Seat assignment with the social distancing

March 7, 2022

Abstract

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

1 Introduction

Social distancing, a physical way to contain the spread of an infectious disease, has been broadly recognized and practiced. As a result, extensive research has emerged on social distancing with respect to its effectiveness and impact. What lags behind 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 seat plan with a given set of seats. In a pandemic, 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 constraint, 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 introduce a brand new concept and consider ... Most literature on social distancing in seat assignments, highlight the importance of social distance in controlling the spread of the virus and focus the model too much, there is not much work on the operational significance behind the social distance [2] [6]. Recently, some scholars studied the effects of social distance on health and economics, mainly in aircrafts [11] [7]. Especially, Our study provides another perspective to help the authority adopt a mechanism setting seat assignments to control the spread of the virus.

Second, we establish the model to analyze the effects of ..., and we ... However, rather than .., ... Then, We develop the theorems to guide us to design effective algorithms to solve this problem. Next we consider the dynamic form.

Third, as a generalization, we apply this method on ... and show the similar conclusions. With this new .., we illustrate how to assign the seats by the govenment/stakeholder to balance health and economic issues. In addition, we also provide managerial guidance for the government on how to publish the related policy to make the tradeoff between economic maintenance and risk management.

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 concludsion that IRU holds for a given A if and only if IRU holds for C(A,w) for every fractional solution.

[5] 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 $s^1, \dots, s^r \in P(A, d, e)Z^n$ 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.

[10] 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.

[13] 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.

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

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

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

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

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

[12] 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, $(d_1, d_2, d_3, d_5, d_6) = (3, 5, 7, 10, 6)$, 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:

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

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

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

Then the corresponding sub-problem is:

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

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.

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

$y_k = 1$ 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.

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

Similarly, we consider the linear relaxation of the master problem and the optimal dual variable vector λ, μ. Using λ as the value assigned to the first constraint (2) and μ 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:

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

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.

4.3 Branch And Price

Rather than solving IP directly to obtain the optimal integer solution, the commonly used method is to branch the fractional solution.

General branch is commonly as follows: $\sum_{k\in K(k^j)} x_k = \alpha$, where $K(p) = \{k \in K : k \ge p\}$ (column subsets) for $p \in Z_+^m$, α is fractional. $K = \{k \in N^m : \sum_{i=1}^m s_i k_i \le S\}$ indicate all feasible patterns, and x_k is the number of times pattern k selected. Given a feasible solution x^* of LP that is not integral, take k^* to be any maximal element of the set $\{k \in K : x_k^* \notin Z_+\}$. Then the only fractional term in this sum $\sum_{k\in K(k^j)} x_k^*$ is $x_{k^*}^*$. Maximal means that the space left after cutting this pattern is less than the smallest size of the group.

Then the generic branching constraints will be: $\sum_{k\in K(k^j)} x_k \le U^j, \forall j \in G^u$ and $\sum_{k\in K(k^j)} x_k \ge L^j, \forall j \in H^u$, where G^u and H^u are sets of branching constraints of the form

$$\sum_{k\in K(k)} x_k \le \lfloor \alpha \rfloor \tag{5}$$

and

$$\sum_{k\in K(k)} x_k \geq \lceil \alpha \rceil \tag{6}$$

, respectively.

If we can always generate the maximal pattern, then any fractional column defines a branching set which contains only one member.

The corresponding restricted master problem will be reformulated as:

$$\begin{aligned}
\max \quad & \sum_{k\in K} (\sum_{i=1}^{m} (s_i - 1) t_i^k) x_k \\
\text{s.t.} \quad & \sum_{k\in K} x_k \leq N \\
& \sum_{k\in K} t_i^k x_k \leq g_i, \quad i = 1, \ldots, m \\
& \sum_{k\in K(k^j)} x_k \leq U^j, \forall j \in G^u \\
& \sum_{k\in K(k^j)} x_k \geq L^j, \forall j \in H^u \\
& x_k \geq 0, \quad k \in K
\end{aligned}$$

Let (π, λ, μ, v) be optimal dual variables associated with the added constraints. The pricing problem(subproblem) is:

$$\begin{aligned}
\max \quad & [(s_i - 1) - \lambda_i] y_i - \pi - \sum_{j\in G^u} \mu_j z^j - \sum_{j\in H^u} v_j z^j \\
\text{s.t.} \quad & \sum_{i=1}^{m} s_i y_i \leq S \\
& z^j = 1 \text{ if } y \geq k^j; z^j = 0 \text{ otherwise} \\
& y_i \geq 0, \text{integer} \quad \text{for } i = 1, \ldots, m.
\end{aligned}$$

$Y_k = \{(y, z) : z = 1, \text{ if } y \geq k, z = 0 \text{ otherwise}\}$ can be formulated as MIPs.

