# Seat assignment with the social distancing

Dis· count

February 3, 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 and stochastic 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 assignment based on the column generation algorithm with given rows of seats and demands of groups. We also develop a column-and-cut method to obtain an assignment with stochastic demands of groups under scenarios. With these results, we can provide a guideline for policies related to seat utilization rate.

Keywords: Social distancing, Cutting stock problem, Combinatorial optimization, COVID-19.

## 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 will be seated 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] [4]. Recently, some scholars studied the effects of social distance on health and economics, mainly in aircraft [8] [5]. 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. The column generation method is used to obtain the minimal number of rows. 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 several possible integral solutions 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

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

[9] 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. [3] [2] [7]

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 Description

## 3.1 Basic Concept

At first, we will introduce some preliminary knowledge about our problem as follows.

We consider a set of groups to be assigned to a set of seats $S$. For the purpose of illustration, we consider the layout of $S$ as a rectangle formed by $m$ rows and $n$ columns. But our model and formulation allow for a more general layout of the seats.

The customers from the same group can sit together while different groups should sit with the social distancing. Considering the actual situation, the social distancing is one seat in our paper and different rows have no effect each other, i.e., a person from one group can sit directly behind a person from another group.

For dealing with the social distancing together, we add one to the original size of each group as the new size of the group and one dummy seat to each row. Then the seat assignment for one row can be illustrated as 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.

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

## 3.2 Deterministic Model

We are given a demand, for example, $(d_1, d_2, d_3, d_4, d_5, d_6) = (3, 5, 7, 0, 10, 6)$, where $d_i$ indicates the number of group containing $i$ people. Suppose each group has to leave a seat to maintain social distancing with the adjacent groups. Regard the groups as items in the CSP, and rows as stocks to be cut.

Suppose that the number of rows is fixed, we hope to accommodate as many as people possible.

And the IP formulation can be shown as below:

$$\begin{aligned} \max \quad & \sum_{j=1}^{N}\sum_{i=1}^{m}(s_i - 1)x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{m} s_i x_{ij} \le L_j, j = 1, \ldots, N \\ & \sum_{j=1}^{N} x_{ij} \le d_i^u, i = 1, \ldots, m \\ & x_{ij} \ge 0, i = 1, \ldots, m, j = 1, \ldots, N. \end{aligned} \tag{1}$$

$m$ indicates the number of rows. $x_{ij}$ indicates the number of group type $i$ placed in each row $j$.

### 3.3 Property

Although the solver can solve this problem easily, the analyses on the property of the solution to this problem can help us generate the useful 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 $\alpha_k, \beta_k$ to indicate the number of groups and the left seat, respectively. Denote by $l(k) = \alpha_k + \beta_k$ the loss for pattern $k$.

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. Recall that we use the vector $(t_1, t_2, \ldots, t_m)$ to represent a pattern, where $t_i$ is the size of group $i$.

For example, take the length of each row be S = 21, the size of group types be $s = [2, 3, 4, 5]$. Thus these patterns, $(5, 5, 5, 5, 1), (5, 4, 4, 4, 4), (5, 5, 5, 3, 3)$, belongs to $I_1$. Notice that the pattern, $(0, 0, 0, 4)$, is not full because there is one left space.

Suppose $(u - 1)$ is the size of the largest group allowed, all possible seats will be taken are the consecutive integers starting from 2, $[2, 3, \ldots, u]$. 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 occupied by the group with the corresponding size. The loss is $q + 1$, where $q$ is the number of times $u$ selected. Let $S = u \cdot q + r$, when $r > 0$, we will have at least $\lfloor \frac{r+u}{2} \rfloor - r + 1$ largest patterns with the same loss. When $r = 0$, we have only one possible largest pattern.

**Lemma 1.** *If all patterns associated with an integral feasible solution belong to $I_1$, then this solution is optimal.*

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

When the number of given rows is small, we can construct a solution in the following way. Every time we can select one pattern from $I_1$, then minus the corresponding number of group type from demand and update demand. Repeat this procedure until we cannot generate a largest pattern. Compare the number of generated patterns with the number of rows.

But how could we know if the number of rows is small enough? We can consider the relation between the demand and the number of group types in patterns. Then we develop the following proposition:

**Proposition 1.** *Let $k^* = \arg\max_{k \in I_1} \min_i \{\lfloor \frac{d_i}{b_i^k} \rfloor\}$, When $N \leq \max_{k \in I_1} \min_i \{\lfloor \frac{d_i}{b_i^k} \rfloor\}$, select $k^*$-th pattern from $I_1$ and it is the optimal solution. $N$ is the number of rows, $i = 1, 2, \ldots, m$, $d_m$ is the demand of the largest size, $b_m^k$ is the number of group $m$ placed in pattern $k$.*

In the light of the Proposition 1, when the number of given rows is small, we just need to select some patterns from $I_1$. Continuing with the above example, we just take $(5, 5, 5, 5), (5, 4, 4, 4, 4), (5, 5, 4, 4, 3)$ as the alternative patterns. For each $k$, $\min_i \{\lfloor \frac{d_i}{b_i^k} \rfloor\}$ will be $2, 3, 5$ respectively. So when $N \leq 5$, we can always select the pattern $(5, 5, 4, 4, 3)$ five times as the optimal solution.

