GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

OPTIMIZATION OF BOARDING GATE ASSIGNMENTS IN AIRPORTS

Autor: Guillermo Rodríguez-Auñón Molina

Director: Ali Haghani

Madrid

Julio de 2019

damages, which may be brought against the University by third parties who believe that their rights and interests have been infringed by the assignment. d) Take responsibility in the event that the institutions are found guilty of a rights infringement regarding the work subject to assignment.

6. Institutional Repository purposes and functioning.

The work shall be made available to the users so that they may use it in a fair and respectful way with regards to the copyright, according to the allowances given in the relevant legislation, and for study or research purposes, or any other legal use. With this aim in mind, the University undertakes the following duties and reserves the following powers: a) The University shall inform the archive users of the permitted uses; however, it shall not guarantee or take any responsibility for any other subsequent ways the work may be used by users, which are non-compliant with the legislation in force. Any subsequent use, beyond private copying, shall require the source to be cited and authorship to be recognized, as well as the guarantee not to use it to gain commercial profit or carry out any derivative works. b) The University shall not review the content of the works, which shall at all times fall under the exclusive responsibility of the author and it shall not be obligated to take part in lawsuits on behalf of the author in the event of any infringement of intellectual property rights deriving from storing and archiving the works. The author hereby waives any claim against the University due to any way the users may use the works that is not in keeping with the legislation in force. c) The University shall adopt the necessary measures to safeguard the work in the future. d) The University reserves the right to withdraw the work, after notifying the author, in sufficiently justified cases, or in the event of third party claims.

Madrid, on 2nd of July, 2019

HEREBY ACCEPTS

Signed……[illegible]……………………………………………

Reasons for requesting the restricted, closed or embargoed access to the work in the Institution's

Repository

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título OPTIMIZATION OF BOARDING GATE ASSIGNMENTS IN AIRPORTS

……………………………………………………………………………………………

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2018/2019 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Guillermo Rodríguez-Auñón Molina Fecha: 03/ 07/ 2019

[illegible]

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Ali Haghani Fecha: 14/ 07/ 2019

[illegible]

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

OPTIMIZATION OF BOARDING GATE ASSIGNMENTS IN AIRPORTS

Autor: Guillermo Rodríguez-Auñón Molina

Director: Ali Haghani

Madrid

Julio de 2019

OPTIMIZATION OF BOARDING GATE ASSIGNMENTS IN AIRPORTS

Autor: Rodríguez-Auñón Molina, Guillermo

Director: Haghani, Ali

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En los tiempos actuales, el método de transporte más rápido es el avión. Se realizan una inmensa cantidad de vuelos, que a su vez estos cubren una cantidad enorme de kilómetros. Es por ello y por la estructura tan compleja del avión que es normal que se produzcan retrasos debido a problemas técnicos tanto de mantenimiento como de reparación. Estos retrasos muchas veces son un incordio para el pasajero que ve como se convierte en una incertidumbre. El programa diseñado en este trabajo se encarga de eliminar este periodo de desconocimiento.

El objetivo del código creado es optimizar en real-time la asignación de las puertas de embarque teniendo en cuenta los retrasos que van surgiendo en los vuelos. La meta de esta optimización consiste en minimizar la distancia que recorre el pasajero desde la entrada hasta la puerta de embarque. Esto aumentará el grado de satisfacción del pasajero con respecto al aeropuerto.

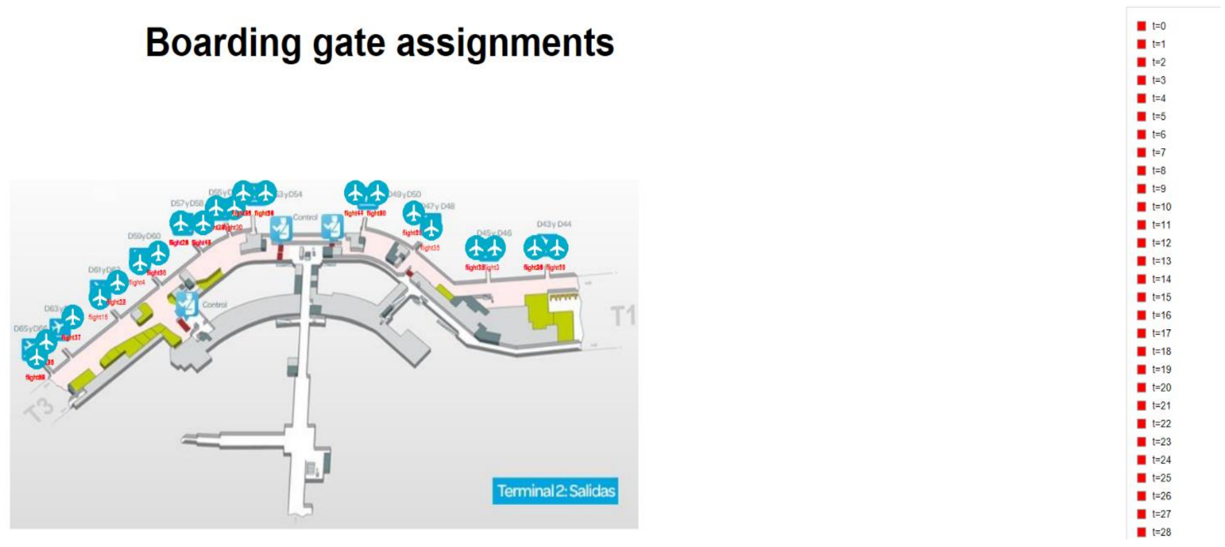
Para conseguir esta optimización, primero se guardarán los datos de los vuelos que se lleven a cabo en el espacio de tiempo que deseamos estudiar en una hoja de Excel, al igual que las distancias a las puertas de embarque desde la entrada, sus nombres y las coordenadas en las que están localizadas estas puertas. Una vez escrito todos estos datos los copiamos en un archivo de texto para que pueda ser leído por el programa.

Una vez conseguidos los datos, el programa se encargará de lo demás. Primero creará una matriz que será optimizada sin tener en cuenta los retrasos. Esta primera matriz se optimiza sola debido a que el código escrito para su optimización difiere de aquellas que utilicen también la información de los retrasos. Después el programa copiará las soluciones de este supuesto en otra matriz más grande, y las asignará al caso 0.

