Lab assignment 1:
Linear Programming

Heuristics and Optimization
2020-2021

Eva Huertas Maldonado
100405799 G88

Carlos Samper Prieto
100383357 G88

Table of contents

1. Introduction 3

2. Problem 1 4

2.1. Overview. 4

2.2. Modelling 4

2.2.1. Decision variables 4

2.2.2. Objective function 4

2.2.3. Constraints 5

2.3. Analysis and Conclusion 6

3. Problem 2 8

3.1. Overview 8

3.2. Modelling 8

3.2.1. Decision variables 8

3.2.2. Objective function 8

3.2.3. Constraints 9

3.3. Analysis and Conclusion 10

4. Final Problem 12

4.1. Modelling 12

4.1.1. Objective function 12

4.2. Analysis and Conclusions 12

1. Introduction

In this document we will be explaining the steps taken to solve the problems proposed in the first assignment.

We will specify what the problem is intended for, the different parts of it and the Linear Programming model used to implement it . Moreover, we will discuss the final results obtained after solving each task. After finishing with the explanation, we will analyse the final result and the usefulness of the tools utilized during this assignment, talking about different scenarios that could happen and would change the model slightly, like adding more planes and eliminating runaways. This means that we will also include anything we consider worth mentioning about any part of the assignment on this section.

These techniques are going to be implementing for the first and second problem. The very last part of problem 2 requires us to merge both models in order to obtain the "real" final results of the lab, as both problems are related. This will be discussed separately at the end of the report, after interpreting the results of both problems separately.

Finally, in the conclusion we will include our personal opinion about the lab, as well as the different things we have learnt.

Before introducing problem 1, it is important to clarify that this report by no means represents a model to be implemented on neither LibreOffice calc nor MathProg, only the formal definition of it.

2. Problem 1

2.1. Overview.

In this problem we are asked to **maximize** the profit acquired from the sale of tickets pertaining to five airplanes. Each with different passenger counts and carrying capacities. There are three different types of ticket, each with a different price.

To correctly solve this problem, we will have to calculate how many tickets of each type will be sold for each plane, and then get the profit from those results. We need to consider that the linear programming task is of integer type, since you cannot sell one and a half tickets to someone. This limitation, along with various constraints, like the number of people and capacity in each plane will limit the solution of our model.

2.2. Modelling

We will now define the problem formally, presenting the model used to solve the linear programming task.

2.2.1. Decision variables

First of all, we have to define the decision variables. For that purpose, we consider the number of tickets of each type for each plane and the number of planes.

The three types of tickets are:

- Standard tickets
- Leisure plus tickets
- Business plus tickets

There are 5 planes:

-AV1 -AV2 -AV3 -AV4 -AV5

From this we conclude that our decision variables are defined as the elements of a 5x3 matrix, with the planes being the rows and the tickets types the columns:

	Standard	Leisure plus	Business plus
AV1	x_{11}	x_{12}	x_{13}
AV2	x_{21}	x_{22}	x_{23}
AV3	x_{31}	x_{32}	x_{33}
AV4	x_{41}	x_{42}	x_{43}
AV5	x_{51}	x_{52}	X_{53}

2.2.2. Objective function

With the decision variables known, we can finally define the objective function (z) . In this particular case, we want to **maximize** the profit. That maximization will be achieved by adding the amount of tickets of each type on all planes and then multiplying those additions by their respective cost (19€ for standard, 49€ for leisure plus and 69€ for business plus).

With that, the z function will be defined as:

$$\max z = 19 * \sum_{i=1}^{5} x_{i1} + 49 * \sum_{i=1}^{5} x_{i2} + 69 * \sum_{i=1}^{5} x_{i3}$$

2.2.3. Constraints

Without any constraints, this problem becomes trivial, since you would only need to make all the tickets for every plane business and add, that is the reason why there are constraints added to the problem.

To make the constraints easier to understand, we will include the statement that defines them and an explanation on how we have understood and implemented them.

- *"It is not allowed to sell more tickets for an airplane than its number of available seats."*

To implement this constraint, we have had to get the number of seats of each plane (data provided in the statement (Table 2)), and make sure that the addition of the tickets for every type did not surpassed that amount for each plane. Therefore, we had to create this constraint five times, one for each plane.

$$\sum_{j=1}^{3} x_{ij} \leq SeatsPerAirplane$$

**Notice how we sum the entries by columns for each corresponding row. In that way we are calculating the number of passengers for each plane. (SeatsPerAirplane corresponds to the total number of airplane seats of the column Seats of table 2).*

$$Example: \sum_{j=1}^{3} x_{1j} \leq 90$$

*(We will only include one of the constraints as an example because all five would take too much space. All constraints that must be divided into various equations will also be presented this way.)