# 4 Stochastic Demands Situation

## 4.1 Two-stage Stochastic Programming

Consider the decision maker who has to act in two consecutive stages. The first stage involves the choice of cutting patterns denoted by decision vector $\mathbf{x}$. At the second stage, some new information about demands is obtained, then a new vector $\mathbf{y}$ of decisions is to be chosen.

$\mathbf{x} \in \mathbb{Z}_+^n$ is the vector of first-stage scenario-independent decision variables, each component $x_k$ stands for the pattern $k$.

Use $\omega$ to index the different scenarios, each scenario $\omega \in \Omega, \Omega = \{1, 2, \ldots, S\}$ corresponds to a particular realization of the demand vector, $\mathbf{D}_s = (d_{1s}, d_{2s}, \ldots, d_{ms})$.

$\mathbf{y} \in \mathbb{Z}_+^m$ is the vector of second-stage scenario-dependent decision variables, which include the number of holding groups, $y_{i\omega}^+$, when the supply overestimates the actual demand and the number of short groups, $y_{i\omega}^-$, when the supply understimates the actual demand for group type $i$ in scenario $\omega$.

Regarding the nature of the obtained information, we assume that there are $S$ possible scenarios, and that the true scenario is only revealed after $\mathbf{x}$ is chosen.

Let $p_\omega$ denote the probability of any scenario $\omega$, which we assume to be positive.

We use the same definition as above, the size of group type $i$, $s_i$.

The assignment will be determined before the realization of the random demand, here-and-now policy.

Considering that the seats assigned to some group type can be taken by the smaller group type, we assume that the holding group type $i$ can be utilized by the smaller group type $j = i - 1$. Then we have the following scenario-based optimization problem:

$$
\begin{aligned}
\max \quad & E_\omega \left[ \sum_{i=1}^{m-1} (s_i - 1)(\sum_{j=1}^{N} x_{ij} + y_{i+1,\omega}^+ - y_{i\omega}^+) + (s_m - 1)(\sum_{j=1}^{N} x_{mj} - y_{m\omega}^+) \right] \\
\text{s.t.} \quad & \sum_{j=1}^{N} x_{ij} - y_{i\omega}^+ + y_{i+1,\omega}^+ + y_{i\omega}^- = d_{i\omega}, \quad i = 1, \ldots, m-1, \omega \in \Omega \\
& \sum_{j=1}^{N} x_{ij} - y_{i\omega}^+ + y_{i\omega}^- = d_{i\omega}, \quad i = m, \omega \in \Omega \\
& \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\
& y_{i\omega}^+, y_{i\omega}^- \in \mathbb{Z}_+, \quad i \in I, \omega \in \Omega \\
& x_{ij} \in \mathbb{Z}_+, \quad i = 1, \ldots, m, j = 1, \ldots, N.
\end{aligned}
\tag{2}
$$

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

Reformulate the objective function,

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

When $s_i = i + 1$, the objective function is $\sum_{j=1}^{N}\sum_{i=1}^{m} i x_{ij} - \sum_{\omega=1}^{S} p_\omega \sum_{i=1}^{m} y^+_{i\omega}$.

This programming is in a deterministic equivalent form.

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 number of seats for row $j$. 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

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

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

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

and $\mathbf{I}$ is the identity matrix.

$$
\mathbf{y}_s = \begin{bmatrix} \mathbf{y}^+_s \\ \mathbf{y}^-_s \end{bmatrix}, \quad s \in \Omega
$$

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

As we can find, this formulation is a large-scale problem even if the number of possible scenarios $\Omega$ is moderate. Thus, we need to reformulate the problem and use a decomposition method.

## 4.2 Solve The Second Stage Problem

Let $c'\mathbf{x} = \sum_{j=1}^{N}\sum_{i=1}^{m}(s_i - 1)x_{ij}$, $f'y_\omega = -\left((s_1 - 1)y^+_{1\omega} + \sum_{i=2}^{m}(s_i - s_{i-1})y^+_{i\omega}\right)$. The objective function of problem (2) can be expressed as $c'\mathbf{x} + \sum_\omega p_\omega f' y_\omega$.

Once $x$ is fixed, the optimal second stage decision $y_\omega$ can be determined by solving for each $\omega$ the following problem:

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

Let $z_\omega(x)$ be the optimal value of the problem (3), together with the convention $z_\omega(x) = \infty$ if the problem is infeasible.

