# Seat assignment with the social distancing

Dis· count

January 1, 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 seating assignment with social distancing. 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 [2] [9]. Recently, some scholars studied the effects of social distance on health and economics, mainly in aircraft [15] [10]. Especially, our study provides another perspective to help the authority adopt a mechanism for setting seat assignments to control the spread of the virus.

Second, we establish the deterministic model to analyze the effects of social distancing. We construct an algorithm based on the column generation method to obtain the maximal supply when the demand is known. 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 and column-and-cut generation methods to obtain the optimal linear solution. Then we develop several possible integral solutions from linear solutions according to the trait 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 dynamic programming 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

[4] gives a method to check IRU and IRD property and give the conclusion that IRU holds for a given A if and only if IRU holds for C(A,w) for every fractional solution.

[7] gives a general decomposition property. We shall say that polyhedron $P(A, d, e)$ has the real decomposition property (RDP) if for any positive real $T$ and any real $x \in TP(A, d, e)$, there exists an integer $r$, positive coefficients $\lambda_1, \dots, \lambda_r$ and vectors $s^1, \dots, s^r \in P(A, d, e)$ such that $x = \lambda_1 s^1 + \cdots + \lambda_r s^r$, $T = \lambda_1 + \cdots + \lambda_r$.

An important property: $Q(A, b)$ has the RDP iff is integral.

[3] gives that IRD holds for $M$(A matrix) if and only if $P$ satisfies the integral decomposition property.

[14] demonstrates that CSP has the IRU property if and only if P, the corresponding knapsack polyhedron, has the integral decomposition property.

Cutting stock problems of the form $(a_1, a_2; b)$ have the IRU property.

[17] compares two branch-and-price approaches for the cutting stock problem. Both algorithms employ column generation for solving LP relaxations at each node of a branch-and-bound tree to obtain optimal integer solutions.

[18] transforms the integer knapsack subproblems into 0-1 knapsack problems and use branch-and-bound procedure to solve them.

[12] sloves CSP based on enumerating the possible cutting patterns.

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

[19] gives the branch and column generation for general IP.

[1] uses branch-and-price to solve huge integer programs.

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

And shows that for all instances an integer solution fulfills the MIRUP(Modified Integer Round-Up Property). Moreover, in most cases the optimal value equals the round-up optimal value of the corresponding LP relaxation.

## 3 Problem Description

### 3.1 Basic Concept

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

### 3.2 Inspired Example

# 4 Deterministic Model

## 4.1 Obtain Minimum Number of Rows to Cover Demand

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. With considering the social distancing, the size of each group should be treated as the actual size plus one. Then each row of seats should also add a dummy(virtual) seat for the same reason, and the number of all seats in each row is called the length of the row.

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.

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

$y_k = 1$ 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.

## 4.2 Provide The Maximal Supply When Given Rows

Let us review this problem in another way. In most cases where the number of rows is fixed, we hope to accommodate as many as people possible. That is, we should minimize the space loss.

Then minimizing $NS - \sum_{i=1}^{m} r_i(s_i - 1)$ equals to maximize $\sum_{i=1}^{m} r_i(s_i - 1)$ and maximze $\sum_{i=1}^{m}(g_i - d_i)(s_i - 1)$.

Notice that $\sum_{k=1}^{K} t_i^k x_k + d_i = g_i$, by substituting this equation we can obtain the objective function of the following master problem.

$$\begin{aligned} (D) \quad \max \quad & \sum_{k=1}^{K} \left(\sum_{i=1}^{m} (s_i - 1) t_i^k\right) x_k \\ \text{s.t.} \quad & \sum_{k=1}^{K} x_k \leq N \qquad (2) \\ & \sum_{k=1}^{K} t_i^k x_k \leq g_i, \quad i = 1, \ldots, m \qquad (3) \\ & x_k \geq 0, \quad k = 1, \ldots, K \end{aligned}$$

