ISSN-e: 1856-9811

TABU SEARCH ALGORITHM: THE METROPOLITAN UNIVERSITY CLASSROOM ASSIGNMENT CASE

TABU SEARCH ALGORITHM: CASE ASSIGNMENT OF CLASSROOMS OF THE METROPOLITAN UNIVERSITY

SIRO TAGLIAFERRO
stagliaferro@unimet.edu.ve
Universidad Metropolitana de Caracas. (Venezuela)

MANUEL MARTÍNEZ
manuel.martinez@uss.cl
Universidad de San Sebastián. (Chile)

JOSÉ ALGUÍNDIGUE RUIZ
jose.alguindigue@correo.unimet.edu.ve
Universidad Metropolitana de Caracas. (Venezuela)

ALBERTO PEREIRA
alberto.pereira@correo.unimet.edu.ve
Universidad Metropolitana de Caracas. (Venezuela)

Summary

The Tabu Search algorithm allows to perform an optimization methodology whose differential characteristic is the use of adaptive memory and special problem solving strategies, especially in the distribution of resources. Within the Metropolitan University there have been several problems with the allocation of classrooms, due to the student demand, which has had an erratic and difficult to predict behavior, which required the use of optimization tools or programs. For this purpose, a mathematical model was developed, where the strong restrictions were proposed based on the characteristics of the Metropolitan University and the objective function was determined, to later use it in the tabu search algorithm to evaluate the solutions generated. After developing the model, we proceeded to develop the tabu search algorithm, first processing the data on the academic offerings and classrooms provided by the university and developing the initial solution to the problem, which is used as a starting point for the search. Subsequently, the model was implemented in the algorithm, so that the tabu search seeks to reduce the number of sections where the assigned room does not have the required resources and optimizes the assigned spaces based on the number of students enrolled per section. Subsequently, a system was developed to visualize the obtained solution more easily

and, finally, it was validated that the obtained solution complies with the characteristics of the allocation of classrooms of the Metropolitan University in the academic period 2022.

Summary

The Tabu Search algorithm makes it possible to carry out an optimization methodology whose differential characteristic is the use of adaptive memory and special problem-solving strategies, especially in the distribution of resources. In the Metropolitan University there have been various problems with the classroom assignment, due to student demand, which has been an erratic and difficult for predict behavior, which required the use of optimization tools or programs. For this, a mathematical model was made, where the strong constraints were raised based on the characteristics of the Metropolitan University and the objective function was set up, to later use it in the tabu search algorithm to evaluate the solutions. After developing the model, the tabu search algorithm was developed, first the data of the academic offer and that of the classrooms provided by the university were processed and the initial solution of the problem was developed, which is used as a starting point. search game. Subsequently, the model was implemented in the algorithm, so that the tabu search seeks to reduce the number of sections where the assigned room does not have the required resources and optimizes the assigned spaces based on the number of students enrolled per section. Subsequently, a system was implemented to be able to visualize the obtained solution more easily and, finally, it was validated that the obtained solution complied with the characteristics of the classroom reform of the Metropolitan University in the 2022 academic period.

RECEIVED: 09-03-2024 ACCEPTED: 11-05-2024 PUBLISHED: 15-12-2024

How to quote: Tagliaferro et al. (2024). Tabu Search Algorithm: The Metropolitan University Classroom Assignment Case. *Anales*, 40, 1 - 34. https://doi.org/10.58479/acbfn.2024.68

ISSN-e: 2244-8276

INDEX

Introduction

Space allocation has been a problem present in all educational institutions around the world. This is caused by the fact that each educational unit has its own characteristics and different spaces, causing that each solution generated must take into consideration different variables and restrictions. Currently, the assignment of classrooms at the Metropolitan University is done manually, which means that the process takes several days. Previously, the assignment was done automatically, but it did not take into consideration the different characteristics of the university, which generated a series of complaints that made the process extremely long.

The general objective of this project is to develop a system that uses the Tabu Search algorithm to assign classrooms at the Metropolitan University.

The tabu search algorithm, starting from an initial solution and through a series of strategies, such as the tabu list, allows to perform an efficient search in the solution space, with the objective of finding a solution close to the optimum. Tabu Search allows for bad choices to be made during exploration, as it considers that one can learn by making a bad choice and thus target more promising areas.

Mathematical Model Design

Based on the characteristics of the Metropolitan University, a series of parameters were proposed to mathematically translate the hard constraints and establish the objective function.

Parameters

- S: set of all sections (index i).
- $SINS_i$number of students enrolled in section i.
- $SHOR_i$time blocks required by section i.
- SVVirtual section: subset of all the virtual sections.
- SP: subset of all face-to-face sessions.
- SSP: subset of all semi-face-to-face sections.

