Heuristics and Optimization

Lab Report 01

Ignacio Aguado Sobradillo Pedro Navedo Martín

Group 121

Applied Mathematics and Computing

NIA: 100496813, 100495748

Contents

Introduction

This lab report will analyze the solutions of two linear programming problems aimed at optimizing different real-world scenarios using distinct tools.

The first part involves minimizing the total distance that the buses should travel to get to their respective workshop. For this task we will use LibreOffice Calc´s Solver.

The second part is divided in two tasks. In the first one we will minimize the total cost of breakdowns, deciding if it is less costly to go directly to the workshop or to finish the service. In the second task we just want to maximize the customer´s satisfaction. Here the challenge will be to really make this problem linear. For solving both this problems we will use MathProg and GLPK (GNU Linear Programming Kit).

1 Part 1

In the first part of the project, we address an assigment problem between five different buses and five available workshops. Each workshop is located at a certain distance from each bus and the goal is to assign buses to workshops while minimizing the total distance traveled. We will formulate the model as a binary integer linear programming problem, where the decision variables indicate if a given bus is assigned to a given workshop. The constraints will ensure that each bus is assigned to exactly one workshop, that no workshop is assigned to more than one bus and that the decision variables are binary. Here is a better representation of the whole problem:

Sets:

- $A = \{1, \ldots, m\}$: set of buses
- $T = \{1, \ldots, n\}$: set of workshops

Parameters:

- c_{ij} : distance from bus $i \in A$ to workshop $j \in T$

Decision Variables:

$$x_{ij} = \begin{cases} 1 & \text{if bus } i \text{ is assigned to workshop } j \\ 0 & \text{otherwise} \end{cases}$$

Objective Function:

$$\min Z = \sum_{i \in A} \sum_{j \in T} c_{ij}\, x_{ij} \tag{1}$$

Constraint	Expression
Each bus assigned once	$\sum_{j \in T} x_{ij} = 1 \quad \forall i \in A$
At most one bus per workshop	$\sum_{i \in A} x_{ij} \leq 1 \quad \forall j \in T$
Binary constraint	$x_{ij} \in \{0, 1\} \quad \forall i \in A,\ j \in T$

Table 1: Constraints of the assignment problem

The model was implemented in LibreOffice Calc by constructing the assignment matrix and defining the objective function as a weighted sum of distances using the SUMPRODUCT function. The constraints were incorporated into the Solver tool, enforcing binary decision variables. The Solver was then used to obtain the optimal assignment and the minimized total distance, confirming that all model conditions were satisfied.

2 Part 2

2.1 Task 1

In task 1 of part 2, we want to minimize the impact of unexpected issues at the beginning of service. Here, we only consider one workshop t, but this workshop offers n different slots and there are m different buses available to participate. We will formulate the model as a mixed integer linear programming task. Here we have a detailed representation of the whole problem:

Sets:

- $A = \{1, \ldots, m\}$: set of buses.
- $S = \{1, \ldots, n\}$: set of timeslots.

Parameters:

- d_i : distance from bus $i \in A$ to workshop.
- p_i : number of passengers that have booked service on bus $i \in A$.
- k_d : constant representing the amount in euros per kilometer that must be assumed if a bus i is assigned to some slot in the workshop.
- k_p : constant representing the amount in euros per passenger that must be assumed if a bus i is not assigned to any slot.

Decision Variables:

$$x_{i,s} = \begin{cases} 1 & \text{if bus } i \text{ is assigned to timeslot } s \\ 0 & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if bus } i \text{ is left unassigned} \\ 0 & \text{otherwise} \end{cases}$$

Objective Function:

$$\min Z = \sum_{i \in A} \sum_{s \in S} k_d \, d_i \, x_{is} \; + \; \sum_{i \in A} k_p \, p_i \, y_i \tag{2}$$

We want to present some clarifications on the objective function and the decision to have two different variables, one (x_{is}) that takes the value 1 if the bus i is assigned to the time slot s and y_i that takes the value 1 if bus i is left unassigned. The binary variable y_j is introduced to explicitly represent the decision to not assign bus i, which simplifies the formulation of the penalty cost in the objective function.

