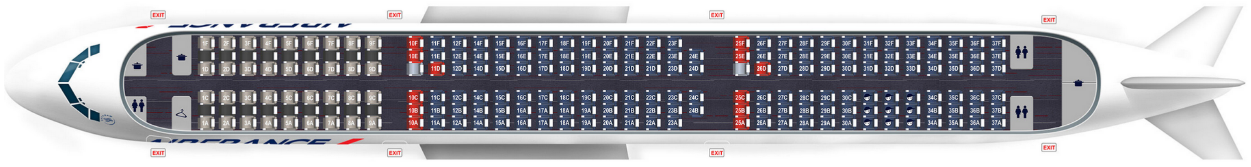
ST7 - Systemes de transport passagers

Optimizing passenger seating project (Air France)

https://github.com/DelphineSalzmann/Opti_AirFrance

Lucas Tramonte
Delphine Salzmann
Iman Tachrift
Kiyoshi Araki
Eléonore Belloni



March 2024

Contents

1 Introduction **3**

2 Overview of the data **3**

3 Static model **4**

3.1 Modeling of the plane . . . 4

3.2 Decision Variables . . . 4

3.3 Constraints . . . 4

3.3.1 Maximum one passenger per seat . . . 4

3.3.2 Each passenger gets exactly one seat . . . 5

3.3.3 The central aisle is left free . . . 5

3.3.4 Aircraft centering . . . 5

3.3.5 Passengers with reduced mobility . . . 6

3.4 Objective Functions . . . 7

3.4.1 Front seating for connecting passengers . . . 7

3.4.2 Allocate passengers to the same group . . . 8

3.5 Global objective . . . 9

3.6 Post Processing . . . 9

4 Solution evaluation **9**

4.1 Solution visualization . . . 9

4.1.1 Graphic visualization of the plane . . . 9

4.1.2 Barycenters . . . 16

4.2 Customer satisfaction . . . 17

4.2.1 An approach passenger by passenger . . . 17

4.2.2 A rigid approach considering different cases . . . 17

4.2.3 An general approach using a "bounding box" . . . 18

4.3 Impact of the running time on the satisfaction . . . 20

4.4 Customer satisfaction when the plane is full . . . 21

4.5 Customer satisfaction results for the given instances . . . 22

5 Dynamic model **25**

5.1 Principle . . . 25

5.2 Possible choices of seats for one group . . . 25

5.2.1 First approach by considering permutations of groups of same size . . . 25

5.2.2 Improvement of our offer by adding combinations of seats . . . 26

5.3 Filters in the choices offered to the group . . . 26

5.4 Group choices . . . 28

5.5 Cases of groups with WCHR passengers . . . 28

5.6 Case when passengers choose their seats in advance . . . 29

6 Sensitivity analysis **31**

6.1 Visualization . . . 31

6.2 Number of choices offered . . . 34

6.3 Comparison between the static and dynamic model . . . 36

7 Conclusion **38**

8 Algorithm Overview **39**

1 Introduction

For an airline such as Air France, providing a satisfactory travel experience to its customers is essential for maximizing their satisfaction. This includes the need to offer seats that meet passengers' needs, especially by providing seats that are close or adjacent if they are traveling in a group. However, safety standards impose strict constraints on passenger placement, particularly regarding aircraft balancing. Another important aspect is to minimize the time passengers spend on board the aircraft to avoid compromising their connections to other flights.

In this context, we address an optimization problem which aims at placing passengers on a commercial aircraft in a way that maximizes their satisfaction while respecting mandatory safety constraints. An additional objective is to solve this problem in minimal time, so that the method can be implemented under real conditions.

This study will unfold in two phases. First, we will develop a static model in which all passengers will be placed simultaneously, without offering them a choice of seat. Then, for future analysis, we will develop a dynamic model, more realistic, in which passengers will have the option to choose from different seats during check-in. This will ensure the feasibility of passenger placement in the cabin while maintaining an optimal layout compared to the static model.

2 Overview of the data

The analysis of flight instances from October 21st to November 7th reveals a diverse range of scenarios on the Paris to Nice route. During these dates, we observed instances ranging from planes with low and moderate occupancy on October 21st and October 22nd to a significant increase in passenger volume on October 23rd, attributed to a large group leading to a fully occupied aircraft. Furthermore, October 24th showcased a highly filled plane, emphasizing fluctuations in passenger demand. The subsequent instances on October 30th, November 5th, and November 7th focused on passengers with reduced mobility, highlighting a specific aspect of travel dynamics. This comprehensive analysis provides valuable insights into the varying factors influencing flight scenarios during this period, facilitating a more nuanced understanding of passenger demographics and their specific needs.

For each date entered, there is a dataset with the following data structure:

Table 1: Data Structure

Column	**Description**
Numéro du groupe	Group number associated with passengers
Femmes	Number of women in the group
Hommes	Number of men in the group
WCHR	Number of passengers using wheelchairs in the group
TransitTime	Transit time for each passenger in the group

For easier use, we create new dictionaries :

- Passengers : {passenger ID: {'gender': str, 'group': int , 'weight': int , 'connection_time': int}}

- group_dict() returns a dictionary that assigns to a group the list of the IDs of the passengers in that group : {group1 : [Passenger1 ID, Passenger2 ID, ...], group2 : [Passenger3 ID, Passenger4 ID, ...] }

3 Static model

Firstly, we considered the simplification that all passengers check in at the same time, which leads us to the modeling of the problem below.

3.1 Modeling of the plane

We choose to represent the seats in our plane as a list from seat 1 to seat n_s (with seat n_s the final seat). Moreover, we consider ranks with 7 seats, the 4th being a 'fantom seat' representing the central aisle.

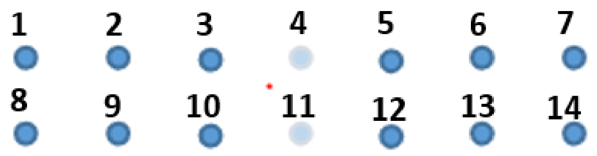


Figure 1: Seats modeling

We write a function seat_coordinates(seats) in order to access the position of a seat. The function takes a list of seats and returns a list of tuples corresponding to the coordinates in terms of rows and places in the plane. For instance, seat_coordinates([1, 54]) returns [(1,1), (5,8)]. Seat 54 is the 5th seat in the 8th row.

3.2 Decision Variables

We can see that the problem of allocating passengers on an airplane requires a binary variable S_{ij}, which associates each passenger i with a seat j.

$$S_{ij} = \begin{cases} 1 & \text{if Passenger i is in seat j} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

3.3 Constraints

3.3.1 Maximum one passenger per seat

To meet the restriction of having a maximum of one passenger per seat, we modeled the following equation:

$$\sum_{i=1}^{n_p} S_{ij} \leq 1 \quad \forall j \in [1, n_s] \tag{2}$$

where n_p is the number of passengers and n_s the number of seats.

3.3.2 Each passenger gets exactly one seat

To meet the restriction that every passenger has one and only one seat, we modeled the following equation:

$$\sum_{j=1}^{n_s} S_{ij} = 1 \quad \forall i \in [1, n_p] \tag{3}$$

3.3.3 The central aisle is left free

$$\forall i \in [1, n_p], \forall j \in [1, n_s] : \quad \text{if } (j\%7) = 4, \text{ then} \quad S_{ij} = 0 \tag{4}$$

3.3.4 Aircraft centering

To ensure safe takeoff and stable flight, the aircraft's center of gravity must lie within a very specific area known as the "centering zone". This is a critical safety constraint that must be strictly adhered to. Therefore, it is important that the plane has a balanced weight distribution around its center of mass, based on the definition of the barycenter shown in Figure 2.

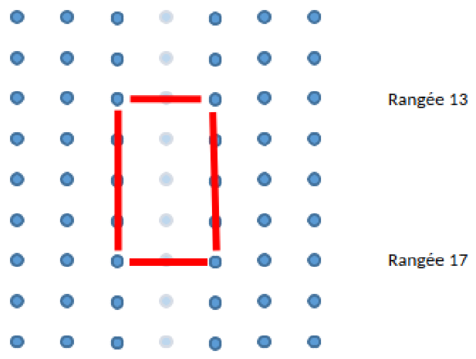


Figure 2: barycenter in the centering zone

In view of this, the following mathematical modeling was carried out:

$$x_g = \sum_{i=1}^{n_p} \sum_{j=1}^{n_s} \frac{(j\%7) S_{i,j} w_i}{\sum_{i=1}^{n_p} w_i} \tag{5}$$

$$y_g = \sum_{i=1}^{np} \sum_{j=1}^{ns} \frac{(j\%n_{rows}) S_{i,j} w_i}{\sum_{i=1}^{np} w_i} \tag{6}$$

With x_g and y_g the barycenters on x and y respectively. These barycenters are calculated with the function *calculate_ center_ of_ mass.*

If n_{rows}[1] is the number of rows in the plane, we then have to put these following constraints :

$$3 \leq x_g \leq 5 \tag{7}$$

$$\lceil n_{rows} \rceil // 2 - 2 \leq y_g \leq \lceil n_{rows} \rceil // 2 + 2 \tag{8}$$

For example if we take a plane of 29 rows, the constraints will be the following.

$$x_g = \sum_{i=1}^{n_p} \sum_{j=1}^{n_s} \frac{(j\%7) S_{i,j} w_i}{\sum_{i=1}^{n_p} w_i} \tag{9}$$

$$y_g = \sum_{i=1}^{n_p} \sum_{j=1}^{n_s} \frac{(j\%29) S_{i,j} w_i}{\sum_{i=1}^{n_p} w_i} \tag{10}$$

To fix the central zone between rows 13 and 17 and seats 3 and 7 :

$$3 \leq x_g \leq 5 \tag{11}$$

$$13 \leq y_g \leq 17 \tag{12}$$

3.3.5 Passengers with reduced mobility

The reduced mobility constraint is a crucial consideration in aircraft seating arrangements, ensuring that passengers with reduced mobility, such as wheelchair users, can safely and comfortably access the aircraft and its facilities.

These constraints involve allocating sufficient space for wheelchair maneuverability, including aisle width and seat arrangement, to facilitate easy boarding, movement within the cabin, and access to lavatories. Moreover, compliance with regulations and standards regarding wheelchair-accessible facilities is paramount to accommodate passengers with disabilities and ensure a seamless travel experience for all individuals aboard the aircraft. Thus, incorporating and adhering to these space arrangement constraints is fundamental in promoting inclusivity and accessibility in air travel.

In this modeling we consider that a passenger in wheelchair occupies 4 seats on the central aisle. Because of equation (3), we arbitrary decide to allocate to him a seat j in the 2^{nd} or 6^{th} column. Then we condemn the seat next to him and the two seats behind :

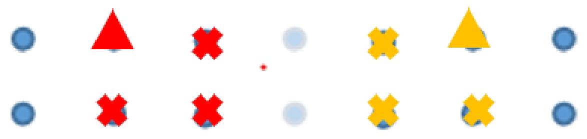


Figure 3: Representation of the two possible placements of a passenger in wheelchair

[1]This is a good way to generalize the problem according to the number of rows in the plane. However, as this parameter was only changed for November 7th, only an if-else condition was implemented in the code, instead of leaving it parameterized by the number of rows.

Let $x1, x2, x3, x4, x5, x6, x7$ be the seats for a given rank. Let i be a passenger in wheelchair. We impose:

$$S_{ix_1} + S_{ix_3} + S_{ix_5} + S_{ix_7} = 0 \tag{13}$$

Passenger i can only seat in seat x_2 or x_6. If passenger i seats in seat x_2, then we forbid seat x_3 to be taken by anyone. This can be written :

$$S_{ix_2} + \sum_{j=1}^{n} S_{jx_3} \leq 1 \tag{14}$$

Indeed, if passenger i seats in seat x_2, then $S_{ix_2} = 1$ and $\sum_{j=1}^{n} S_{jx_3}$ must be equal to 0 to respect the constraint. If not, one of the S_{jx_3} can be equal to 1. Similarly :

$$S_{ix_2} + \sum_{j=1}^{n} S_{j(x_2+7)} \leq 1 \tag{15}$$

and

$$S_{ix_2} + \sum_{j=1}^{n} S_{j(x_2+8)} \leq 1 \tag{16}$$

We proceed in the same way if passenger i seats on seat x_6.

3.4 Objective Functions

3.4.1 Front seating for connecting passengers

The objective function defined in the static model aims to optimize the placement of transit passengers at the front of the aircraft, thereby minimizing their connection time to ensure they can catch their next flight on time. To achieve this, the function assigns a value to each seat based on the associated passenger's connection time and their position in the aircraft. The shorter the connection time, the higher the value assigned to the seat, thereby favoring the placement of transit passengers in strategic seats at the front of the aircraft. By minimizing this objective function, the model seeks to maximize the efficiency of transit passenger connections while adhering to safety and customer satisfaction constraints.

$$f = \sum_{k=1}^{n_t} \sum_{j=1}^{n_s} S_{k,j} \times \frac{1}{T_k} \times q \tag{17}$$

where:

- f represents the objective function.
- n_t is the total number of transit passengers.
- n_s is the total number of seats.
- T_k represents the connection time of transit passenger k.

- q is the row associated with the seat's position in the aircraft. We start with row 0. It is defined as:

$$q = (j-1)//7 \tag{18}$$

Thus, the lower the transit time of a passenger is, the bigger the value $\frac{1}{T_k}$ is and the smaller q is forced to be. The passenger will seat in the front rows.

3.4.2 Allocate passengers to the same group

In order to satisfy the passengers, we want the groups to stay together. We choose to define an objective function to do so : if two passengers are in the same group, we calculate the distance between them as the sum of their distance along x and their distance along y, ponderated to give a bigger importance to the distance along y (we want passengers to be in the same row). The axis x and y are represented Figure 4:

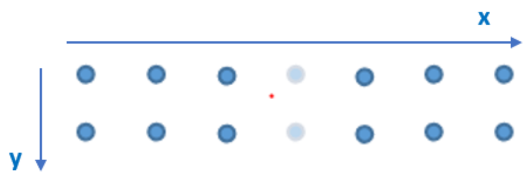


Figure 4: Representation of the axis used to calculate the distance between two seats

Thus we want :

$$minimize \sum_{i=1}^{n} \sum_{j \in group(i)} \alpha \times d_{x_i j} + (1-\alpha) \times d_{y_i j} \tag{19}$$

with $\alpha \leq 0.5$ in order to minimize in priority the distance along y.

However, distances are hiding an absolute value. We have to linearize our function for Gurobi to solve it. We introduce new variables X_i and Y_i representing the distance between i and the members of his/her group. We add the the constraints :

$$X_i \geq d_{x_i j} \tag{20}$$

$$X_i \geq -d_{x_i j} \tag{21}$$

$$Y_i \geq d_{y_i j} \tag{22}$$

$$Y_i \geq -d_{y_i j} \tag{23}$$

Then we resolve the linear problem :

$$minimize \sum_{i=1}^{n} \alpha \times X_i + (1-\alpha) \times Y_i \tag{24}$$

At this point we can see that the code is running for a long time. A solution to have a correct answer without taking to much time is to set a time limit :

```
m.Params.timeLimit = 120
```

Listing 1: Time limit

We will analyse next the influence of time on the satisfaction of the passengers.

