical way to control the spread of infectious disease, has been broadly recognized and practiced. As a result, extensive research has emerged on social distancing concerning its effectiveness and impact. What lags is operational guidance for implementation, an issue particularly critical to social distance measures of which the implementation involves operations details. One typical example is how to make social distancing ensured seating plans.

We will start by considering the seating plan with a given set of seats. In a pandemic, the government may issue a minimum physical distance between people, which must be implemented in the seating plan. The problem is further complicated by the existence of groups of guests who can sit together. To achieve such a goal, we develop a mechanism for seat planning, which includes a model to characterize the riskiness of a seating plan, and a solution approach to make the tradeoff between seat utilization rate and the associated risk of infection.

In this paper, we are interested in finding a way to implement a seating plan with social distancing constraints instead of solving the IP model directly. After knowing the group portfolio structure, we can obtain the minimum number of seat rows inspired by the cutting stock problem. And we can formulate the corresponding model with a given number of rows to maximize the capacity.

Our main contributions are summarized as follows.

First, this paper is the first attempt to consider how to arrange the seat assignment with social distancing under dynamic arrivals. Most literature on social distancing in seat assignments highlights the importance of social distance in controlling the virus's spread and focuses too much on the model. There is not much work on the operational significance behind the social distance [1] [5]. Recently, some scholars studied the effects of social distance on health and economics, mainly in aircraft [9] [6]. Especially, our study provides another perspective to help the government adopt a mechanism for setting seat assignments to protect people in the post-pandemic era.

Second, we establish the deterministic model to analyze the effects of social distancing. When the demand is known, we can solve the IP model directly because of the medium size of this problem. Then we consider the stochastic demand situation when the demands of different group types are random. With the aid of two-stage stochastic programming, we use Benders decomposition methods to obtain the optimal linear solution. Then we develop an integral solution from linear solutions according to the traits of our problem.

Third, to solve the dynamic demand situation, we apply the result of a scenario-based problem. We generate scenarios from multinomial distribution and use nested policy to decide whether to accept or reject each group arrival.

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

[7] gives the well-known initial compact formulation.

[10] carries out the computational experiments with a lot of randomly generated instances of the one-dimensional cutting stock problem.

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. [4] [2] [8]

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.

Every time we can regenerate the scenario based on the realized demands. (Use the conditional distribution or the truncated distribution)

Suppose that the groups arrive from small to large according to their size. Once a larger group comes, the smaller one will never appear again.

When a new group arrives (suppose we have accepted n groups with the same size), we accept or reject it according to the supply (when $n + 1 <$ supply, we accept it), then update the scenario set according to the truncated distribution. We can obtain a new supply with the new probability and scenario set.

With the conclusion of section , we know how to reject a request. Once we reject one group, we will reject all groups of the same size.

3 Problem Formulation

In this section, we give the description of the problem, then present the stochastic formalution.

3.1 Problem Description

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, $j = 1, \ldots, N$.

According to the epidemic prevention requirements, the customers from the same group can sit together, while different groups should sit with social distancing. Suppose that each group has to leave one seat to maintain social distancing from the adjacent groups and different rows have no effect on each other, i.e., a person from one group can sit directly behind a person from another group.

To simplify the social distancing in the seat assignment, we add one to the original size of each group as the new size of the group and one dummy seat to each row. Take $s_i = i + 1$, $L_j = S_j + 1$, s_i is the new size of group type i and L_j is the length of row j.

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

Figure 1: Problem Conversion

On the left side, the blue squares stand for the empty seats as the social distancing. The orange squares represent the seats sat by the groups. On the right side, one dummy seat is added at the end of the row. The orange squares surrounded by the red line are the seats taken by groups. Here, this row is placed two group of 1, one group of 2 and one group of 3.

In this way, the social distancing can be integrated by solving this new seat assignment problem.

4 Dynamic Demand

We can use a vector to record the state of each row, each time we can decide

Dynamic programming:

$DP(S_1, S_2, \ldots, S_N) =$

The dimension is too large,

We also study the dynamic seating plan problem, which is more suitable for commercial use. In this situation, customers come dynamically and the seating plan needs to be made without knowing the number and composition of future customers.

It becomes a sequential stochastic optimization problem where conventional methods fall into the curse of dimensionality due to many seating plan combinations. To avoid this complexity, we develop

an approach that aims directly at the final seating plans. As stated above, we can obtain such a seating plan from stochastic programming. However, it only gives the initial seat assignment without handling the dynamic situation. We discuss two dynamic situations in the following section to further know how to make decisions when the group arrives dynamically.