Constraint	Expression
Each bus assigned once or is left unassigned	$\sum_{s \in S} x_{is} \; + \; y_i = 1 \quad \forall i \in A$
At most one bus per timeslot	$\sum_{i \in A} x_{is} \leq 1 \quad \forall s \in S$
Binary constraint	$x_{is} \, , \; y_i \in \{0, 1\} \quad \forall i \in A, \; s \in S$

Table 2: Constraints of the assignment problem 2

With this implementation, if n < m some buses will be left unassigned and we will have a penalization $k_p p_i$ as requested.

2.2 Task 2

In this part, the goal is to optimize the allocation of buses to available workshop slots in order to *maximize passenger satisfaction*, or equivalently, to *minimize the total dissatisfaction* produced when buses that share many common passengers are sent to the same time slot but in different workshops.

When we first approached the problem, our intention was to directly penalize pairs of buses that were assigned to the same slot in different workshops. So, the most intuitive formulation involved multiplying binary assignment variables. Something like:

$$x_{isw} \cdot x_{jsw'}$$

However, this introduces nonlinear terms, which are not allowed for this project, and when we tried to run GLPK with this approach, an error was raised. Therefore, we needed to reformulate the model in a linear way. The final linear version we implemented is the following:

Sets

- $B = \{1, \ldots, m\}$: set of buses
- $S = \{1, \ldots, n\}$: set of time slots
- $W = \{1, \ldots, u\}$: set of workshops

Parameters

- c_{ij} : number of passengers shared between bus i and bus j
- o_{ws} : binary parameter, 1 if slot s is available in workshop w, 0 otherwise

Decision Variables

$$x_{isw} = \begin{cases} 1 & \text{if bus } i \text{ is assigned to slot } s \text{ of workshop } w, \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ijs} = \begin{cases} 1 & \text{if buses } i \text{ and } j \text{ are both assigned to the same slot } s \\ 0 & \text{otherwise} \end{cases}$$

Objective Function

$$\min \text{TotalDissatisfaction} = \sum_{i<j} \sum_{s \in S} c_{ij}\, y_{ijs}$$

We want to maximize passenger satisfaction, which is equivalent to minimizing the dissatisfaction of passengers. Dissatisfaction occurs only when two buses that share passengers are repaired during the same time slot but not in the same workshop. The parameter c_{ij} represents how many passengers buses i and j have in common. If buses are assigned to different slots or if they end up in the same workshop, no dissatisfaction is generated. To linearize the condition "buses share slot s but are in different workshops," we use the auxiliary variable y_{ijs}, which is linked to the assignment variables x_{isw}. So dissatisfaction is only generated when y_{ijs} is 1 and we multiply it by the number of passengers affected.

Constraint	Expression
Each bus assigned once	$\sum_{s \in S} \sum_{w \in W} x_{isw} = 1 \quad \forall i \in B$
At most one bus per slot per workshop	$\sum_{i \in B} x_{isw} \leq o_{ws} \quad \forall s \in S, w \in W$
Linking y to shared slots	$\begin{cases} y_{ijs} \leq \sum_{w \in W} x_{isw} \\ y_{ijs} \leq \sum_{w \in W} x_{jsw} \\ y_{ijs} \geq \sum_{w \in W} x_{isw} + \sum_{w \in W} x_{jsw} - 1 \end{cases} \quad \forall i < j, s \in S$
Binary constraint	$x_{isw}, y_{ijs} \in \{0, 1\}$

Table 3: Constraints for minimizing dissatisfaction due to overlapping buses in the same slot

The first constraint ensures that every bus is assigned to exactly one slot in one workshop. The second constraint enforces the workshop capacity, preventing more than one bus from occupying the same slot within the same workshop. The linking constraints ensure that y_{ijs} takes the value 1 if and only if both buses i and j are assigned to slot s. Because the model already forbids two buses from sharing the same (s, w) pair, assigning them to the same slot automatically implies they are in different workshops, which is precisely the condition that generates dissatisfaction. Finally, the binary constraints ensure all decisions are strictly yes/no, with no fractional assignments.