Similarly, we consider the linear relaxation of the master problem and the optimal dual variable vector $\lambda, \mu$. Using $\lambda$ as the value assigned to the first constraint (2) and $\mu$ to the second constraints (3). This master problem is to find a feasible pattern $(t_1^{k_0}, t_2^{k_0}, \ldots, t_m^{k_0})$ that maximizes the reduced cost. The corresponding reduced cost is $c_{k_0} - c_B B^{-1} A_{k_0}$, where $c_{k_0} = \sum_{i=1}^{m} (s_i - 1) t_i^{k_0}$, $c_B B^{-1} = (\lambda, \mu)$, $A_{k_0} = (1, t_1^{k_0}, t_2^{k_0}, \ldots, t_m^{k_0})^T$. Use $y_i$ indicate the feasible pattern instead of $t_i^{k_0}$, we can obtain the sub-problem:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{m} \left[(s_i - 1) - \mu_i\right] y_i - \lambda \\
\text{s.t.} \quad & \sum_{i=1}^{m} s_i y_i \le S \\
& y_i \ge 0, \text{integer} \quad \text{for } i = 1, \dots, m.
\end{aligned}
$$

Use column generation to generate a new pattern until all reduced costs are negative.

And the IP formulation can be shown as below:

$$
\begin{aligned}
\max \quad & \sum_{j=1}^{m} \sum_{i=1}^{n} (s_i - 1) x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{n} s_i x_{ij} \le S, \quad j = 1, \dots, m \\
& \sum_{j=1}^{m} x_{ij} \le g_i, \quad i = 1, \dots, n \\
& x_{ij} \ge 0 \text{ and integer}, \quad i = 1, \dots, n, j = 1, \dots, m
\end{aligned} \tag{4}
$$

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

For our new problem, the column generation will give the upper bound (LP relaxation) and lower bound (restricted IP). After obtaining the patterns with column generation, restricted LP equals LP relaxation, $LP^r = LP$, which provides an upper bound. Thus, we have the relation, $\max LP \ge \max LP^r \ge \max IP \ge \max IP^r$.

To obtain an optimal solution, we should implement branch and bound into column generation. This method is called branch-and-price.

## 5 Stochastic demands situation

### 5.1 Deterministic equivalent form

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 $\omega \in \Omega, \Omega = \{1, 2, \dots, S\}$ corresponds to a particular realization of the demand vector, $\mathbf{D}_s = (d_{1s}, d_{2s}, \dots, d_{ms})$.

$\mathbf{y} \in \mathbb{Z}_+^m$ 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:

$$\begin{aligned}
\max \quad & E_\omega \left[ \sum_{i=1}^{m-1} (s_i - 1)(\sum_{j=1}^{N} x_{ij} + y_{i+1,\omega}^{+} - y_{i\omega}^{+}) + (s_m - 1)(\sum_{j=1}^{N} x_{mj} - y_{m\omega}^{+}) \right] \\
\text{s.t.} \quad & \sum_{j=1}^{N} x_{ij} - y_{i\omega}^{+} + y_{i+1,\omega}^{+} + y_{i\omega}^{-} = d_{i\omega}, \quad i = 1, \ldots, m-1, \omega \in \Omega \\
& \sum_{j=1}^{N} x_{ij} - y_{i\omega}^{+} + y_{i\omega}^{-} = d_{i\omega}, \quad i = m, \omega \in \Omega \\
& \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\
& y_{i\omega}^{+}, y_{i\omega}^{-} \in \mathbb{Z}_{+}, \quad i \in I, \omega \in \Omega \\
& x_{ij} \in \mathbb{Z}_{+}, \quad i = 1, \ldots, m, j = 1, \ldots, N.
\end{aligned} \tag{5}$$

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 $N$. $\mathbf{V}_s = [\mathbf{W}\ \mathbf{I}]$.

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

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.