Esta matriz es la que guardará las soluciones finales teniendo en cuenta los retrasos que se hayan dado a conocer. Consiste en una matriz binaria de 4 dimensiones (Y(i,j,k,d)), donde las primeras dos hacen referencia a las filas y columnas de la matriz, que en este caso serán los vuelos y las puertas respectivamente. Si la celda contiene un 1, significa que el vuelo i está asignado a la puerta j. La dimensión k hace referencia al tiempo. En cada momento de tiempo, el cual está dividido en escalones de 15 minutos (por ejemplo, de 9:00 a 9:15 sería el momento 1), se mostrará el estado de las puertas de embarque del aeropuerto. A su vez, la dimensión d va aumentando conforme se van dando a conocer los retrasos. Es decir, cada vez que se recibe la información de que un vuelo sufre un retraso, se optimizará la matriz en tiempo real. Para ello, conforme se comuniquen los retrasos se irá corrigiendo la hoja de datos de la que se nutre el

programa para la asignación. Por ejemplo, si a las 14:00 se comunica el retraso de un vuelo que aterriza a las 14:30, se implementará el cambio en la hoja de Excel y se optimizará la asignación para actualizar la puerta de embarque de aquellos vuelos que se puedan ver afectados, pero sin modificar aquellos ya aparcados en estas puertas. De esta manera se evita que un avión sea mandado a otra puerta de embarque cuando ya se ha enganchado a una y por consiguiente se haga desplazarse a los pasajeros también. En caso de que en un mismo momento se comunique más de un retraso, la dimensión d solo se optimizará una vez, y estos retrasos coincidentes serán optimizados a la vez. Cada valor diferente de esta dimensión contiene una matriz diferente de tres dimensiones, es decir, contiene la información de todas las puertas asignadas a los vuelos en todos los instantes de tiempo que se hayan elegido para estudiar. Por ejemplo, en el caso base que se estudió en 12 momentos diferentes se comunicaban retrasos, y estudiábamos 62 escalones de tiempo. Por lo tanto, para cada uno de los doce momentos el programa optimizó el estado de las puertas durante esos 62 escalones. El objetivo es obtener una predicción para organizar el aeropuerto con los datos obtenidos de retrasos en los vuelos en tiempo real.

Finalmente, cuando se tienen los datos de la solución final con todos los retrasos, el programa se encargará de mostrar una imagen de la terminal objeto de estudio (en este caso, la T2 del aeropuerto Madrid-Barajas Adolfo Suárez) con los resultados obtenidos de esta manera:



Esta imagen es interactiva. En la columna de la derecha aparece un menú con todos los momentos de tiempo estudiados y se puede seleccionar aquellos que se deseen. Con esta manera de presentar se facilita en gran medida la capacidad de comprobar y corregir aquellos datos que lleven a errores. A su vez ayuda a la correcta visualización de la disposición óptima del aeropuerto. Si se desea se puede disponer de varios momentos para comprobar que el avión no cambia de puerta en el tiempo que está enganchado a la puerta de embarque o para corroborar que todo lleve su correcto funcionamiento.

El hecho de que el programa realice optimizaciones en tiempo real conlleva que las dimensiones del programa aumenten considerablemente. Por ejemplo, en el caso base, con 58 vuelos, 22 puertas de embarque, 63 escalones de tiempo y 12 retrasos, la matriz solución contiene 1.045.044 valores, de los cuales a la solución final pertenecen 80.338. Esto significa que un 7,7% de los resultados son significativos.

OPTIMIZATION OF BOARDING GATE ASSIGNMENTS IN AIRPORTS

Author: Rodríguez-Auñón Molina, Guillermo

Director: Haghani, Ali

Collaborating Entity: ICAI – Universidad Pontificia Comillas

PROYECT SUMMARY

Currently, the fastest way of transport is the airplane. We perform a huge amount of flights, and we cover an stratospheric number of kilometers as well. This and the complexity of an airplane’s structure are the reason why we can usually fly delays in flights due to maintenance or repair of this machines. These delays cause stress to the passengers as the details of their flight become unknown for them.

The model that has been designed in this thesis has marked as goal to eliminate this period of uncertainty. The aim of the code that I created is to optimize in real-time the assignment of boarding gates taking into consideration the delays that are ocurrying to the flight.

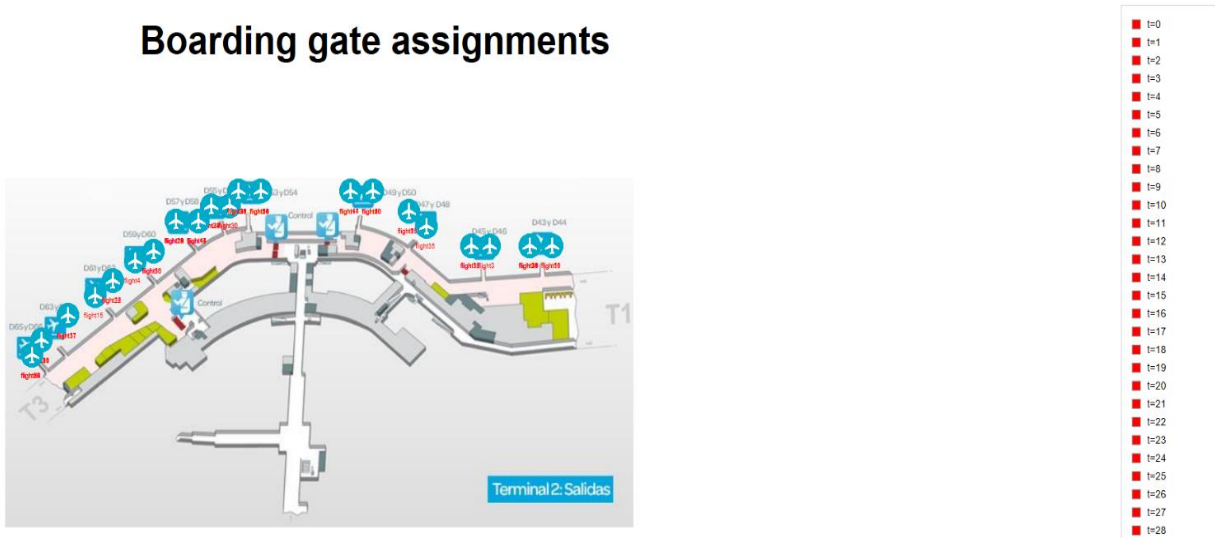
In order to achieve this Optimization, first we need to sabe the information of the flights that will take place in the time that we desire to study in an Excel sheet, as well as the distances from the main entrance to the boarding gates, their names and the coordinates in which the gates are located. Once all of the data is written we copy it in a text file so that the program is able to read it.

When we have all the information collected in the text file, the code will do the rest. Firstly it will create a matrix that will be optimized without considering the delays. This matrix is optimized alone because the code to reach it is written in a different way from the one we use when there are delays. After this, the program will copy the solutions of this case in a bigger matrix, and will be assigned to the case 0.