4.1 Dynamic Demand Situations

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, ticket reservation without seat selection and seat assignment before arrival.

The seat assignment will not be made immediately for reservations without seat selection. The decision-maker must only reject or accept each request during the making-reservation stage. After the reservation deadline, the seller will inform the seat layout information to the customers before admission. For example, singing concert venues have more seats and higher 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.

For seat assignment before arrival, the specific procedure will be changed under the requirement of social distancing. The seat assignment will be arranged before the groups book the tickets, the groups only need to choose the corresponding-size seats when booking tickets. For example, movie theaters or small concert have relatively few seats, the attendance rate is usually low enough to allow free selection of seats directly online. The early seat planning can satisfy the requirement of social distancing and also 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.

4.2 Generate scenarios by discrete periods

In the dynamic situation, we can use stochastic information to generate the sequences of group arrivals. The sequences can be integrated to obtain the scenario of demands.

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 $\sum_j p_j = 1$. Then a sequence can be expressed as $\{y_1, y_2, \ldots, y_T\}$. (It can be modeled as a multinomial distribution, $p(\mathbf{Y} \mid \mathbf{p}) = \prod_{j=1}^{J} p_j N_j$).

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 set of counts N_j (scenarios) follows 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 $D(T,2) = T+1$, $D(T,3) = \frac{(T+2)(T+1)}{2}$, $D(T,J) = O(T^{J-1})$.

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

Remark: this approach still works under the assumption that time is continuous.

4.3 Nested Policy with Given Supply

Once we obtain a solution from stochastic programming, 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 1 Nested policy under given supply

Step 1. Obtain a supply, $\mathbf{X}^0 = [x_1, \ldots, x_J]$, from the stochastic programming.

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.

4.4 Dynamic Situation with Stochastic Information

In this section, we will present methods to assign seats with stochastic information.

4.4.1 Ticket Reservation without Seat Selection

We develop a DP-based method by relaxing all rows to one row with the same capacity. Suppose there always exists one seat assignment under the total capacity. Then we can use DP to make the decision in each period.

$$V(S, T) = \sum_{i \in N} p_i \max\{[V((S - s_i - 1), T - 1) + s_i], V(S, T - 1)\}$$

After obtaining the pre-accepted sequence, we still need to check whether this sequence is feasible for the seat assignment. In most cases, it is feasible. That is the reason why we use relaxation. When it is not feasible, we should delete the group one by one from the last arrival of the sequence until it is feasible. In practice, we reject the request in the pre-accepted sequence until we cannot find a feasible seat assignment.

4.4.2 Seat Assignment before Arrivals

The stochastic seat assignment(SSA) method can give a seat assignment before the group arrivals. The specific procedures are demonstrated in the above sections. The first step is to obtain the feasible seat assignment from Algorithm 4. Then accept or reject group arrivals according to the nested policy in section 4.3.

4.5 Benchmark

The benchmarks for the above two situations are described below.

4.5.1 FCFS

For ticket reservation without seat selection, 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.

4.5.2 Largest Patterns Planning

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 2 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.

4.6 Scenario-based Stochastic Programming

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 $\omega \in \Omega, \Omega$ corresponds 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:

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

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,

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

The objective is to obtain the maximal number of people placed according to the demand scenarios. It will not provide an appropriate seat assignment when the number of people associated with scenario demands is way less than the number of available seats because there are multiple optimal solutions and

the solution given by solver probably does not utilize all the empty seats. For example, there are 10 rows of seating arrangements, with 20 seats in each row. Suppose the size of group is up to 4, when the scenario demands are $[2,3,2,3],[2,3,3,3],[3,3,2,3]$, any solutions can provide a supply more than $[3,3,3,3]$ will be the optimal, but the corresponding seat assignment obviously leaves many seats vacant. We will address this problem in Section 5.3.

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 $\mathbf{c}'\mathbf{x} = \sum_{j=1}^{N} \sum_{i=1}^{m} i x_{ij}$, $\mathbf{f}'\mathbf{y}_\omega = -\sum_{i=1}^{m} y_{i\omega}^+$. 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 5, we obtain an optimal linear solution by decomposition approach and generate a near-optimal seat assignment.

5 Solve The Scenario-based Two-stage Problem

At first, we generate a closed-form solution to the second-stage problem in section 5.1. 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.

5.1 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.