## 5.2 Two-stage stochastic programming

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 (5) 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:

$$\begin{aligned} \max \quad & f'y_\omega \\ \text{s.t.} \quad & \mathbf{x1} + \mathbf{V}\mathbf{y}_\omega = \mathbf{d}_\omega \\ & y_\omega \geq 0 \end{aligned} \tag{6}$$

Let $z_\omega(x)$ be the optimal value of the problem (6), 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:

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

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 (6) does not depend on $x$. We can form its dual, which is

$$\begin{aligned} \min \quad & \alpha'_\omega(\mathbf{d}_\omega - \mathbf{x}) \\ \text{s.t.} \quad & \alpha'_\omega \mathbf{V} \geq f' \end{aligned} \tag{8}$$

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 (8) has an optimal solution and $z_\omega(x)$ is finite, or the primal problem (6) 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.

**Theorem 1.** *The feasible region, $P$, is bounded. And the extreme points are all integral.*

(Proof of theorem 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. And $\bar{s}_i = s_i - s_{i-1}, s_0 = 1$ are integral, so the extreme points are all integral.* □

Remark: we only need to consider the optimality cut in the following decomposition method.

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

$$\begin{aligned}
&0 \leq \alpha_1 \leq 1, \\
&0 \leq \alpha_2 \leq \alpha_1 + 1, \\
&\cdots, \\
&0 \leq \alpha_m \leq \alpha_{m-1} + 1
\end{aligned}$$

When $s_i$ is integral, we can still use the same way to obtain the dual optimal solution.

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 (6), we can obtain the value of $y_{i\omega}$ recursively:

$$\begin{aligned}
y^-_{m\omega} &= (d_{m\omega} - d_{m0})^+ \\
y^+_{m\omega} &= (d_{m0} - d_{m\omega})^+ \\
y^-_{i\omega} &= \left(d_{i\omega} - d_{i0} - y^+_{i+1,\omega}\right)^+, i = 1, \ldots, m-1 \\
y^+_{i\omega} &= \left(d_{i0} - d_{i\omega} + y^+_{i+1,\omega}\right)^+, i = 1, \ldots, m-1
\end{aligned} \tag{9}$$

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

**Theorem 2.** *An optimal solution to problem (8) is given by*

$$\begin{aligned}
\alpha_{i\omega} &= 0, i = 1, \ldots, m \quad \textit{if } y^-_{i\omega} > 0 \\
\alpha_{i\omega} &= \alpha_{i-1,\omega} + 1, i = 1, \ldots, m \quad \textit{if } y^+_{i\omega} > 0
\end{aligned} \tag{10}$$

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 2)**.** *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.* $\square$

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.

We know that $z_\omega(x)$ is finite and it is the optimal value of the problem (6) and this value will be attained at extreme points of the set $P$. Thus, we have $z_\omega(x) = \min_{\alpha^j \in \mathcal{O}} (\alpha^k)'(\mathbf{d}_\omega - \mathbf{x1})$.

## 5.3 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 (7), and take into account the optimality condition, we can conclude the master problem (7) will have the form:

$$\begin{aligned} \max \quad & c'x + \sum_{\omega \in \Omega} p_\omega z_\omega \\ \text{s.t.} \quad & \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\ & (\alpha^k)'(\mathbf{d}_\omega - \mathbf{x1}) \geq z_\omega, \alpha^k \in \mathcal{O}, \forall \omega \\ & \mathbf{x} \geq 0 \end{aligned} \tag{11}$$

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 (11):

$$\begin{aligned} \max \quad & c'x + \sum_{\omega \in \Omega} p_\omega z_\omega \\ \text{s.t.} \quad & \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\ & (\alpha^k)'(\mathbf{d}_\omega - \mathbf{x1}) \geq z_\omega, \alpha^k \in \mathcal{O}^t, \forall \omega \\ & \mathbf{x} \geq 0 \end{aligned} \tag{12}$$