Now go back to consider the optimization of $x$, we are faced with the problem:

$$\begin{aligned} \max \quad & c'\mathbf{x} + \sum_{\omega \in \Omega} p_\omega z_\omega(\mathbf{x}) \\ \text{s.t.} \quad & \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\ & \mathbf{x} \geq 0 \end{aligned} \tag{4}$$

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

$$\begin{aligned} \min \quad & {\alpha'}_\omega(\mathbf{d}_\omega - \mathbf{x1}) \\ \text{s.t.} \quad & {\alpha'}_\omega \mathbf{V} \geq f' \end{aligned} \tag{5}$$

Let $P = \{\alpha | \alpha' V \geq f'\}$. We assume that $P$ is nonempty and has at least one extreme point. Then, either the dual problem (5) has an optimal solution and $z_\omega(x)$ is finite, or the primal problem (3) is infeasible and $z_\omega(x) = \infty$.

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 2.** *The feasible region of problem (5), $P$, is bounded. In addition, all the extreme points of $P$ are integral.*

(Proof of lemma 2)**.** *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.

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

**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 $0 \leq \alpha_i \leq \alpha_{i-1} + 1$, $i = 1, \ldots, m$.*

When we are given $x^*$, the demand that can be satisfied by the arrangement 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:

$$\begin{aligned} y^-_{m\omega} &= (d_{m\omega} - d_{m0})^+ \\ y^+_{m\omega} &= (d_{m0} - d_{m\omega})^+ \\ y^-_{i\omega} &= \left(d_{i\omega} - d_{i0} - y^+_{i+1,\omega}\right)^+, i = 1, \ldots, m-1 \\ y^+_{i\omega} &= \left(d_{i0} - d_{i\omega} + y^+_{i+1,\omega}\right)^+, i = 1, \ldots, m-1 \end{aligned} \tag{6}$$

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

**Theorem 1.** *The optimal solutions to problem (5) are given by*

$$\begin{aligned} &\alpha_{i\omega} = 0, i = 1, \ldots, m \quad \text{if } y^-_{i\omega} > 0 \\ &\alpha_{i\omega} = \alpha_{i-1,\omega} + 1, i = 1, \ldots, m \quad \text{if } y^+_{i\omega} > 0 \end{aligned} \tag{7}$$

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 $\Delta d$ will be negative integer, positive integer and zero. The value of $\alpha$ associated with zero will not affect objective function directly, only affect the value of $\alpha$ associated with negative integer. The larger the value of $\alpha$ associated with 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, starting from $\alpha_{1\omega}$ to $\alpha_{m\omega}$, to obtain the value of $\alpha_\omega$ rather than solving a linear programming.

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

### 4.3 Solve The Two-stage Problem by Benders Decomposition

Let $z_\omega(x)$ be the upper bound of $z_\omega$ such that $(\alpha^k)'(\mathbf{d}_\omega - \mathbf{x1}) \geq z_\omega, \alpha^k \in \mathcal{O}$, which is the optimality cut.

Use the characterization of $z_\omega(x)$ in the problem (4), and take into account the optimality condition, we can conclude the master problem (4) will have the form:

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

Now use a small subset of $\mathcal{O}$, $\mathcal{O}^t$, to substitute the original one, then we have a relaxation of the master problem (8):

$$\begin{aligned} \max \quad & c'x + \sum_{\omega \in \Omega} p_\omega z_\omega \\ \text{s.t.} \quad & \sum_{i=1}^{m} s_i x_{ij} \le L_j, j = 1, \ldots, N \\ & (\alpha^k)'(\mathbf{d}_\omega - \mathbf{x1}) \ge z_\omega, \alpha^k \in \mathcal{O}^t, \forall \omega \\ & \mathbf{x} \ge 0 \end{aligned} \tag{9}$$

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

**Lemma 3.** *When each scenario has at least any one optimality cut, the problem (9) is always bounded.*

(Proof of lemma 3)**.** *Suppose we have one extreme point $\alpha^\omega$ for each scenario. Then we have the following problem.*

$$\begin{aligned} \max \quad & c'x + \sum_{\omega \in \Omega} p_\omega z_\omega \\ s.t. \quad & \sum_{i=1}^{m} s_i x_{ij} \le L_j, j = 1, \ldots, N \\ & (\alpha^\omega)'\mathbf{d}_\omega \ge (\alpha^\omega)'\mathbf{x1} + z_\omega, \forall \omega \\ & \mathbf{x} \ge 0 \end{aligned} \tag{10}$$

*Problem (10) reaches its maximum when $(\alpha^\omega)'\mathbf{d}_\omega = (\alpha^\omega)'\mathbf{x1} + z_\omega, \forall \omega$. Substitute $z_\omega$ with these equations, we have*

$$\begin{aligned} \max \quad & c'x - \sum_{\omega} p_\omega (\alpha^\omega)'\mathbf{x1} + \sum_{\omega} p_\omega (\alpha^\omega)'\mathbf{d}_\omega \\ s.t. \quad & \sum_{i=1}^{m} s_i x_{ij} \le L_j, j = 1, \ldots, N \\ & \mathbf{x} \ge 0 \end{aligned} \tag{11}$$

*Notice that $\mathbf{x}$ is bounded by $\mathbf{L}$, then the problem (10) is bounded. Adding more optimality cuts will not make the optimal value larger. Thus, the problem (9) is bounded.* □

Given the initial $\mathcal{O}^t$, we can obtain the optimal $\mathbf{x}^*$ and $\mathbf{z}^* = (z_1^*, \ldots, z_S^*)$. By solving problem (5), we can check whether $(\mathbf{x}^*, \mathbf{z}^*)$ is an optimal solution to the full problem.