3 Analysis

3.1 Part 1

To obtain a solution, we entered the provided distance data and defined a table of binary decision variables in **LibreOffice Calc**. The complete distance matrix is shown in Table 4.

	a1	a2	a3	a4	a5
t1	206	66	263	104	154
t2	285	86	111	300	110
t3	164	251	109	220	177
t4	264	175	217	147	149
t5	141	75	218	87	228

Table 4: Distance matrix (in km) from each workshop t_i to each bus line a_j.

All constraints and the objective function were added to the Solver tool, indicating where the decision variables were placed. The Solver returned an optimal value for the objective function of **573 km**. This means that the total distance traveled by all buses is minimized to 573 km. The corresponding optimal assignment of workshops to buses is shown in Table 5.

	x_1	x_2	x_3	x_4	x_5
Workshop 1	0	1	0	0	0
Workshop 2	0	0	0	0	1
Workshop 3	0	0	1	0	0
Workshop 4	0	0	0	1	0
Workshop 5	1	0	0	0	0

Table 5: Optimal assignment of workshops to buses (Base Case).

In this base case, bus x_1 is assigned to workshop 5, bus x_2 to 1, bus x_3 to 3, bus x_4 to 4, and bus x_5 to 2. All constraints are satisfied: each bus is assigned to exactly one workshop, each workshop serves at most one bus, and all variables are binary. Because the number of workshops equals the number of buses, these conditions are easily satisfied.

Case 1: Fewer Buses than Workshops

We next explored how the solution changes when the number of buses decreases. Eliminating bus number 5, while keeping the same number of workshops, leads to the assignment shown in Table 6. The total distance decreases to **463 km**.

	x_1	x_2	x_3	x_4
Workshop 1	0	1	0	0
Workshop 2	0	0	0	0
Workshop 3	0	0	1	0
Workshop 4	0	0	0	1
Workshop 5	1	0	0	0

Table 6: Binary decision variable values for the optimal workshop–bus assignment (4 buses, 5 workshops).

As expected, all constraints remain satisfied. Each bus is assigned to one workshop, while workshop 2 remains unused. The smaller total distance (463 km) is reasonable, since with fewer buses, the total distance traveled is less.

Case 2: Fewer Workshops than Buses

Finally, we considered the opposite case, with fewer workshops than buses. In this situation, the constraint requiring each bus to be assigned to exactly one workshop makes the problem infeasible because there are not enough workshops to host all buses. This demonstrates that the requirement for every bus to be assigned is a critical limiting constraint that makes the problem unsolvable when workshop capacity is insufficient.

To further explore the model's behavior, we relaxed the constraint that each workshop can serve at most one bus and allowed each workshop to accommodate up to two buses.This makes the problem feasible again. The new results are presented in Table 11.

	x_1	x_2	x_3	x_4	x_5
Workshop 1	0	1	0	1	0
Workshop 2	0	0	0	0	1
Workshop 3	1	0	1	0	0
Workshop 4	0	0	0	0	0

Table 7: Binary decision variable values for the optimal workshop–bus assignment after relaxing the capacity constraint (allowing up to two buses per workshop).

In this case, no buses are assigned to workshop 4, while workshops 1 and 3 each serve two buses. The total distance is **553 km**, slightly less than the base case. This demonstrates that relaxing the capacity constraint allows a more flexible but less realistic allocation, as some workshops handle multiple buses.