- *"It is strictly forbidden to exceed the maximum capacity of each airplane."*

Each plane has a maximum carrying capacity, also provided in the statement (Table 2), and each type of ticket has a maximum weight they can get in the plane, so we have to multiply the weight allowed for each ticket type by the number of tickets of that type and the add all the results.

$$x_{ij} + 20 * x_{ij+1} + 40 * x_{ij+2} \leq CapacityPerPlane$$

**Notice how we sum the entries by columns for each corresponding row previously multiplied by its capacity. In that way we are calculating the capacity for each plane. (CapacityPerAirplane corresponds to the total number of airplane capacity of the column Capacity of table 2).*

$$Example: x_{11} + 20 * x_{12} + 40 * x_{13} \leq 1700$$

- *"At least, 20 leisure plus airplane tickets should be offered for each airplane, and at least 10 business plus airline tickets for each airplane as well."*

This particular constraint is divided into ten different ones, five for the leisure plus tickets of each plane and five for the business plus. To implement it we have to consider the Leisure Plus and Business Plus tickets for each plane:

$$x_{i2} \geq 20$$

** This is the general form of the inequation for Leisure Plus tickets, where i is the row of the matrix(plane for which we are calculating the inequation).*

The inequation of the business plus tickets remains mostly the same, with a few changes:

$$x_{i3} \geq 10$$

** This is the general form of the inequation for Business Plus tickets, where i is the row of the matrix(plane for which we are calculating the inequation).*

- *"Because this is a low-cost company, the number of standard air tickets should be at least the 60 % of the overall number of airline tickets offered."*

This constraint forces the number of standard tickets being at the very least 60% of the total amount of tickets, not for every plane, but for the entire company. To calculate this, we created this constraint:

$$\frac{\sum_{i=1}^{5} x_{i1}}{\sum_{i=1}^{5}(\sum_{j=1}^{3} x_{ij})} \geq 0.6$$

2.3. Analysis and Conclusion

Here we will analyse the results, why the solution is how it is and any extra details that may need considering.

Due to the nature of this problem the main constraints are the capacity and the number of people in each plane. The constraints related to the amount of tickets sold helps to maximize benefits.

Another thing to consider is the percentage of standard tickets that must be sold. If this percentage was lower, more incomes with the same number of people could be obtained. For instance, if the percentage was reduced to 20% there would be an increase of 1040€ with respect to our actual result.

The result of this problem is 26190€, with most of the planes at full capacity and full of clients. The number of standard tickets is 450 which is exactly the 60% of the total number of airline passengers. This joined with the minimum number that must be sold to Leisure and Business Plus in the original terms helps to increase the margin of benefits.

This happens because we want to maximize the profit of the company, so we will have the maximum amount possible of tickets of non-standard fairs to earn more money. This way not only will we have the maximum amount of clients, but the planes will also be almost, if not fully, loaded.

Moreover, every time we have calculated the objective function in LibreOffice, we have observed that although the maximize benefit remains the same, as well as the total number of Standard, Leisure Plus and Business Pluss tickets sold. There were different values for each of the matrix decision variables. This clearly shows how the different solvers can lead to different solutions for the problem, allowing us to know for sure that the solution set for this one is indeed larger than a unique solution.

3. Problem 2

3.1. Overview

In the second problem we are given five planes and several tables. The first one shows the runaways and slots planes can occupy. In addition, it shows the times for each available slot. Another table shows the schedule time for planes landing and the maximum time they can be on air, as well as the money the delays cost to the company. We are asked to minimize the amount of money spent during plains delays.

To solve this problem, we will calculate the time each plane is in the air before arriving at a runaway and multiply that amount by the cost of keeping them flying per minute. To do so we will first have to take every single time stamp in the problem and transform it into minutes.

Lastly, we are asked to take this problem and fuse it with the previous one so that both of them are solved at the same time with the GLPK solver.

3.2. Modelling

As in the previous problem, we will now present the problem formally and define the model used to solve it.

3.2.1. Decision variables

To calculate the minimization, we used a matrix of binary numbers with the rows as the planes and the columns as the available slots in the different runaways.

$$x = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x_{19} & x_{110} & x_{111} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} & x_{28} & x_{29} & x_{210} & x_{211} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} & x_{38} & x_{39} & x_{310} & x_{311} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} & x_{48} & x_{49} & x_{410} & x_{411} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} & x_{58} & x_{59} & x_{510} & x_{511} \end{pmatrix}$$

This is a binary matrix, where x_{ij} will only take the values 1 or 0. 1 if the slot is occupied by a plane and 0 if not. We will use that to compute the objective function easily.