According to theorem 1, we can obtain the optimal solution, $\alpha_\omega^*$, to problem (5). When $\mathbf{x}_0$ is given,

$z^0_\omega = \alpha^*_\omega(d_\omega - \mathbf{x}_0\mathbf{1})$ will give a minimal upper bound of $z_\omega$, thus all the left constraints associated with other extreme points are redundant.

Notice that $(x_0, z^0_\omega)$ is a feasible solution ($c'x_0 + \sum_{\omega\in\Omega} p_\omega z^0_\omega$ is the lower bound) when the extreme points are $\alpha_\omega$. The problem (10) associated with $\alpha_\omega$ will give an optimal solution $(x_1, z^1_\omega)$. (Upper bound)

The steps of the algorithm can be described as below,

---

**Algorithm 1** The benders decomposition algorithm

---

**Step 1.** Solve LP (10) with all $\alpha^0_\omega = \mathbf{0}$ for each scenario. Then, obtain the solution $(x_0, z^0_\omega)$ and dual solution $(\pi, \lambda)$.

**Step 2.** Set the upper bound $UB = c'x_0 + \sum_{\omega\in\Omega} p_\omega z^0_\omega$

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

**Step 4.** If $(\alpha^1_\omega)'(\mathbf{d}_\omega - \mathbf{x}_0\mathbf{1}) < z^{(0)}_\omega$, add one new constraint, $(\alpha^1_\omega)'(\mathbf{d}_\omega - \mathbf{x}\mathbf{1}) \geq z_\omega$, to LP (9).

**Step 5.** Solve the updated LP (9), obtain a new solution $(x_1, z^1_\omega)$ and update UB.

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

---

**Remark 1.** *From the lemma 3, we can set $\alpha^0_\omega = \mathbf{0}$ 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 $x^*$. 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 (6).

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

## 4.4 Obtain Near-optimal Seat Assignment

The stochastic model only gives the maximal people that can be served, not the optimal seat assignment. It will not give an appropriate solution when the number of arriving people of the scenarios is way less than the number of total seats because it does not utilize all the empty seats.

We make the following changes to obtain the near-optimal seat assignment in the once-decision method.

Before that, we give the deterministic model with a lower bound and upper bound of demand as

below,

$$\begin{aligned}
\max \quad & \sum_{j=1}^{N}\sum_{i=1}^{m}(s_i-1)x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{m} s_i x_{ij} \le L_j, j=1,\dots,N \\
& \sum_{j=1}^{N} x_{ij} \ge d_i^l, i=1,\dots,m \\
& \sum_{j=1}^{N} x_{ij} \le d_i^u, i=1,\dots,m \\
& x_{ij} \ge 0, \text{integer}, i=1,\dots,m, j=1,\dots,N.
\end{aligned} \tag{12}$$

$d_i^l$ is the lower bound of the demand. (It can be the number of group type $i$ we have accepted.) $d_i^u$ is the upper bound of the demand.

When the lower/upper bound is not needed, the corresponding constraints can be removed. The solver can solve this deterministic formulation with bounds quickly.

**Algorithm 2** Near-optimal seat assignment algorithm

**Step 1.** Obtain the solution from stochatic programming by benders decomposition.

**Step 2.** Aggregate the solution to the supply for each group type.

**Step 3.** Set the supply as the upper bound of demand in problem (12). The lower bound is not included.

**Step 4.** Aggregate the solution from problem (12) to a supply for each group type.

**Step 5.** Obtain the near-optimal assignment algorithm from (12) by setting the supply as the lower bound.

**Remark 3.** *Step 3 can give a feasible integer supply. Then problem (12) with this supply as the lower bound can always give a integer solution. Thus, we can obtain the near-optimal seat assignment by setting the upper and lower bound alternately.*

# 5 Dynamic demand situation

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. Specifically, we define the concept of target seating plans deemed satisfactory. In making the dynamic seating plan, we will try to maintain the possibility of achieving one of the target seating plans as much as possible.

## 5.1 Dynamic Situation without Stochastic Information

## 5.2 Solve the dynamic situation with the largest patterns

Partially dynamic: at the beginning stage, the capacity is sufficient, thus we will accept all arrivals. Suppose that we don't know the stochastic information, we can use the multiple planning approach (Apply the largest pattern for each row). When the largest patterns cannot accept more groups, we can change the largest pattern to a second largest pattern according to the realized arrivals.