5.2 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:

$$
\begin{aligned}
\max \quad & c'x + \sum_{\omega \in \Omega} p_\omega z_\omega \\
\text{s.t.} \quad & \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\
& (\alpha^k)'(\mathbf{d}_\omega - \mathbf{x1}) \geq z_\omega, \alpha^k \in \mathcal{O}, \forall \omega \\
& \mathbf{x} \geq 0
\end{aligned} \tag{7}
$$

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.*

$$
\begin{aligned}
\max \quad & c'x + \sum_{\omega \in \Omega} p_\omega z_\omega \\
s.t. \quad & \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\
& (\alpha^\omega)'\mathbf{d}_\omega \geq (\alpha^\omega)'\mathbf{x1} + z_\omega, \forall \omega \\
& \mathbf{x} \geq 0
\end{aligned} \tag{8}
$$

Problem (8) reaches its maximum when $(\alpha^\omega)'\mathbf{d}_\omega = (\alpha^\omega)'\mathbf{x1} + z_\omega, \forall \omega$. Substitute z_ω with these equations, we have

$$
\begin{aligned}
\max \quad & c'x - \sum_{\omega} p_\omega (\alpha^\omega)'\mathbf{x1} + \sum_{\omega} p_\omega (\alpha^\omega)'\mathbf{d}_\omega \\
s.t. \quad & \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\
& \mathbf{x} \geq 0
\end{aligned} \tag{9}
$$

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 $\mathbf{z}^0 = (z_1^0, \ldots, z_S^0)$. 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, $(\alpha_\omega^1)'(\mathbf{d}_\omega - \mathbf{x1}) \geq z_\omega$, to RBMP.

The steps of the algorithm are described as below,

Algorithm 3 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.

5.3 Obtain The Feasible Seat Assignment

The decomposition method only gives a fractional solution and the stochastic model does not provide an appropriate seat assignment when the number of people in scenario demands is smaller than the number of the seats. Thus, we change the linear solution from the decomposition method to obtain a feasible seat assignment. 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 [3] 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 4 Feasible seat assignment 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 assignment.

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.*

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

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 Different Probabilities

We discuss the effect of different probabilities in this sub-section. Suppose that the group of up to 4 can sit together. Then we have $E(D) = (p_1*1+p_2*2+p_3*3+p_4*4)T$. Let $c = p_1*1+p_2*2+p_3*3+p_4*4$. Choose $T(E(D)/c)$ such that the supply is near the demand. Then we compare the number of people served under different values of c.

Sample p_1, p_2, p_3 from 0.05 to 0.95 with increment of 0.5, 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 10, and the number of seats for each row is 21 (including one dummy seat). The number of periods is 200. The number of people served with respect to the value of c is shown below:

Each blue point stands for the average number of people of 50 instances. Each red point stands for the expected number of people.

Suppose we accept D_a people with T arrivals and the sum of D_a and T is equal to the total number of seats.($D_a = c*T$) Then the estimation of occupancy rate is $\frac{c*T}{(c+1)*T} = \frac{c}{c+1}$ when there are full patterns for all rows.

The number of people served is near $\frac{c}{c+1} * 210$ on average.(Red points showed in the figure.)

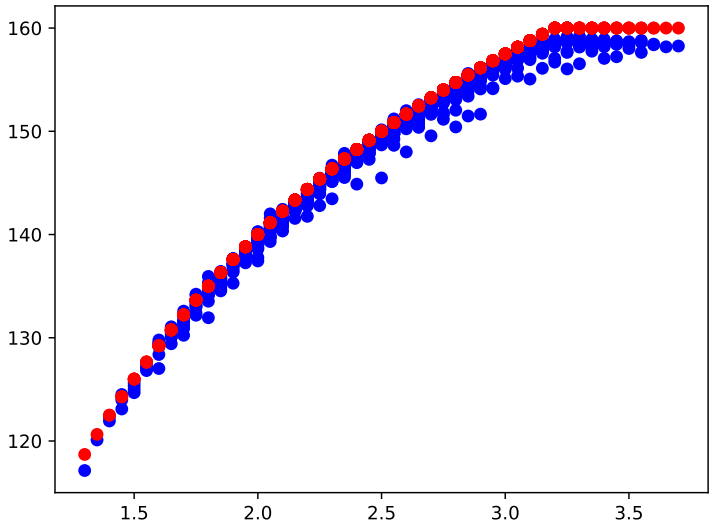


Figure 2: The number of people served versus c

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

From the above figure, we can also find that some blue points are far from the red points. 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 $[2, 2, 47, 1]$. We cannot construct all full patterns for every row with these demands. It violates the assumption, thus in this case there is a large gap between blue point and red point.

Multiple linear regression:

Because we have $p_1 + p_2 + p_3 + p_4 = 1$, we just select three of four probabilities as the dependent variables. Select y, the number of people served, as the independent variable.