This examples show how constraint changes affect both feasibility and total distance. When fewer buses are available, the objective value decreases naturally. When there are fewer workshops, the model becomes infeasible unless we relax capacity restrictions. Additionally, if two distances were equal, multiple optimal solutions could arise, yielding the same minimum total distance but different individual assignments. This illustrates the sensitivity of assignment models to both data and constraint structure.

From the previous results we can conclude that the most restrictive constraint is the one ensuring that *each bus is assigned to exactly one workshop*

This condition enforces full coverage of all buses and directly causes infeasibility when the number of workshops is smaller than the number of buses. In contrast, the constraint limiting each workshop to at most one bus can be relaxed to restore feasibility, as demonstrated in Case 2. Therefore, the requirement that every bus must be assigned is the binding and most restrictive condition in this problem.

3.2 Part 2 - Task 1

To perform the analysis of Task 1 and put our system under pressure, we have considered different cases and studied the results obtained when we executed the code that we have developed. Here we emphasize in each of them:

3.2.1 Example 1: Basic Case

For the first example, we try our system with standard data. We set the same number of buses and time slots. The cost per kilometer k_d is set at 5.0 and the cost per passenger left without a service k_p is set at 4.0. The values of distances from each bus to the workshop is $[10, 25, 5, 10, 20]$ and the number of passengers of each bus is $[20, 40, 15, 10, 50]$. With these values as input, we obtain 30 decision variables and 10 constraints, and the value of the objective function, that is, the minimum cost that we could obtain with this input, is **340**. The buses are arranged as stated in Table 8.

	a_1	a_2	a_3	a_4	a_5
Slot 1	0	0	0	0	1
Slot 2	0	1	0	0	0
Slot 3	0	0	1	0	0
Slot 4	1	0	0	0	0
Slot 5	0	0	0	0	0

Table 8: Distribution of buses in the different time slots

3.2.2 Example 2: Less time slots than buses and higher values

In this case, we tried choosing values for n and m such that $n < m$, that is, there are more buses than time slots. The system remains consistent and it does not raise any problem. It chooses the buses that minimize the cost per kilometer and leaves unassigned the ones that minimize the cost per passenger without service. Also we wanted to try the time that would take for the system to solve an input of higher values: we tried with 17 time slots and 19 buses. We set k_d at 4.0 and k_p at 3.0. In general lines, we obtained **342** decision variables and **36** constraints, and the value of the objective function was **1568**.

3.2.3 Example 3: Extreme case

Here we wanted to consider one of the extreme cases that we could find. In this case, the penalization per kilometer k_d is really high, 150.0, and the cost per passenger without a service k_p is set to a normal value. Considering the same data for distances and number of passengers as in Example 1, we can see that, since the objective function is in form

of minimization, it prefers to leave all the passengers without a service than give them a seat in a bus and pay the cost per kilometer for that bus. Therefore, all buses are left unassigned.
We could consider other extreme cases, like having a cost per passenger k_p very high, or distances between where are located the buses and the workshop also very high, but we thought that this one was the most interesting.

3.2.4 Example 4: Relaxation and Hardening of constraints

To finalize our analysis of part 2, we are going to relax and harden our constraints. For all the examples presented here, we consider these values:

1. Number of slots n: 4
2. Number of buses m: 5
3. Set of distances for buses: $[10, 25, 5, 10, 20]$
4. Set of passengers per bus: $[20, 40, 15, 10, 50]$

Now, firstly, we relax the second constraint: "At most one bus per slot". We set that each slot can take at most 2 buses instead of one. $\sum_{s \in S} x_{is} \leq 2 \, \forall i$ This has no physical meaning, but it might help our system minimize the cost. We obtain **340** from the objective function, 30 decision variables and 9 constraints. These are the same values that we obtain if we run the system with the same data with the original constraint. We see the difference between the original system and the one with the relaxed constraint in the distribution of buses over the time slots.

	a_1	a_2	a_3	a_4	a_5
Slot 1	0	0	0	0	1
Slot 2	0	1	0	0	0
Slot 3	0	0	1	0	0
Slot 4	1	0	0	0	0