This matrix is the one that will save the final solutions taking into account the delays that have been acknowledged. It consists of a binary matrix of four dimensions (Y(i,j,k,d), where the first two indexes refer to the rows and columns of the matrix, which are flights and gates. If the slot has the value 1, it means that th flight i is assigned to gate j. The dimension k is set to indicate the timestep. For each timestep, which is divided into steps of 15 minutes (for example, from 9:00 to 9:15 would be the step 1), it will be shown the status of the boarding gates of the airport. On the other hand, dimensión d will be increasing when the number of delays known rises. In other words, each time the Airport controllers receive a notification that a flight has been delayed, the matrix will be optimized in real time. In order for this to happen, we will need to edit the data file where it obtains the flight information. For instance, if at 14:00 a delay is communicated related to a flight that lands at 14:30, a change in the Excel sheet will be implemented and the assignment of boarding gates will be updated for each of the flights that will be affected, but without modifying the Assignment of the previous flights that are already attached to the gates. This way we can avoid that an airplane is changed from one boarding gate to another with the following displacement

of passengers. In the evento that in a same timestep, more than one delay is detected, the dimensión d will optimize just once, and these delays that coincide will be optimized at the same time. For each different value of this index,the matrix contains a three dimensión matrix which contains the information of all the boarding gate assigned to each flights in all the timesteps that have been chosen to study. For example, in the basic case that have been studied in twelve different moments the delays were acknowledged. Therefore, for each and every one of this twelve timesteps the program optimized the status of the gates during the sixty-two timesteps. The goal of this process is to obtain an exact prediction to organize the airport with the data obatained on the delay of the flights in real time.

Finally, when we obtain the information on the final feasible solution taking into consideration every delay we acknowledged, the program will carry out the task of displaying an image of the final solution of the terminal that is being our object of study (in this case it is the second terminal of the airport Madrid-Barajas Adolfo Suárez) with the results collected in this way:



This is an interactive image. In the column that is to the right of the picture there is an index with all the timesteps studied and it can be chosen any timestep that we want. This way of displaying the solution facilitates in big amount the ability to correct and check that information that is not right. Furthermore, it helps tu visualize the optimal disposition of the airplanes in the airport. If desired, we can display more than one timestep with the objective of double checking that the same airplane is not moved between boarding gates while it is already attached to one of them or to corroborate that everything is functioning as it should be.

The fact that this program optimizes in real time signifies that the dimension of the feasible solution increases considerably. For instance, in the case that is being studied here, we account for 58 flights, 22 boarding gates, 63 timesteps and 12 delays. The matrix that collects the final solutions contains 1.045.044 slots, from which the final solution only is reflected in 80.338 of them. This means that bearly an 8% of the results are truly significant.

INDEX

INTRODUCTION AND APPROACH OF THE PROJECT

One of the important aspects of airline operations is the assignment of flights to gates. From a planning perspective, given the flight schedule and the available gates, a preliminary assignment can be made using mathematical optimization and employing a variety of objectives. This preliminary assignment of flights to gates can work well if all flights operate on schedule. However, due to unforeseen circumstances, delays occur in both arrival and departure of flights. These delays mean that sometimes the arriving flights cannot arrive at the pre-assigned gates because the gate is occupied by a delayed flight. By the same token, departing flights cannot leave from pre-assigned gates. During the course of a day there are a lot of delays in flights due to numerous circumstances, and sometimes the airport crew fails to solve it quickly and creates a lot of uncertainty around the passengers. In these circumstances, a reassignment of flights to gates in real-time is necessary.

The aim of this project is to develop a model that can reassign flights to gates in real time. This project is focused on creating a program that automatically solves the gate assignment due to delays on flights, so people do not have to wait until a decision is reached. The way to proceed would be just in the moment that the airport crew gains acknowledge of the delay, write it on the data file so the program can optimize the assignments with this new information.

STATE OF THE ART

Due to the secrecy policy implemented in airports because of security reasons I was not able to find current information of the operating systems of airports. Nonetheless I was able to find a document dated on 2012 that describes the software used on that date. Those will be the softwares I will describe below.

On 2012, airports had various softwares available to assign the airport resources (including boarding gates) like (FERN12):

- Sadama: this software is in charge of automatically assign the resources but does not prevent delays. With this software delays would have to be handled manually.
- Quintiq: in this case, it optimizes the resources in a automatic way on real time so it can maximize the productivity and minimize the cost.
- SITA workbridge: it also optimizes the resources automatically and on real time so there can be a situation analysis on each time and it can optimize the resources accordingly.
- Sistema de Terminal Systems: it is a simulator in real time that shows the status of the airport. It is useful because the coordinator can visualize the airport, although the changes and assignments are left for the person encharged to implement them manually.

DESCRIPTION OF THE DEVELOPED MODEL

GOALS AND SPECIFICATIONS

The goals I set for this project were:

1. Apply real airport data with delays to the case in order to solve real life problems that occurred in the past.
2. Optimize the assignment of boarding gates to airplanes to minimize the distance covered by passengers in T2 of Adolfo Suárez-Barajas airport.
3. Accomplish to write a code that from some data written on Excel can optimize the assignment of boarding gates to airplanes with their delays.
4. Display the airport image situation for each moment in time.
5. Transform the code from a simple optimizer to a real time optimizer.

The model will be written on Xpress IVE software. This software belongs to FICO and it is made specifically to code optimization models and run it to obtain a feasible solution.

For the specifications of the model it has four matrices of decision variables. These four matrices will be obtained from the data collected on terminal T2 in Adolfo Suárez-Barajas airport that will be shown afterwards. Using the data collected we will minimize the objective function in order to achieve the smallest distance covered in the airport by the passengers, applying the delays recorded in real time. Finally, when the solution is obtained, a map of the terminal will be displayed with the gate assignments shown for each timestep.

The type of the problem will be a Binary Integer Program. This means that all the decision variables in this model will only assume 0 or 1 values. As this is an assignment problem, they will only be required to indicate if the gate is occupied or not. In spite of being a Binary Integer Program, the data collected is not binary, it is made of integer numbers, but it will be used to achieve the binary results

DATA

Through the airport website I managed to obtain the flight information related to the basic case. In the data we can differentiate between the flight information and the gate information.

The flight data collected belongs to the flights that arrive to and depart from the terminal that is being studied from 9am to 12am on April 10th, 2019. The flights selected where the ones where we could determine the time they were spending in the airport. I discarded those flights that it could only be known the arrival time because there is no reason to take them into account as they only leave passengers, so it won't affect the optimal solution.