Data points: $x = (p_{1i}, p_{2i}, p_{3i}), i \in I$. $|I| = n_p \times n_s$, n_s is the number of sequences generated for each probability combination.

For each probability combination, we select 1, 20, 50 sequences to carry on the multiple linear regression.$(y = c_1p_1 + c_2p_2 + c_3p_3 + \epsilon)$

Table 3: Multiple linear regression results

# of sequences	c1	c2	c3	Intercept	R-squared
1	-50.7086	-29.1060	-12.9784	171.3059	0.867
20	-51.2565	-29.4447	-13.1312	171.6110	0.875
50	-51.3270	-29.3098	-13.2226	171.5823	0.874

Let $y = -51.327p_1 - 29.3098p_2 - 13.2226p_3 + 171.5823$,

When $p_1 = 1$, we know the number of people served is $1/2 * 210 = 105$, then we fix the intercept at 105.

$y = 15.2553p_1 + 37.2725p_2 + 53.3597p_3 + 66.5823p_4 + 105$

Fix the intercept at 160, we can get $y = -39.7447p_1 - 17.7275p_2 - 1.6403p_3 + 11.5823p_4 + 160$.

For $[0.05, 0.05, 0.85, 0.05]$, $y = 156.31125$. True value: 151.5.

Let $y = \frac{c}{c+1} * 210 + c_1 p_1 + c_2 p_2 + c_3 p_3 + \epsilon$

Table 4: Regression results

# of sequences	c1	c2	c3	Intercept
1	1.3163	3.5121	0.4751	-2.2182
20	2.7481	3.4819	2.0804	-2.9253

But the R-squared is 0.041, indicating that independent variables can not explain much in the variation of dependent variable.

If we confine the region where c is lower than 3.2, the R-squared is 0.008.

Let $E(D) = 150$.

Two experiments: When $E(D) = 2.5T$, which means on average 2.5 people arrive for each group.

$T = 75$, the number of rows is 9, the number of seats each row is 25.

Probabilities: $p_1 = p_3 + 2p_4$. p_3 is from 0.05 to 0.45 with step size of 0.1. p_4 is from 0.05 to 0.3 with step size of 0.05.

When $E(D) = 2T$, which means on average 2 people arrive for each group.

$T = 60$, the number of rows is 10, the number of seats each row is 21.

Probabilities: $2p_2 + 4p_3 + 6p_4 = 3$. p_2 is from 0.05 to 0.95 with step size of 0.1. p_3 is from 0.05 to 0.75 with step size of 0.1.

6.4 Different Periods

We discuss the effect of the number of periods this sub-section.

Parameters: T = 10-100, step size =1.

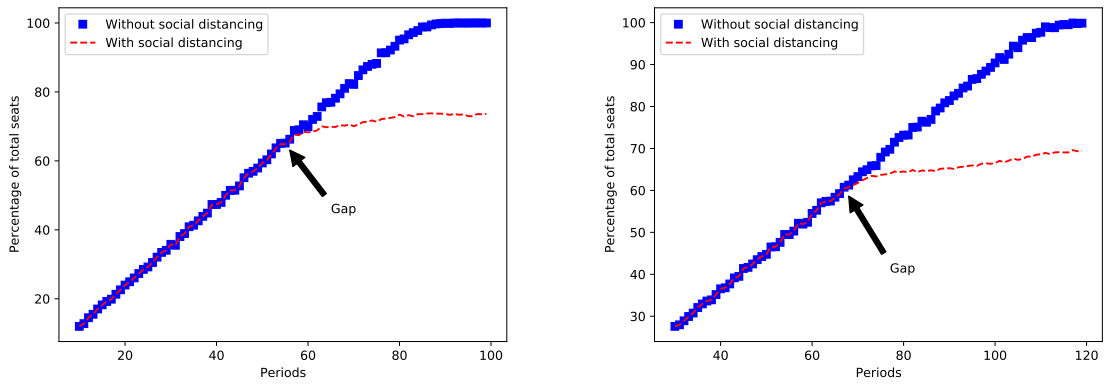
The expected number of period: 60 The expected number of demand(people): 150 Number of rows: 10 Number of seats each row: 21 Probabilities: $[0.25, 0.25, 0.25, 0.25]$.

Parameters: T = 30-120, step size =1.

The expected number of period: 72 The expected number of demand(people): 137 Number of rows: 10 Number of seats each row: 21 Probabilities: $[0.4, 0.4, 0.1, 0.1]$

We can find that the difference between the number of accepted people and the number of total people will increase with the period.