When we have the stochastic information, we can use nested policy based on the multiple planning approach. For each row, we can choose the pattern 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 use the nested policy to make the decision.

---

**Algorithm 3** 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 $i$ and go to step 5; otherwise, go to step 4.

**Step 4.** Add up the remaining supply for all rows, use the nested policy to make the decision.

**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.)

**Lemma 4.** *Any largest patterns can be generated by the largest pattern constructed from the greedy method.*

Results compared with other methods:

We can compare M3 with M5, the improvement from M3 to M5 results from the stochastic information. But the result of M5 is not ideal compared with other methods.

M3 can be used without stochastic information.

## 5.3 Generate Scenarios with Stochastic Information

At first, we can use the stochastic information to generate scenarios.

### 5.3.1 Mapping sequences to scenarios by discrete periods

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

Suppose there are $T$ independent periods, at most one group will arrive in each period.

There are $J$ different group types(including the group with no people). Let $\mathbf{y}$ be a discrete random variable indicating the number of people in the group. Let $\mathbf{p}$ be the vector probability, where $p(y = j) =$

$p_j$, $j = 0, 1, \ldots, J-1$ 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, $p(\mathbf{Y} \mid \mathbf{p}) = \prod_{j=0}^{J-1} p_j N_j$).

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,

$$p\left(N_0, \ldots, N_{J-1} \mid \mathbf{p}\right) = \frac{T!}{N_0!, \ldots, N_{J-1}!} \prod_{j=0}^{J-1} p_j^{N_j}, T = \sum_{j=0}^{J-1} N_j$$

Show the complexity: the number of different sequences $J^T$, and the number of scenarios is $O(T^{J-1})$.

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

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

#### 5.3.2 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 5.3.

### 5.4 Nested policy when given supply

Once we obtain a solution from the stochastic programming, we need to follow some basic rules to assign the 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 accpected to occupy the larger-size seats, the rest empty seat(s) can be reserved for the 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 assign this group to the larger-size seats? Now we demonstate the nested policy for this problem.

If we accept a group of $i$ to take over $j$-size seats, then 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, $P(D_{j-i-1} \geq x_{j-i-1} + 1)$ indicates the probability of 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 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, the group will be rejected.

The algorithm is shown below:

**Algorithm 4** 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.

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

## 5.5 Seat assignment scenarios and methods

In this section, we will present several methods to assign seats under different scenarios.

At first, we give the general method to deal with the dynamic situation.

**Algorithm 5** General method to deal with the Dynamic situation

**Step 1.** Obtain a linear solution from stochastic programming (8).

**Step 2.** Obtain the near-optimal integral solution from deterministic model with lower and upper bounds.

**Step 3.** Accept or reject group arrivals according to the nested policy in section 5.4.

**Step 4.** According to different scenarios, we can fix the supply or re-calculate to give a new supply.

In step 4, the scenarios include the ticket reservation without seat selection, ticket reservation with only row selection, ticket reservation with restricted seat selection.

These senarios are described as follows:

1. Seat assignment will not be made immediately, we only need to reject or accept each request during the making-reservation stage. After the deadline of the reservation, we will give the seat assignment and tell the groups their seats when they check in. This scenario applies for the reservation without seat selection. For example, some theaters and concerts only provide the seat reservation for the audiences.

2. The groups can only choose which row to sit. This scenario appears in the on-site seminar.

We need to assign seats to the group for each arrival. In each period, the group can select the row they want to sit when the capacity is enough.

FCFS will be more appropriate. But M1 and M3 can also be used.

3. Online booking. (Can select 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.

4. The seat assignment will be arranged before the arrival of groups, the groups only need to find the corresponding-size seats. This scenario applies for the reservation without seat selection. For example, the movie halls can assign the seats in advance to save costs. And seat assignment could be remained in one day because the same film genre will attract the same feature of different group types.

There are six methods basically:

M1: The seat assignment(supply) is obtained from stochastic model. Then use Algorithm 4 to make the decision.

M1 can be used in scenario 1,2,3,4. The seats can be placed in the cinema hall in advance.

M2: DP-based. We relax several rows to one row with the same capacity. Suppose there always exists one assignment under the 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 accpected sequence, we still need to check whether this sequence is feasible for the seat assignment. For most cases, it is feasible. That is the reason why we choose DP. 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 accept the request until we cannot find the seat assignment. It can only be applied in scenario 1.

M3: Set the largest pattern in each row. See section 5.2

M4: The inital supply is obtained from stochastic model, update the supply from deterministic

model by setting accpected demand as the lower bound. / It can be used in scenario 1,2. (The initial supply can be other demands.)

M5: The intuitive but trivial method will be first-come-first-serve. Each request will be assigned row by row. When the capacity of one row is not enough for the request, we arrange it in the next row. If the following request can take up the remaining capacity of some row exactly, we place it in the row immediately. We check each request until the capacity is used up.