3.5 Global objective

We can next resolve our linear problem for the global objective function :

$$minimize \sum_{i=1}^{n} \alpha \times X_i + (1-\alpha) \times Y_i + \sum_{k=1}^{n_t} \sum_{j=1}^{n_s} S_{k,j} \times \frac{1}{T_k} \times q \tag{25}$$

Results will be displayed and analysed below.

3.6 Post Processing

In our endeavor to optimize passenger seating arrangements within an aircraft, we initially set a cap of two minutes for our optimization algorithm to run. Surprisingly, we observed an increase in overall passenger satisfaction simply by swapping the positions of two passengers. Recognizing this potential for quick improvement, we ventured into developing a post-processing function that involved testing individual passenger swaps.

However, this approach proved to be time-consuming due to the exhaustive nature of testing each possible permutation. Despite the temptation to reduce complexity by limiting exchanges to nearby passengers, we made the strategic decision to extend the optimization algorithm's runtime instead.

4 Solution evaluation

4.1 Solution visualization

4.1.1 Graphic visualization of the plane

In order to visualize the solution implemented, the following figures were constructed, with the dots of the same color representing the passengers belonging to the same group. For November 7th, we need a plane with more rows to allocate the high number of passengers. For October 24th, in order to achieve a higher level of satisfaction, we've set the time limit to 8 minutes.