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

**Lemma 1.** *When each scenario has at least any one optimality cut, the problem (12) is always bounded.*

(Proof of lemma 1)**.** *Suppose we have one extreme point $\alpha^\omega$ for each scenario. Then we have the following* *problem.*

$$
\begin{aligned}
\max \quad & c'x + \sum_{\omega \in \Omega} p_\omega z_\omega \\
s.t. \quad & \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\
& (\alpha^\omega)' \mathbf{d}_\omega \geq (\alpha^\omega)' \mathbf{x1} + z_\omega, \forall \omega \\
& \mathbf{x} \geq 0
\end{aligned}
\tag{13}
$$

*Problem (13) 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*

$$
\begin{aligned}
\max \quad & c'x - \sum_{\omega} p_\omega (\alpha^\omega)' \mathbf{x1} + \sum_{\omega} p_\omega (\alpha^\omega)' \mathbf{d}_\omega \\
s.t. \quad & \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j = 1, \ldots, N \\
& \mathbf{x} \geq 0
\end{aligned}
\tag{14}
$$

*Notice that $\mathbf{x}$ is bounded, then the problem (13) is bounded. Adding more optimality cuts will not make the optimal value larger. Thus, the problem (12) is bounded.* □

When some scenario only includes one optimality cut, we can substitute $z_\omega$ into the objective function.

According to theorem 2, we can obtain the optimal solution, $\alpha^*_\omega$, to problem (8). 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 (13) 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 (13) 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, $(\alpha^1_\omega)'(\mathbf{d}_\omega - \mathbf{x1}) \geq z_\omega$, to LP (12).

**Step 5.** Solve the updated LP (12), 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: Step 1 results from the lemma 1. Step 6, notice that only contraints are added, thus $LB$ and $UB$ are both monotone, then we can use $UB - LB < \epsilon$ to terminate the algorithm.

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 (9).

### 5.4 Benders decomposition versus IP

In view of the fact that IP of the deterministic model can be solved quickly, the column generation method does not show an obvious advantage. We can revise the stochastic model as follows:

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

| # 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.

### 5.5 Scenario-based method

The stochastic model only gives the maximal people that can be served, not the maximal 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,

$$
\begin{aligned}
\max \quad & \sum_{j=1}^{N}\sum_{i=1}^{m}(s_i-1)x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{m} s_i x_{ij} \leq L_j, j=1,\ldots,N \\
& \sum_{j=1}^{N} x_{ij} \geq d_i^l, i=1,\ldots,m \\
& \sum_{j=1}^{N} x_{ij} \leq d_i^u, i=1,\ldots,m \\
& x_{ij} \geq 0, i=1,\ldots,m, j=1,\ldots,N.
\end{aligned}
\tag{15}
$$

$d_i^l$ 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.

Firstly, we obtain the solution from stochatic programming. This solution corresponds to a supply for each group type. Then solve the deterministic model by setting the supply as the upper bound of demand to obtain another feasible supply. Finally, solve the deterministic model by setting this supply as the lower bound to obtain the seat assignment.

**Algorithm 2** The scenario-based method to deal with the dynamic situation

```
Step 1. Obtain a linear solution from stochastic programming (11).
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 5.7.
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 limited seat selection.

In part 6, different methods are derived to deal with these situations.

## 5.6 Mapping sequences to scenarios

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.

Remark: 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 5.6.

## 5.7 Nested policy under 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 3** 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.
```