Table 9: Distribution of buses in the different time slots in the original system

	a_1	a_2	a_3	a_4	a_5
Slot 1	0	1	0	0	1
Slot 2	1	0	1	0	0
Slot 3	0	0	0	0	0
Slot 4	0	0	0	0	0

Table 10: Distribution of buses in the different time slots in the modified system

We can see that the buses are assigned to time slots 1 and 2, and the other time slots remain empty.
On the other hand, in order to harden the constraints we can take two different paths:

1. In the first, we require that all buses must be assigned. That is, we change the second constraint to $\sum_{i \in A} x_{i,s} = 1 \forall s \in S$ and set all y_i to 0, that is, $\sum_{i \in A} y_i = 0$. In this way, the problem will be unfeasible whenever $n < m$, that is, when we have more buses than time slots.

2. In the second, we add another constraint: a maximum distance D_{max}. Buses that are at a distance greater than D_{max} are now not eligible, so this reduces the amount of assigned buses and increases the cost per passenger left without service if k_p is high enough. Let´s see an example. Considering the data in the first example, but adding the constraint $d_i x_{is} \leq 15$, where $D_{max} = 15$, we obtain 25 decision variables and 9 restrictions, but the value of the objective function rises to **475**. Since buses 2 and 5 now can not be assigned to a time slot because the distances between them and the workshop are higher than D_{max}, the distribution of the buses is now the one represented in the following table.

	a_1	a_2	a_3	a_4	a_5
Slot 1	0	0	1	0	0
Slot 2	1	0	0	0	0
Slot 3	0	0	0	0	0
Slot 4	0	0	0	0	0

Table 11: Distribution of buses in the different time slots in the modified system

If we study which of these two cases has the most restrictive constraints, we can see it is the first one, where every bus must be assigned to one time slot. Why? Because this constraint makes the problem unfeasible whenever $n < m$, and forces a perfect assignation when $n \geq m$ which reduces the feasible region. In the second case, the maximum distance constraint only reduces the number of buses that can be assigned (the ones satisfying $d_i \leq D_{max}$) but it doesn´t impede $y_i = 1$. Normally feasibility is preserved and we "only" have to increment the cost.

3.3 Part 2 - Task 2

For the analysis of the second task, we will consider different cases to ensure the correct application of the constraints, to measure the run time in different cases and to try the system under edge cases.

For the first three examples, we will try increasingly complex examples to see the change on the number of variables and the number of constraints used. Also, because we are only focused on the increase of complexity, we will have all the slots available for the first three cases. Now lets see these cases:

3.3.1 Simple case

For the simpler case we used just 3 buses and two workshops with two slots each. For these case and all the rest we decided to create examples with at most 20 coinciding

passengers between buses to have a more normal value for the objective function. For the previous data, we obtained a value for the objective that in this case is irrelevant, a total of 18 decision variables created and a total of 25 constraints created.

3.3.2 Medium case

For this case we augmented the complexity of the input data. Now we have 5 buses and 3 workshops with 3 slots each. Now we obtain a given objective function, we have a total of 75 decision variables and 104 constraints. We can see that, as expected, the number of decision variables and of constraints increase as it increases the complexity of the problem.

3.3.3 Complex case

For the more complex case we will use 8 buses and 4 workshops with 6 slots each. As we have increased a lot the complexity we notices that the runtime also increased a lot. It was taking several minutes so we decided to stop and try with a smaller example. For this one we used 6 buses and 3 workshops with 4 slots each. We obtained a given objective function and a total of 132 decision variables and 198 constraints created. We had the following results:

Slot \Workshop	1	2	3
1	1	-	-
2	-	2	5
3	3	-	-
4	4	6	-

Table 12: Bus assignment grid: rows = slots, columns = workshops, numbers = bus IDs. A dash (-) indicates no bus assigned.