- $SDOB$: subset of the sections that have a double time block in a row.
- $STRI$: subset of the sections that have three time blocks in a row.
- $SLAB$: subset of the sections that require a laboratory.
- $SESP$: subset of the sections that require a specific classroom (music room, thespis and judgment room).
- $SLABS_i$: laboratory requested by section i.
- $SESPS_i$specific classroom requested by section i.
- SD_idays required by section i.
- A: set of all classrooms (index j).
- $ACAP_j$: capacity of the classroom j.
- AD_{iw}: classroom assigned to section i on day w.
- AS_{ik}: classroom assigned to section i in time block k.
- $ALAB$: subset of all laboratories.
- $AESP$: subset of all specific classrooms.
- $ALABS_i$: laboratory assigned to section i.
- $AESPS_i$:specific room assigned to section i.
- B: set of all available time blocks in a day (index k).
- BSi time block assigned to section i.
- D: set of all the days of the week on which classes are held (index w).
- DS_idays assigned to section i.
- C_1cost associated with the difference between the number of people enrolled in the section and the capacity of the classroom assigned to that section.
- C_2Cost associated with the type of classroom requested by the section does not correspond to the type of classroom in the room to which it was assigned.
- NSAnumber of sections where the type of classroom required by the section does not match the type of classroom in the room to which it was assigned, in the case that the type of classroom is T (theoretical), M (multimedia) and B (systems laboratory).

Variable

A single binary variable was proposed to control the assignment of classrooms; the variable takes the value of "1" if the section has a classroom assigned in the requested time block and day. However, it takes the value "0" in the opposite case, i.e. the section has not been assigned a classroom, time block or day in the timetable. The variable is described below:

X_{ijkw} = 1 if section i is assigned to room j in time block k and on day w; 0 if not.

Restrictions

Thanks to the collaboration of the personnel of the Study Control area, the necessary restrictions were obtained. Subsequently, they were put into mathematical form using the variable and parameters previously determined. The purpose of these hard constraints is to prevent the solutions obtained by the tabu search from containing assignments that do not meet the characteristics of the Metropolitan University.

The hard restrictions determined were as follows:

1. The assigned classroom capacity cannot be less than the number of students enrolled in the section (R1).
2. A classroom can only have one section assigned to it in the same time block and on the same day (R2).
3. The time block required by the section must be the same as the one assigned to it in the classroom (R3).
4. Virtual type sections should not be assigned any room in any time block on any day (R4).
5. In face-to-face sections, the assigned room must be the same on both days (R5).
6. All face-to-face sections must be assigned to a classroom, a time block and two days of the week (R6).
7. All semi-face-to-face sections must be assigned to a classroom, a time block and a day (R7).
8. The days required by the section must be the same days to which it is assigned (R8).
9. Sections that require two time blocks on the same day in a row must be assigned the same room (R9).

10. Sections that require three time blocks on the same day in a row must be assigned the same room (R10).
11. Sections requiring a laboratory must have the requested laboratory assigned (R11).
12. Sections requiring a specific classroom must be assigned the specific classroom requested (R12).

The mathematical form of each of the hard constraints can be seen in Figure 1.

$$\textbf{R1: } X_{ijkw}\, SINS_i \leq ACAP_j \quad ; \quad \forall i \in S,\ \forall j \in A,\ \forall k \in B,\ \forall w \in D$$

$$\textbf{R2: } \sum_{i \in S} X_{ijkw} \leq 1 \quad ; \quad \forall j \in A, \forall k \in B,\ \forall w \in D$$

$$\textbf{R3: } X_{ijkw}\, SHOR_i = BS_i \quad ; \quad \forall i \in S,\ \forall j \in A,\ \forall k \in B,\ \forall w \in D$$

$$\textbf{R4: } \sum_{j \in A} \sum_{k \in B} \sum_{w \in D} X_{ijkw} = 0 \quad ; \quad \forall i \in SV$$

$$\textbf{R5: } X_{ijkw}\, AD_{iw} = X_{ijk(w+2)}\, AD_{i(w+2)} \quad ; \quad \forall i \in SP,\ \forall j \in A,\ \forall k \in B\ ,\forall w \in D$$

$$\textbf{R6: } \sum_{j \in A} \sum_{k \in B} \sum_{w \in D} X_{ijkw} = 2 \quad ; \quad \forall i \in SP$$

$$\textbf{R7: } \sum_{j \in A} \sum_{k \in B} \sum_{w \in D} X_{ijkw} = 1 \quad ; \quad \forall i \in SSP$$

$$\textbf{R8: } X_{ijkw}\, DS_i = SD_i \quad ; \quad \forall i \in S,\ \forall j \in A,\ \forall k \in B,\ \forall w \in D$$

$$\textbf{R9: } X_{ijkw}\, AS_{ik} = X_{ij(k+1)w}\, AS_{i(k+1)} \quad ; \quad \forall i \in SDOB,\ \forall j \in A,\ \forall k \in B\ ,\ \forall w \in D$$

$$\textbf{R10: } X_{ijkw}\, AS_{ik} = X_{ij(k+2)w}\, AS_{i(k+2)} \quad ; \quad \forall i \in STRI,\ \forall j \in A,\ \forall k \in B\ ,\ \forall w \in D$$

$$\textbf{R11: } X_{ijkw}\, ALABS_i = SLABS_i ; \quad \forall i \in SLAB,\ \forall j \in ALAB,\ \forall k \in B\ ,\ \forall w \in D$$

$$\textbf{R12: } X_{ijkw}\, AESPS_i = SESPS_i ; \quad \forall i \in SESP,\ \forall j \in AESP,\ \forall k \in B\ ,\ \forall w \in D$$

Figure 1. Hard constraints in mathematical form.
Source: Own elaboration.