Table 1: Decomposion and IP with 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 | 0.12 | 0.12 |
| 1000 | 50 | [0.4,0.4,0.1,0.1] | 8 | 1.86 | 1.86 |
| 1000 | 55 | [0.4,0.4,0.1,0.1] | 8 | 5.39 | NA |
| 1000 | 60 | [0.4,0.4,0.1,0.1] | 8 | 7.41 | NA |
| 1000 | 65 | [0.4,0.4,0.1,0.1] | 8 | 9.03 | 9.07 |
| 1000 | 37 | [0.25,0.25,0.25,0.25] | 8 | 0.10 | 0.10 |
| 1000 | 40 | [0.25,0.25,0.25,0.25] | 8 | 1.54 | 1.54 |
| 1000 | 45 | [0.25,0.25,0.25,0.25] | 8 | 7.17 | 7.17 |
| 1000 | 50 | [0.25,0.25,0.25,0.25] | 8 | 8.83 | 8.95 |
| 1000 | 55 | [0.25,0.25,0.25,0.25] | 8 | 11.10 | 11.10 |
| 5000 | 300 | [0.25,0.25,0.25,0.25] | 30 | 0.00 | 0.00 |
| 5000 | 350 | [0.25,0.25,0.25,0.25] | 30 | 5.28 | 5.28 |
| 5000 | 400 | [0.25,0.25,0.25,0.25] | 30 | 8.16 | 8.20 |
| 5000 | 450 | [0.25,0.25,0.25,0.25] | 30 | 11.00 | 11.04 |

Need to update the table to show benders decomposition and IP.

Fix the number of seats for each row: 20, 40. 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.

## 6 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.

### 6.1 Seat assignment methods

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

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. We set the result as the baseline.

(1,2,3,4,5,6) in the 'people served in the table': 1: The supply is obtained from stochastic model.(LP) can be used in scenario 1,2,3.

The seat assignment can be used in all scenarios, and the cinema can place the seat in advance, unsettle the useless seats.

2. DP/ Because we relax several rows to one row, this method can only be applied in scenario 1.

3: Set the mean demand as the initial supply, update the supply from deterministic model by setting accpected demand as the lower bound. / can be used in scenario 1,2.

4: The inital supply is obtained from stochastic model, update the supply from deterministic model by setting accpected demand as the lower bound. / Similar to the above.

5: First-come-first-serve / can be used in scenario 1,2,3. For scenario 3, the performane will be worse without restriction.

6: Based on First-come-first-serve, but use the deterministic model to update. can be used in scenario 1,2.

If we need to assign seats to the group each period.

1.1 Every group can only choose which row to sit. In each period, every group can choose to sit in some row with the corresponding capacity. After certain periods, we update the remaining seats in each row, then solve a sub-problem.

1.2 Every group can choose where to sit according to the assignment. In each period, every group can choose to sit everywhere as long as the assignment allows. If one group choose to sit in the middle of some row, then this row is divided into two rows. After certain periods, we update the remaining seats and rows, then solve a similar problem.

Once: Obtain the supply from the stochastic model by benders decomposition. Use the deterministic model to obtain a heuristi supply. Then use the multi-class rule to decide whether to accept the group at each period.

Several: Initially, set the mean demand for all periods as the upper bound of demand. Then obtain the supply from the deterministic model. Set the accepted demand as the lower bound of demand, the upper bound of demand will be the sum of accepted demand and mean demand for the remaining periods. Update the lower bound and upper bound when some supply runs out.

## 6.2 DP

Set several rows as one row with the same capacity. Suppose there always exists one assignment when the total demand equals the capacity. Then we can use DP to accept or reject in each period.

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

Use a buffer to contain the accpected groups, when the groups can be assigned to a full row, then remove them from the buffer.

In fact, direct DP will have a gap, we should use a buffer to improve this method.

For example, the number of seats for 10 rows is 21. The demand is $[1, 2, 41, 16]$. The optimal assignment is $[0, 0, 40, 10]$. But DP will give $[0, 0, 35, 14]$.

When a full pattern is reached, then delete the related groups and the row. Update the remaining demand.

We know $(0, 0, 4, 1)$ is a largest and full pattern, thus an assignment constructed with these 10 patterns is an optimal assignment.

| # 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) |

# 7 Results

## 7.1 Result

Merit: The plan will be always feasible.

Demerit: Cannot cover all possible demands.

Improvement: For $\mathbf{X}^0$, we introduce one empty seat, $x_1$. But it cannot provide the feasibility.

## 7.2 How to generate scenario demands

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. [8] [6] [13]

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.

Solving the deterministic formulation with a large set of scenarios is not tricky in our case.