It can be used in scenario 1,2,3. For scenario 3, the performane will be worse without restriction.

M6: Based on first-come-first-serve. For the arrival sequence, find the target arrival when the total number of seats taken by the preceding arrivals does not exceed the capacity. And use nested policy to accept or reject one group in the remaining arrivals.

Then we obtain a new sub-sequence, which includes the arrivals from the first one to the target one and a possible arrival.

Similar to M2, when it is not feasible for the seat assignment, we should delete the group one by one from this sub-sequence until it is feasible.

It can be used in scenario 1.

The relation between the scenarios and the methods is showed in the following table:

Table 1: Relation between the scenarios and the methods

| | M1 | M2 | M3 | M4 | M5 | M6 |
|---|---|---|---|---|---|---|
| Scenario 1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Scenario 2 | ✓ | | ✓ | ✓ | ✓ | |
| Scenario 3 | ✓ | | ✓ | | ✓ | |
| Scenario 4 | ✓ | | | | | |

The results of different methods on scenario 1 are shown below.

| # samples | T | probabilities | # rows | performance(%) compared to the optimal |
|---|---|---|---|---|
| 1000 | 50 | [0.4,0.4,0.1,0.1] | 8 | (99.72, 100.00, 100.00, 100.00, 98.11, 100.00) |
| 1000 | 55 | [0.4,0.4,0.1,0.1] | 8 | (97.75, 99.83, 99.76, 99.76, 93.15, 99.76) |
| 1000 | 60 | [0.4,0.4,0.1,0.1] | 8 | (95.78, 99.20, 97.80, 97.80, 89.35, 97.65) |
| 1000 | 65 | [0.4,0.4,0.1,0.1] | 8 | (95.61, 99.10, 96.23, 96.23, 87.80, 96.12) |
| 1000 | 40 | [0.25,0.25,0.25,0.25] | 8 | (99.94, 100.00, 100.00, 100.00, 98.22, 100.00) |
| 1000 | 45 | [0.25,0.25,0.25,0.25] | 8 | (97.19, 99.51, 99.09, 99.09, 91.31, 99.29) |
| 1000 | 50 | [0.25,0.25,0.25,0.25] | 8 | (95.23, 98.98, 97.21, 97.21, 87.73, 96.88) |
| 1000 | 55 | [0.25,0.25,0.25,0.25] | 8 | (94.84, 99.05, 95.70, 95.70, 85.49, 95.13) |

The numbers in the 'performance compared to the optimal' represent M1, M2, M3, M4, M5, M6 respectively in order.

The maximal number of people served can be obtained by (1) with a realized sequence of arrival.