Target function

The weak restrictions determined by the study were as follows:

1. Difference between the number of people enrolled in the section and the assigned classroom capacity.

2. Classroom type assigned to the section does not match the requested classroom type. Only applies to T (theoretical), M (multimedia) and B (systems laboratory) classrooms.

These two weak constraints were selected because the personnel in charge of classroom allocation at the university indicated that they should try to optimize the space used and that there are not enough multimedia rooms to meet the current demand. These constraints were set out in the objective function to be minimized and a cost or penalty was assigned to each constraint, the value of the penalty being obtained through experimentation.

It is important to note that the objective function is used in the tabu search algorithm to evaluate each solution, determine if one solution is better than the other and thus move through the search space. Figure 2 shows the objective function.

$$\textbf{Min} \left(\sum_{i \in SP} \sum_{j \in A} \sum_{k \in B} \sum_{w \in D} X_{ijkw} \, X_{ijk(w+2)} \, C_1 \left(ACAP_j - SINS_i \right) \right) +$$

$$\left(\sum_{i \in SSP} \sum_{j \in A} \sum_{k \in B} \sum_{w \in D} X_{ijkw} \, C_1 \left(ACAP_j - SINS_i \right) \right) +$$

$$C_2 \, NSA$$

Figure 2. Objective function.
Source: Own elaboration.

Data processing

It should be noted that the data provided pertains to a quarter of 2017. The data contains the academic offer of that quarter and the allocation of classrooms that was made.

The following is a detailed explanation of the processing applied to the data provided by the Metropolitan University.

1. The data obtained was separated into two Excel© files, one containing information on the sections and the other on the classrooms, each with its corresponding information.

2. The variable classroom/type was separated into two (classroom and type of classroom), in order to easily recognize the type of classroom corresponding to each of them. In addition, classrooms with the name N/A were eliminated.

3. Regarding section data, sections that had no students enrolled were eliminated, as well as those that closed.

4. Both Excel files are saved as *Comma Separated Value* (.csv) files.

5. Within Python the team created two *dataframes* corresponding to each of the files.

6. After the creation of the sections *dataframe*, it underwent the following changes:

    a. Division of the type of classroom required (sections requiring a type of classroom M, T and B were separated over sections requiring a specific laboratory).

    b. Division of the *dataframe* for classrooms which will be considered in the tabu algorithm versus classrooms which will not be considered (note that specific classrooms such as laboratories need a specific section).

    c. Division depending on the pair of days on which the sections require classes to be taught (*dataframes* were divided into LM = Monday and Wednesday, MJ = Tuesday and Thursday, V = Friday, S= Saturday and D = Sunday.

    d. Subsequently, after the initial assignment of the classrooms, each group of sections was divided into smaller *dataframes* based on the time block in which they started and ended.

Tabu Search Algorithm Design

The design of the tabu algorithm was developed in a modular way. According to William (2012) programming in a modular way grants different benefits, about which we can mention:

- Simplify the design.
- Reduce the complexity of the algorithms.
- Decrease the overall size of the program. Since it promotes code reusability.
- Facilitates debugging and testing.
- Facilitates maintenance.

In addition, the beginning of a solution to then start with the search for the optimal combination.

Initial Solution Design

The initial is a must be simple, as it is not required to be complex to start the tabu search. In this solution, the assignment of classrooms to the sections was done trying to comply with the constraints. Mainly it was assigned to take into account constraint 1 (the capacity of the assigned classroom cannot be less than the number of students enrolled in the section) and constraint 2 (a classroom can only have one section assigned in the same time block and on the same day).

To carry out the generation of the initial solution, the team performed the following steps.

1. Sort the section *dataframe* (already divided into pairs of days) by number of students enrolled from oldest to youngest and by the time it started from earliest to latest.