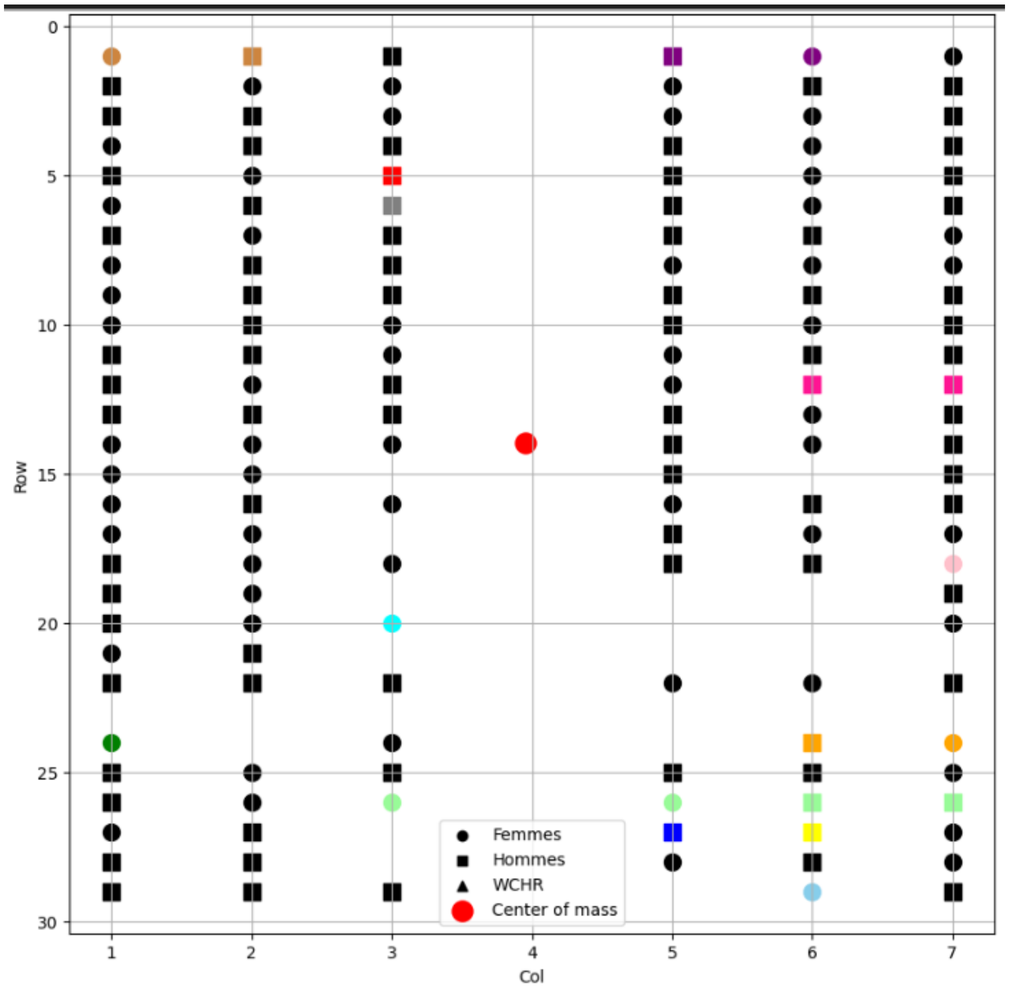


Figure 5: Passenger allocation for October 21st

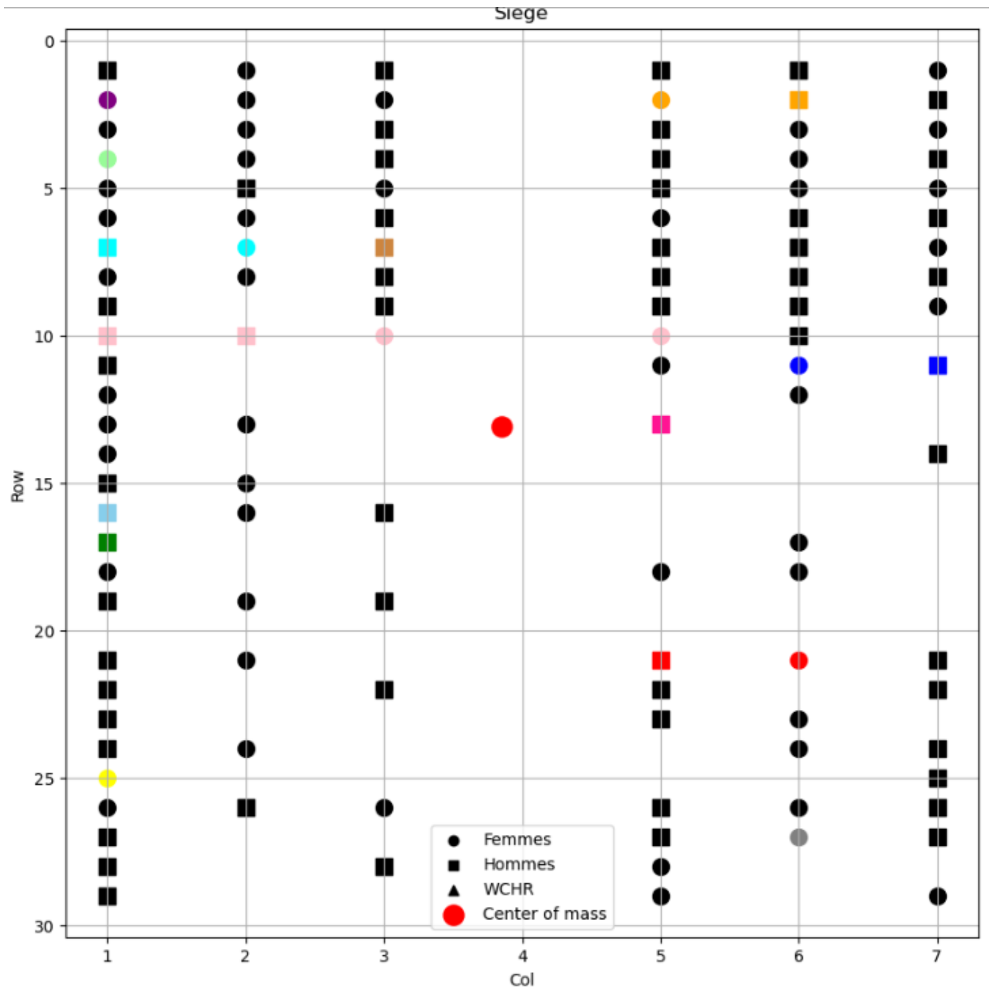


Figure 6: Passenger allocation for October 22nd

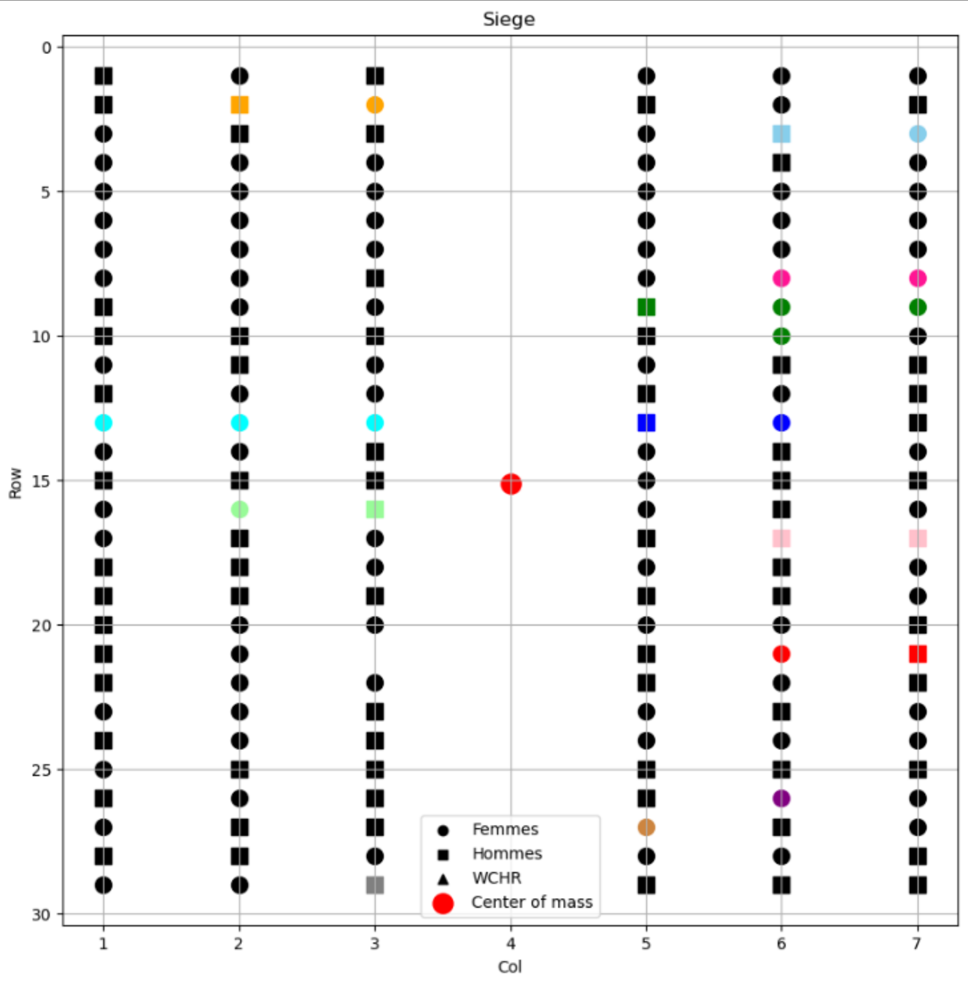


Figure 7: Passenger allocation for October 23rd

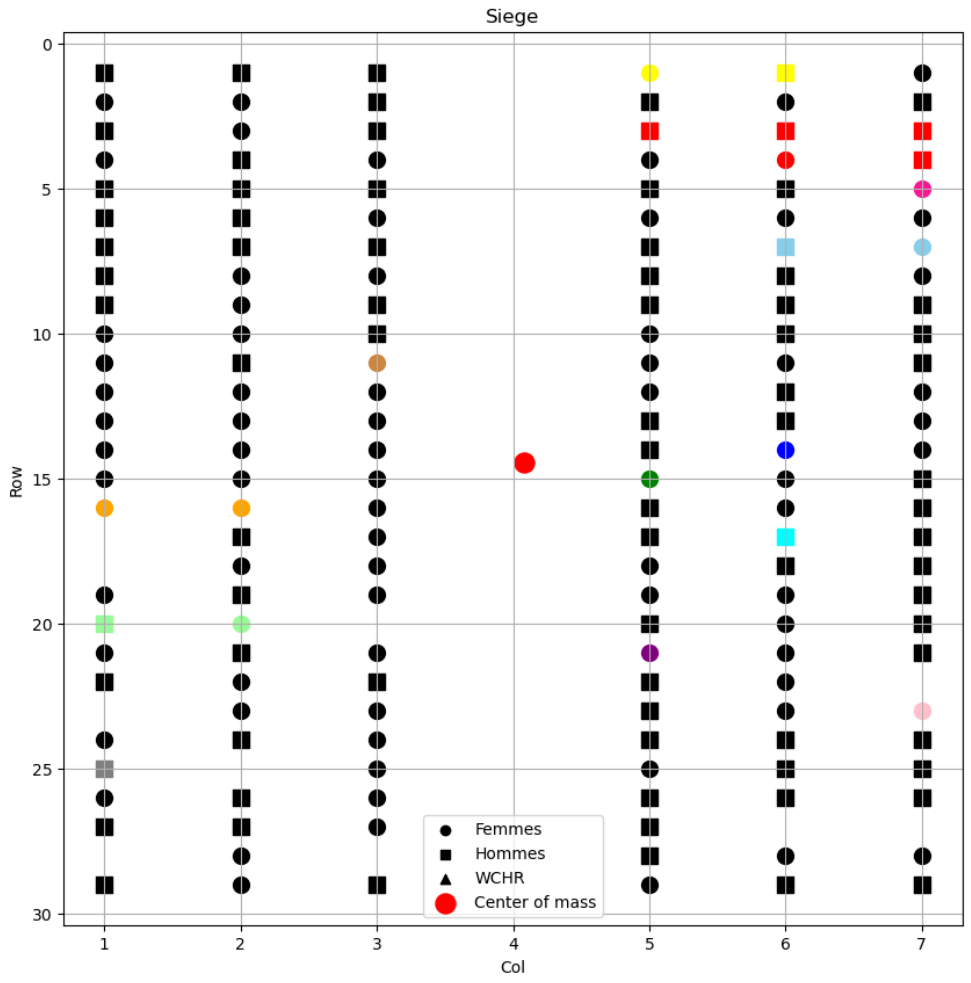


Figure 8: Passenger allocation for October 24th

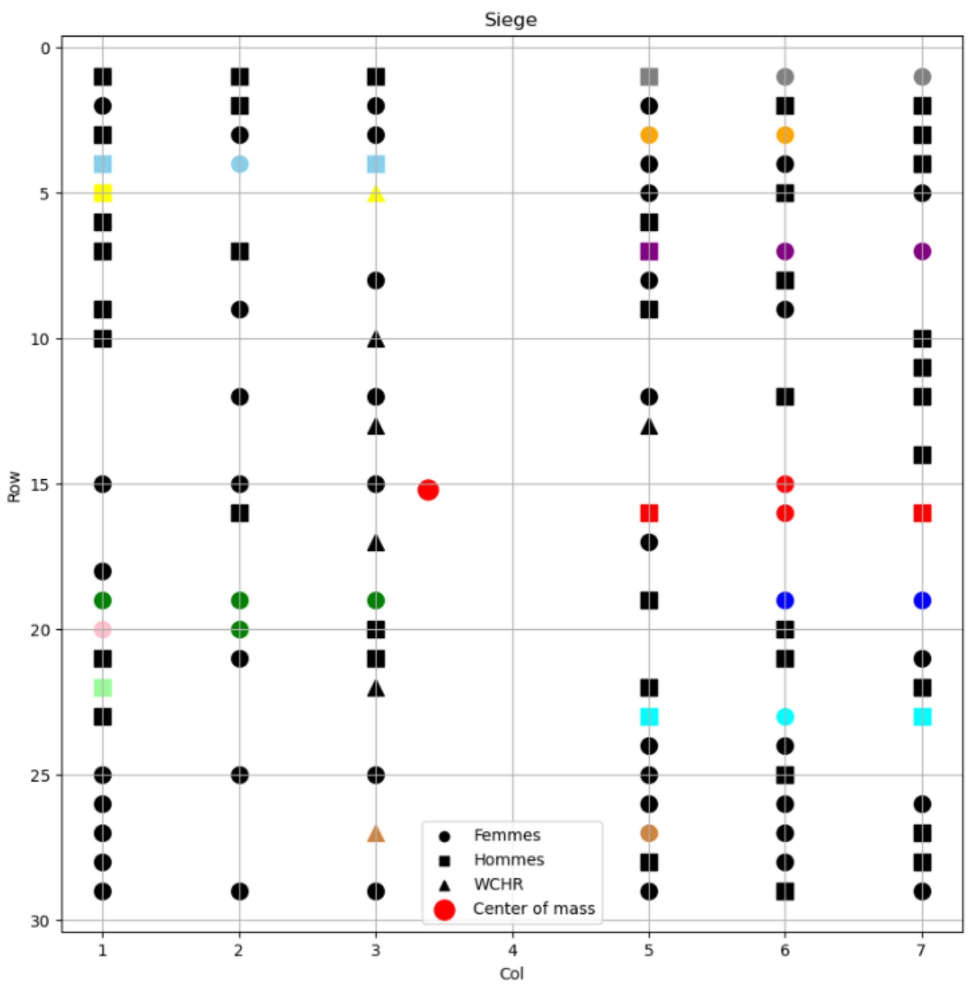


Figure 9: Passenger allocation for October 30th

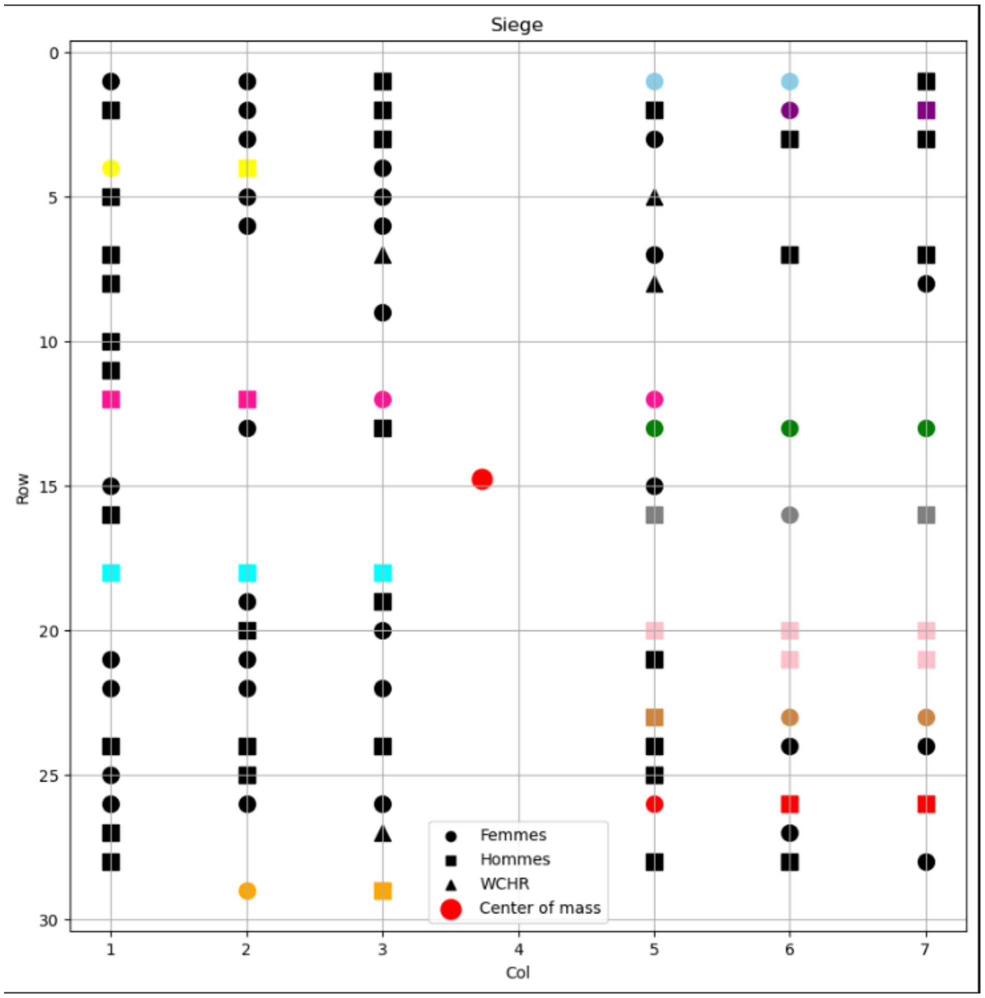


Figure 10: Passenger allocation for november 5th

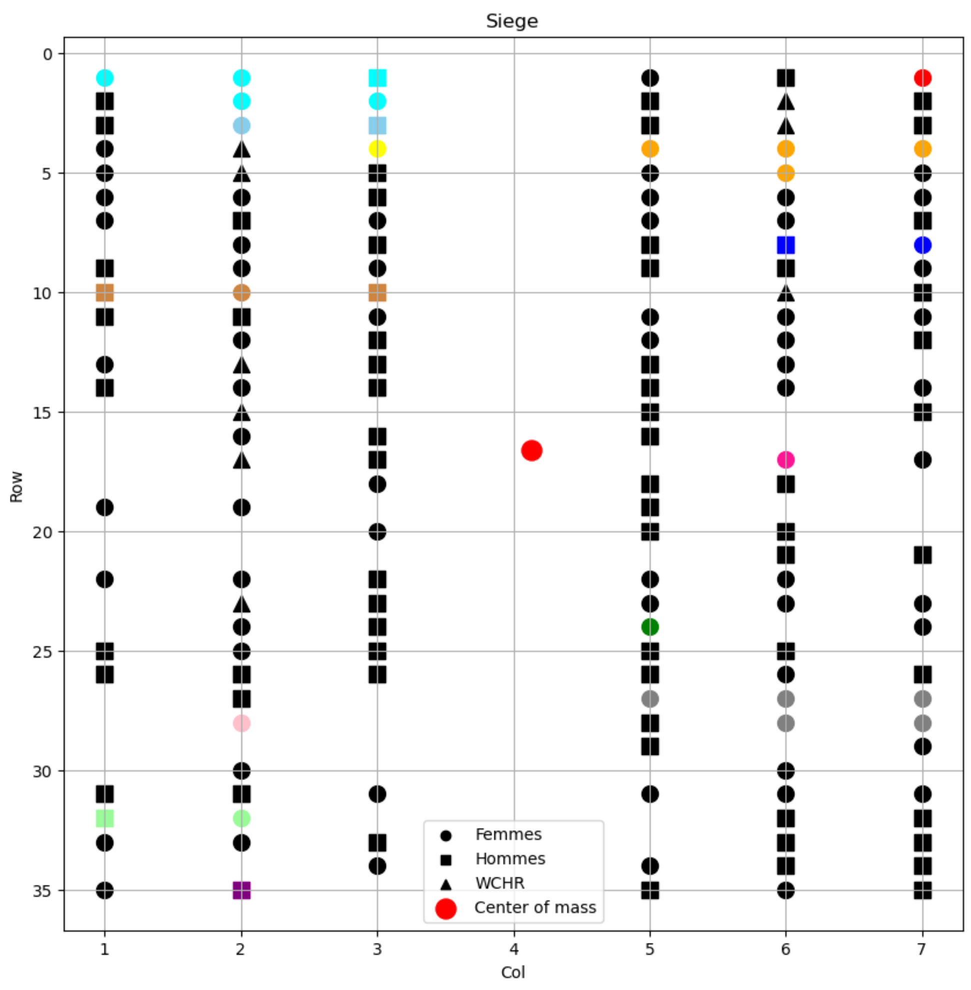


Figure 11: Passenger allocation for november 7th

As a fisrt approach, we can see that groups are staying quite well together. Moreover we manually check if passengers in transit are seating in the front of the plane. We will next define functions to evaluate more precisely the satisfaction of the clients.

4.1.2 Barycenters

On the figures above, we can see that the constraints on the barycenter is respected for each flight.

For example the barycenters for the flight on October 23rd are graphically perfectly centered, located in the central alley, at the 15th row.

For the flights on October 30th and November 5th, the barycenters are slightly deported from the center of the plane, but stay in the central alley.

4.2 Customer satisfaction

We will define different metrics in order to evaluate the satisfaction of the passengers. We get a score per passenger given over 1. Thus we only have to multiply by 100 to get a percentage.

4.2.1 An approach passenger by passenger

Fisrt we will consider a very simple approach verifying if a passenger in transit is seating in the front rows and if he/her is seating next to a member of his/her group. The obvious limit of this function is that it does not take into account the global allocation of the group. We are expecting high percentages of satisfaction. For the different instances we get :

Date	**Score**
Oct21	97.35
Oct22	97.41
Oct23	80.49
Oct24	95.12
Oct30	97.17
Nov5	99.08

4.2.2 A rigid approach considering different cases

The evaluation of the satisfaction of passengers having a given connection time remains similar to the one previously described. However we consider now a score per group.
We define the satisfaction as follows :

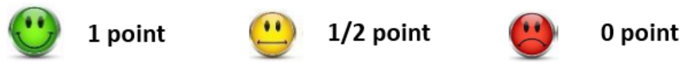


Figure 12: Points attribution

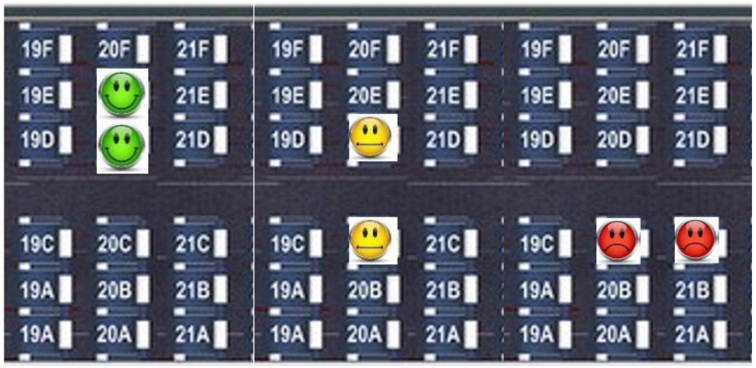


Figure 13: Group of 2

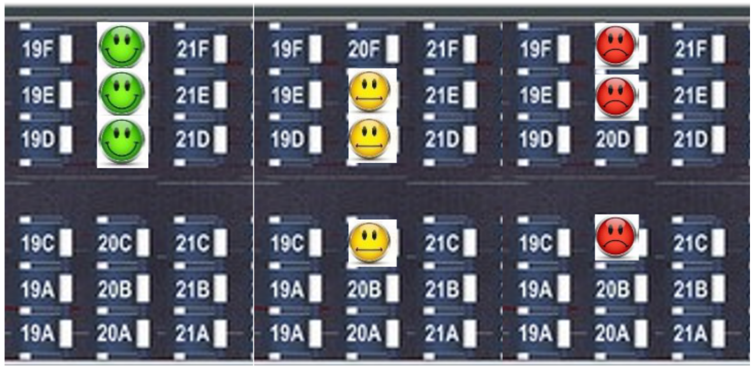

19F 21F 19F 20F 21F 19F 21F
19E 21E 19E 21E 19E 21E
19D 21D 19D 21D 19D 20D 21D
19C 20C 21C 19C 21C 19C 21C
19A 20B 21B 19A 20B 21B 19A 20B 21B
19A 20A 21A 19A 20A 21A 19A 20A 21A

Figure 14: Group of 3

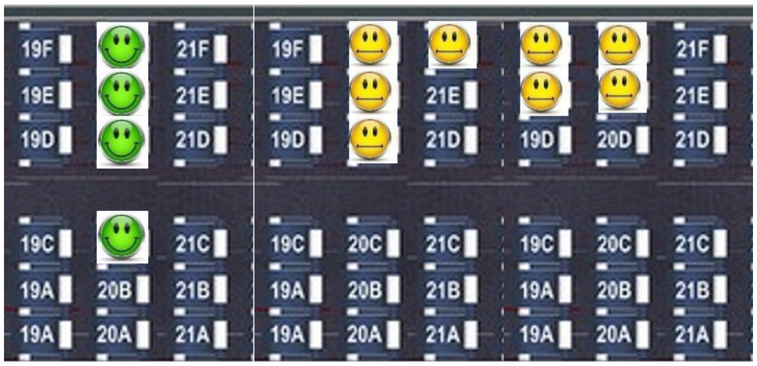

19F 21F 19F 21F
19E 21E 19E 21E 21E
19D 21D 19D 21D 19D 20D 21D
19C 21C 19C 20C 21C 19C 20C 21C
19A 20B 21B 19A 20B 21B 19A 20B 21B
19A 20A 21A 19A 20A 21A 19A 20A 21A

Figure 15: Group of 4

Groups with more than 4 people follow the same principle as groups of 4 : when are together in the same rank we give the maximum score (1 point), when their are sitting one behind another we give half a point, 0 point otherwise.

For the different instances we get :

Date	**Global Score**	**Transit Score**	**Group Score**
Oct21	98.68	100	97.35
Oct22	97.15	100	94.3
Oct23	83.53	73.4	96.7
Oct24	89,98	92,3	96,44
Oct30	93,04	100	93,67
Nov5	87,11	100	86,5

We can see that the scores are good, even excellent for the transit. However, like for the first evaluation method, the score is lower on Oct23. Indeed this instance has a group of 19 people in transit. They could not fit in the five first ranks because of their number.

4.2.3 An general approach using a "bounding box"

That method considers that passengers should be within a rectangle where proximity along the x-axis is prioritized over the y-axis. In other words, it evaluates whether they are close together, by checking if the rectangle bounded by the four vertices with the leftmost, rightmost, foremost, and rearmost passengers forms a smaller rectangle. This metric is more generalized, focusing less on the individual positions of passengers within the same group, and more on an overall score of the region they occupy.