This data collected was saved into a matrix of that has as many rows as flights and six columns. Each column of this matrix represents a different value:

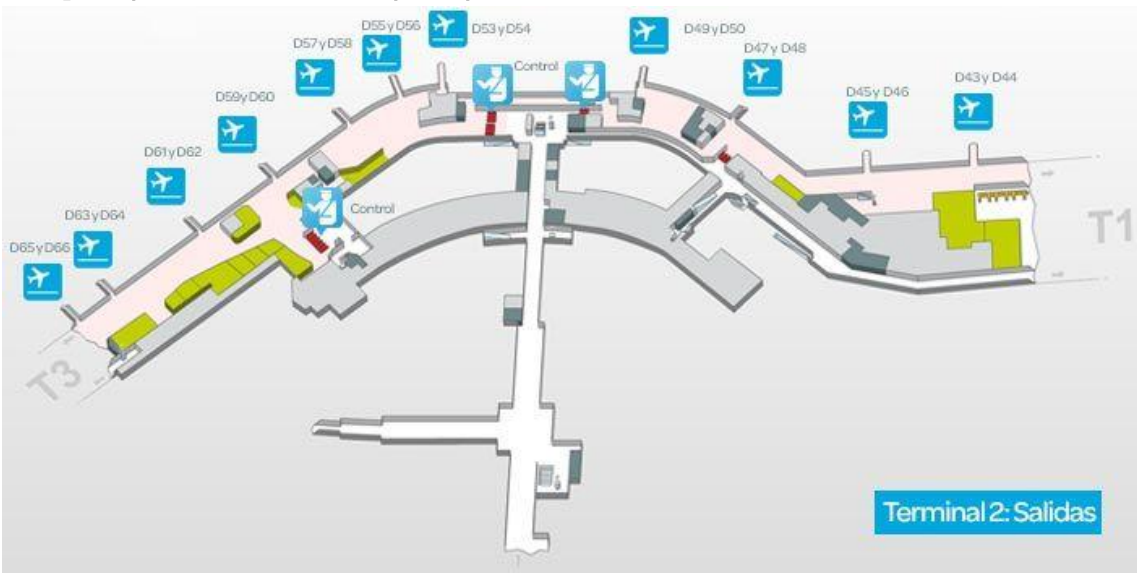
- Column 1: arrival time.
- Column 2: timesteps spent in the gate.
- Column 3: contains the information related to the delay on arrival
- Column 4: same as column 3 but related to the departure
- Column 5: timestep at which the delay is acknowledged
- Column 6: number of passengers transported

For coding reasons, I divided the time into timesteps of 15 mins starting at 9am, so if an airplane arrives at 9:10, it will belong to timestep 1. All time data is displayed the same way.

The passenger information was retrieved supposing that they were flying at full capacity. This is, with the airplane model I was able to find the number of seats it accounted for. Although it is probable that there are always empty seats, I supposed it was full to run the most extreme situation where there was more chaos in the airport.

Also, there were flights that its matching arrival landed the previous day. In that case, I supposed that it needed to be assigned to a gate an hour before the flight (4 timesteps).

Then we have the gate information which is formed by data that reflect distances between the terminal entrance and the different boarding gates (using Google Maps I was able to measure them) and saved in a vector. I was also able to find the gate names, which were also located in a vector, and using Xpress ploting code I created a matrix where I stored the coordinates of the different gates into a matrix in order to display the occupied gates in the following image:



MODEL

IMPORTED DATA

Each problem before being solved needs to have its basic information imported. For this problem we import the flight information fl(x,6), the distance between boarding gate and entrance D(x), the name of the gates gates(x) and the coordinates of their location C(x,2). All this data is imported from a text file, which obtains the information from an excel sheet (copy and paste the information from the sheet to the file).

For the basic case, the number of rows will be 57 for the flight information, meaning that there are 57 flights in that time lapse. The number of gates is 22, which reflects the number of rows for the vectors and matrices that gather the information related to the distance, the names and the coordinates.

VARIABLES

1. **X(x,y,z):** binary. This matrix represents the assignment of boarding gates to flights in real time in the utopic situation where we did not incur in delays. The x value represents the number of flights, y the number of gates and z the timestep.
2. **P(x,y):** binary. This matrix represents the assignment of boarding gates to flights not taking into account the time in the utopic situation where we did not incur in delays. The x value and the y value represent the same as in matrix X.
3. **K(x,y,z):** binary. In this case, matrix K is equal to matrix X. Its main purpose is to be able to copy matrix X solution into matrix Y.
4. **Y(x,y,z,d):** binary. Matrix Y has the same function as matrix X. They differ that matrix Y will be optimized counting with the delays in real time. Indexes x, y and z mean the same as in matrix X. The index d reflects the number of delays happened in the period of time subject to study.
5. **Q(x,y,d):** binary. This matrix is the equal to matrix P for matrix X. The difference with matrix P is that, as well as matrix Y, it will take delay into consideration.

CONSTRAINTS

First, I will show the constraints that affect the utopic case where delays don't affect the assignment:

1. $\sum_{i=1}^{i} X(i,j,z) \leq 1 \; for\; all\; j\; and\; z$: this constraint secures that at each timestep, one gate is only assigned to one flight.

2. $\sum_{j=1}^{j} X(i,j,z) \leq 1 \, for \, all \, i \, and \, z$: this constraint implements that at each timestep one flight is only occupying one gate.
3. $\sum_{j=1}^{j} P(i,j) \leq 1 \, for \, all \, i$: same as constraint number 2.
4. $\sum_{j=1}^{j} X(i,j,z) = 1 \, for \, all \, i \, and \, z \, that \, fulfill \, fl(i,1) = z$: this enables a flight to be assigned to a boarding gate when it lands.
5. $\sum_{j=1}^{j} X(i,j,z) = X(i,j,z-1) \, and \, for \, all \, i, z \, and \, j \, where \, fl(i,1) < z \, and \, fl(i,1) + fl(i,2) > z$: this ensures that once a flight is assigned to a gate it stays in that gate.
6. $\sum_{j=1}^{j} X(i,j,z) = P(i,j) \, and \, for \, all \, i, z \, and \, j \, where \, fl(i,1) < z \, and \, fl(i,1) + fl(i,2) > z$: this constraint fixes the gate to the flight in order to optimize it
7. In any other case, $X(i,j,z) = 0$.

Now I will show the constraints the affect the matrix Y and Q (I will not include the constraints that are similar to the previous case like ensure that one flight is assigned to only one gate as they are written the same way just adding the new d index):

8. $Y(i,j,z,m) = Y(i,j,z,m-1) \, for \, all \, i, j, z \, and \, d \, only \, if \, z < n$: where n is the time at which the airport personal acknowledges of the delay, this constraint fixes all the assignments previous to that moment because what was previously assigned cannot be changed.
9. $\sum_{j=1}^{j} Y(i,j,z,m) \leq 1 \, for \, all \, i, m \, and \, z \, that \, fulfill \, fl(i,1) + fl(i,3) = z$: this enables a flight to be assigned to a boarding gate when it lands when it is delayed.
10. $\sum_{j=1}^{j} Y(i,j,z,m) = Y(i,j,z-1,m) \, and \, for \, all \, i, m, z \, and \, j \, where \, fl(i,1) + fl(i,3) < z \, and \, fl(i,1) + fl(i,3) + fl(i,2) + fl(i,4) > z$: this ensures that once a flight is assigned to a gate it stays in that gate.
11. $\sum_{j=1}^{j} Y(i,j,z,m) = Q(i,j,m) \, and \, for \, all \, i, m, z \, and \, j \, where \, fl(i,1) + fl(i,3) < z \, and \, fl(i,1) + fl(i,3) + fl(i,2) + fl(i,4) > z$: this constraint fixes the gate to the flight in order to optimize it
12. In any other case, $Y(i,j,z,m) = 0$.