2. Sort the *dataframe* of rooms from highest to lowest capacity.

3. Execution of the initial algorithm for room assignment. Based on the logic shown in Figure 1.

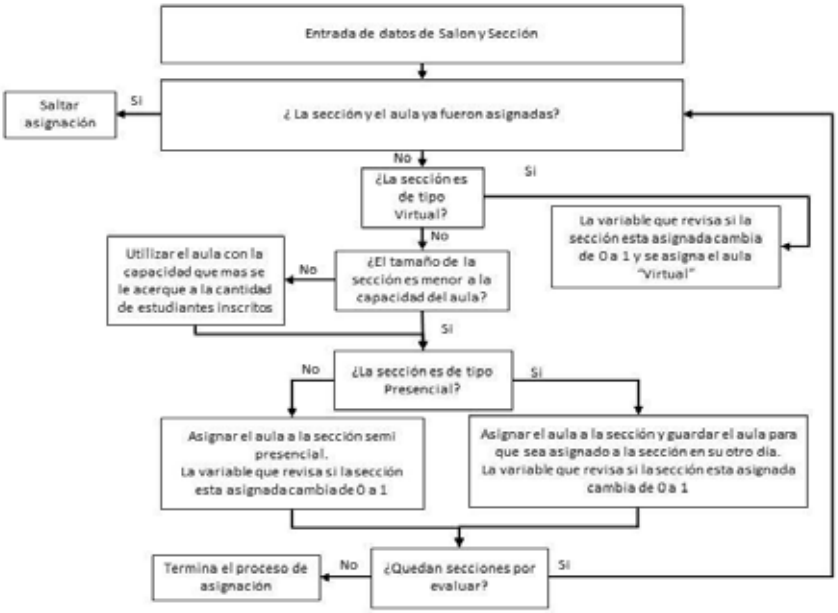


Figure 3. Initial algorithm for assigning classrooms

Source: Own elaboration

Assignment of non-Tabu sections

Due to the fact that at the Metropolitan University there are specific classrooms for certain subjects, it was observed the need to assign certain sections to the classrooms necessary to carry out their activities. Because these sections require specific materials that are only available in certain classrooms.

The reason for assigning these classrooms before using the Tabu algorithm is that since they are classrooms that can only be assigned once per time block and to certain subjects, they will not vary at any time during the tabu search, which would only increase the search time.

In conclusion, the sections that require the use of a laboratory, Thepsis classroom, music room and trial room will be automatically assigned by the system to the required classrooms since it is not possible for those sections to receive classes in another classroom.

Identification of sections with more than 1 time block

Given that at the Metropolitan University there are sections with more than one time block, a validation was carried out to obtain a list of the sections with this quality and to take them into account.

Then, to check if this quality was present in each of the sections, we proceeded to validate the strong constraint validation module in which if the tabu algorithm assigns a classroom to a section and that classroom is occupied by a section with more than one time block, that possible solution is discarded.

Neighborhood generation

This function is in charge of generating all possible movements to move through the neighborhood of the current solution. To carry out this process, the *combinations* method belonging to the Python Itertools module is used, which generates all possible combinations of the sections and their assigned room of the current solution. In addition, since it is possible that not all the available rooms are assigned in the time block being evaluated, it is added before generating the neighborhood.

The movements generated are of two types, the first type is an exchange movement, which consists of exchanging the already assigned rooms of two sections (Figure 2).

```
['BPTMI01-6', 'A1-216', 'BPTMI01-2', 'A2-001']
['BPTMI01-6', 'A1-216', 'BPTMI01-4', 'A2-000']
['BPTMI01-6', 'A1-216', 'FGTMM01-1', 'EMG_19']
['BPTMI01-6', 'A1-216', 'FGTMM01-2', 'A1-110']
['BPTMI01-6', 'A1-216', 'BPTMI02-2', 'A1-201']
['BPTMI01-6', 'A1-216', 'BPTMI02-4', 'A2-007']
['BPTMI01-6', 'A1-216', 'FGTIE01-5', 'A1-203']
['BPTMI01-6', 'A1-216', 'BPTMI11-1', 'A2-309']
['BPTMI01-6', 'A1-216', 'BPTMI11-3', 'A2-206']
['BPTMI01-6', 'A1-216', 'FPTGT04-1', 'A2-003']
['BPTMI01-6', 'A1-216', 'FPTGT04-2', 'A2-207']
['BPTMI01-6', 'A1-216', 'FPTGT04-3', 'A2-310']
['BPTMI01-6', 'A1-216', 'BPTCC15-1', 'A2-204']
```

Figure 4. Example of exchange movements generated.
Source: own elaboration.

Figure 2 shows several exchange movements, for example, the first movement shows how section BPTMI01-6 is assigned room A1-216 and section BPTMI01-2 is assigned room A2-001, if this movement is performed, the rooms will be exchanged, i.e., the solution generated would be assigned room A2-001 in section BPTMI01-6 and room A2-001 in section BPTMI01-2.

