REPORT

Heuristics and Optimization

Modelling

Course 2020/2021

DEGREE IN COMPUTER ENGINEERING

GROUP 88

Carlos Gallego Jiménez 100405937
Andrés Langoyo Martín 100405869

Carlos Gallego Jiménez 100405937
Andrés Langoyo Martín 100405869

TABLE OF CONTENTS

1. INTRODUCTION

The purpose of this document is to describe the work done for the first lab assignment of the course Heuristic and Optimization, that is related with modeling linear programming tasks. Firstly, the models proposed to solve the stated problems will be described. Then, the results obtained after applying the models are discussed and, finally, some conclusions regarding the development of the practice are mentioned.

2. DESCRIPTION OF THE MODELS

2.1 Part 1

2.1.1 Decision Variables

To solve this problem, it was decided to use integer decision variables to represent the amount of tickets of each kind for each plane. The decision variables are described in the following manner:

$$X_{ij}\ ,\ i \equiv plane\ number;\ j \equiv type\ of\ ticket$$

The number of planes needed for the problem are 5 and the amount of tickets are 3. That leaves us with a total of 15 decision variables.

$$i \in \{1,2,3,4,5\};\ j \in \{1,2,3\}$$

In the tickets, 1 stands for standard ticket, 2 stands for leisure plus ticket and 3 stands for business plus ticket.

2.1.2 Constraints

For this problem, and according to the statement we formulated the following constraints that are shown below the given text requirements.

$Parameter \setminus Fair(j)$	1	2	3
Price (€)	19	49	69
Allowed Baggage (kg)	1	20	40

Figure 1. Price and baggage depending on fair**.**

$Parameter \setminus Plane(i)$	1	2	3	4	5

Seats	90	120	200	150	190
Capacity (kg)	1700	2700	1300	1700	2000

Figure 2. Seats and capacity depending on plane**.**

Constraint 1.It is not allowed to sell more tickets for an airline than its number of available seats.

$$\sum_{i=1}^{5}\sum_{j=1}^{3} X_{ij} \leq 750$$

Constraints 2. It is strictly forbidden to exceed the maximum capacity of each airplane.

$$\sum_{j=1}^{3} X_{ij} \bullet Allowed\ Baggage_j \leq Capacity_i \quad \forall\ i \in \{1,2,3,4,5\}$$

Constraint 3. 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.

$$X_{i2} \geq 20\ , \quad \forall\ i \in \{1,2,3,4,5\}$$
$$X_{i3} \geq 10\ , \quad \forall\ i \in \{1,2,3,4,5\}$$

Constraint 4. Because this is a low-cost company, the number of standard air tickets should be at least 60 % of the overall number of airline tickets offered.

$$\sum_{i=1}^{5} X_{i1} \leq 750 \bullet 0.6$$

Extra constraint. It is needed to include some extra constraints to avoid exceeding the maximum capacity of each plane.

$$\sum_{j=1}^{3} X_{ij} \geq 10 \quad \forall\ i \in \{1,2,3,4,5\}$$

2.1.3 Objective function

After declaring our decision variables, it is needed to maximize the profit obtained by selling the tickets:

$$max\ Z = \sum_{i=1}^{5}\sum_{j=1}^{3} X_{ij} \bullet Price_j$$

2.2 Part 2

2.2.1 Decision Variables

In this case, a similar approach is followed. The variables defined are binary variables and represent whether each combination of runway, slot and plane is taken or not. The decision variables are described in the following manner:

$$X_{rps}\ ,\ r \equiv runway\ ;\ p \equiv plane;\ s\ \equiv slot$$

In this problem, there are 4 runways, 5 planes and 6 slots. That amounts to 120 binary variables.

$$r \in \{1,2,3,4\}\ ;\ p \in \{1,2,3,4,5\}\ ;\ s \in \{1,2,3,4,5,6\}$$

2.3.2 Constraints

For this problem, and according to the statement we formulated the following constraints that are shown below the given text requirements.

$Runway\ (r) \setminus Slot\ (s)$	9:00-9:15	9:15-9:30	9:30-9:45	9:45-10:00	10:00-10:15	10:15-10:30
R1	0	0	0	0	1	1
R2	0	0	1	1	0	0
R2	1	1	1	0	0	1
R3	1	0	0	0	1	1

Figure 3. Slots_available(r,s) depending on runway and time slot.