For the stochastic situation, we assume the group sizes are discretized from independent random variables following some distribution.(non-negative)

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

If we need to assign seats before the groups' arrival, we can select any planning supply and fix it.

If we don't need to assign seats immediately, we can wait until all demands are realized. During the process, we only need to decide whether to reject or accept each request.

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.

Fix the supply of this group size, and continue this procedure.

If groups arrive randomly, the procedure will be similar.

We don't care about the arrival sequence; only the number of groups matters. Because as long as the approximation about the number of groups is accurate, we can handle any sequence.

Three senarios:

1. The seat assignment will be arranged in advance, the groups only need to find the corresponding-size seats. This scenario applies for the short admission time. For example, the same film genre will attract the

same feature of different group types. The movie hall can assign the seats without changing to save costs in one day. (Theater, Concert, will not provide the seat positions)

2. The groups can only select which row they sit.

3. Online booking. (Can select arbitrarily but with some constraints.)

At first, the request should give the size of group, then give the possible row number it can sit. Finally give the possible first seat number.

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

The third step need to check whether the group destroies the assignment. Use subset sum problem to check every position in the row.

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

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

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

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

## 7.3 Different probabilities

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

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.

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

## 7.4 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.

## 7.5 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

## 7.6 How to use the stochastic demand to solve the dynamic situation?

[5] this paper connects the stochastic and dynamic VRP.

## 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] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

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

[3] Stephen Baum and LE Trotter, Jr. Integer rounding for polymatroid and branching optimization problems. *SIAM Journal on Algebraic Discrete Methods*, 2(4):416–425, 1981.

[4] Stephen Baum and Leslie E Trotter. Finite checkability for integer rounding properties in combinatorial programming problems. *Mathematical Programming*, 22(1):141–147, 1982.

[5] Russell W Bent and Pascal Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987, 2004.

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

[7] Dominique de Werra. A decomposition property of polyhedra. *Mathematical programming*, 30(3):261–266, 1984.

[8] 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.

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

[10] 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.

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

[12] Constantine Goulimis. Optimal solutions for the cutting stock problem. *European Journal of Operational Research*, 44(2):197–208, 1990.

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

[14] Odile Marcotte. The cutting stock problem and integer rounding. *Mathematical Programming*, 33(1):82–92, 1985.

[15] 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.

[16] 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.

[17] Pamela H Vance. Branch-and-price algorithms for the one-dimensional cutting stock problem. *Computational optimization and applications*, 9(3):211–228, 1998.

[18] François Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, 86(3):565–594, 1999.

[19] François Vanderbeck and Laurence A Wolsey. An exact algorithm for ip column generation. *Operations research letters*, 19(4):151–159, 1996.