The second type of movement refers to delete/add that can be reported in Figure 3, which shows that the first movement that the room A1-104 is not assigned to any section in the current solution, if the first movement is made, the room A1-104 would be assigned to the section BPTMI01-2 and the room A2-001 would not be assigned to any section. This happens because the mentioned room does not meet the requirements for the section of that subject, therefore, it will always be given priority in the key requirements to teach the course, i.e. it is the strong restrictions for the assignment of classrooms.

```
['BPTMI01-2', 'A2-001', 'no', 'A1-104']
['BPTMI01-2', 'A2-001', 'no', 'A1-108']
['BPTMI01-2', 'A2-001', 'no', 'A1-112']
['BPTMI01-2', 'A2-001', 'no', 'A1-208']
['BPTMI01-2', 'A2-001', 'no', 'A1-312']
['BPTMI01-2', 'A2-001', 'no', 'A1-316']
['BPTMI01-2', 'A2-001', 'no', 'A2-308']
['BPTMI01-2', 'A2-001', 'no', 'EMG_12']
['BPTMI01-2', 'A2-001', 'no', 'EMG_14']
['BPTMI01-2', 'A2-001', 'no', 'EMG_17']
['BPTMI01-2', 'A2-001', 'no', 'EMG_20']
['BPTMI01-2', 'A2-001', 'no', 'EMG_21']
['BPTMI01-2', 'A2-001', 'no', 'SL-007']
```

Figure 5. Examples of generated delete add movements.

Source: own elaboration.

Exchange of sections and rooms (Swap)

The module of swapping sections and classrooms is in charge of performing the function of switching between 2 sections and 2 classrooms. This is key in the process of assigning classrooms using a tabu search algorithm. Because it evaluates each possible combination and obtains the smallest objective function. To carry out this process, the module receives the current solution being worked on together with the pair of sections and classrooms to which the change will be made. Once the change is made, the current solution updated with the new change is returned. This module can also assign a room not assigned in the solution to a section, for this it only exchanges the room of the first section it receives.

Function Objective

The calculation of the objective function value of the solution to be evaluated was based on the mathematical model previously explained. To carry out this module, the team took the current solution and divided it into two new *dataframes*, one corresponding to the face-to-face sections and the other to the semi-face-to-face sections. Once both *dataframes* were obtained, the sum of the difference between the number of students enrolled in the section and the capacity of the assigned classroom was calculated in each one. As well as the sum of the classrooms that do not match the type of classroom. Both summations are performed to calculate the final cost of the objective function and then multiplied by the cost assigned in the tabu search function to obtain the total cost of any solution.

Validation of Strong Constraints

To carry out the validation of the hard constraints, the team designed the tabu search algorithm considering them. In addition, validations were designed to be fulfilled within the system itself, as well as there are other constraints which are also validated with code to verify that the solution complies with the constraints.

The following is a detailed explanation of the procedure carried out to validate each of the restrictions mentioned in the mathematical model.

1. The assigned classroom capacity cannot be less than the number of students enrolled in the section. A module was designed which checks that the number of students enrolled is less than the capacity of students in that classroom.
2. A classroom can only have one section assigned to it in the same time block and on the same day. A module was designed which checks that there is only one classroom assigned to a section in a time block.
3. The time block required by the section must be the same as the one assigned to it in the classroom. The classroom assignment system was designed so that there are only classroom and section changes and the system was not given the ability to make changes between time blocks.
4. Virtual sections should not be assigned any classroom in any time block and on any day. The classroom assignment system was designed in such a way that there is only assignment to sections type “P” (Presential) or type “SP” (Semi-Presential) if the system gets any section with any other type it automatically discards it.
5. In the classroom type sections, the room assigned must be the same on both days. The design of the Swap module included in the code that once the assignment is made to a section of a room, the movement will be replicated for the other day.
6. All blended learning sections must be assigned to a classroom, a time block and a day. The classroom assignment system is in charge of assigning each section to a classroom on the corresponding day and the time block is defined from the beginning by the university in the data provided.
7. Sections that require two time blocks on the same day in a row must be assigned the same room. A module was designed which identifies these sections with more than one time block and at the time of evaluating each possible solution in the strong constraints validation module, each solution is checked to ensure that it complies with this requirement, otherwise it is discarded.
8. The sections that require three time blocks in the same day in a row must be assigned the same room. A module was designed which identifies these sections with more than one time block and at the time of evaluating each possible solution in the strong constraint validation module, each solution is checked to ensure that it complies with this requirement, otherwise it is discarded.

9. Sections requiring a laboratory must be assigned a laboratory of the required type. Sections requiring a specific laboratory are assigned using the non-taboo section assignment module.
10. Sections requiring a specific classroom (music, Thespis) must be assigned the specific classroom requested. S was responsible for assigning the sections that require a specific classroom (trial room, music and Thespis), for this purpose the module assignment of non-taboo sections is used.

Taboo Search

The procedure for carrying out the tabu search will be presented below. Initially, the value of the objective function of the initial solution is calculated. Then, the generation of the neighborhood is started. For this procedure, a random sample of 25% of the total size of the neighborhood generated by the neighborhood generation module is obtained, after obtaining this space of solutions, we proceed to verify if it complies with the module of strong constraints. If the movement complies with the restrictions, the movement is saved, otherwise it is discarded. For a neighborhood to be admissible it is necessary that the size of the neighborhood that meets the restrictions is at least 50% of the generated one, otherwise a new neighborhood is searched.