$Plane\ (p) \setminus Parameter$	Arrival_time(min)	Maximum_landing_time (min)	Price_Penalty(€/min)
1	9:10 = 550	10:15 = 615	100
2	8:55 = 485	9:30 = 570	200
3	9:40 = 580	10:00 = 600	150
4	9:55 = 595	10:15 = 615	250
5	10:10 = 610	10:30 = 630	200

Figure 4. Arrival_time, maximum_landing_time and price penalty depending on plane.

Constraint 1. Every airplane should be assigned one slot for landing.

Constraints 2. No more than one slot has to be assigned to each airplane

Only one slot should be assigned to each plane so all the variables that relate to a plane must add up to one. This way we ensure that both first and second constraints are fulfilled at the same time.

$$\sum_{r=1}^{4} \sum_{s=1}^{4} X_{rps} = 1 \quad \forall\, p \in \{1,2,3,4,5\}$$

Extra constraint. We understand that the first constraint and second constraints are not enough and that one additional constraint is needed to check that two planes have not been assigned the same slot. To do that, we need to get that the addition of planes that can be in a slot is maximum 1:

$$\sum_{p=1}^{5} X_{rps} = 1 \quad \forall r \in \{1,2,3,4\},\ \forall s \in \{1,2,3,4,5,6\}$$

Constraint 3. Assigned slots should be available.**Figure 3**
Initially we tried to specify for each runway which slots were not allowed for landing. We did that by forcing the addition of all the slots in the same runway to be equal to zero. We defined the following constraints:

$$\sum_{p=1}^{5} \sum_{s=1}^{4} X_{rps} = 0\ ,\ r = 1$$

$$\sum_{p=1}^{5} \sum_{s=1}^{2} X_{rps} = 0\ ,\ r = 2$$

$$\sum_{p=1}^{5} \sum_{s=5}^{6} X_{rps} = 0,\ r = 2$$

$$\sum_{p=1}^{5} \sum_{s=5}^{6} X_{rps} = 0\ ,\ r = 3$$

$$\sum_{p=1}^{5} \sum_{s=2}^{4} X_{rps} = 0\ ,\ r = 4$$

As we could not find a way to implement this in Mathprog we opted for a more general approach, depending on parameters:

$$\sum_{p=1}^{5} X_{rps} \leq Slot\_available_{rs}$$

$$\forall r \in \{1,2,3,4\},\ \forall\ s \in \{1,2,3,4,5,6\}$$

Constraint 4. The starting time of an assigned slot shall be either equal or greater than the scheduled landing time.
Just as before, we specified the slots for each plane that are not possible to be assigned, in this case, because the start time of the slot happens before the starting time. So we iterate through every slot indicating which ones should add up to 0 taking into account that slots are different for every plane

$$\sum_{r=1}^{4}\sum_{s=1}^{1} X_{rps} = 0 , p = 1$$
$$\sum_{r=1}^{4}\sum_{s=1}^{3} X_{rps} = 0, p = 3$$
$$\sum_{r=1}^{4}\sum_{s=1}^{4} X_{rps} = 0 , p = 4$$
$$\sum_{r=1}^{4}\sum_{s=1}^{5} X_{rps} = 0 , p = 5$$

As we couldn't find a way to implement this in Mathprog, we opted for a more general and simple approach. Here, for each variable that is assigned, the Arrival time of a plane has to be less or equal than the starting time of the slot.

$$\sum_{r=1}^{4}(Arrival\_time_p - Start\_ time_s) \bullet X_{rps} \leq 0$$
$$\forall p \in \{1,2,3,4,5\}, \forall s \in \{1,2,3,4,5,6\}$$

Constraint 5.The starting time of an assigned slot shall be either less or equal than the maximum allotted landing time.
In the same fashion as in the previous constraint, we tried to specify the slots in which each plane would run out of fuel:

$$\sum_{r=1}^{4}\sum_{s=4}^{6} X_{rps} = 0 , p = 2$$
$$\sum_{r=1}^{4}\sum_{s=6}^{6} X_{rps} = 0, p = 3$$

The final modeling design for this constraint was:

$$\sum_{r=1}^{4}(start\_time_s - max\_time_p) \bullet X_{rps} \leq 0$$
$$\forall p \in \{1,2,3,4,5\}, \forall s \in \{1,2,3,4,5,6\}$$

Constraint 6.For safety reasons it is not allowed to assign to consecutive slots in the same track.