In this case, as in the other cases, we can clearly see that all the constraints are satisfied, each bus is assigned exactly once and there is at most one bus per slot per workshop. In this table we can not see the other constraints but they are also satisfied, getting the minimum possible value for the objective function.

Also, we wanted to see if making some of the slots unavailable would make any difference in the number of created constraints. But we modified the availability of some of those slots and the number of decision variables and of created constraints stayed the same, as the structure of the model is independent of the specific occupancy pattern.

Our initial analysis of the GLPK output across all cases (Simple, Medium, and Complex) showed a variable count that matched the product of buses, slots, and workshops ($m \times n \times u$). This number, however, represents the presolved problem, which the solver internally simplifies.

The model we actually formulated is significantly larger and defines the true complexity. We defined not only the $O(mnu)$ assignment variables (x_{isw}) but also the $O(m^2n)$ auxiliary variables (y_{ijs}) required for linearization.

Thus, our actual variable count is:

$$\text{Total Variables} = (m \times n \times u) + \left(\binom{m}{2} \times n \right)$$

This analysis reveals the true source of the problem's difficulty. The complexity is dominated by the $O(m^2 n)$ terms from the linearization. This $O(m^2 n)$ component, not the $O(mnu)$ term reported by the presolver, is the limiting factor for the model's scalability and the most complex part of our formulation.

Now lets explore other cases:

3.3.4 Infeasible case

For this case, we just wanted to see if the tool will detect correctly an infeasible case. So we picked 4 buses and 2 workshops with 2 slots each. However some of this slots were unavailable. As expected, we do not get any solution and, if we open the *glpsol.out* file, at the end it is stated that the solution is infeasible.

3.3.5 Restrictive Slot Availability case

For this case, we wanted to see the behavior of the system even when some buses can not be assigned to which could be their most favorable slot. So, we used 3 buses and 2 workshops with 3 slots each. However, only a total of 3 slots were available. We prepared two examples, on where some of this slots would coincide, and one where each available slot was a different one. This is the table for the first case:

Passenger Shared Matrix c_{ij}			
	Bus 1	Bus 2	Bus 3
Bus 1	0	10	5
Bus 2	10	0	8
Bus 3	5	8	0

Workshop Slot Occupancy o_{ij}			
	Slot 1	Slot 2	Slot 3
Workshop 1	1	0	1
Workshop 2	0	1	0

Table 13: Data for 3 slots, 3 buses, 2 workshops

And this is the data for the second case:

Passenger Shared Matrix c_{ij}			
	Bus 1	Bus 2	Bus 3
Bus 1	0	10	5
Bus 2	10	0	8
Bus 3	5	8	0

Workshop Slot Occupancy o_{ij}			
	Slot 1	Slot 2	Slot 3
Workshop 1	1	0	1
Workshop 2	0	0	1

Table 14: Data for 3 slots, 3 buses, 2 workshops

For the first case, we obtained an objective function of 0 (as expected because no two buses coincide), 27 decision variables and 36 constraints created. For the second case, we obtained a value of the objective function of 5 (as in the solution bus 1 and bus 3 coincide) and we obtained the same number of decision variables and of created constraints as the previous case. These examples illustrate the correct behavior of the system by carefully managing slot assignments to minimize conflicts and maximize passenger satisfaction.

4 Conclusions

This project successfully applied Mixed Integer Linear Programming to solve a series of practical optimization tasks, demonstrating proficiency in two methods: the Solver tool on LibreCalc Office and the GLPK modeling language.

We developed models of increasing complexity, starting from a basic assignment and culminating in a more advanced model. The primary modeling challenge was encountered in the final task, where we successfully linearized an inherently non-linear objective related to passenger satisfaction, a critical step to ensure a solvable MILP.

Our analysis validated that all models produced correct and optimal solutions under various test conditions, including infeasible and edge-case scenarios. This project provided a comprehensive experience in MILP: from formal problem definition and modeling to implementation, solving, and the analysis of the results.