The pricing problem is only affected when an upper bound has been placed on the value of the variable associated with the selected pattern. Because this variable could be a nonbasic variable for the LP. In this case, we have avoid regenerating this maximal pattern. Thus, we should solve a knapsack problem with a forbidden pattern set. Besides, the drawback of this method will be that the branch is unbalanced.

The procedure of branch and price is as follows:

1) Solve the restricted master problem with the initial columns.

2) Use the pricing problem to generate columns.

3) When the column generation method terminates, is the solution integral?

Yes, update lower bound.

No, update upper bound. And fathom node or branch to create two nodes.

4) Select the next node until all nodes are explored.

4.4 Occupancy

For any pattern k, the occupancy is $\frac{S-l(k)}{S-1}$ and the largest occupancy is $\frac{S-l(k)}{S-1}, k \in I_1$. Thus, the largest occupancy for multiple rows is also $\frac{S-l(k)}{S-1}, k \in I_1$ when all rows are largest patterns.

Lemma 1. *Suppose the size of group types be $s = [2, 3, \ldots, u]$, the largest occupancy will be $\frac{S-q-\text{sign}(r)}{S-1}$.*

Suppose the size of group types be $s = [2, 3, \ldots, u]$, the maximal group size be u and $S = u \cdot q + r$. When $r = 0$, the minimal loss is q, the largest occupancy will be $\frac{S-q}{S-1}$. When $r > 0$, the minimal loss will be $q+1$, the largest occupancy will be $\frac{S-q-1}{S-1}$. Thus, the largest occupancy will be $\frac{S-q-\text{sign}(r)}{S-1}$, where $q = \lfloor \frac{S}{u} \rfloor$, $\text{sign}(r) = 1$ if $r > 0$, $\text{sign}(r) = 0$ if $r = 0$.

If we want the largest occupancy is no more than $\frac{1}{2}$, we have $\frac{S-\lfloor \frac{S}{u} \rfloor - 1}{S-1} \leq \frac{1}{2} \Rightarrow \frac{S-1}{2} \leq \lfloor \frac{S}{u} \rfloor$. It is clear that when $u = 2$ this inequality holds. When $u = 3$, S should be no larger than 3. In other cases, this inequality doesnot hold.

When $uq \leq S \leq uq + (u-1)$, $\lfloor \frac{S}{u} \rfloor$ equals q and this ratio $\frac{S-q-1}{S-1}$ increases in S when q is fixed. Thus, when $S = uq + (u-1)$, the occupancy can reach maximum value, $\frac{S-q-1}{S-1}$.

When the social distancing is 2, the size of group types will be $s = [3, 4, \ldots, u]$, the occupancy will be $\frac{S-1-2*\lceil \frac{S}{u} \rceil}{S-1}$

4.5 The Property

In view of the complexity and uncertainty of branch scheme, we should analyze the property of this problem and use it to obtain a solution.

At first, we consider the types of pattern. For each pattern k, we use α_k, β_k to indicate the number of groups and the left space, respectively. Denote $(\alpha_k + \beta_k)$ as the loss for pattern k, $l(k)$.

Let I_1 be the set of patterns with the minimal loss. Then we call the patterns from I_1 are largest. And the pattern with zero left space is called full pattern. 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, $(0, 0, 4, 1), (0, 0, 0, 4), (0, 2, 0, 3)$, belongs to I_1. Notice that the pattern, $(0, 0, 0, 4)$, is not full because there is one left space.

Now consider this special case, $[2, 3, \ldots, u]$, the group sizes are consecutive integers starting from 2. 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 2. *If all patterns from 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. If the number of rows is small, this method is useful.

Corollary 1. *When the left updated demand can form a largest pattern, the optimal solution is the combination of patterns from I_1.*

For example, when given the demand $d = (10, 11, 12, 10)$ and three rows. By column generation, we will obtain the solution $2.333 \times (0, 0, 0, 4), 0.667 \times (0, 0, 4, 1)$. But we can construct an integral solution $2 \times (0, 0, 0, 4), 1 \times (0, 1, 2, 2)$ or $3 \times (0, 0, 4, 1)$, that depends on which pattern we choose at the beginning.

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 theorem:

Theorem 1. *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 Theorem 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 $(0, 0, 0, 4), (0, 0, 4, 1), (0, 1, 2, 2)$ 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 $(0, 1, 2, 2)$ five times as the optimal solution.

When column generation method gives an integer solution at the first time, we obtain the optimal solution immediately. Now suppose that we have a fractional solution. Divide the solution into a pure integral part and the fractional part. The fractional solution will have a corresponding integral supply occupying the same space size. When the rest groups can be placed in the rest rows(given rows minus the integral rows), then the total groups can be placed in the given rows.