## 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). □

```
Run Start Time: Sun Jan  1 17:36:13 2023
Probability: [0.3, 0.5, 0.1, 0.1]

The number of periods:  70
M1: 99.17 ;M2: 99.97 ;M3: 99.96 ;M4: 99.96 ;M5: 97.43 ;M6: 99.96
;Number of accepted people: 140.30      Number of people: 140.40
The number of periods:  71
M1: 99.01 ;M2: 99.96 ;M3: 99.95 ;M4: 99.95 ;M5: 96.91 ;M6: 99.95
;Number of accepted people: 141.98      Number of people: 142.08
The number of periods:  72
M1: 98.72 ;M2: 99.99 ;M3: 99.89 ;M4: 99.89 ;M5: 96.18 ;M6: 99.99
;Number of accepted people: 143.92      Number of people: 143.96
The number of periods:  73
M1: 98.42 ;M2: 99.87 ;M3: 99.76 ;M4: 99.76 ;M5: 95.28 ;M6: 99.82
;Number of accepted people: 145.16      Number of people: 145.70
The number of periods:  74
M1: 98.66 ;M2: 99.84 ;M3: 99.67 ;M4: 99.67 ;M5: 94.49 ;M6: 99.75
;Number of accepted people: 145.08      Number of people: 145.72
The number of periods:  75
M1: 98.26 ;M2: 99.79 ;M3: 99.78 ;M4: 99.78 ;M5: 93.97 ;M6: 99.75
;Number of accepted people: 146.42      Number of people: 147.42
The number of periods:  76
M1: 97.35 ;M2: 99.67 ;M3: 99.46 ;M4: 99.46 ;M5: 92.63 ;M6: 99.46
;Number of accepted people: 148.96      Number of people: 151.16
The number of periods:  77
M1: 97.01 ;M2: 99.54 ;M3: 99.24 ;M4: 99.24 ;M5: 92.55 ;M6: 99.20
;Number of accepted people: 149.82      Number of people: 153.16
The number of periods:  78
M1: 96.92 ;M2: 99.48 ;M3: 98.87 ;M4: 98.87 ;M5: 91.47 ;M6: 98.90
;Number of accepted people: 151.30      Number of people: 156.28
The number of periods:  79
M1: 96.47 ;M2: 99.42 ;M3: 98.46 ;M4: 98.46 ;M5: 91.06 ;M6: 98.41
;Number of accepted people: 152.44      Number of people: 159.30
The number of periods:  80
M1: 96.59 ;M2: 99.42 ;M3: 98.07 ;M4: 98.07 ;M5: 91.03 ;M6: 97.99
;Number of accepted people: 152.80      Number of people: 161.16
The number of periods:  81
M1: 96.33 ;M2: 99.48 ;M3: 98.01 ;M4: 98.01 ;M5: 91.03 ;M6: 97.92
;Number of accepted people: 152.76      Number of people: 162.22
The number of periods:  82
M1: 96.16 ;M2: 99.35 ;M3: 97.55 ;M4: 97.55 ;M5: 90.50 ;M6: 97.49
;Number of accepted people: 153.80      Number of people: 165.32
The number of periods:  83
M1: 95.87 ;M2: 99.26 ;M3: 97.34 ;M4: 97.34 ;M5: 89.84 ;M6: 97.29
;Number of accepted people: 153.98      Number of people: 166.64
The number of periods:  84
M1: 95.78 ;M2: 99.44 ;M3: 97.38 ;M4: 97.38 ;M5: 90.15 ;M6: 97.32
;Number of accepted people: 153.24      Number of people: 166.20
The number of periods:  85
M1: 96.15 ;M2: 99.44 ;M3: 96.97 ;M4: 96.97 ;M5: 89.63 ;M6: 96.90
;Number of accepted people: 153.72      Number of people: 168.12
Total Runtime 693.916252
```

```
Run Start Time: Sun Jan  1 17:50:02 2023
Probability: [0.4, 0.3, 0.2, 0.1]

The number of periods:  70
M1: 99.69 ;M2: 100.00 ;M3: 100.00 ;M4: 100.00 ;M5: 98.29 ;M6: 100.00
;Number of accepted people: 138.56	Number of people: 138.56
The number of periods:  71
M1: 98.28 ;M2: 99.92 ;M3: 99.90 ;M4: 99.90 ;M5: 96.43 ;M6: 99.89
;Number of accepted people: 143.88	Number of people: 144.32
The number of periods:  72
M1: 98.61 ;M2: 99.95 ;M3: 99.95 ;M4: 99.95 ;M5: 96.48 ;M6: 99.95