Here we want to avoid two consecutive slots from being taken at the same time. For this reason we have created a matrix, **consecutive(c,t),** in which there are 5 groups consisting of couples of
consecutive slots.

Couple (c) \ *Slots* (t)	1º	2º	3º	4º	5º
Slot 0	1	2	3	4	5
Slot 1	2	3	4	5	6

Figure 5. Table representing consecutive slots. **consecutive(c,t)**

Now we create a constraint for every combination of runway and group of consecutive slots. In each constraint, we make sure that at most one plane is assigned to each couple.

$$\sum_{p=1}^{5} \sum_{t=0}^{1} X_{r\,p\,consecutive\,[c][t]} = 0 \;\; \forall r \in \{1,2,3,4\}, \;\; \forall c \in \{1,2,3,4,5\}$$

2.3.3 Objective function

It is needed to obtain the lowest possible price penalty so we have to minimize. The price penalty is calculated by obtaining the delay of each plane, which is the difference in time between the time that the slot starts and the time the plane is supposed to land. Once we have that difference in minutes, it is multiplied by each plane's penalty.

In the objective function we iterate for every combination of variables but only calculate for the variables that have been assigned. That is why the formula is multiplied by X_{rps}.

$$min\ Z = \sum_{r=1}^{4} \sum_{p=1}^{5} \sum_{s=1}^{6} X_{rps} \bullet [\ (Slot\_start - Arrival\_time_p) \bullet Price\_penalty_p\ ]$$

2.3 Combination of problems

When combining the first and second problems into a single model, we realized that each problem was independent. Then, the model would be defined by simply using together the decision variables and the constraints defined in the first and the second models. The only difference would be in the objective function, which combines both of them.

Secondly, after analysing each problem we discovered that we only needed to include in the data section one set of Planes as both problems could use it without repercuting in the final result.

For the objective function, as the first problem was maximizing and the second was minimizing. To solve this, we added both objective functions and inverted the sign of the second objective function.

Carlos Gallego Jiménez 100405937
Andrés Langoyo Martín 100405869

$$\max Z = \left(\sum_{i=1}^{5} \sum_{j=1}^{3} X_{ij} \bullet Price_j \right)$$

$$- \left(\sum_{r=1}^{4} \sum_{p=1}^{5} \sum_{s=1}^{6} X_{rps} \bullet [\, (Slot\_start - Arrival\_time_p) \bullet Price\_penalty_p \,] \right)$$

3. ANALYSIS OF RESULTS

We will firstly describe solutions for each part. For this, we will use the same notation as explained in the first part of this report.
In the first part of the assignment, the solutions obtained for the tickets and planes:

LibreOffice solution:

x11	x12	x13	x21	x22	x23	x31	x32	x33	x41	x42	x44	x51	x52	x53
34	29	27	37	33	50	159	25	16	100	20	30	120	46	24

Mathprog solution:

x11	x12	x13	x21	x22	x23	x31	x32	x33	x41	x42	x44	x51	x52	x53
19	58	13	40	27	53	160	23	17	100	20	30	131	25	34

Even if we get slightly different values for the decision variables when using different tools, the value of the objective function remains the same, 26910. From these results, we can infer that the problem has infinite solutions.

We have verified the constraints as the number of tickets does not exceed the number of available seats, the maximum capacity of each airplane is not exceeded, in each airplane more than 20 leisure tickets are sold and at least 10 business plus. Finally, the standard air tickets are more than 60% of all sold tickets. All this can be checked in the output files or in the LibreOffice document attached.

For the first part the number of decision variables is 15 and the number of constraints we have written is 6. However, from plane_max_tickets,business_tickets, leisure_tickets and plane_capacity 5 extra constraints are derived from each one.

In the second part, the assignment of planes of planes obtained is:

$Runway\ (r)$ \ $Slot\ (s)$	9:00-9:15	9:15-9:30	9:30-9:45	9:45-10:00	10:00-10:15	10:15-10:30
R1					AV4	🟩

Carlos Gallego Jiménez 100405937
Andrés Langoyo Martín 100405869

R2			🟩	AV3		
R2	🟩	AV1	🟩			🟩
R3	AV2				🟩	AV5

The minimum value for the objective function is 4500.