Even though these constraints seem like organized in a chaotic order, later on in the code entry I will show the order and explain how I implemented them.

OBJECTIVE FUNCTION

1. For the utopic case, the objective function to minimize will be $\sum_{i=1}^{i} \sum_{j=1}^{j} fl(i,6) * P(i,j) * D(j)$. This will inimize the total distance covered by all the passengers who use the airport.

2. For the case where we study the delays, the objective function will be $\sum_{i=1}^{i}\sum_{j=1}^{j} fl(i,6) * P(i,j,m) * D(j)$. This function has the same purpose as the previous one just differing in the delay of flights. Here m will represent the greatest value of index d, as it will be the one that takes into account all the delays that occur within our time of study.

CODE

The code is divided into five parts. First, we need to call the functions of the program we will need to use. After this, we will declare all the variables and assign them their values (in case of matrices and vectors their dimensions). Then, we will save into some vectors and matrices created the data collected on the case we are going to study. The next step will be to write all the constraints that are required for the program to carry out the optimization. Finally, we will display the results using the plotting tool. Now, I will show the code part by part:

In the initial part we find this:

```
model ModelName
uses "mmxprs"; !gain access to the Xpress-Optimizer solver
uses "mmsvg"   !gain access to the plotting function
```

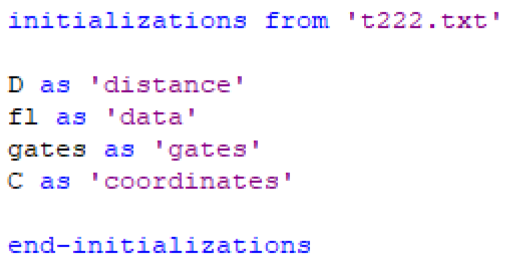
The function of this part of the code is to enable the access to the different tools to optimize (“mmxprs”) and plot (“mmsvg”).

Secondly, we need to write the variables:

```
!sample declarations section
declarations

    g=1..22 !number of gates
    f=1..58 !number of flights
    gates: array(g) of string !name of gates
    t=0..62 !total timesteps
    d=0..12 !number of delays
    fl: array(f,1..6) of integer !data of flights
    D: array(g) of integer !distance from entrance to gates
    C: array(g,1..2)of integer  !coordinates of the gates
    X: array(f,g,t) of mpvar !occupancy of gates in real time without delay
    P: array(f,g) of mpvar  !assignment of gates without delay
    K: array(f,g,t) of real  !bridge
    Y: array(f,g,t,d) of mpvar  !assignment of gates in real time with delays
    Q: array(f,g,d) of mpvar  !assignment of gates with delays

  Objective:linctr
end-declarations
```

In this part of the code we will need to establish the size of the problem, that wil be the number of flights, the number of gates, the number of timesteps and the number of delays. In this case, the matrix of solutions will have a size of 58 flights, 22 gates, 62 timesteps and 12 delays. The rest of the declarations are the matrices ad vectors that we will use in the next steps.

With all our variables created, now we need to start filling them out and we will start with the data collected:



```
initializations from 't222.txt'

D as 'distance'
fl as 'data'
gates as 'gates'
C as 'coordinates'

end-initializations
```

This step is used to initialize the information vectors and matrices. All the data that is being saved it is being read from a text file (t222) which obtains the information from an excel sheet (copy and paste from the excel sheet to the text file). For example, the text file will look like this:

```
distance:

[

345

346

282

283

…]


data:

[

0       1       0       0       0       164

0       1       0       0       0       110

0       3       0       0       0       110

1       4       0       0       0       180

…]
```

```
gates:

[

D43

D44

D45

D46

…]


coordinates:

[112    79

107     79

98      79

94      79

…]
```

In this example, the first position of the distance vector refers to the distance between the main entrance and the first gate in the gates vector, in this case is D43. The coordinates matrix indicates the horizontal and vertical position of this same gate in order to display it in a map of 400x400 (the image shown with the plotting tool is divided into 1600 squares).

The data matrix indicates the flight information. If we observe it, we can understand that the first flight will arrive at timestep 0 (it was already at the gate when out time of study starts) and it is leaving at timestep 1 without any delays (columns 3, 4 and 5 are equal to 0). Finally, it carries 164 passengers.

All of this information will be collected by the program and will organize the disposition of the boarding gates to satisfy the consumer and enable them to walk the minimum distance possible.

After we have set all the information that we needed, we proceed to write the main script. This will be separated in three parts. First, we will have the optimization of the utopic case:

```
forall(i in f, j in g, z in t)do !make X and P matrix binary
    X(i,j,z) is_binary
    P(i,j) is_binary
end-do

forall(j in g,z in t)do !constraint to make that each gate
    sum(i in f) X(i,j,z)<=1 !is assigned to only one flight at each time
end-do

forall(i in f, z in t)do  !constraint to make that each flight
    sum(j in g) P(i,j)<=1  !is only assigned to one gate
    sum(j in g) X(i,j,z)<=1
end-do

forall(z in t)do

forall(i in f| fl(i,1)<=z)do !formulation to assign gates

    if(fl(i,1)=z)then
        sum(j in g) X(i,j,z)=1
    elif((fl(i,1)<z) and ((fl(i,1)+fl(i,2))>z))then
        forall(j in g)do
            X(i,j,z)=X(i,j,z-1)
            P(i,j)=X(i,j,z)
        end-do
    else
        sum(j in g) X(i,j,z)=0
    end-if

end-do

end-do


O:=sum(x in f, y in g) fl(x,6)*P(x,y)*D(y)

minimize(O)
```

In order to write the code, we used the constraints described in the previous pages. The only part left to explain is the forall loop. The X matrix is like a picture of the status of the airport for each timestep. Therefore, the function of this loop is to take into account the timestep where the case is being studied and depending on whether the flight has just arrived, had already arrived or didn’t arrive yet it decides to assign a gate, assign the same gate that was given to it in the previous timestep or doesn’t assign a gate to it. After everything is ready we optimize it so we obtain the results of the utopic case.