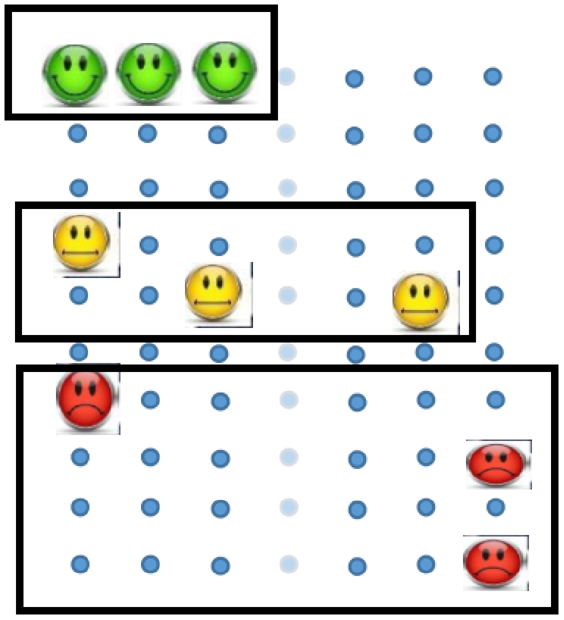


Figure 16: Client satisfaction depending on bounding box

$$X = x^i_{max} - x^i_{min} \tag{26}$$

$$Y = y^i_{max} - y^i_{min} \tag{27}$$

$$S(X, Y) = \frac{1}{\beta \cdot X + (1 - \beta) \cdot Y + 1} \qquad \beta = 0.95 \tag{28}$$

For each group of passengers, it calculates the minimum and maximum x and y coordinates to determine the bounding box around the group. It then calculates the x and y distances within this bounding box. If all passengers in a group are seated consecutively along the x-axis and have no y-axis displacement, each passenger in the group is given a satisfaction score of 1. Otherwise, it calculates a satisfaction score using the *ponderation function* based on the x and y distances. The total satisfaction score is updated based on these individual scores.

The mathematical formula employed to compute satisfaction scores incorporates this notion by utilizing a weighted sum of the distances along these axes, with a weighting parameter denoted by β. The resulting score, S, reflects how closely passengers are grouped together within the defined rectangle, with higher scores indicating greater satisfaction:

where x represents the distance along the x-axis and y represents the distance along the y-axis.

4.3 Impact of the running time on the satisfaction

We can first use our evaluation functions in order to fix the time limit of our code so that it does not take too much time and that we get a satisfying solution. We use the metrics defined in section 4.2.2. Results for two instances are given Figure 17 and 18:

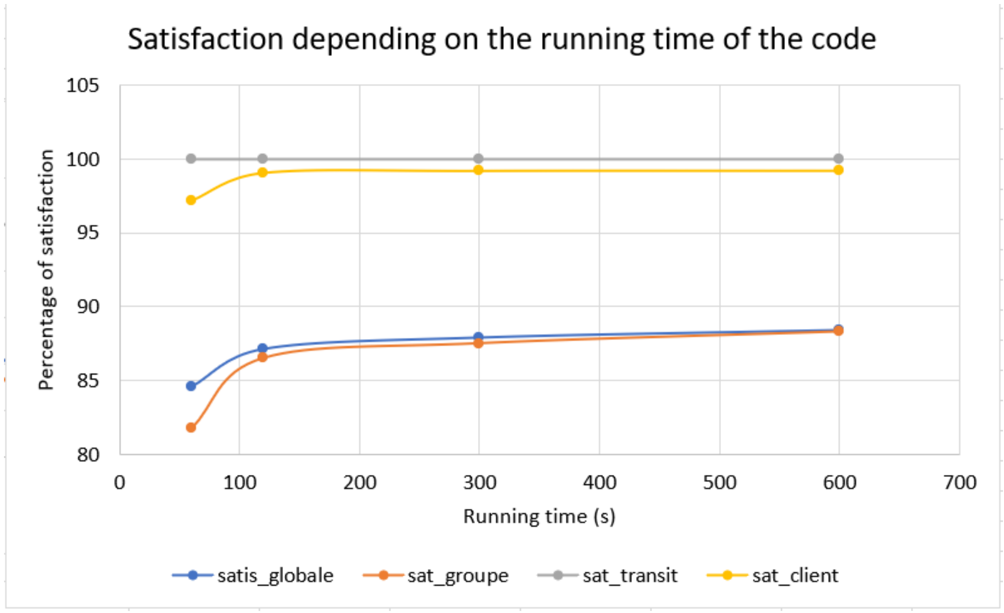


Figure 17: Client satisfaction for Nov5 depending on the time allocated to the code to run

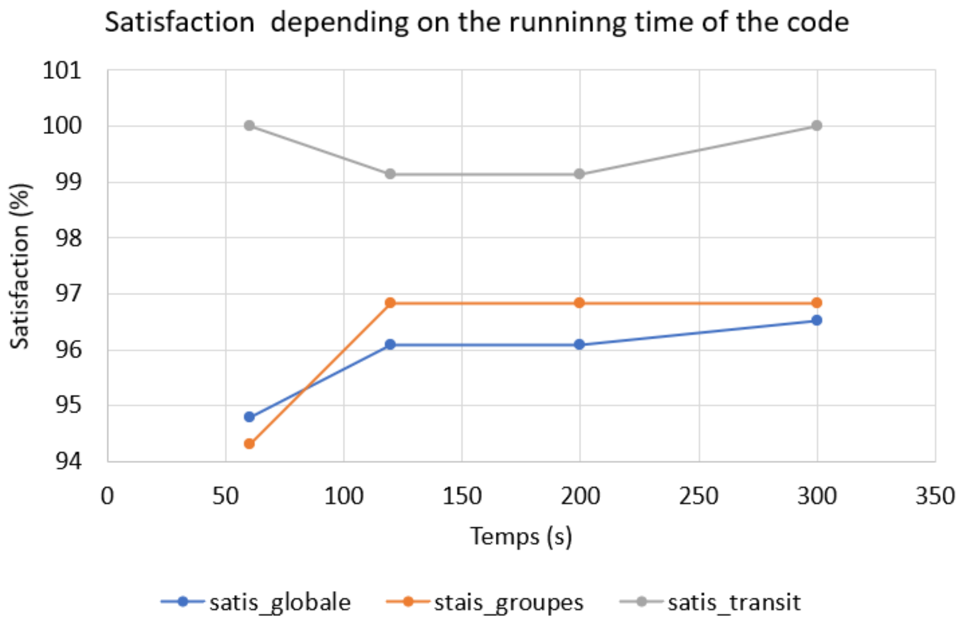


Figure 18: Client satisfaction for Oct30 depending on the time allocated to the code to run

We can see that there is no huge difference on the result between 120 s et 10 minutes. We can set our time limit to 2 minutes without losing information.

4.4 Customer satisfaction when the plane is full

In this section, we will conduct a sensitivity analysis to examine the influence of a fully occupied plane on the seating optimization solutions. This analysis will help us understand how the seating arrangements are affected when the aircraft reaches maximum capacity.

In examining the impact of seating occupancy on customer satisfaction, our analysis utilized an 'imaginary' dataset to simulate a scenario where the plane is fully occupied. This allowed us to assess the direct influence of seating availability on passenger contentment.

In this analysis, we utilized the "bounding box" metric as defined in section 4.2.3 to assign a binary value to each groupe: 1 indicating satisfaction and 0 indicating dissatisfaction. This is determined by if the passengers are seated next to a member of their group. Subsequently, we depicted the sum of these values in a graph to visualize overall satisfaction.

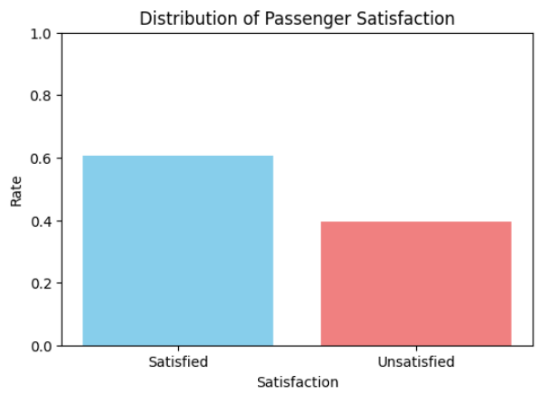


Figure 19: Client satisfaction in a full plane

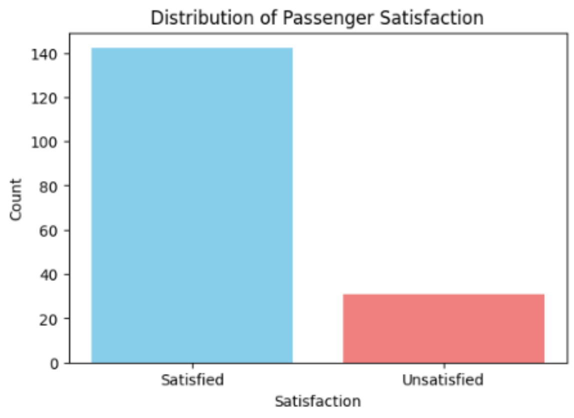


Figure 20: Client satisfaction in a semi full plane

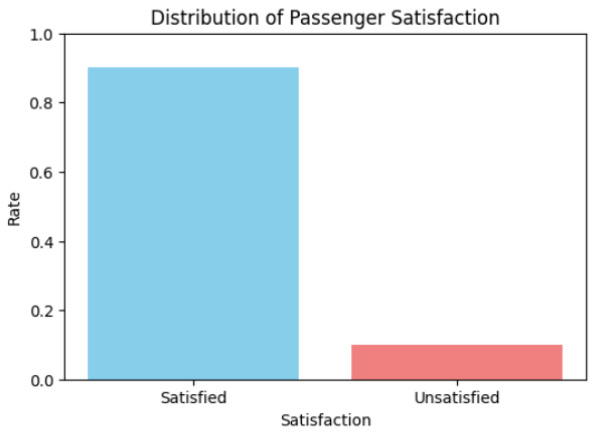


Figure 21: Client satisfaction in an empty-ish plane

As expected, the satisfaction decreases the fuller the plane gets. However, even in a fully occupied plane, the number of satisfied customers remains notable compared to unsatisfied ones.

4.5 Customer satisfaction results for the given instances

The boxplot, generated from the provided dataset, allows the comparison of satisfaction scores between groups of different sizes, helping to identify any disparities or trends in passenger satisfaction using the rectangle metrics.

Summarizing the rectangle metrics it evaluates passenger satisfaction within the same group, with them being in an area delimited by a rectangle. When the rectangle is smaller, priority is given to the x-axis, meaning that if they are arranged in the same row, it assigns a higher evaluation score than if they are lined up one behind the other, in other words, the distance in y.

The accompanying bar chart offers insights into the distribution of group sizes, providing an overview of how passengers are clustered within the aircraft. Finally, the pie chart complements these visuals by illustrating the relative proportions of each group size, aiding in understanding the composition of passenger groups onboard.

By integrating these visual elements directly into the code, analysts gain a more intuitive and interactive means of exploring and interpreting the data, enabling them to derive actionable insights to improve the passenger experience.

For this dataset from October 23, we have an overall score of 98%. With only groups of 1, 2, 3, and 4 passengers, all with average scores higher than 90%, and some groups achieving a perfect average, indicating a very good evaluation.

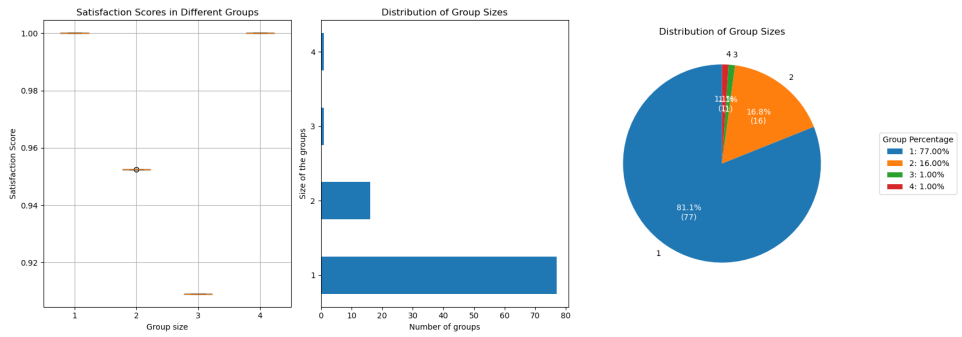


Figure 22: Group Satisfaction for October 23th

For this data set from October 24, we have an overall score of 75.45% and the individual evaluations for each group.

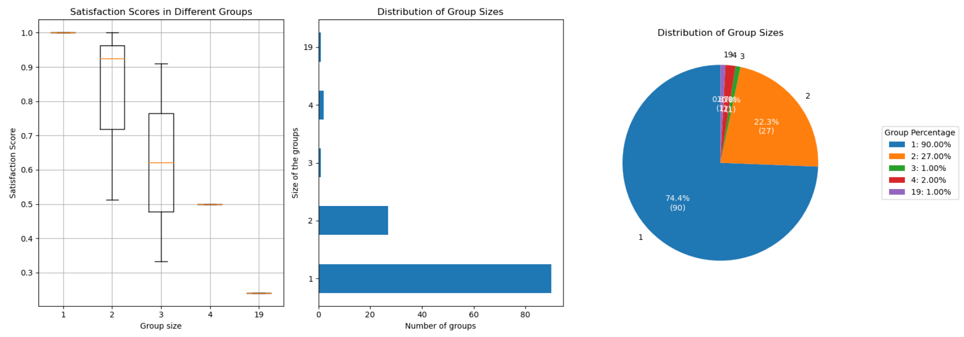


Figure 23: Group Satisfaction for October 24th

For this data set from October 30, we have an overall score of 92,71% and the individual evaluations for each group.

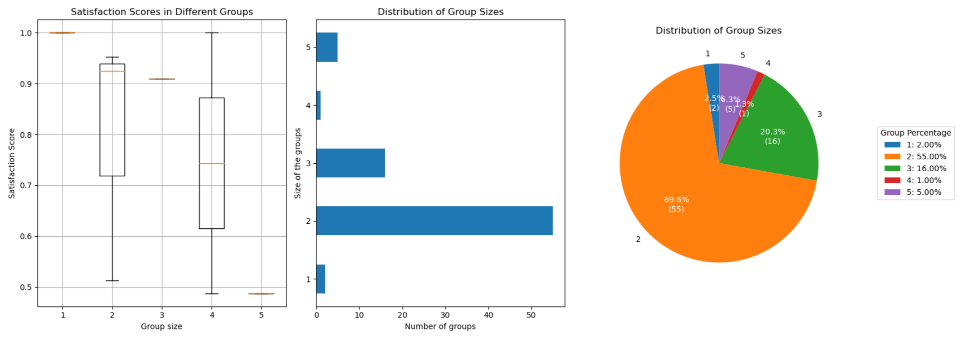


Figure 24: Group Satisfaction for October 30th

For this data set from November 5, we have an overall score of 87.81% and the individual evaluations for each group.