Probability 1: $[0.25, 0.25, 0.25, 0.25]$, Probability 2: $[0.4, 0.4, 0.1, 0.1]$



(a) Probability 1

(b) Probability 2

Figure 3: The number of people served versus periods

There are three stages,

First stage: the capacity is sufficient. The social distancing will not cause any effect.

Second stage: the gap is becoming larger as T increases.

Third stage: As T continues to increase, the gap will converge when the capacity is limited.

The gap point represents the first point where the people without social distancing is larger than that with social distancing.

The government can make the policy about how much attendance rate.

6.5 Group of Different Sizes

7 Extension

7.1 Obtain Minimum Number of Rows to Cover Demand

To find the minimum number of rows to satisfy the demand, we can formulate this problem as a cutting stock problem form and use the column generation method to obtain an approximate solution.

Similar to the concept of pattern in the CSP, let the k-th placing pattern of a line of seats with length S into some of the m group types be denoted as a vector $(t_1^k, t_2^k, \ldots, t_m^k)$. Here, t_i^k represents the number of times group type i is placed in the k-th placing pattern. For a pattern $(t_1^k, t_2^k, \ldots, t_m^k)$ to be feasible, it must satisfy: $\sum_{i=1}^{m} t_i^k s_i \leq S$, where s_i is the size of group type i. Denote by K the current number of placing patterns.

This problem is to decide how to place a total number of group type i at least g_i times, from all the available placing patterns, so that the total number of rows of seats used is minimized.

Immediately we have the master problem:

$$\begin{aligned} \min \quad & \sum_{k \in K}^{K} x_k \\ \text{s.t.} \quad & \sum_{k \in K}^{K} t_i^k x_k \geq d_i \quad \text{for } i = 1, \ldots, m \\ & x_k \geq 0, \text{integer} \quad \text{for } k \in K, \ldots, K. \end{aligned}$$

If K includes all possible patterns, we can obtain the optimal solution by solving the corresponding IP. But it is clear that the patterns will be numerous, considering all possible patterns will be time-consuming.

Thus, we need to consider the linear relaxation of the master problem, and the optimal dual variable vector λ. Using λ as the value assigned to each group type i, the next problem is to find a feasible pattern $(y_1, y_2, \ldots, y_m)$ that maximizes the product of λ and y.

Then the corresponding sub-problem is:

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} \lambda_i y_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} w_i y_i \leq S \\ & y_i \geq 0, \text{integer} \quad \text{for } i = 1, \ldots, m. \end{aligned}$$

This is a knapsack problem, its solution will be used as an additional pattern in the master problem. We should continue to add new pattern until all reduced costs are nonnegative. Then we have an optimal solution to the original linear programming problem.

But note that column generation method cannot gaurantee an optimal solution. If we want to reach the optimal solution, we should tackle with the integer formulation.

$$
\begin{aligned}
&\min \sum_{k\in K}^{K} y_k \\
&\sum_{k=1}^{K} x_{ik} \geq d_i \quad i = 1, \ldots, n \\
&\sum_{i=1}^{n} a_i x_{ik} \leq S y_k \quad k = 1, \ldots, K \\
&y_k \in \{0, 1\} \quad k = 1, \ldots, K \\
&x_{ik} \geq 0 \text{ and integer } i = 1, \ldots, n; k = 1, \ldots, K
\end{aligned} \tag{13}
$$

$y_k = 1$ if line k is used and 0 otherwise, x_{ik} is the number of times group i is placed in row k, and K is the upper bound on the number of the rows needed.

7.2 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.

7.2.1 Mapping Sequences to Scenarios by Continuous Time

When the time is continuous, we only need to count the number of different group types, the method will be the same.

For the nonhomogeneous Poisson process, $N_i(T) \sim Poisson(\int_0^T p_i(s)\lambda(s)\mathrm{d}s)$. If the Poisson process is homogeneous, $N_i \sim Poisson(\lambda p_i T)$ during the interval $[0, T]$. Then for a realization of the Poisson process, we can still apply the method in Section 4.2.

7.3 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.

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

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

[4] 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.

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

[6] 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.

[7] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

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

[9] 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.

[10] Guntram Scheithauer and Johannes Terno. The modified integer round-up property of the one-dimensional cutting stock problem. *European Journal of Operational Research*, 84(3):562–571, 1995.

Proof

(Theorem 1). □

(Lemma 1). □

(Lemma 2). □

(Theorem 2). □

(Theorem 2). □

(Theorem 3). □

(Theorem 4). □

(Theorem 5). □

(Theorem 6). □

(Lemma 2). □

(Theorem 7). □

(Lemma 4). □