The second part is where we prepare the transition from the utopic case to the delayed case:

```
forall(i in f, j in g, z in t)do
    K(i,j,z):=getsol(X(i,j,z))
end-do

forall(i in f, j in g, z in t, a in d)do
    Y(i,j,z,a) is_binary
    Q(i,j,a) is_binary
end-do

forall(j in g,z in t,a in d)do
    sum(i in f) Y(i,j,z,a)<=1
end-do

forall(i in f, z in t,a in d)do
    sum(j in g) Q(i,j,a)<=1
    sum(j in g) Y(i,j,z,a)<=1
end-do

m:=0

forall(i in f,j in g, z in t)do
    Y(i,j,z,m)=K(i,j,z)
end-do
```

Here we copy the X values to K values. This step is a formality, as Xpress solver does not allow to copy from one decision variable to another, we need to use K matrix as a bridge. Then we set the first constraints for matrices Y and Q and then we copy from K the solutions without delays to the index 0 of matrix Y.

Finally, following all the previous preparations, we will write the constraints for the delayed case and optimize it:

```
m:=1
n:=0

while(m<=12)do

p:=0

while(p<1)do

forall(i in f)do

    if(fl(i,5)=n and ((fl(i,3)>0) or (fl(i,4)>0)))then
        p:=1
    end-if

end-do

if(p<1)then
    n:=n+1
end-if

end-do

forall(i in f, j in g, z in t| z<n)do

    Y(i,j,z,m)=Y(i,j,z,m-1)

end-do

forall(z in t, i in f| fl(i,1)<=z)do

    if(fl(i,5)<=n and ((fl(i,3)>0) or (fl(i,4)>0)))then


         if(fl(i,1)+fl(i,3)=z)then
             sum(j in g) Y(i,j,z,m)=1
         elif((fl(i,1)+fl(i,3)<z) and ((fl(i,1)+fl(i,2)+fl(i,3)+fl(i,4))>z))then
             forall(j in g)do
                 Y(i,j,z,m)=Y(i,j,z-1,m)
                 Q(i,j,m)=Y(i,j,z,m)
             end-do
         else
             sum(j in g) Y(i,j,z,m)=0
         end-if

    else

         if(fl(i,1)=z)then
             sum(j in g) Y(i,j,z,m)=1
         elif((fl(i,1)<z) and ((fl(i,1)+fl(i,2))>z))then
             forall(j in g)do
                 Y(i,j,z,m)=Y(i,j,z-1,m)
                 Q(i,j,m)=Y(i,j,z,m)
             end-do
         else
             sum(j in g) Y(i,j,z,m)=0
         end-if

    end-if

end-do

M:=sum(x in f, y in g) fl(x,6)*Q(x,y,m)*D(y)

minimize(M)
```

```
m:=m+1

end-do

forall(z in t) do

writeln('time is ',z)

forall(x in f, y in g| getsol(Y(x,y,z,12))<>0) do
    writeln('Gate ',gates(y),' is occupied by flight ', x)
end-do

end-do
```

So we arrive to the most complex part of the code. At the beginning we can find m, n and p. M and n variables are used as counters. m will count the times at which delays that are being detected. This way if two delays are acknowledged at the same time, it will only count as one for m. For example, if m=2, this means that at time n there have been detected two timesteps at which there have been delays transmitted to the airport controllers, but there could be 4 delays already. n will count the timestep at which is being detected the delay and afterwards it will be used to copy all the solutions prior to that timestep from the previous matrix solutions (m-1). P will be used as a binary variable to show if a delay has been detected or not in that timestep.

When the program detects a delay (p=1), then it proceeds to copy the results previous to that timestep n [Y(x,y,z,m)=Y(x,y,z,m-1)]. This way the status of the airport in the moments before the acknowledgement of the delay will remain the same (obviously as it wasn't known before the airport cannot predict it). The reason to do this is to only change future decisions.

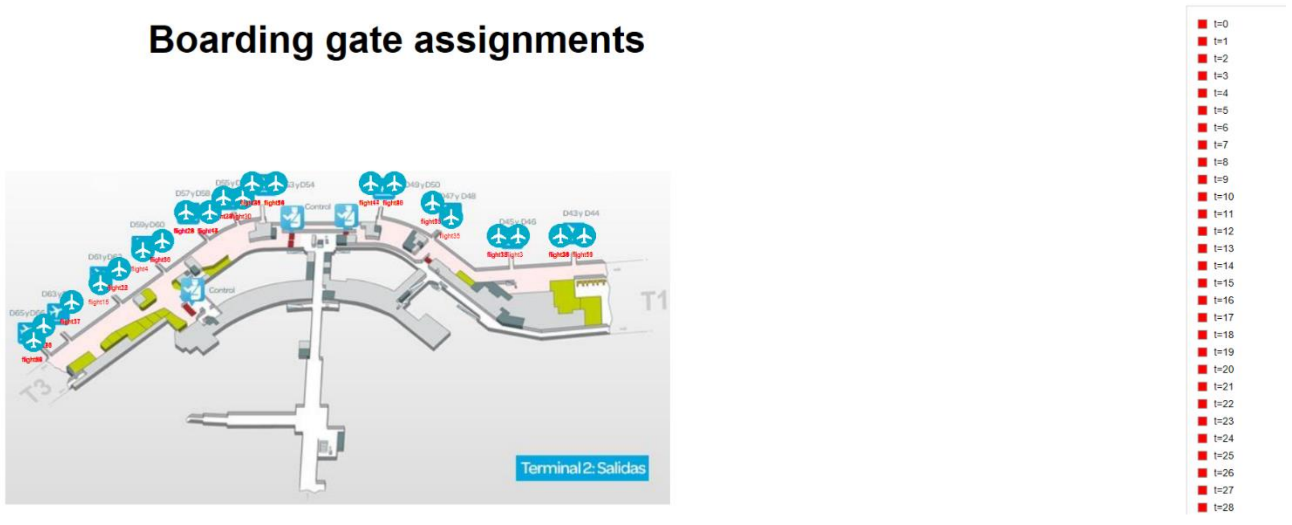
After everything in the previous timesteps is fixed, then we proceed to create the constraints. Here the program differentiates between flights with delays are flights on time. If the flight has delays (only delays that have been already acknowledged), it will optimize it using the information on landing and take off delays. Apart from that, the way to implement the constraints is the same as the utopic case. If it just landed it will assign a gate. If it had already landed it will still assign the same ate as before. If it hasn't landed it will not assign a gate. For flights with delays that have not been already detected, it will treat them as if they were on time flights (due to the fact that at that time it is thought that they are on time).

Finally, it will optimize the model and display in the output screen how the final status of the airport will look like.