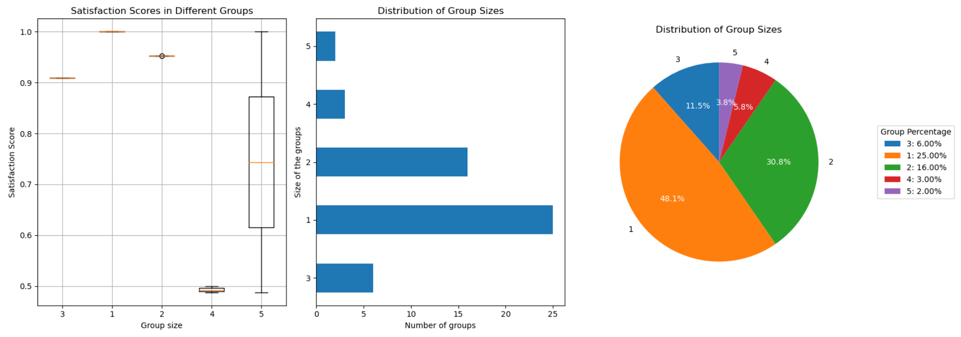


Figure 25: Group Satisfaction for November 5th

For this data set from November 7, we have an overall score of 89.45% and the individual evaluations for each group.

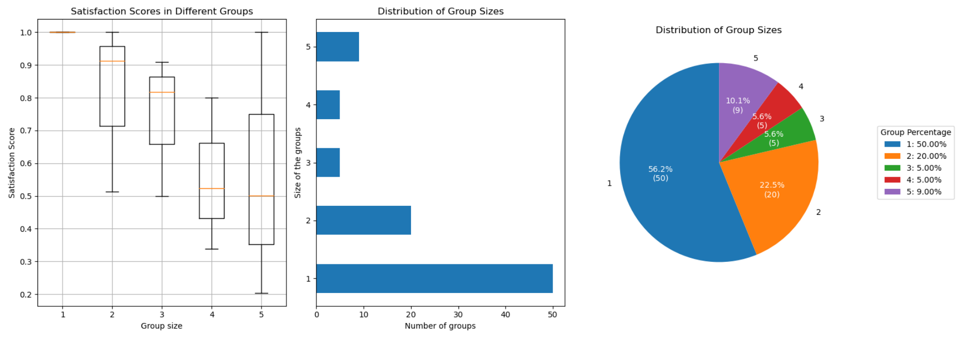


Figure 26: Group Satisfaction for November 7th

5 Dynamic model

5.1 Principle

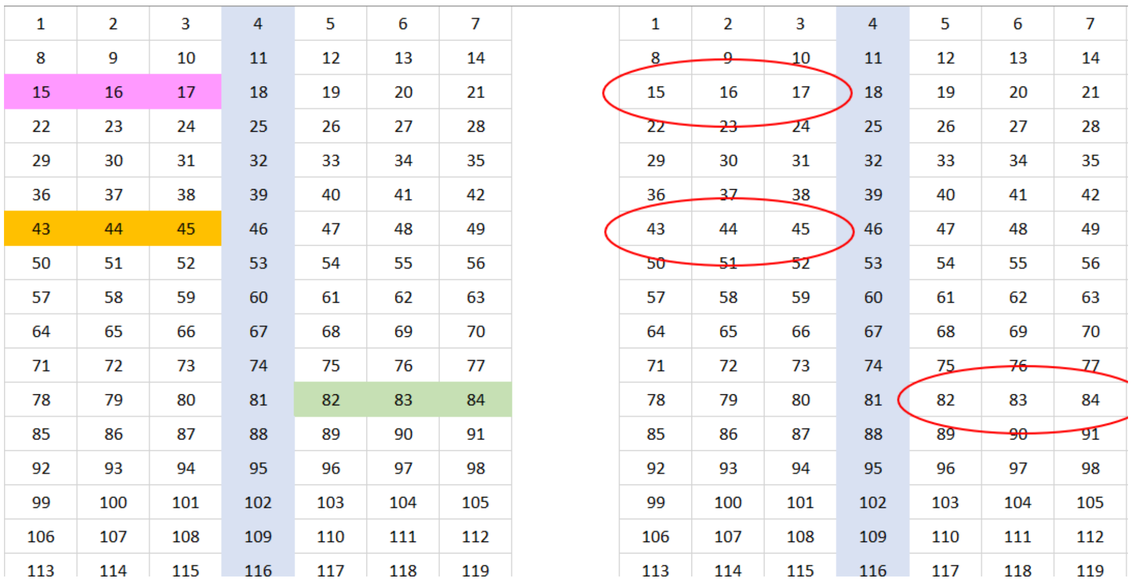
We now consider a dynamic process for customer registration, which we choose to base on our static model. Thus we begin with running the static model. On that base an offer of seats will be made to the groups, depending on their size and on their order during registration.
We will use :

- A dictionary temoin_S which is the control allocation given by the static model
- New variables S_dyn that are initialized with the results of the static model
- A dictionary Choices_dict we update everytime a new group has made a choice
- A dictionary satis_choice_dict where keys are the order of registration and values the lengh of the group and the number of choices available

5.2 Possible choices of seats for one group

5.2.1 First approach by considering permutations of groups of same size

We decide, in order to offer a choice to the different groups, to present to them seats corresponding to permutations ; we mean by that that a group of 3 will be offered the choice between the seats occupied by all groups of 3 in the static model.
The corresponding function is possible_choice (group_number) which returns a list of lists of seats.



1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49
50	51	52	53	54	55	56
57	58	59	60	61	62	63
64	65	66	67	68	69	70
71	72	73	74	75	76	77
78	79	80	81	82	83	84
85	86	87	88	89	90	91
92	93	94	95	96	97	98
99	100	101	102	103	104	105
106	107	108	109	110	111	112
113	114	115	116	117	118	119

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49
50	51	52	53	54	55	56
57	58	59	60	61	62	63
64	65	66	67	68	69	70
71	72	73	74	75	76	77
78	79	80	81	82	83	84
85	86	87	88	89	90	91
92	93	94	95	96	97	98
99	100	101	102	103	104	105
106	107	108	109	110	111	112
113	114	115	116	117	118	119

Figure 27: Right : places occupied by group of 3 in the static model. Left : Possibilities offered to a group of 3

For instance, in that case the function returns for groups of 3 [[15,16,17],[43,44,45],[82,83,84]].

5.2.2 Improvement of our offer by adding combinations of seats

However, when there are few groups of the same size, the offer is reduced. In order to be able to present more possibilities of choices, we decide to add to the list of possibilities the concatenations of groups for which the sum of their sizes is equal to the size of the group considered. Thus we don't cut groups.

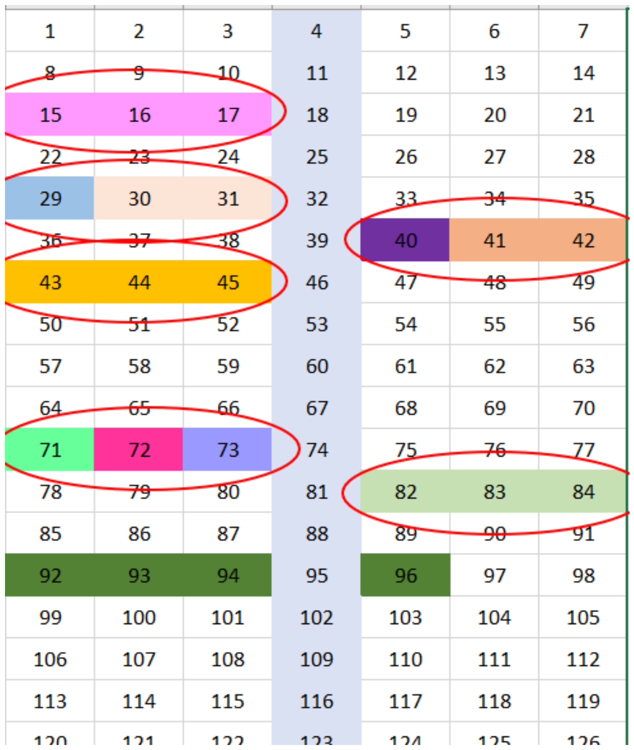


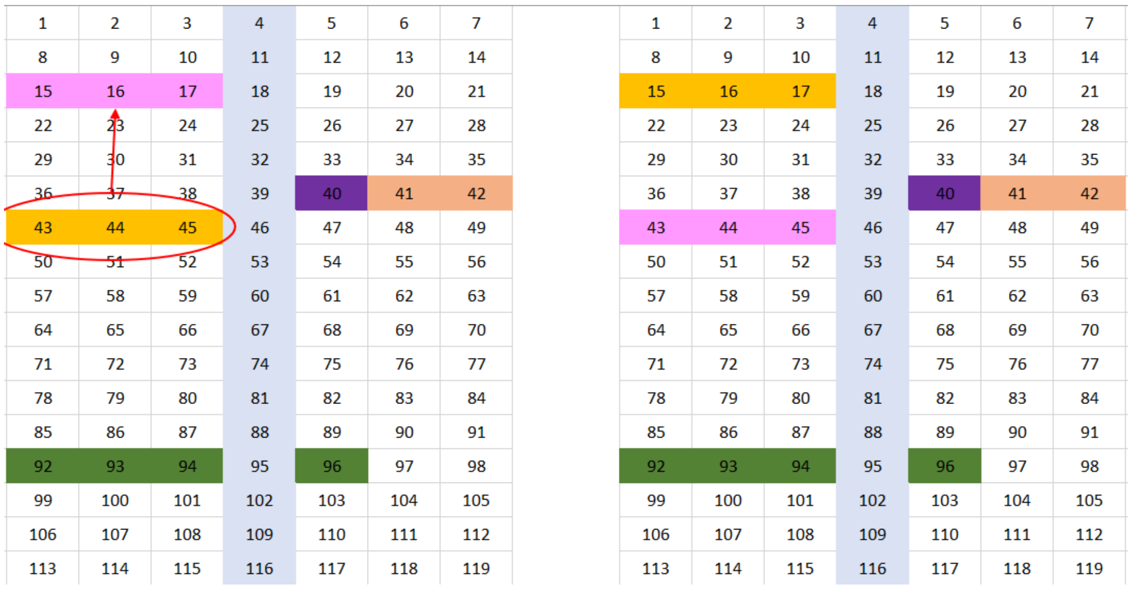
Figure 28: Possibilities offered to a group of 3

5.3 Filters in the choices offered to the group

Once we get the possible choices, we filter them in three ways :

- First, we check if a choice has not already been made by a previous group, i.e. if the choice appears in Choices_dict.
- Then we add a transit condition : a group which is not in transit can't be offered seats in the front rows. Otherwise the actual groups in transit won't be able to have these seats.
- Finally we verify that for every choice an allocation exists that satisfies the barycenter constraint.

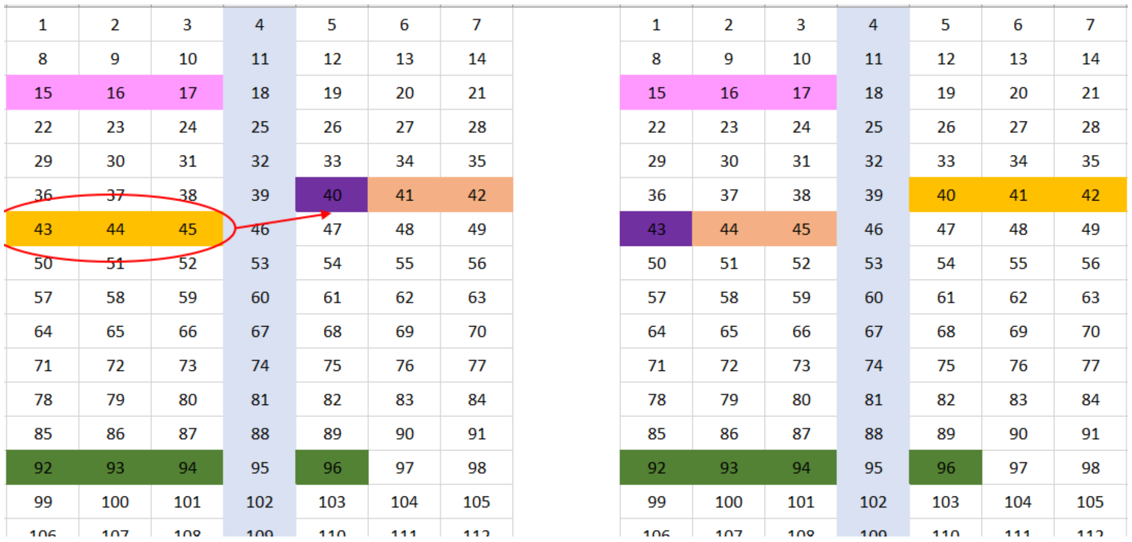
Figures 29 and 30 illustrate with an example the barycenter filter :



1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49
50	51	52	53	54	55	56
57	58	59	60	61	62	63
64	65	66	67	68	69	70
71	72	73	74	75	76	77
78	79	80	81	82	83	84
85	86	87	88	89	90	91
92	93	94	95	96	97	98
99	100	101	102	103	104	105
106	107	108	109	110	111	112
113	114	115	116	117	118	119

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49
50	51	52	53	54	55	56
57	58	59	60	61	62	63
64	65	66	67	68	69	70
71	72	73	74	75	76	77
78	79	80	81	82	83	84
85	86	87	88	89	90	91
92	93	94	95	96	97	98
99	100	101	102	103	104	105
106	107	108	109	110	111	112
113	114	115	116	117	118	119

Figure 29: Assumption 1: the group in yellow chooses the places previously occupied by the group in pink. They invert their places and the new barycenter is calculated.



1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49
50	51	52	53	54	55	56
57	58	59	60	61	62	63
64	65	66	67	68	69	70
71	72	73	74	75	76	77
78	79	80	81	82	83	84
85	86	87	88	89	90	91
92	93	94	95	96	97	98
99	100	101	102	103	104	105
106	107	108	109	110	111	112

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49
50	51	52	53	54	55	56
57	58	59	60	61	62	63
64	65	66	67	68	69	70
71	72	73	74	75	76	77
78	79	80	81	82	83	84
85	86	87	88	89	90	91
92	93	94	95	96	97	98
99	100	101	102	103	104	105
106	107	108	109	110	111	112

Figure 30: Assumption 2: the group in yellow chooses the places previously occupied by the groups in purple and beige. They invert their places and the new barycenter is calculated.

Concretely, we browse the list of possibilities and for each possibility we assume that it is the one chosen by the group. So we swap the passengers and calculate the new centre of mass. If it satisfies our constraint, then the option remains possible. If not, it is deleted from the list of possible choices.

We could have chosen to execute this verification later, only when the choices are ordered by score, in order not to have to check every possibility. The result of our code would have been the same. However, even if it would had saved some calculations, it would not have been realistic; indeed it doesn't make sense to erase a possibility after it has already been chosen by a group.

At the end, a group will be offered only places available that respect the transit condition and the barycenter constraint.

5.4 Group choices

The group can afterwards make a choice among the possibilities given to it.
At first, a score will be given to each possibility accordingly to the metrics defined in section 4.2.2. Then we define a function called group choices. It's a function that takes a list of choices with associated values as input. These choices are sorted based on their values in descending order using Python's sorted function with a lambda function as the sorting key. The sorted choices are then extracted, leaving only the choice names in the group choices list. This code facilitates the process of selecting and organizing group choices based on their associated values, aiding in the subsequent allocation of seats in accordance with the dynamic model's principles.