Combining both problems, the values for the decision variables are the same and the objective function for the optimal solution is 21690.

Regarding the constraints, it can be checked that all the planes are on available slots, no two planes are assigned to contiguous slots, all the planes are only assigned to one slot and all the slots have only one plane. In addition, all planes respect their time restrictions and are delayed only five minutes.

Furthermore, the constraint that checks that one slot has been assigned only to one plane and the constraint that prevents two slots from being taken are redundant. The second one already includes the first one in the way we have implemented it.
The second constraint forces the addition of all the planes in two contiguous slots to be 1. However, the first one checks that all the planes in one slot add up to one. Therefore, if the second one is fulfilled, the first one is also fulfilled.

In addition, the same happens with constraints forbidden_slots and one_slot_per_plane . forbidden_slots already includes one_slot_per_plane as it is more restrictive.

In the second model, the number of decision variables are 120. This number is given by the multiplication of $r \cdot p \cdot s$. This second part adds the majority of constraints because all constraints but the first one depend on two different sets. The number of constraints depends on the dimension of those sets.

one_plane_per_slot: p (in this case 5)
one_slot_per_plane: $r \cdot s$ (in this case 20)
forbidden_slot: $r \cdot s$ (in this case 20)
after_starting_time: $p \cdot s$ (in this case 30)
before_max_time: $p \cdot s$ (in this case 30)
not_contiguous: $r \cdot g$ (in this case 20)

The total number of constraints that we obtain in the global model is 156 but we could reduce it more if we removed the redundant constraint regarding the slots.

The modelling of the problem has been done in a general way so, to scale the problem or modify it, we do not have to change the constraints, only the parameters. However, as more planes and/or runways are added the constraints will grow in a significant manner.

If we add a new runway we would have to modify the parameter slot_available.
However, if we add a new plane we would have to include new data in the following parameters: max_time, price_penalty, arrival_time, plane_capacity and plane_seats.
As seen, adding a new plane would result in a more complex model as this would affect both problems and the addition of several data in different parameters would lead to an increase in the number of constraints.

what if the AV1 is delayed 20 minutes then passengers will arrive late
(provide solution)
If we assume some of the planes have delays when it comes to arrival time this could be easily solved as adding the delay to the arrival_time parameter. As long as these delays do not exceed the max_time parameter or arrive after the last time slot, this problem is feasible.
In fact, we provide the solution to the proposed delay, in which AV1 is delayed one slot more:

Slot (s) \ *Runway* (r)	9:00-9:15	9:15-9:30	9:30-9:45	9:45-10:00	10:00-10:15	10:15-10:30
R1					AV4	🟩
R2			🟩	AV3		
R2	🟩	🟩	AV1			🟩
R3	AV2				🟩	AV5

The objective function in this case is 4000.
pros and cons of LibreOffice and GLPK
Finally, about the tools we have been working with we think that LibreOffice is more accessible and visual than GLPK when it comes to modelling a solution.

It provides an intuitive and easy to learn solver without the need of programming languages. In addition, we have found problems when trying to model the problems in GLPK as we would have liked to have found more examples in the Internet about the use of this program and how to use it at its full potential.

However, we also think that GLPK is better suited for larger and more scalable problems as it has powerful tools like sets and iterables.
A problem like the second model of this assignment would have required to write manually many constraints and variables in the sheet document, worsening its readability and increasing the probability of committing mistakes.

4. CONCLUSIONS

In this practice we have learnt how to use GLPK and model Linear Programming Tasks, we have realized that it requires a lot of reasoning to optimize a model and how this optimization can significantly reduce the number of constraints. In addition we have also learnt that modelling a problem in the most general way can help with the scalability and this is useful for the future changes that a system may experience.

We think that an improvement that could be made is the attachment of more advanced tutorials regarding GLPK as this would help in the understanding of this language. As there is not much information on the internet about this language, it would be really helpful. For instance, it took us several hours to discover how to define sets of sets to implement our last constraint.The examples in the tutorial given were not really complete and we couldn't find much information on the internet. In the end we made it by trial and error.

We think that the problem proposed is adequate and the difficulty of the optimization has encouraged us to try different ways of modelling the problems, this way, learning about how slight changes in constraints can reduce them significantly.