;Number of accepted people: 142.66	Number of people: 142.78
The number of periods:  73
M1: 97.81 ;M2: 99.87 ;M3: 99.77 ;M4: 99.77 ;M5: 94.45 ;M6: 99.76
;Number of accepted people: 146.64	Number of people: 147.38
The number of periods:  74
M1: 97.12 ;M2: 99.72 ;M3: 99.52 ;M4: 99.52 ;M5: 93.64 ;M6: 99.59
;Number of accepted people: 148.14	Number of people: 149.78
The number of periods:  75
M1: 97.89 ;M2: 99.77 ;M3: 99.56 ;M4: 99.56 ;M5: 93.66 ;M6: 99.52
;Number of accepted people: 147.34	Number of people: 148.76
The number of periods:  76
M1: 96.71 ;M2: 99.53 ;M3: 99.02 ;M4: 99.02 ;M5: 91.80 ;M6: 98.98
;Number of accepted people: 151.38	Number of people: 155.20
The number of periods:  77
M1: 96.59 ;M2: 99.45 ;M3: 98.95 ;M4: 98.95 ;M5: 91.90 ;M6: 98.89
;Number of accepted people: 150.64	Number of people: 154.58
The number of periods:  78
M1: 96.46 ;M2: 99.41 ;M3: 98.64 ;M4: 98.65 ;M5: 91.64 ;M6: 98.56
;Number of accepted people: 152.10	Number of people: 158.20
The number of periods:  79
M1: 96.59 ;M2: 99.50 ;M3: 98.64 ;M4: 98.64 ;M5: 90.80 ;M6: 98.58
;Number of accepted people: 151.46	Number of people: 157.46
The number of periods:  80
M1: 96.37 ;M2: 99.27 ;M3: 98.43 ;M4: 98.43 ;M5: 91.22 ;M6: 98.30
;Number of accepted people: 151.92	Number of people: 159.30
Total Runtime 446.648286
```

```
Run Start Time: Sun Jan  1 18:12:40 2023
Probability: [0.25, 0.25, 0.25, 0.25]

The number of periods:  55
M1: 99.64 ;M2: 100.00 ;M3: 100.00 ;M4: 100.00 ;M5: 96.44 ;M6: 100.00
;Number of accepted people: 137.74      Number of people: 137.74
The number of periods:  56
M1: 99.21 ;M2: 99.88 ;M3: 99.88 ;M4: 99.88 ;M5: 96.18 ;M6: 99.88
;Number of accepted people: 138.06      Number of people: 138.12
The number of periods:  57
M1: 98.29 ;M2: 99.89 ;M3: 99.75 ;M4: 99.75 ;M5: 94.05 ;M6: 99.81
;Number of accepted people: 142.34      Number of people: 142.72
The number of periods:  58
M1: 97.85 ;M2: 99.82 ;M3: 99.64 ;M4: 99.64 ;M5: 92.71 ;M6: 99.77
;Number of accepted people: 143.96      Number of people: 144.88
The number of periods:  59
M1: 97.43 ;M2: 99.67 ;M3: 99.54 ;M4: 99.54 ;M5: 92.17 ;M6: 99.57
;Number of accepted people: 145.36      Number of people: 146.12
The number of periods:  60
M1: 97.46 ;M2: 99.71 ;M3: 99.54 ;M4: 99.54 ;M5: 90.77 ;M6: 99.47
;Number of accepted people: 147.24      Number of people: 148.46
The number of periods:  61
M1: 97.82 ;M2: 99.44 ;M3: 99.16 ;M4: 99.16 ;M5: 90.79 ;M6: 99.26
;Number of accepted people: 147.50      Number of people: 149.40
The number of periods:  62
M1: 96.79 ;M2: 99.26 ;M3: 98.73 ;M4: 98.73 ;M5: 89.12 ;M6: 98.76
;Number of accepted people: 150.00      Number of people: 153.62
The number of periods:  63
M1: 95.97 ;M2: 99.04 ;M3: 98.32 ;M4: 98.32 ;M5: 89.09 ;M6: 98.16
;Number of accepted people: 151.76      Number of people: 157.72
The number of periods:  64
M1: 96.51 ;M2: 98.91 ;M3: 97.97 ;M4: 97.98 ;M5: 88.14 ;M6: 97.86
;Number of accepted people: 152.78      Number of people: 160.28
The number of periods:  65
M1: 95.82 ;M2: 99.06 ;M3: 97.54 ;M4: 97.55 ;M5: 87.64 ;M6: 97.29
;Number of accepted people: 153.74      Number of people: 163.80
Total Runtime 644.346994
```