5.5 Cases of groups with WCHR passengers

Groups with WCHR passengers are treated separately. With the same logic as the one used for the other groups we offer them possibilities based on the result of the static model. However, we consider only groups of same size and with the same number of WCHR passengers.

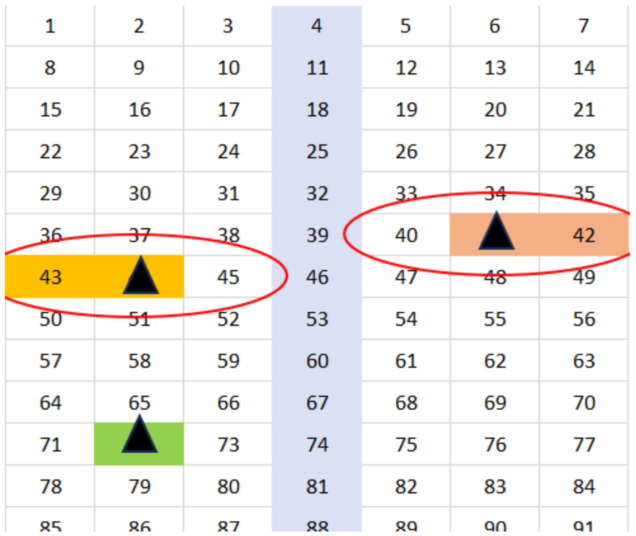


Figure 31: Possibilities for the yellow group

This modeling does not offer many options to these groups, however it is expected given their status. An improvement of this model would be to consider concatenating groups with the same number of WCHR passenger :

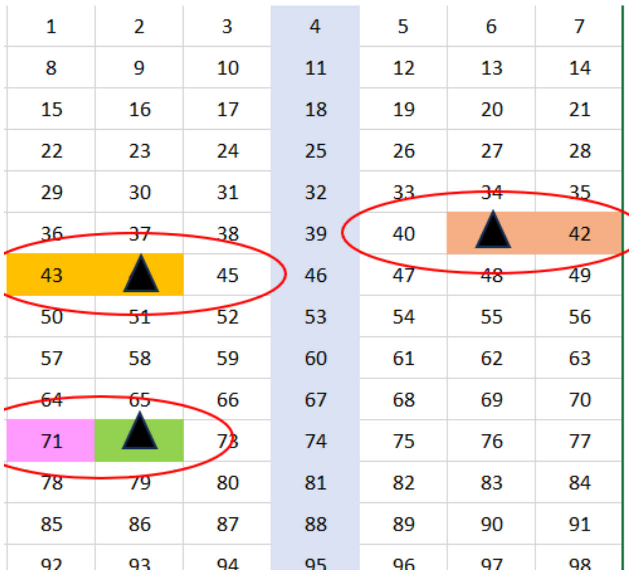


Figure 32: Possibilities for the yellow group

However we did not have time to implement this improvement.

5.6 Case when passengers choose their seats in advance

In that case we consider that some passengers choose their seats in advance, before registration.

In our model, it means adding new constraints to the static model by setting the variables corresponding : if passenger i chooses seat j we impose $S_{ij} = 1$. Then we run the static model with these new variables.

The only change that appears in the dynamic model is the initialization of the dictionary of choices, Choices_dict : it is initialized with the choices already made by the different groups.

Here is the impact of the number of passengers who chose their seats in advance on the global satisfaction (Note: we only had time to measure this impact on one instance, Oct30) :

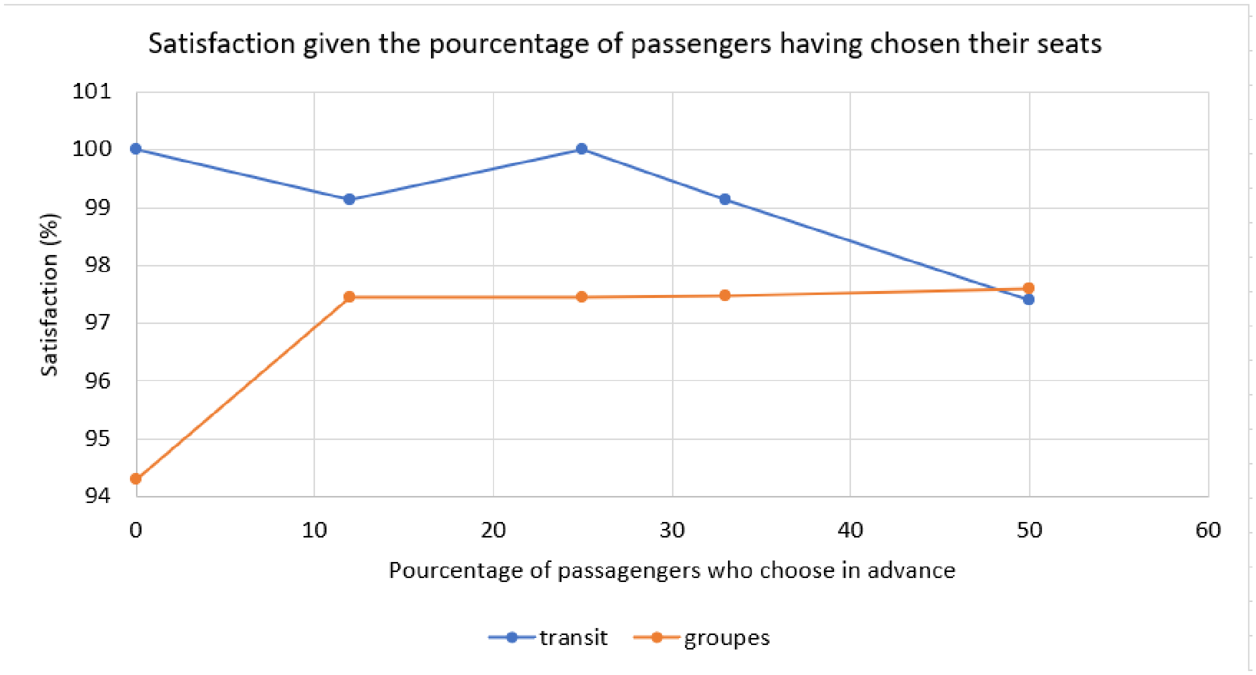
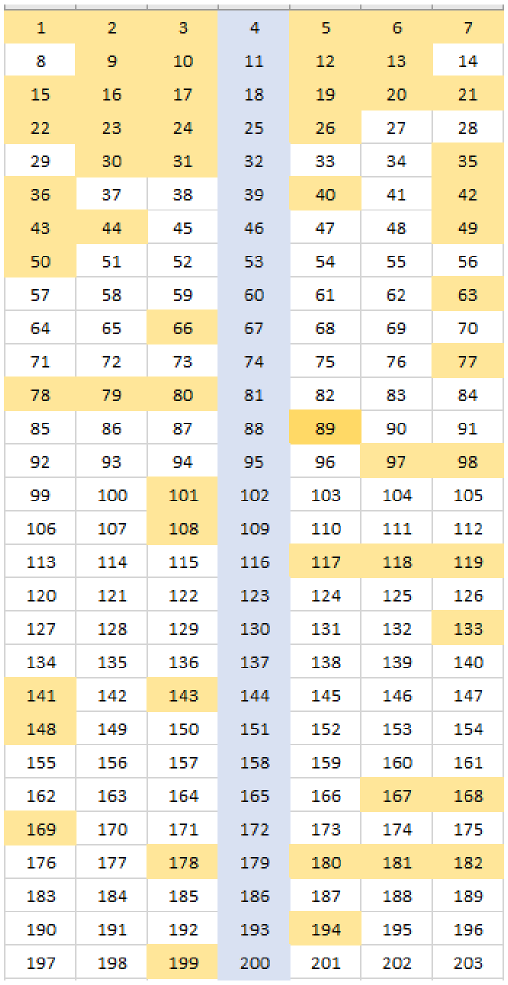


Figure 33: Impact of the percentage of passengers who chose their seats on the satisfaction

We can see that the transit satisfaction decreases seriously from 25%. Indeed in our simulation people tend to choose seats in the front rows. So people in transit who did not choose their places have more and more difficulties to find seats in the front. On the other hand, group satisfaction increases : indeed many of them could choose their places and those remaining had no problem to stay together given that the plane is not full.

We expect the impact to be bigger in the case of a full plane. The next step would be to test it on a plane full.

In the case of 50% of the passengers having chosen their seats in advance, the initialization plane is the following :



1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49
50	51	52	53	54	55	56
57	58	59	60	61	62	63
64	65	66	67	68	69	70
71	72	73	74	75	76	77
78	79	80	81	82	83	84
85	86	87	88	89	90	91
92	93	94	95	96	97	98
99	100	101	102	103	104	105
106	107	108	109	110	111	112
113	114	115	116	117	118	119
120	121	122	123	124	125	126
127	128	129	130	131	132	133
134	135	136	137	138	139	140
141	142	143	144	145	146	147
148	149	150	151	152	153	154
155	156	157	158	159	160	161
162	163	164	165	166	167	168
169	170	171	172	173	174	175
176	177	178	179	180	181	182
183	184	185	186	187	188	189
190	191	192	193	194	195	196
197	198	199	200	201	202	203

Figure 34: Example of the plane when 50% of the passengers have chosen their seats (before running the static model)

6 Sensitivity analysis

6.1 Visualization

We get a first idea of the results of our dynamic model thanks to these visualizations. We can check that groups are together, that the barycenter constraint is respected and that WCHR verify their placements conditions.

We present results for two instances, one with WCHR passengers and one without.

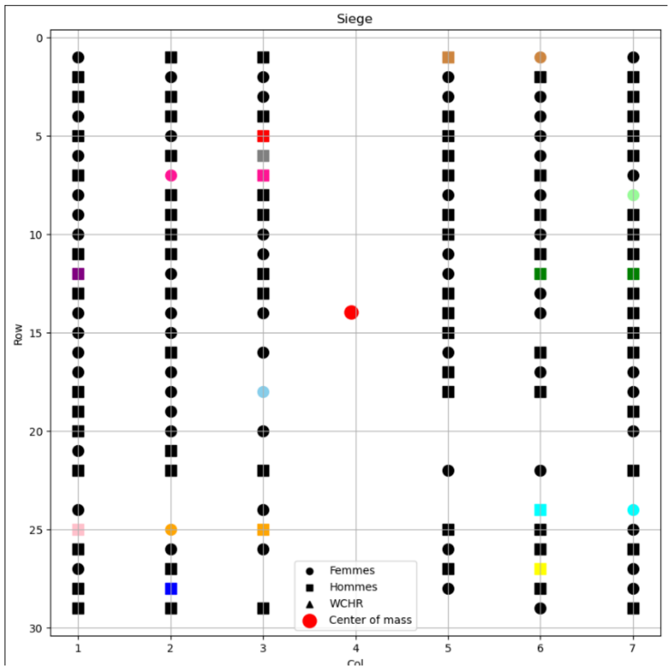


Figure 35: Passenger allocation for October 21st with the static model

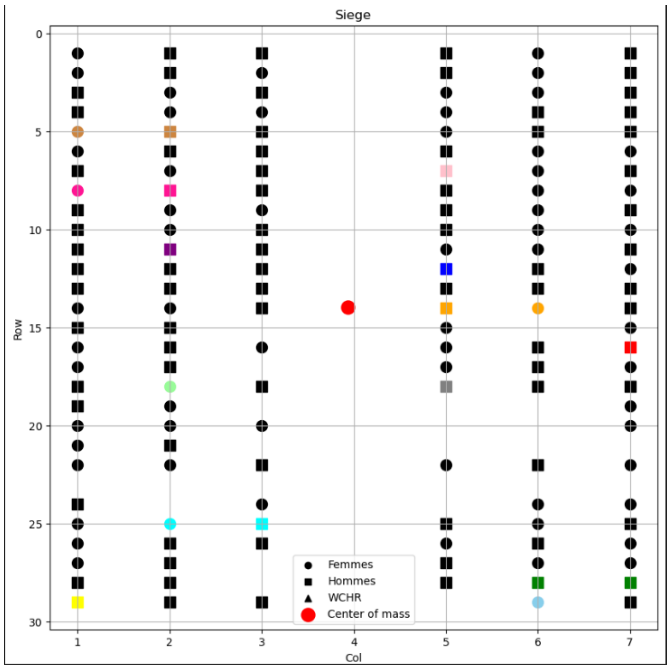


Figure 36: Passenger allocation for October 21st with the dynamic model

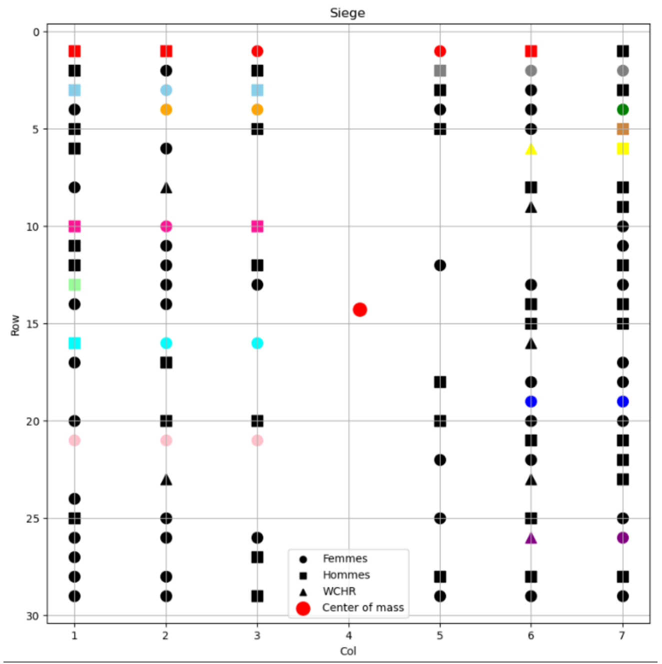


Figure 37: Passenger allocation for October 30th with the static model

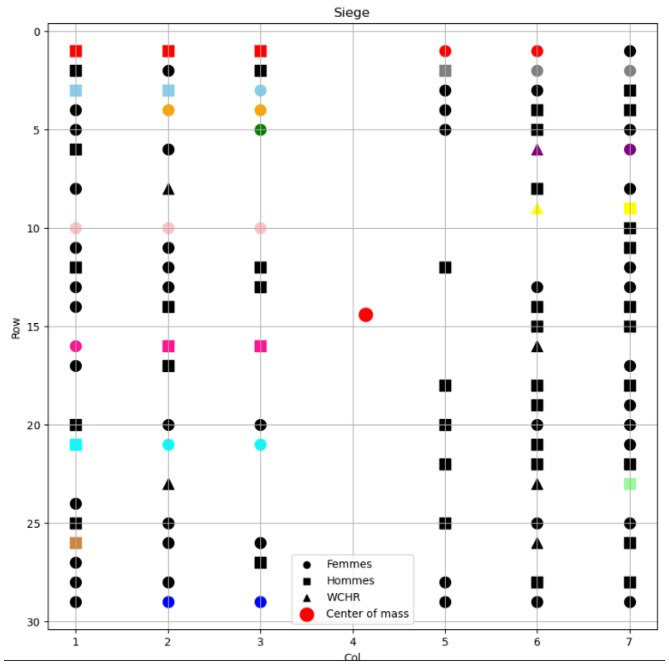


Figure 38: Passenger allocation for October 30th with the dynamic model

6.2 Number of choices offered

In general, our model allows a large amount of choices for the groups. They decrease with the order of registration and become particularly reduced (less than 10) from the 60^{th} position, that is 76% of the global number of groups.