It was decided to randomly select 25% of the movements of the neighborhood of the solution under evaluation, because reviewing the entire neighborhood would take too long and was not effective, since it took many iterations without obtaining an improvement in the value of the objective function of the best solution found.

Once the new neighborhood has been obtained, the best movement is selected, the one with the lowest objective value. Once this movement has been obtained, we proceed to check whether this change will continue to comply with the hard constraints. If it proceeds satisfactorily, it is evaluated if the movement belongs or not to the taboo list. If it does not belong, it is added to the list and the solution is updated with the new change. If it does, it is also evaluated whether that move could improve the current best solution. If the move improves the solution, aspiration criterion is applied, which is responsible for skipping the taboo constraint and performing the move. If the movement does not improve the current best solution, it is discarded.

Additionally, it can be mentioned that every certain number of iterations that occur in the tabu search the neighborhood is renewed in the same way as explained above. This is because, it was achieved that the value of the objective function is significantly improved by renewing every certain iteration the neighborhood and not every interaction.

The number of iterations necessary to renew the neighborhood varies according to the size of the neighborhood of the current solution, because the value selected is 10% of the neighborhood length divided by 10 if it is greater than 100, otherwise it is not divided. This method was obtained after trying different values to select each iteration, which must be renewed in the neighborhood and it was chosen to vary according to the neighborhood

because in certain periods of Sunday or Friday the neighborhood can be very small. This is because the university has very few courses on these days.

It can also be mentioned that, there is a stop condition of the algorithm which occurs if the objective function value does not improve after 200 iterations. Finally, a solution is obtained which meets all the constraints and also significantly decreases the value of the objective function. For a detailed explanation see Figure 4 which graphically represents a diagram of the logic.

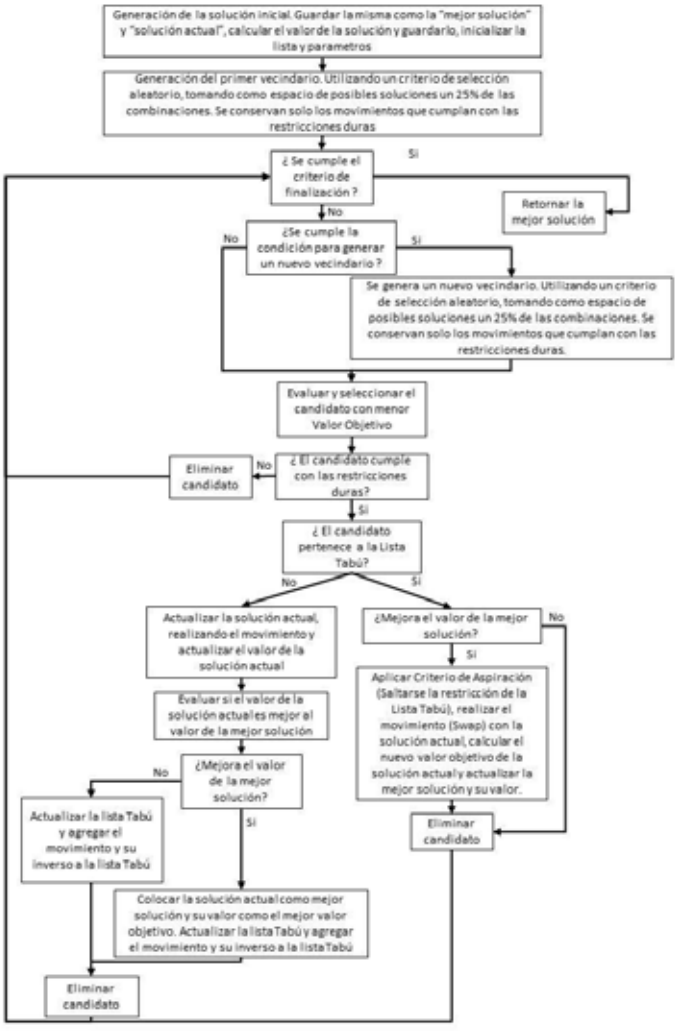


Figure 6. Tabu search algorithm logic diagram.

Source: Own elaboration.

For the code of the tabu search algorithm, see Appendix A.

Classroom Assignment System Design

In order to visualize more easily the mapping performed with the tabu search algorithm, a system was developed using the Python *tkinder* library to create the graphical interface.

The following user requirements were defined in order to know what the system needs to do:

- The system should automatically assign classrooms.
- You must be able to visualize the assignment of classrooms.
- The system must be able to extract the classroom assignment from the system in order to save it.
- The system must have an intuitive interface.