Run Start Time: Mon Jan 30 16:37:12 2023
probabilities: [0.45 0.05 0.05 0.45]
M1: 98.49 ;M2: 99.38 ;M3: 91.44 ;M4: 99.01 ;M5: 90.88 ;M6: 99.09
Number of accepted people: 147.12 Number of people: 149.64
probabilities: [0.41666667 0.05 0.15 0.38333333]
M1: 97.70 ;M2: 99.44 ;M3: 93.27 ;M4: 99.07 ;M5: 90.39 ;M6: 99.15
Number of accepted people: 148.40 Number of people: 151.04
probabilities: [0.38333333 0.05 0.25 0.31666667]
M1: 98.46 ;M2: 99.78 ;M3: 94.26 ;M4: 99.54 ;M5: 91.86 ;M6: 99.44
Number of accepted people: 145.90 Number of people: 147.20
probabilities: [0.35 0.05 0.35 0.25]
M1: 98.14 ;M2: 99.46 ;M3: 93.64 ;M4: 99.43 ;M5: 90.97 ;M6: 99.47
Number of accepted people: 147.42 Number of people: 149.14
probabilities: [0.31666667 0.05 0.45 0.18333333]
M1: 97.78 ;M2: 99.45 ;M3: 92.42 ;M4: 99.09 ;M5: 90.20 ;M6: 99.15
Number of accepted people: 148.46 Number of people: 150.58
probabilities: [0.28333333 0.05 0.55 0.11666667]
M1: 97.96 ;M2: 99.52 ;M3: 92.88 ;M4: 99.22 ;M5: 91.49 ;M6: 99.27
Number of accepted people: 147.50 Number of people: 149.36
probabilities: [0.25 0.05 0.65 0.05]
M1: 98.23 ;M2: 99.12 ;M3: 92.53 ;M4: 98.55 ;M5: 91.85 ;M6: 98.88
Number of accepted people: 147.74 Number of people: 150.78
probabilities: [0.38333333 0.15 0.05 0.41666667]
M1: 97.73 ;M2: 99.48 ;M3: 92.29 ;M4: 98.80 ;M5: 90.70 ;M6: 99.19
Number of accepted people: 147.34 Number of people: 149.44
probabilities: [0.35 0.15 0.15 0.35]
M1: 97.43 ;M2: 99.53 ;M3: 93.34 ;M4: 99.08 ;M5: 91.15 ;M6: 99.22
Number of accepted people: 147.30 Number of people: 149.60
probabilities: [0.31666667 0.15 0.25 0.28333333]
M1: 97.07 ;M2: 99.58 ;M3: 92.68 ;M4: 99.22 ;M5: 90.37 ;M6: 99.34
Number of accepted people: 148.22 Number of people: 150.46
probabilities: [0.28333333 0.15 0.35 0.21666667]
M1: 97.89 ;M2: 99.59 ;M3: 92.21 ;M4: 99.49 ;M5: 91.01 ;M6: 99.43
Number of accepted people: 147.52 Number of people: 149.14
probabilities: [0.25 0.15 0.45 0.15]
M1: 97.07 ;M2: 99.50 ;M3: 91.13 ;M4: 99.35 ;M5: 90.79 ;M6: 99.41
Number of accepted people: 148.42 Number of people: 149.96
probabilities: [0.21666667 0.15 0.55 0.08333333]
M1: 97.69 ;M2: 99.67 ;M3: 91.18 ;M4: 99.53 ;M5: 91.47 ;M6: 99.47
Number of accepted people: 147.84 Number of people: 149.14
probabilities: [0.18333333 0.15 0.65 0.01666667]
M1: 98.10 ;M2: 99.65 ;M3: 91.23 ;M4: 99.69 ;M5: 91.40 ;M6: 99.65
Number of accepted people: 147.84 Number of people: 149.44
probabilities: [0.31666667 0.25 0.05 0.38333333]
M1: 97.62 ;M2: 99.56 ;M3: 92.33 ;M4: 98.94 ;M5: 90.17 ;M6: 99.20
Number of accepted people: 148.94 Number of people: 151.04
probabilities: [0.28333333 0.25 0.15 0.31666667]
M1: 96.87 ;M2: 99.39 ;M3: 92.39 ;M4: 99.19 ;M5: 90.38 ;M6: 99.32
Number of accepted people: 148.70 Number of people: 150.82
probabilities: [0.25 0.25 0.25 0.25]
M1: 97.28 ;M2: 99.66 ;M3: 91.92 ;M4: 99.30 ;M5: 90.15 ;M6: 99.36
Number of accepted people: 148.72 Number of people: 150.76
probabilities: [0.21666667 0.25 0.35 0.18333333]
M1: 97.35 ;M2: 99.65 ;M3: 91.41 ;M4: 99.25 ;M5: 90.29 ;M6: 99.41
Number of accepted people: 148.72 Number of people: 150.44
probabilities: [0.18333333 0.25 0.45 0.11666667]
M1: 96.92 ;M2: 99.48 ;M3: 90.75 ;M4: 99.14 ;M5: 90.38 ;M6: 99.16

# 6 Results

## 6.1 Benders decomposition versus IP

At first, we compare the running time of these two methods.

The running times of solving IP directly and using Benders decomposition are shown in the table below.

Table 2: Running time of Decomposion and IP

| # of Scenarios | running time of IP(s) | Benders (s) | # of rows | # of groups |
|---|---|---|---|---|
| 1000 | 5.1 | 0.13 | 30 | 8 |
| 5000 | 28.73 | 0.47 | 30 | 8 |
| 10000 | 66.81 | 0.91 | 30 | 8 |
| 50000 | 925.17 | 4.3 | 30 | 8 |
| 1000 | 5.88 | 0.29 | 200 | 8 |
| 5000 | 30.0 | 0.62 | 200 | 8 |
| 10000 | 64.41 | 1.09 | 200 | 8 |
| 50000 | 365.57 | 4.56 | 200 | 8 |
| 1000 | 17.15 | 0.18 | 30 | 16 |
| 5000 | 105.2 | 0.67 | 30 | 16 |
| 10000 | 260.88 | 1.28 | 30 | 16 |
| 50000 | 3873.16 | 6.18 | 30 | 16 |

The parameters of the experiments are listed below.

The first one: The number of rows is 30. The number of groups is 8. The number of seats for each row L is generated from (21, 50) randomly, about 1000 seats. The scenarios of demands are generated from (150, 350) randomly.

The second one: The number of rows is 200. The number of groups is 8. The number of seats for each row L is generated from (21, 50) randomly, about 7000 seats. The scenarios of demands are generated from (1000, 2000) randomly.

The third one: The number of rows is 30. The number of groups is 16. The number of seats for each row L is generated from 41-60 randomly, about 1500 seats. The scenarios of demands are generated from (150, 250) randomly.

Then we compare the results of benders decomposition and IP under nested policy.

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. Thus, we can substitute IP with the Benders decomposition.