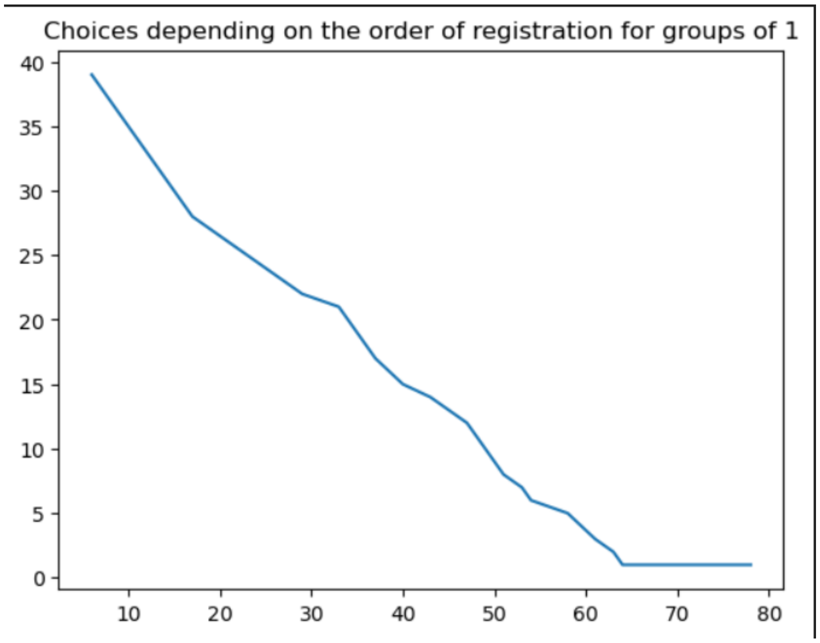


Figure 39: Number of choices offered to groups of 1 not in transit depending on their order of registration (Oct30)

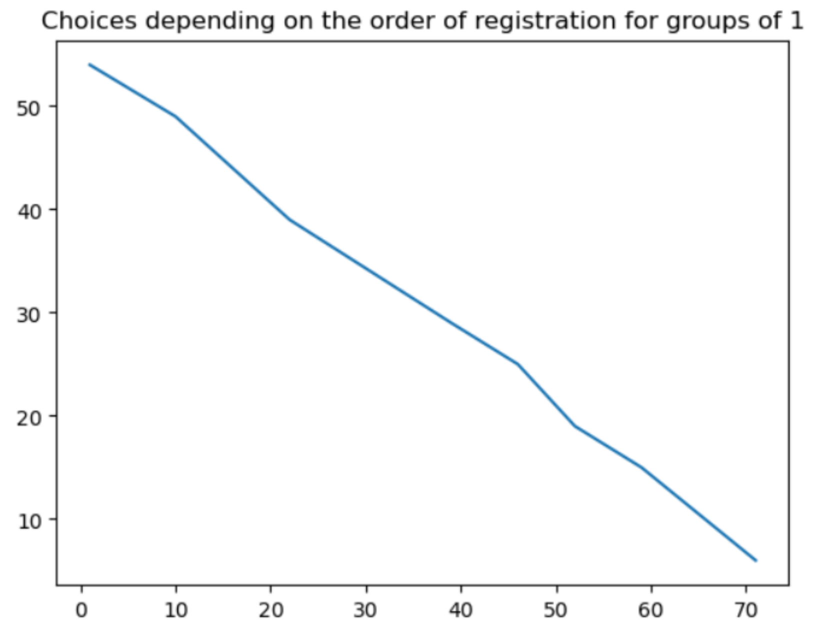


Figure 40: Number of choices offered to groups of 1 in transit depending on their order of registration (Oct30)

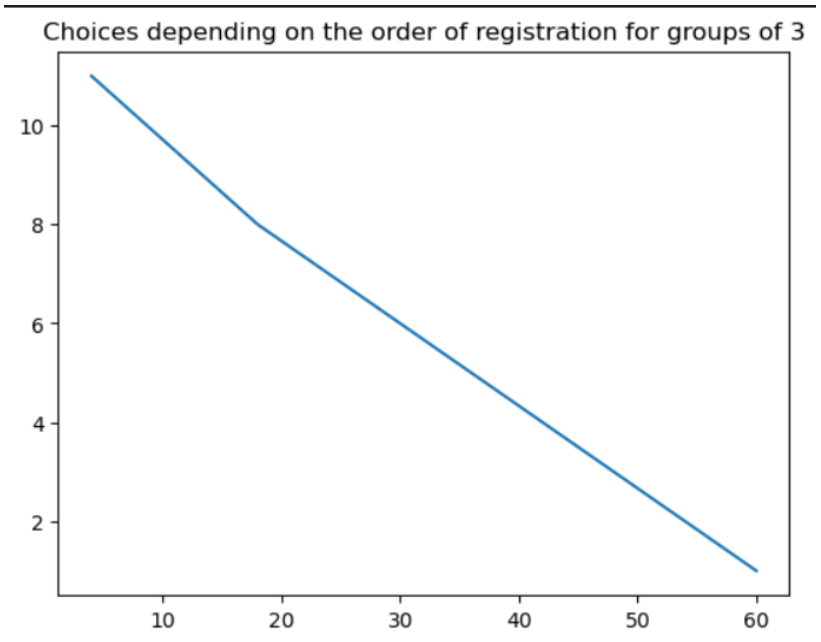
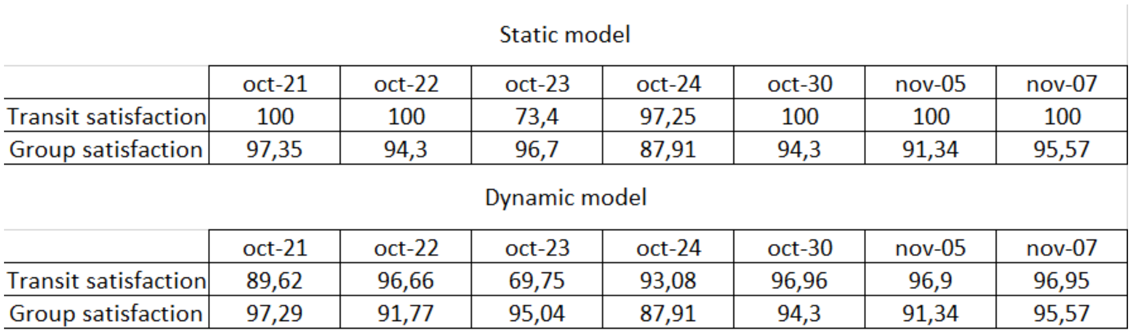


Figure 41: Number of choices offered to groups of 3 not in transit depending on their order of registration (Oct30)

There are only three groups of 3 not in transit and without WCHR passengers in Oct30. However we can propose to them up to 11 possibilities for the first one.

6.3 Comparison between the static and dynamic model

Here we present the transit and group satisfactions for the static and dynamic models for every instance :



Static model

	oct-21	oct-22	oct-23	oct-24	oct-30	nov-05	nov-07
Transit satisfaction	100	100	73,4	97,25	100	100	100
Group satisfaction	97,35	94,3	96,7	87,91	94,3	91,34	95,57

Dynamic model

	oct-21	oct-22	oct-23	oct-24	oct-30	nov-05	nov-07
Transit satisfaction	89,62	96,66	69,75	93,08	96,96	96,9	96,95
Group satisfaction	97,29	91,77	95,04	87,91	94,3	91,34	95,57

Figure 42: Tabular of transits et groups satisfaction for the different instances

For better visualization, we present some of these results in graphs :

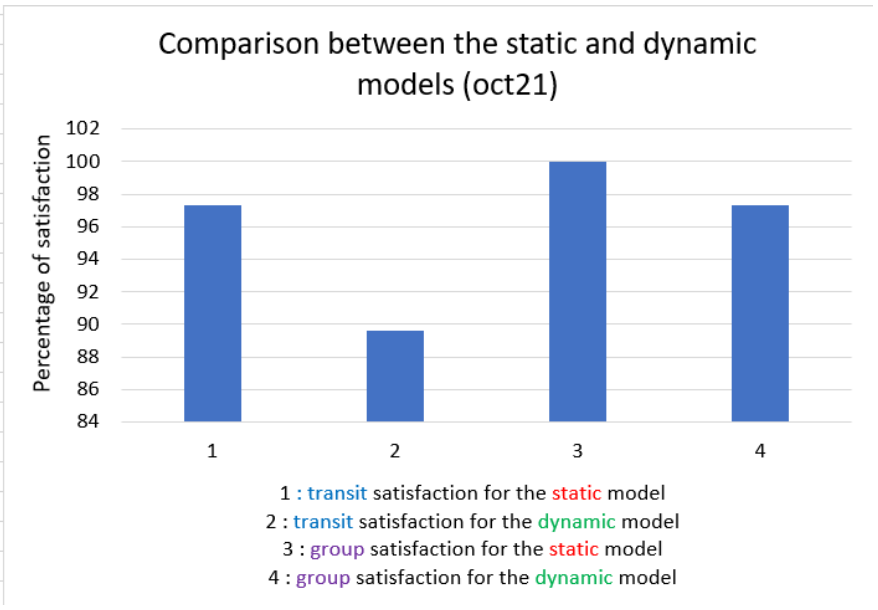


Figure 43: Transit and group satisfaction for the static and dynamic models

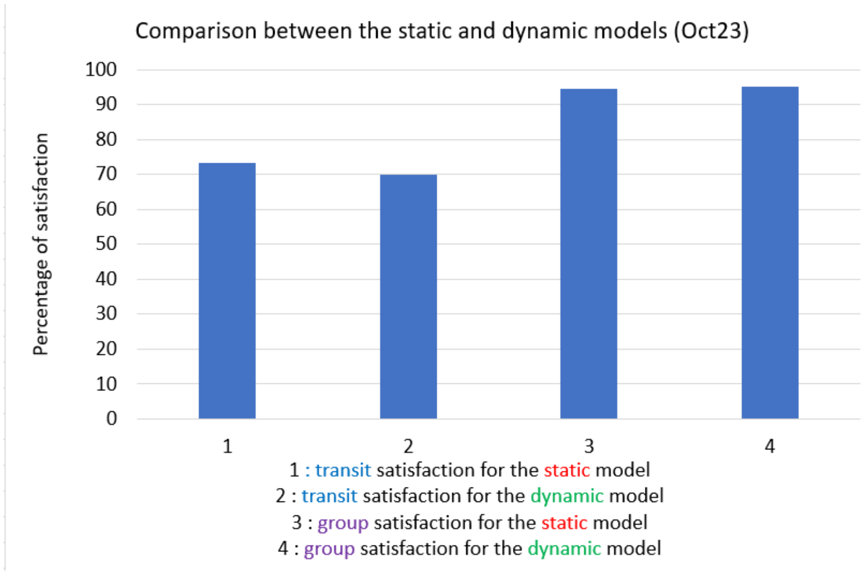


Figure 44: Transit and group satisfaction for the static and dynamic models

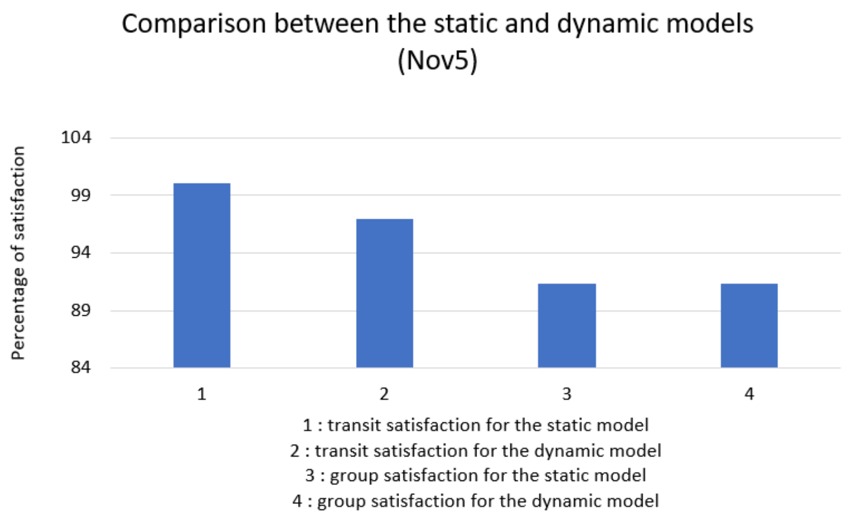


Figure 45: Transit and group satisfaction for the static and dynamic models

We can see that, as expected, that the global satisfaction is smaller for the dynamic model for the last groups don't necessarily have a choice that satisfies them. It is particularly true for transit satisfaction. The group satisfaction remains the same; indeed, by construction of our dynamic model, the disposition of groups don't change from one model to another.

However, the satisfaction score does not decreases a lot and the dynamic model allows passengers to choose seats and place them in a satisfying way.

7 Conclusion

Based on the analysis and development of the static model for passenger seating optimization in the context of Air France, prioritizing passenger satisfaction while adhering to safety constraints and operational efficiency is crucial. By considering diverse scenarios and passenger demographics, the proposed model aims to maximize customer experience by offering optimal seat placements.

Moreover, adhering to reduced mobility constraints is essential not only for ensuring the safety and well-being of passengers with disabilities but also for regulatory compliance and upholding standards of accessibility and inclusion in air travel.

Our evaluation methods gave satisfying results regarding transit time and the conservation of groups. However some cases need to be analysed more precisely, particularly the case of big groups. Furthermore, the dynamic model provides satisfying results as well, and is offering the passengers the right of choice and including wheelchair passengers in the analysis, satisfying all the restrictions, which is a great improvement. This constitutes a great added value for Air France by showing its customers its flexibility.

We also had the opportunity to implement early registration, and obtained plausible results, which respect all the previously mentioned constraints. On the other hand, in future analyses, the test of the dynamic model and global satisfaction could be made including all possible dates, which would add positively for Air France.

In conclusion, this study provides valuable insights into the complexities of passenger seating optimization in commercial aircraft, emphasizing the importance of balancing passenger satisfaction, safety standards, and operational efficiency. By addressing these key aspects, airlines like Air France can enhance the overall travel experience for passengers while meeting regulatory requirements and industry standards.