The following data is required for the system to work:

- Academic offer in a csv file.
- University classroom information, three separate csv files should be used as follows:
  - First file with the information of the T (theoretical), M (multimedia) and B (system laboratory) classrooms.
  - Second file with information on laboratories and classrooms with specific uses.
  - Third file only with the names of the T (theoretical), M (multimedia) and B (system laboratory) classrooms.

It is important to note that, when replacing the files already placed in the system with the new ones, these must be adapted to the structure of the old ones.

The system manipulates the data through the tabu search algorithm to generate the classroom assignment and the system output would be:

- Classroom assignment in an Excel file.
- Assignment of classrooms of the selected time block and days in an Excel file.

Analysis and interpretation of results

With the execution of Tabu Search was performed taking into consideration the requirements of the university, to successfully fulfill a series of tests were performed to the algorithm to evaluate its effectiveness on the following parameters: the value of the objective function and the execution time.

Initially, the results corresponding to the calculation of the taboo period will be presented. As well as, the calculation of the best cost associated with the type of classroom requested by the section does not correspond to the type of classroom of the room on which it was assigned (C2), which were obtained through experimentation. In addition, with respect to the cost associated with the difference between the number of students enrolled in the section and the capacity of the classroom assigned to that section (C1), it was established that it will be set at 1, this is due to the value of the difference between the capacity of the classroom and the number of students enrolled in the section.

These results are presented in tabular form, taking into account each of the parameters set above in order to understand the behavior of the Tabu Search algorithm.

It is important to note that the assignment provided pertains to a quarter of 2017 and the academic offering for that quarter was used in the tabu search algorithm and the initial solution.

The following table 1 contains a series of abbreviations used in this chapter to better understand the results obtained.

Table 1. Abbreviations used.

Abreviacion	Significado
LM	Conjunto de secciones presentes los dias Lunes y Miercoles
MJ	Conjunto de secciones presentes los dias Martes y Jueves
V	Conjunto de secciones presente los Viernes
S	Conjunto de secciones presente los Sabado
D	Conjunto de secciones presente los Domingo
1	Bloque horario 1. Referente a clases comprendidas desde 7:00AM - 8:30AM
2	Bloque horario 2. Referente a clases comprendidas desde 8:45AM - 10:15AM
3	Bloque horario 3. Referente a clases comprendidas desde 10:30AM - 12:00PM
4	Bloque horario 4. Referente a clases comprendidas desde 12:15PM - 1:45PM
5	Bloque horario 5. Referente a clases comprendidas desde 2:00PM - 3:30PM
6	Bloque horario 6. Referente a clases comprendidas desde 3:45PM - 5:15PM
7	Bloque horario 7. Referente a clases comprendidas desde 5:30PM - 7:00PM
8	Bloque horario 8. Referente a clases comprendidas desde 7:15PM - 8:45PM

Source: Own elaboration.

Table 2 shows the results obtained corresponding to the calculation of the Tabu period. The remaining calculations were performed under the Tabu period obtained.

Table 2. Calculation of the best tabu period.

Dataframe	Periodo Tabu	Valor funcion objetivo	Tiempo (Segundos)
LM2	10	567	276,7490416
LM2	20	562	381,0043287
LM2	30	567	257,6215920
LM2	40	567	381,2482698
LM2	50	562	258,1157598
LM2	60	567	380,3369820
LM2	70	562	380,9133692
LM2	80	562	383,8990729
LM2	90	569	423,2832067
LM2	100	567	291,5418239
LM2	√(N)	567	271,9696901

LM2 Posee un Valor funcion objetivo inicial de: 951

Source: Own elaboration.

IV.2 Results

IV.2.3 According to the time required to obtain each solution

To calculate the time needed to obtain each solution, the team used a Python library called *timeit*, which measures the time from the beginning of the execution of a certain function until its end. Table 3 shows the results obtained when calculating the solutions of the Tabu Search Algorithm.

Table 3. Time required to obtain the solution of the Tabu Search Algorithm.

Tiempo necesario para obtener la solución	
Dataframe	Algoritmo de Búsqueda Tabú
LM Total	31,74
MJ Total	41,24
V Total	21,72
S Total	0,00
D Total	0,33
Tiempo Total (Minutos)	95,04

Source: own elaboration.

According to the results shown in Table 13, it can be observed that from the calculation of each of the *dataframe* times, a feasible solution was generated in 95.04 minutes.

On the other hand, according to E. Oberto, "Universidad Metropolitana currently assigns classrooms manually, respecting the requirements of each subject and of the professors, thus minimizing the number of complaints generated by the previous system. They also mentioned that, the time dedicated to perform this process is two to three days generally on weekends." (personal communication June 2, 2021).

IV.3 Classroom Assignment System

Figure 7 shows the initial screen of the system developed by the team. Here, the dataframe to work with is selected, as well as the time block.