The last part of the code contains the plotting script. It will transform the solution that is saved into the matrix Y when d reached its maximum value (all delays have been detected) into a visual solution. The code is written like this:

```
svgsetgraphviewbox(-10,-10,140,140)
svgsave('AIRPORT_T2.svg')
svgaddfile('T2-salidas.jpg', 'T2-salidas')
svgaddfile('avionsitos.png', 'avionsitos')

svgaddtext(20,120, 'Boarding gate assignments')
a:=svggetlastobj
svgsetstyle(a, SVG_FONTSIZE, '6pt')
svgsetstyle(a, SVG_FONTWEIGHT, 'bold')

svgaddimage('T2-salidas',-10,-10,140,140)

forall(z in 10..10)do

    svgaddgroup(''+z,'t='+z, SVG_RED)

    forall(i in f,j in g | getsol(Y(i,j,z,12))<>0)do

    svgaddimage('avionsitos', C(j,1)-2.5, C(j,2), 5,5)
    svgaddtext(C(j,1)-2.5, C(j,2)-2,'flight'+i)
    pepi:=svggetlastobj
    svgsetstyle(pepi, SVG_FONTSIZE, '1pt')

    end-do

end-do

svgrefresh
svgwaitclose

end-model
```

The display of the visual solution will showcase an image of the terminal T2 that is being studied and then there is the option of selecting the desired timestep and it will display only that timestep. There is also the option to see the display of different timesteps at the same time as well. Either way the image displayed will look like this:



As it can be appreciated, in the right side there is an index of the timesteps and by clicking on them they can be activated or deactivated. In the terminal image we can see how the airplanes have been located on their assigned gates with their name written below each airplane image.

The option of displaying different timesteps at the same time is really useful to double check that the airplanes are assigned to only one gate. It can be even corroborated that it follows the instructions of the delays that are saved in the text file. It is a tool that really enables the user to go over the solution in the easiest way possible. Also the visualization is optimal to examine any mistake that the program made.

ANALYSIS OF RESULTS

When I started this project, I set five goals that I wanted to achieve during the process of creating this code:

1. **Apply real airport data with delays to the case in order to solve real life problems that occurred in the past:** this was a task that due to the importance it has for the code is was the longest to achieve. This was caused by the transformation of real-life information into the data that my code demanded. For instance, the arrival time needed to be changed into the timesteps. It was really helpful that the airport web itself had all information on flights, including delays, and boarding gates.
2. **Optimize the assignment of boarding gates to airplanes to minimize the distance covered by passengers in T2 of Adolfo Suárez-Barajas airport:** this belongs to what I called the 'utopic case' in this document. It is the first optimization where the delays are not being processed. The hardest part of this optimization was getting used to the timesteps and once I was confident on understanding them everything came along faster than expected.
3. **Accomplish to write a code that from some data written on Excel can optimize the assignment of boarding gates to airplanes with their delays:** definitely this goal was more complicated than the previous one. Although some parts were similar I had a hard time trying to develop a system so that the delays were reflected properly in the code.
4. **Display the airport image situation for each moment in time:** this task was really satisfying to achieve as it facilitated a lot the visualization of the final results. I thought it helps to people to understand what the code is really doing and is a really helpful tool when it comes to comparing results or checking that everything is in its place.
5. **Transform the code from a simple optimizer to a real time optimize:** definitely, albeit the hardest, it is the task that I enjoyed the most. It has been really difficult to come up with this idea of implementing a fourth dimension in the Y matrix so it can optimize the code delay by delay so it is a real-time optimizer.

After proving that all the milestones set at the beginning of the project have been reached, I can conclude that the result is satisfactory.

CONCLUSIONS

METHODOLOGY

I decided to use Xpress IVE because in the previous semester I was able to solve an optimization problem of bus routes with this same program without any problem. The truth is that I felt really comfortable using it and I had real confidence when it came to use it due to my experience with it. Therefore, I went along with the decision of using Xpress. Probably there were other programs that would have made the problem smaller (the final matrix is made of 79112 results but in order to achieve it I need almost around a million solutions).

In relation with the way the code is written, I am really satisfied. As I wasn't able to overwrite the constraints, I was forced to reimagine the problem creating enormous matrices with more than two dimensions. I think I managed to achieve a well-structured program that leads to a real-time feasible solution.

RESULTS

The results I achieved I believe that were the ones my director asked for when he offered me the assignment. The program really optimizes in real time the assignment of boarding gates taking into consideration the delays that cannot be predicted due to unforeseen circumstances. Moreover, the ability to display the results to ease the task of study and comparison I think that adds value to the program. All over, the results fulfill the task I was awarded.

RECOMMENDATIONS FOR FUTURE STUDIES

In airports there are a lot of aspects which can be used as optimization: saving of gas, airport services (employees, cars, trucks…), etc. But I Think it would be interesting an optimization implementing the ability to overcome the malfunction of boarding gates, as sometimes they can break. This would take the program to the next level.

From my point of view, another improvement could be the management of more than one terminal or even the whole airport. It would create a huge group of datasets but as well would increase the quality.

And last but not least, the decrease of the size of the program so it can save space and be run in less time.

BIBLIOGRAPHY

Aena. (2019, Abril 10). *Aeropuerto Madrid-Barajas*. Retrieved from https://www.aeropuertomadrid-barajas.com/llegadas.html

Pérez, J. F. (2012). *Creación de un simulador aeroportuario. Sistema experto de asignación en el desembarque de pasajeros y generación de reglas a partir del conocimiento adquirido* . Madrid: Universidad Carlos III de Madrid .

APPENDICES

DATA COLLECTED

distance:

[

345

346

282

283

182

183

132

133

129

130

156

157

186

187

239

240

296

297

360

361

383

384

]

data:

[

0	1	0	0	0	164
0	1	0	0	0	110
0	3	0	0	0	110
1	4	0	0	0	180
1	3	10	10	0	180
2	3	0	0	0	180
3	3	0	1	3	164
4	2	0	0	0	110
8	3	0	0	0	165
9	3	0	0	0	166
10	4	1	1	12	212
10	4	0	0	0	85
11	3	0	0	0	132
12	3	1	2	10	180
13	3	0	1	14	190
13	3	0	0	0	199
13	3	0	0	0	164
13	9	0	0	0	164
14	3	0	1	15	164
17	1	0	2	18	164
18	7	0	0	0	120
19	4	0	0	0	120
19	6	0	0	0	299
19	5	1	1	17	120
19	6	0	0	0	196
20	5	1	0	18	299
20	4	0	0	0	186
20	4	0	0	0	156
20	13	0	2	25	174
21	4	0	1	23	180

```
22      4       0       0       0       166
22      3       0       0       0       180
24      3       0       1       25      186
25      4       0       2       27      186
27      4       0       0       0       180
27      6       2       1       25      166
29      4       0       0       0       270
30      3       0       0       0       164
30      4       0       0       0       253
31      4       1       1       29      141
31      4       0       0       0       165
31      4       0       0       0       180
32      3       0       0       0       164
34      4       0       0       0       186
35      3       0       1       36      190
37      4       0       0       0       85
38      4       0       0       0       156
39      3       0       0       0       110
39      4       0       0       0       180
41      2       0       0       0       110
41      5       0       0       0       144
43      3       0       0       0       164
43      5       0       0       0       190
45      4       0       0       0       186
46      3       0       0       0       145
46      3       0       0       0       141
49      3       0       0       0       132
54      3       0       0       0       110
]

gates:
27
```

```
[
D43
D44
D45
D46
D47
D48
D49
D50
D53
D54
D55
D56
D57
D58
D59
D60
D61
D62
D63
D64
D65
D66
]

coordinates:
[112    79
107     79
98      79
94      79
84      83
```