## 6.2 Different probabilities

Discuss the effect of different probabilities. $E(D) = (p_1 * 1 + p_2 * 2 + p_3 * 3 + p_4 * 4)T$

Let $c = p_1 * 1 + p_2 * 2 + p_3 * 3 + p_4 * 4$. Notice that different $c$ need different $T$ such that the supply is close to the demand. Different $T$ has little effect on the number of people served using M6 because it

Table 3: 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 |

is based on first-come-first-served.

Thus, we use M6 to 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. 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:

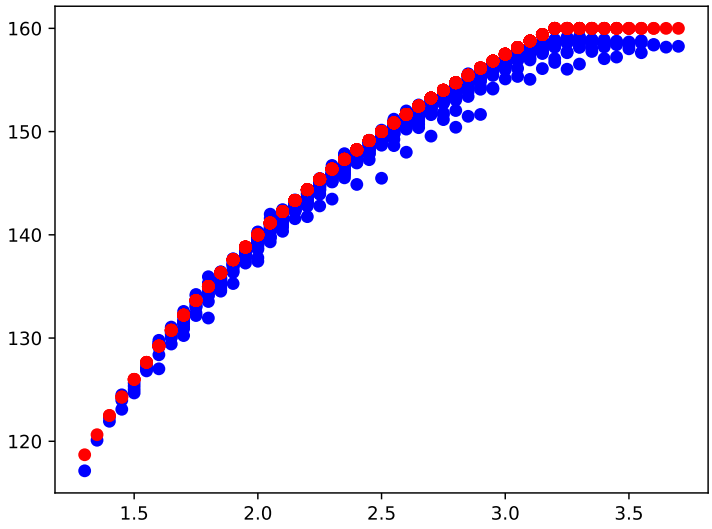


Figure 2: The number of people served versus $c$

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. Then the occupancy rate is $\frac{c*T}{(c+1)*T} = \frac{c}{c+1}.(D_a = c * T)$

$\frac{c}{c+1}$ is the estimation of occupancy rate 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.)

Why some blue points are far from the red points?

Explanation: For the case of probability $[0.05, 0.05, 0.85, 0.05](c = 2.9)$, the demand can be $[4, 1, 45, 2]$ or $[2, 2, 47, 1]$. We cannot construct all full patterns for every row. It violates the assumption, thus in this case there is a large gap between blue point and red point.

We can find that 160 is the upper bound of the number of people served.

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 $210 * \frac{16}{21} = 160$.

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.

Results: M1-M6, the number of accepted people, the number of total people.

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.

Results: M1-M6, the number of accepted people, the number of total people.

### 6.3 Different periods

Discuss the effect of the number of periods: Parameters: T = 70-80, step size =1.

The expected number of period: 75 The expected number of demand(people): 150 Number of rows: 9 Number of seats each row: 25 Probabilities: $[0.4, 0.3, 0.2, 0.1], [0.3, 0.5, 0.1, 0.1]$.

Results: M1-M6, the number of accepted people, the number of total people.

T = 55-65, 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: $[0.25, 0.25, 0.25, 0.25]$.

Results: M1-M6, the number of accepted people, the number of total people.

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

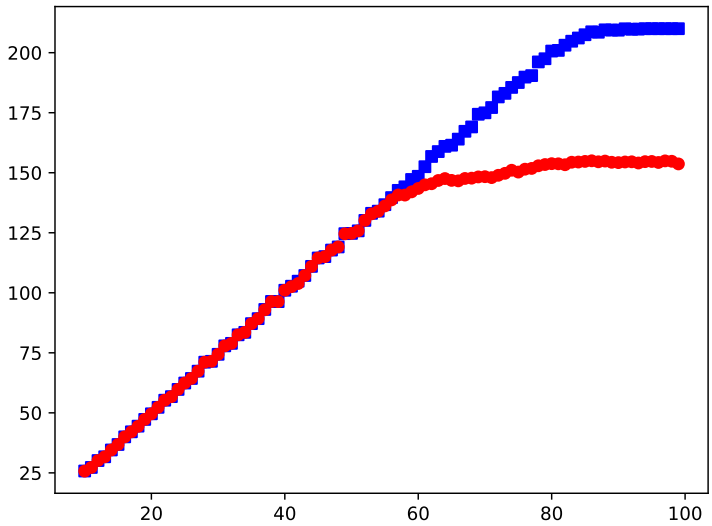


Figure 3: The number of people served versus $c$

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 increases, the gap will converge when the capacity is limited.

## 6.4 Measurement

Suppose a real scenario with a fixed sequence, $s^r$. Solving the following program can obtain the optimal value, $V_{s^r}$. (Offline)

Then the difference is $V_{s^r}$ − our result.

WS(the value under wait-and-see policy with all possible scenarios)

EVPI(Expected Value of Perfect Information) = WS - the value of deterministic equivalent form

Regret.

## 6.5 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.

## 6.6 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 $\lambda$. Using $\lambda$ 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 $\lambda$ 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 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] 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.

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

[5] 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.

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

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

[8] 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.

[9] 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). □