3.2.2. Objective function

With the decision variables already defined, we can model the objective function. We want to **minimize** the cost of keeping the planes in the air, so we have to calculate the time they spend on the air until they arrive at the airport and land in each slot. Then we multiply that by the corresponding column of the matrix and after that multiply it by the cost per minute, so that only the times that have been chosen, the ones multiplied by the ones in the matrix, are counted.

$$\min z = (\sum_{i=1}^{5}(\sum_{j=1}^{11} x_{ij} * (slotTime_j - arivalTime_i)) * cost_i)$$

Being *cost* the specific cost per minute in the air for each plane, *slotTime* the time at which each slot starts and *arrivalTime* the time at which the plane arrives at the air space of the airport.

3.2.3. Constraints

As in the previous part, we will take the statements for the constraints and copy them here along with how we understood and implemented them.

- *"Every airplane should be assigned one slot for landing."*

This constraint was easy to model, since it was a matter of taking the matrix and making sure that each of the rows of the matrix only had one x that was less or equal to1

$$\sum_{j=1}^{11} x_{ij} \leq 1$$

This constraint has been divided into six, one for each plane and a special one to make sure that the number of total slots occupied is just five. The one above is a generalization of the first five constraints, with i changing depending on which airplane we are calculating and the one below is the sixth one, to make sure the addition is just five and there are no planes which still have not landed.

$$\sum_{i=1}^{5} \left(\sum_{j=1}^{11} x_{ij} \right) = 5$$

- *"No more than one slot has to be assigned to each airplane."*

This constraint is the same as the last one with a small difference, instead of making sure the rows have a 1, we make sure that the columns have at most one 1.

$$\sum_{i=1}^{5} x_{ij} \leq 1$$

As before, this formula is a generalization of the constraint, since there are eleven slots and j is the letter we use to refer to all of them, as in the constraint before.

- *"Assigned slots should be available, i. e., shown in green in Figure 1."*

For this constraint we have taken a different approach. We have taken each of the slots available and used them as sets to calculate the x in the problem, making them the columns of the decision variables matrix. This way we do not have to create a matrix of runaways and times and just have to consider the number of available slots and their starting time.

- *"The starting time of an assigned slot shall be either equal or greater than the scheduled landing time."*

This constraint forces the planes to land in the available slots only after they have arrived at the airport, making it so that the planes cannot take a slot without even being there in the first place. It also means that the time at which the planes land must be the time of beginning of an available slot. Since all the planes arrive five minutes before the starting time of the next slot, this time has to be grater or equal to five.

$$\sum_{j=1}^{11} x_{ij} * (slotTime_j - arivalTime_i) \geq 5$$

As with the ones above, the i on the x is a generalization of the row since there are five of these constraints.

- *"The starting time of an assigned slot shall be either less or equal than the maximum allotted landing time."*

This constraint is limiting our time so that no plane stays in the air after running out of gas. It is modelled similarly to the previous one since the left part of the inequation is the same.

$$\sum_{j=1}^{11} x_{ij} * (slotTime_j - schedulelandingTime_i) \geq maxLandingTime_i - schedulelandingTime_i$$

The difference is easily spotted. It is the change that has happened in the right part of the inequation. The *maxLandingTime* is the generalization for the time limits each plane has, and that appears in Table 3.

- *"For safety reasons it is not allowed to assign to consecutive slots in the same track."*

Since we have taken the available slots as the columns of the 5x11 matrix, this constraint is simply making sure that the columns of the consecutive slots only add up to 1.

$$\sum_{i=1}^{5} x_{ij} + x_{ij+1} \leq 1$$

$$Example: \sum_{i=1}^{5} x_{i1} + x_{i2} \leq 1$$

** This is an example of the first pair of consecutive slots, the first and the second, and the rest of the possibilities will be the same with the only difference being the change in their column number.*

3.3. Analysis and Conclusion

After having solved the problem, it's time to interpret the solution obtained as we did with the first part.

For this part we have to understand how the constraints affect the assignment of airplanes to slots. The most restrictive constraints are the ones that limit strictly the assignment of the planes to the slots like the first and the second constraint. After carrying out some tests to the program, we have realized that $\sum_{i=1}^{5}\left(\sum_{j=1}^{11} x_{ij}\right) = 5$ doesn't alter the optimal result at all. So, in an improved model it could be perfectly expendable. That is because

the firs, the second and the last constraints already limits the possible positions planes can have in our slot.

Subsequently, we have the constraints related to the time planes should arrive. These constraints will have a great impact on the amount of money the company will spend. If we eliminate $\sum_{j=1}^{11} x_{ij} * (slotTime_j - arivalTime_i) \geq 5$ from the constraints, the company will reduced costs as planes will land is slots which available time wasn't greater to scheduling time making an improvement of resources.