Based on the above analysis, we can establish an algorithm below.

Algorithm 1 The algorithm to construct an optimal solution

Step 1. After the column generation gives the fractional LP solution $x^{'}$, calculate the supply quantity $q^{'} = (q_1, \ldots, q_m)$ for each group type. If each element is integral, keep it. If any element of this supply is not integral, we can construct an integer supply vector which can provide the largest integral profit.

Step 2. Construction: calculate the space occupied by fractional supply, increase the corresponding supply having the same space, delete the fractional part. If the space occupied by fractional supply is fraction, find the supply providing the same profit.

Step 3. Take this integral supply vector q as a new demand and obtain a new LP solution x^* with column generation. Divide x^* into a pure integral part x^I and fractional part x^F. Subtract the corresponding supply of the integral part x^I from the new demand q to obtain the rest groups($r = q - q^I$). N is the number of given rows. The number of rest rows equals to $N - \sum x^I$.

Step 4. Use subset sum problem to check if the rest groups can be placed in rest rows.

Step 5. If the rest cannot be placed, go to step 2 to construct a new supply.

Now we need to judge whether we need to tackle step 4 in some cases.

For the case $(2, 3, \ldots, u)$, if the rest space is 1, we can discard 1 or exchange 1 and 3 with two 2. Remember what we need to do is to construct a maxmimal integer supply.

Assume that the demand is large such that the given rows cannot accommodate it and the number of group containing 2 people is large.

When the demand is large, i.e. supply cannot cover the whole demand. The equation $\sum_{k=1}^{K} x_k \leq N$ will be always valid. The sum of all elements of the solution vector always equals to the given number of rows.

Then the rest space for each maximal pattern will be no more than 1 with the extra groups with the size of 2.

Theorem 2. *For the case* $[2, 3, \ldots, u]$*, there exist patterns to contain the rest groups.*

We can use the induction to construct the pattern. When the sum of all elements of the fractional solution is 1, it is clear that the rest group will form a pattern because the summation of space occupied by the rest group will be no more than the length of row. When the sum is 2, suppose the rest groups cannot form a full pattern, the maximal pattern will have a left space. So there will be two rows with one unoccupied space, and we know in this situation there will be a left group with size of 1. When there are two adjacent numbers in the two patterns separately, we can change their position to accommodate these groups. As for the situation without adjacent numbers, it will be removed during the calculation of column generation. For example, we have 6 seats for a row. The rest groups are [2,1,1](number)* [2,3,5](size).

When the sum is 3, if we can form a full pattern for a row, this situation converts to the situation where the sum is 2. Then we suppose that the left group should have the size of 3 and the three rows all contains one occupied space. But this situation will contradict with the results of column generation method.

5 Dynamic Model

We also study the dynamic seating plan problem, which is more suitable for commercial use. In the problem, customers come dynamically and the seating plan needs to be made without knowing the number and composition of future customers. This becomes a sequential stochastic optimization problem where conventional methods fall into the curse of dimensionality due to the large number of seating plan combinations. To avoid this complexity, we develop an approach which aims directly to the final seating plans. Specifically, we define the concept of target seating plans which are deemed as 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 Nested Structure

Firstly, suppose that $1 + V_T(x) \geq V_T(x+1)$ holds. Then we have $2\lambda_2 \leq 1$, $3\lambda_3 \leq 1$, $\cdots$.

Use the inductive method to prove $1 + V_t(x) \geq V_t(x+1)$.

Suppose that $1 + V_t(s) \geq V_t(s-1)$, then we prove $1 + V_t(s+1) \geq V_t(s)$

$V_t(s) = \sum_i \lambda_i (\max\{i + V_{t+1}(s-i-1), V_{t+1}(s)\})$

$V_t(s+1) =$

Then we can develop the Theorem 3.

Theorem 3.

6 Results

To construct the whole diagram of ..., we need to obtain ...

As stated previously, ...

As we all know, ... Now, the question is how to solve the .... problem mentioned above efficiently.

Lemma 3.

Now we know the problem can be solved in ... and the ...

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

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

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

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

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

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

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

[12] 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.

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

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

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

Proof

Proof 1 (Theorem 1). □

Proof 2 (Lemma 1). □

Proof 3 (Lemma 2). □

Proof 4 (Theorem 2). □

Proof 5 (Theorem 2). □

Proof 6 (Theorem 3). □

Proof 7 (Theorem 4). □

Proof 8 (Theorem 5). □

Proof 9 (Theorem 6). □

Proof 10 (Lemma 2). □

Proof 11 (Theorem 7). □

Proof 12 (Lemma 4). □