8 Algorithm Overview

Data Loading

```
1: function LOAD_DATA(file_path)
2:     Load data from CSV files and extract relevant information.
3:     Return a dictionary with group numbers as keys and data as values.
```

Passengers Dictionary Creation

```
1: function CREATE_PASSENGERS_DICT(data)
2:     Create a dictionary of passengers with unique identifiers and attributes.
3:     Assign passengers to groups based on provided data.
4:     Return the passengers dictionary.
```

Model and Variables Creation

```
1: function CREATE_MODEL_AND_VARIABLES(passengers_dict)
2:     Create an optimization model and define decision variables.
3:     Add constraints to limit seat allocations and enforce rules.
4:     Return the optimization model and decision variables.
```

Objective Function for Transit Passengers

```
1: function OBJ_TRANSIT(S, passengers_dict)
2:     Calculate the objective function for transit passengers.
3:     Consider connection time and prioritize front seating.
4:     Return the objective function value.
```

Objective Function for Distance Minimization

```
1: function OBJ_DIST(Passengers, alpha)
2:     Minimize the distance between passengers with a given weight.
3:     Return the objective function value.
```

Applying Objective Functions

```
1: $m$.setObjective(obj_transit($S$, Passengers) + obj_dist(Passengers, $\alpha$), GRB.MINIMIZE)
```

Create Control Dictionary

```
1: function VALEURS_VAR(S)
2:     Create a control dictionary for optimization results.
3:     Input: Decision variables from the optimization model.
4:     Output: Control dictionary of binary values.
```

Compute Center of Mass

```
1: function CALCULATE_CENTER_OF_MASS(seat_coords, Passengers)
2:     Compute the center of mass of passengers on the aircraft.
3:     Input: Seat coordinates and passenger information.
4:     Output: Coordinates of the center of mass.
```

Calculate Passenger Group Satisfaction Metric

```
1: function METRIQUE_GROUPE_PASSAGER
2:     Compute individual passenger satisfaction based on group seating.
3:     Output: Dictionary of passenger satisfaction scores.
```

Calculate Transit Passenger Satisfaction Metric

```
1: function METRIQUE_TRANSIT_PASSAGER
2:     Compute satisfaction for transit passengers.
3:     Output: Dictionary of transit passenger satisfaction scores.
```

Calculate Overall Passenger Satisfaction Metric

```
1: function METRIQUE_GLOBALE_PASSAGER
2:     Compute overall passenger satisfaction metric.
3:     Output: Average satisfaction and dictionary of individual satisfaction scores.
```

Retrieve Passenger Group Indices

```
1: function GROUPE($i, Passengers$)
2:     Return a list containing the indices of passengers in the same group as $i$.
3:     Input: Passenger index $i$ and Passenger dictionary $Passengers$.
4:     Output: List of passenger indices in the same group.
```

Create Group Dictionary

```
1: function GROUP_DICT($Passengers$)
2:     Return a dictionary where keys are group numbers and values are lists of passengers in each
   group.
3:     Input: Passenger dictionary $Passengers$.
4:     Output: Group dictionary.
```

Calculate Strict Group Satisfaction Metric

```
1: function METRIQUE_GROUPE_STRICTE($Passengers$)
2:     Compute satisfaction scores for passengers based on strict group seating criteria.
3:     Input: Passenger dictionary $Passengers$.
4:     Output: Dictionary of group satisfaction scores.
```

Calculate Strict Transit Satisfaction Metric

```
1: function METRIQUE_TRANSIT_STRICTE($Passengers$)
2:     Compute satisfaction scores for transit passengers based on strict criteria.
3:     Input: Passenger dictionary $Passengers$.
4:     Output: Dictionary of transit passenger satisfaction scores.
```

Calculate Strict Overall Satisfaction Metric

```
1: function METRIQUE_GLOBALE_STRICTE($Passengers$)
2:     Compute overall passenger satisfaction metric based on strict criteria.
3:     Input: Passenger dictionary $Passengers$.
4:     Output: Average satisfaction, transit satisfaction, and group satisfaction.
```

Calculate Rectangle Satisfaction Metric

```
1: function METRIQUE_RETANGULE($passengers, seat\_coords$)
2:     Compute satisfaction scores based on the bounding box criterion.
3:     Input: Passenger dictionary $passengers$ and seat coordinates $seat\_coords$.
4:     Output: Average satisfaction and dictionary of group satisfaction scores.
```

Calculate Group Statistics

```
1: function GROUP_STATISTICS($passengers, seat\_coords$)
2:     Calculate group statistics based on passenger data and seat coordinates.
3:     Input: Dictionary of passengers $passengers$ and list of seat coordinates $seat\_coords$.
4:     Output: A dictionary containing group statistics.
```

Update Visualizations

```
1: function UPDATE_VISUALIZATIONS($quantity\_count, group\_data$)
2:     Update boxplot, bar chart, and pie chart visualizations based on group statistics.
3:     Input: Dictionary containing the count of groups with each quantity $quantity\_count$, and
       dictionary containing the satisfaction data for each group $group\_data$.
```

Analyze Group Statistics

```
1: function ANALYZE_GROUP_STATISTICS($group\_data$)
2:     Analyze group statistics based on the provided data.
3:     Input: Dictionary containing group statistics $group\_data$.
4:     Output: Quantity count, average satisfaction, max satisfaction, and min satisfaction.
```

Generation of a random order of registration

```
function RANDOM_ORDER
    $Clefs \leftarrow$ LIST(GROUP_DICT.KEYS())
    $Order \leftarrow$ RANDOM.SAMPLE($Clefs$, $k =$ LEN($Clefs$))
    return $Order$
```

Modeling 1 : Proposition of seats for the group by only considering groups of the same size

```
function POSSIBLE_CHOICE($n\_group$)
    Initialize empty list $Choices$
    $groups\_dict \leftarrow$ GROUP_DICT($Passengers$), $group \leftarrow groups\_dict[n\_group]$
    for each $g$ in $groups\_dict$ do
        if length of $groups\_dict[g]$ equals length of $group$ then
            $C \leftarrow []$
            for each $i$ in $groups\_dict[g]$ do
                for each seat $j$ in range 1 to $ns$ do
                    if seat $(i, j)$ available, add $j$ to $C$ then
            if $C$ not empty, add $C$ to $Choices$ then
    return $Choices$
```

Modeling 2 : Proposition of seats for the group by considering all combinations of sizes

```
function POSSIBLE_CHOICE_SAME_SIZE($group\_size$, $S\_dyn$)
    Initialize an empty list $Choices$
    for each $g$ in $group\_dico$ do
        if length of $group\_dico[g]$ is equal to $group\_size$ and $date[g][2] == 0$ then
            Initialize an empty list $C$
            for each $i$ in $group\_dico[g]$ do
                for each seat $j$ in range from 1 to $ns$ do
                    if $S\_dyn[(i, j)]$ is true, add $j$ to $C$ then
            if $C$ is not empty, add $C$ to $Choices$ then
    return $Choices$
```

Possible Choice

```
1: function POSSIBLE_CHOICE($size, S\_dyn$)
2:     Obtain possible seat choices for groups of the same size as the one being considered using POSSIBLE_CHOICE_SAME_SIZE.
3:     Input: Group size $size$ and dynamic seat availability $S\_dyn$.
4:     Output: List of interesting possibilities.
5:     $Choices \leftarrow$ POSSIBLE_CHOICE_SAME_SIZE($size, S\_dyn$)
6:     for $k$ in range (1, $size//2 + 1$) do
7:         $choices1, choices2 \leftarrow$ POSSIBLE_CHOICE_SAME_SIZE($size - k, S\_dyn$), POSSIBLE_CHOICE_SAME_SIZE($k, S\_dyn$)
8:         $choices1$.sort(), $choices2$.sort()
9:         for $choice$ in $choices1$ do
10:            $coord, X1, Y1 \leftarrow$ SEAT_COORDINATES($choice$), [], []
11:            for $i$ in range (len($coord$)) do
12:                $X1, Y1 \leftarrow X1 + [coord[i][0]]$, $Y1 + [coord[i][1]]$
13:            $X1, Y1 \leftarrow X1$.sort(), $Y1$.sort()
14:            for $places$ in $choices2$ do
15:                $coord2, X2, Y2 \leftarrow$ SEAT_COORDINATES($places$), [], []
16:                for $i$ in range (len($coord2$)) do
17:                    $X2, Y2 \leftarrow X2 + [coord2[i][0]]$, $Y2 + [coord2[i][1]]$
18:                $X2, Y2 \leftarrow X2$.sort(), $Y2$.sort()
19:                if all($y1 == Y1[0]$ for $y1$ in $Y1$) and all($y2 == Y2[0]$ for $y2$ in $Y2$) and $Y1[0] == Y2[0]$ then
20:                    if all($place$ not in $choice$ for $place$ in $places$) then
21:                        $Choices \leftarrow Choices + [choice + places]$
22:    return $Choices$
```

We suppress from the propositions the seats already chosen by previous groups

```
1: function GROUP_CHOICES_AVAILABLE($choices, Choice\_dict$)
2:     Input: List of choices $choices$ and dictionary of choices made by previous groups $Choice\_dict$.
3:     Output: List of available choices.
4:     $choices\_copy \leftarrow choices.copy()$
5:     for $choice$ in $choices$ do
6:         for $taken$ in $Choice\_dict$ do
7:             for $place$ in $choice$ do
8:                 if $place$ in $taken$ and $choice$ in $choices\_copy$ then
9:                     $choices\_copy$.remove($choice$)
10:    return $choices\_copy$
```

We supress from the propositions the seats in the front rows if the group is not in transit

```
function TRANSIT_FILTER($n\_group, Choices$)
    Input: Group number $n\_group$ and list of choices $Choices$.
    Output: Filtered list of choices.
    $Choices\_copy \leftarrow Choices.copy()$
    $group \leftarrow group\_dico[n\_group]$    ▷ List of passengers in group $n\_group$
    $group\_time \leftarrow Passengers[group[0]]['connection\_time']$ ▷ Transit time of group $n\_group$
    if $group\_time > 0$ and $group\_time \leq 120$ then  ▷ If group is in transit, propose all seats
        return $Choices$
    else    ▷ Otherwise, remove seats in the first 7 rows from propositions
        for $choice$ in $Choices$ do
            $coord \leftarrow$ SEAT_COORDINATES($choice$)
            $X, Y \leftarrow [], []$
            for $i$ in range (len($coord$)) do
                $X \leftarrow X + [coord[i][0]]$, $Y \leftarrow Y + [coord[i][1]]$
            if all($y \leq 7$ for $y$ in $Y$) and len($Choices\_copy$) $> 1$ then
                $Choices\_copy$.remove($choice$)
    return $Choices\_copy$
```

We supress from the propositions the seats which don't respect the barycenter constraint

```
function BARYCENTER_FILTER($group, choices, S\_dyn$)
    $group\_passengers \leftarrow$ GROUP_DICO[$group$]
    for $choice$ in $choices$ do
        $S\_bar \leftarrow S\_dyn.copy()$
        $random\_attribution \leftarrow$ random permutation of passenger indices
        for $k$ in $random\_attribution$ do
            $i, j \leftarrow k$-th passenger and seat in $choice$
            Find passenger $p$ with seat $j$ already assigned
            Find old seat $old\_seat$ occupied by $i$
            Swap seats between $i$ and $exchanged\_passenger$
        Calculate the barycenter coordinates for the choice
        Remove the choice if barycenter constraints are not met
    return $choices$
```

We give each proposition a score based on the satisfaction criteria used in the static model

```
function ASSIGN_VALUES($group, choices$)
    $choices\_with\_values \leftarrow []$
    for $choice$ in $choices$ do
        $score \leftarrow 0$
        $coord \leftarrow$ seat coordinates of $choice$
        $X, Y \leftarrow [], []$
        for $i$ in range (length of $coord$) do
            $X \leftarrow X + [coord[i][0]]$, $Y \leftarrow Y + [coord[i][1]]$
        $X$.sort(), $Y$.sort()
        if group in transit and seats in rows 1-6 then
            $score+ = 0.7$
        else if group not in transit then
            $score+ = 0.7$
        if group size is 1 and not by window or central aisle then
            $score \leftarrow$ random value between 0 and 0.5
        else if group size is 2 or 3 and seats are adjacent then
            $score+ = 0.5$
        else if group size is 2 or 3 and seats at 3 and 5 then
            $score+ = 0.25$
        else if group size is 4-6 and seats are adjacent then
            $score+ = 1$
        else if group size is 4-6 and seats all on one side then
            $score+ = 0.5$
        else if group size is greater than 6 and seats form a rectangle then
            $score+ = 0.5$
        $choices\_with\_values$.append(($choice$, $score$))
    return $choices\_with\_values$
```

The group will choose the proposition with the highest score

```
function GROUP_CHOICE($choices\_with\_values$)
    $sorted\_choices \leftarrow$ sort choices with values by score in descending order
    $group\_choices \leftarrow$ list of choices sorted by score
    $group\_choice \leftarrow$ first choice in sorted list
    return $group\_choice$
```

Case of groups with WCHR passengers

```
function CHOICE_WCHR($group, S\_dyn, Choices\_dict$)
    $Choices \leftarrow []$
    $size \leftarrow$ size of group $group$
    $nb\_wchr \leftarrow$ number of WCHR passengers in group $group$
    for $g$ in group_dico do
        $nb\_wchr\_g \leftarrow$ number of WCHR passengers in group $g$
        $passengers\_g \leftarrow$ passengers in group $g$
        $size\_g \leftarrow$ size of group $g$
        $C \leftarrow []$
        if $nb\_wchr\_g \neq 0$ and $nb\_wchr = nb\_wchr\_g$ and $size = size\_g$ then
            $C \leftarrow$ seats for WCHR passengers in group $g$
        if $C \neq []$ and $C$ not in $Choices\_dict$ values then
            $Choices \leftarrow Choices + [C]$
    return $Choices$
```

Dynamic Model with WCHR

```
function DYN_MODEL_WCHR($group, S\_dyn, Choices\_dict, satis\_choice\_dict, order$)
    $possibilities \leftarrow$ choices with WCHR for group $group$
    $choice \leftarrow$ randomly selected choice from $possibilities$
    Add $choice$ to $Choices\_dict$ for group $group$
    Update satisfaction choice dictionary with group information
    for $i$ in passengers of group $group$ do
        $ancient\_seat \leftarrow$ seat previously occupied by passenger $i$
        $i\_gets\_seat \leftarrow$ False
        while not $i\_gets\_seat$ do
            $seat \leftarrow$ next seat in $choice$
            if passenger $i$ can occupy $seat$ then
                Assign $seat$ to passenger $i$
                Mark $seat$ as occupied
                Mark previous seat of passenger $i$ as vacant
                $i\_gets\_seat \leftarrow$ True
    return $S\_dyn, satis\_choice\_dict, Choices\_dict$
```

We run the dynamic model for every group

```
function DYNAMIC_MODEL
    $S\_dyn$ ← copy of static seat assignment dictionary
    $Choices\_dict$ ← {}
    $satis\_choice\_dict$ ← {}
    $order$ ← random order of groups
    for each $group$ in $order$ do
        if group has WCHR passengers then
            Use dynamic model with WCHR for group
        if group has no WCHR passengers then
            Calculate possible choices for group
            Filter choices considering previous group choices
            Filter choices considering transit constraints
            Filter choices considering barycenter constraints
            Assign values to choices
            Select best choice for group
            Update seat assignments and choice dictionary
            Record group satisfaction information
    return updated seat assignment dictionary and satisfaction choice dictionary
```