This table shows the solutions carry out in different implementations. Where the light blue, shows the slots occupied by planes in the first iteration. The light yellow, shows the slots occupied by planes in the second iteration. The light green shows the slots in which planes have landed in both iterations. With these results, we can deduce that as in the first problem there is not a unique solution. Nonetheless, the objective function always gives the same result, 4500€.

	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8	Slot 9	Slot 10	Slot 11
AV 1											
AV 2											
AV 3											
AV 4											
AV 5											

*AV stands for planes.

This is easy to calculate because this problem can be easily solved simply by looking at the different tables you are given, and by making every plane land just five minutes after they arrive, the minimum amount of time needed. In that way, you get the best solution.

If there are delays, planes can be reassigned. For example if the first plane (AV1) has a delay of 20 min AV2 can be assign to slot 9, AV3 can be assign to slot 4, AV4 can be assign to slot 1, AV5 can be assign to slot 11 and AV1 could be assign to slot 7. If more delays take place and reassigning is not possible more runways with available slots should be added. In the matrix this will lead to the addition of just the available slots.

4. Final Problem

The final model is a combination of the first problem, which maximizes the total benefits the airline company obtain from the selling of tickets, and the second problem which minimizes the cost of airplanes. Both problems are to be merged into a single model and objective function which maximizes the benefits of the company.

4.1. Modelling

4.1.1. Objective function

This merge gives us the next objective function as result:

$$\max z = \left(19 * \sum_{i=1}^{5} x_{i1} + 49 * \sum_{i=1}^{5} x_{i2} + 69 * \sum_{i=1}^{5} x_{i3}\right) - \left(\sum_{y=1}^{5}(\sum_{z=1}^{11} x_{yz} * (slotTime_z - arivalTime_y)) * cost_y\right)$$

The result of this function is 21690€ and can be easily check by subtracting the result of the second function to the first one.

The constraints applied are the union of the two sets of constraints and remain unchanged and the decision variables remain unchanged as well.

The only thing that is going to suffer a bigger transformation is the objective function. In which we have to join both objective functions into a single one to maximize the benefits of the company instead of having two different ones.

4.2. Analysis and Conclusions

Now we are going to discuss the general results of the complete problem, but firstly, we will add some interesting things we found while carrying this merging of models, regarding problem part 1 and 2.

In part 1, we provided a model to solve using LibreOffice Calc, which gave us an optimal solution to the problem by allocating the extras we had in a specific way. In that part, we clearly specify the existence of several values for the decision variables that lead to an optimal value. However, this is firmly confirmed when translating the model implementation into MathProg and solving it. We noticed that despite the objective function leading to the same result, the passengers were allocated in different planes with different tickets type here too. This clearly shows how the different solvers can lead to different solutions for the problem, allowing us to know for sure that the solution set for this one is indeed larger than a unique solution.

In part 2, we want to underline the nature of the problem. This as we have already seen in class is an “assignment problem” where a plane just can be assigned to a single slot and vice versa per resolution. This is clearly seen with the restrictions. If we want an individual to be assign to 1 specific place, the summation over all tasks must be equal to 1. Moreover, while considering one specific place, if we sum over all the different feasible assignments of planes to it , this must be one.

When talking about the complexity of the final project we realize several things. We have in total 37 constraints, one which is unnecessary, as previously stated. 5 come from the first problem with 5 planes and 3 tickets types matrix 5x3 as decision variables. The other 32 constraints come from the second part with a 5x11 matrix of decision variables where 5 stands for the planes and 11 stands for the slots. Interestingly if we add more planes or tickets to the model, the problem complexity would not increase much. However, in the second part if we increase the number of planes, we should add more constraints for checking the adequate arriving time, increasing its complexity.

Finally, we will formalize the strengths and weaknesses as well as our opinion of the different tools used in the practice.

LibreOffice Calc provides an attractive interface to model a linear programming problem. The resolution of problems as well as the spot of solutions become clearer and more visual when painting the different spreadsheet regions. Moreover, it allows you to place decision variables , constraints, resources , costs and objective function. Also, the solver is very intuitive and efficient at showing the output of the problem. However, LibreOffice Calc requires you to assign to each cell a decision variable which can become a tedious process when having a large number of decision variables.

With MathProg there is no need to modify the model programmed every time a change is required as it allows you to separate the problem modelling from its data. The biggest benefit of using MathProg are the sets. These data structures allow you combine data in regard to a relation. Sets are traverse through iterators. All these functionalities combined make this tool really useful when working with many decision variables. MathProg offers many more tools, like complete programming language expressions and the fact it can be used in other programs as a callable programming library. Overall, this makes MathProg a well-equipped tool to solve linear programming tasks. However, for small problems it might be more useful to use LibreOffice Calc for the visual presentation of the problem.