80	86
72	90
67	90
47	90
42	90
40	87
36	87
33	84
28	84
23	78
19	76
14	72
10	69
4	65
1	60
-2	60
-4	57

]

FULL CODE

```
!@encoding CP1252
model ModelName
uses "mmxprs"; !gain access to the Xpress-Optimizer solver
uses "mmsvg"   !gain access to the plotting function

!sample declarations section
declarations

        g=1..22 !number of gates
        f=1..58 !number of flights
        gates: array(g) of string !name of gates
```

```
	t=0..62 !total timesteps
	d=0..12 !number of delays
	fl: array(f,1..6) of integer !data of flights
	D: array(g) of integer !distance from entrance to gates
	C: array(g,1..2)of integer  !coordinates of the gates
	X: array(f,g,t) of mpvar !occupancy of gates in real time without delay
	P: array(f,g) of mpvar  !assignment of gates without delay
	K: array(f,g,t) of real  !bridge
	Y: array(f,g,t,d) of mpvar  !assignment of gates in real time with delays
	Q: array(f,g,d) of mpvar  !assignment of gates with delays

  Objective:linctr
end-declarations

initializations from 't222.txt'

D as 'distance'
fl as 'data'
gates as 'gates'
C as 'coordinates'

end-initializations

forall(i in f, j in g, z in t)do !make X and P matrix binary
	X(i,j,z) is_binary
	P(i,j) is_binary
end-do

forall(j in g,z in t)do !constraint to make that each gate
	sum(i in f) X(i,j,z)<=1 !is assigned to only one flight at each time
end-do

forall(i in f, z in t)do  !constraint to make that each flight
	sum(j in g) P(i,j)<=1  !is only assigned to one gate
```

```
        sum(j in g) X(i,j,z)<=1
end-do

forall(z in t)do

forall(i in f| fl(i,1)<=z)do !formulation to assign gates

        if(fl(i,1)=z)then
                sum(j in g) X(i,j,z)=1
        elif((fl(i,1)<z) and ((fl(i,1)+fl(i,2))>z))then
                forall(j in g)do
                        X(i,j,z)=X(i,j,z-1)
                        P(i,j)=X(i,j,z)
                end-do
        else
                sum(j in g) X(i,j,z)=0
        end-if

end-do

end-do

O:=sum(x in f, y in g) fl(x,6)*P(x,y)*D(y)

minimize(O)

forall(i in f, j in g, z in t)do
        K(i,j,z):=getsol(X(i,j,z))
end-do

forall(i in f, j in g, z in t, a in d)do
        Y(i,j,z,a) is_binary
        Q(i,j,a) is_binary
end-do
```

```
forall(j in g,z in t,a in d)do
        sum(i in f) Y(i,j,z,a)<=1
end-do

forall(i in f, z in t,a in d)do
        sum(j in g) Q(i,j,a)<=1
        sum(j in g) Y(i,j,z,a)<=1
end-do

m:=0

forall(i in f,j in g, z in t)do
        Y(i,j,z,m)=K(i,j,z)
end-do

m:=1
n:=0

while(m<=12)do

p:=0

while(p<1)do

forall(i in f)do

        if(fl(i,5)=n and ((fl(i,3)>0) or (fl(i,4)>0)))then
                p:=1
        end-if

end-do

if(p<1)then
```

```
        n:=n+1
end-if

end-do

forall(i in f, j in g, z in t| z<n)do

        Y(i,j,z,m)=Y(i,j,z,m-1)

end-do

forall(z in t, i in f| fl(i,1)<=z)do

        if(fl(i,5)<=n and ((fl(i,3)>0) or (fl(i,4)>0)))then

                if(fl(i,1)+fl(i,3)=z)then
                        sum(j in g) Y(i,j,z,m)=1
                elif((fl(i,1)+fl(i,3)<z) and ((fl(i,1)+fl(i,2)+fl(i,3)+fl(i,4))>z))then
                        forall(j in g)do
                                Y(i,j,z,m)=Y(i,j,z-1,m)
                                Q(i,j,m)=Y(i,j,z,m)
                        end-do
                else
                        sum(j in g) Y(i,j,z,m)=0
                end-if

        else

                if(fl(i,1)=z)then
                        sum(j in g) Y(i,j,z,m)=1
                elif((fl(i,1)<z) and ((fl(i,1)+fl(i,2))>z))then
                        forall(j in g)do
                                Y(i,j,z,m)=Y(i,j,z-1,m)
                                Q(i,j,m)=Y(i,j,z,m)
                end-do
            else
                sum(j in g) Y(i,j,z,m)=0
            end-if

        end-if

end-do

M:=sum(x in f, y in g) fl(x,6)*Q(x,y,m)*D(y)

minimize(M)

m:=m+1

end-do

forall(z in t) do

writeln('time is ',z)

forall(x in f, y in g| getsol(Y(x,y,z,12))<>0) do
        writeln('Gate ',gates(y),' is occupied by flight ', x)
end-do

end-do

svgsetgraphviewbox(-10,-10,140,140)
svgsave('AIRPORT_T2.svg')
svgaddfile('T2-salidas.jpg', 'T2-salidas')
svgaddfile('avionsitos.png', 'avionsitos')

svgaddtext(20,120, 'Boarding gate assignments')
a:=svggetlastobj
```

```
svgsetstyle(a, SVG_FONTSIZE, '6pt')
svgsetstyle(a, SVG_FONTWEIGHT, 'bold')

svgaddimage('T2-salidas',-10,-10,140,140)

forall(z in 10..10)do

        svgaddgroup(''+z,'t='+z, SVG_RED)

        forall(i in f,j in g | getsol(Y(i,j,z,12))<>0)do

        svgaddimage('avionsitos', C(j,1)-2.5, C(j,2), 5,5)
        svgaddtext(C(j,1)-2.5, C(j,2)-2,'flight'+i)
        pepi:=svggetlastobj
        svgsetstyle(pepi, SVG_FONTSIZE, '1pt')

        end-do

end-do

svgrefresh
svgwaitclose

end-model
```