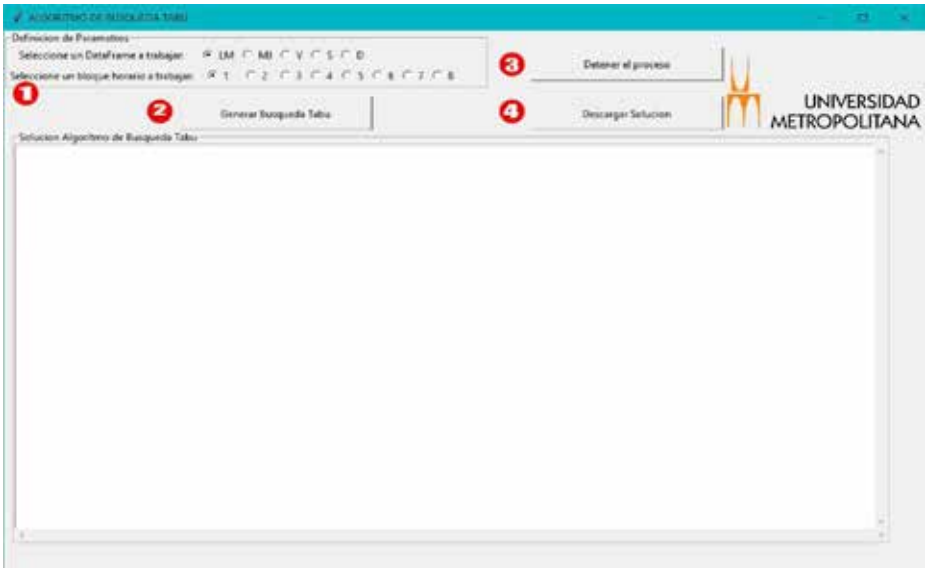


Figure 7. Initial system screen.
Source: own elaboration.

The following are the steps to follow for system startup and the correct way to run the tabu search.

1. The dataframe and the time block to be worked on are selected.
2. The "Generate Tabu Search" button is in charge of interacting with the Tabu Search

Algorithm and performing the search according to the parameters selected in point 1. Note that, when clicked, the process starts, but it may take several minutes to generate a solution.

3. The “Stop process” button is in charge of stopping and closing the program if it is necessary to stop the system.

4. The “Download Solution” button is responsible for downloading the solution generated by the Tabu Search Algorithm in an Excel file. It has validations so that the file cannot be downloaded if a solution has not been generated.

Figure 8 shows the system screen with a solution generated using the MJ *dataframe* and period 2. Using the scroll bars, the generated solution can be fully appreciated. At this point, the “Download Solution” button will allow you to download the generated solution in an Excel file which will be stored in the folder where you have the system saved.

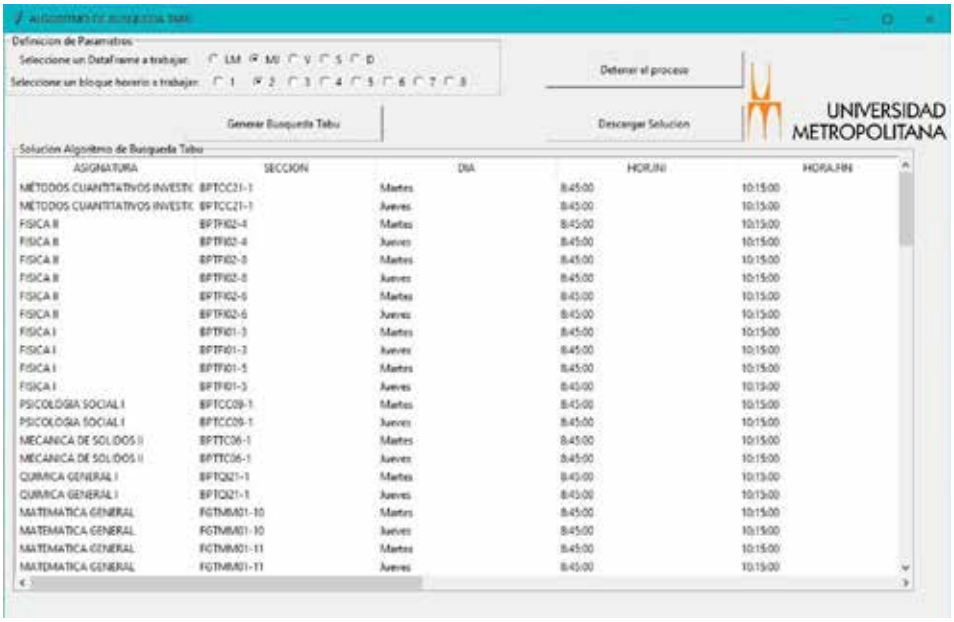


Figure 8 Sample solution of the system.
Source: own elaboration.

Figure 9 shows the system screen once the “Download solution” button is clicked. It will generate a window for the user to confirm if he/she wants to download the solution.

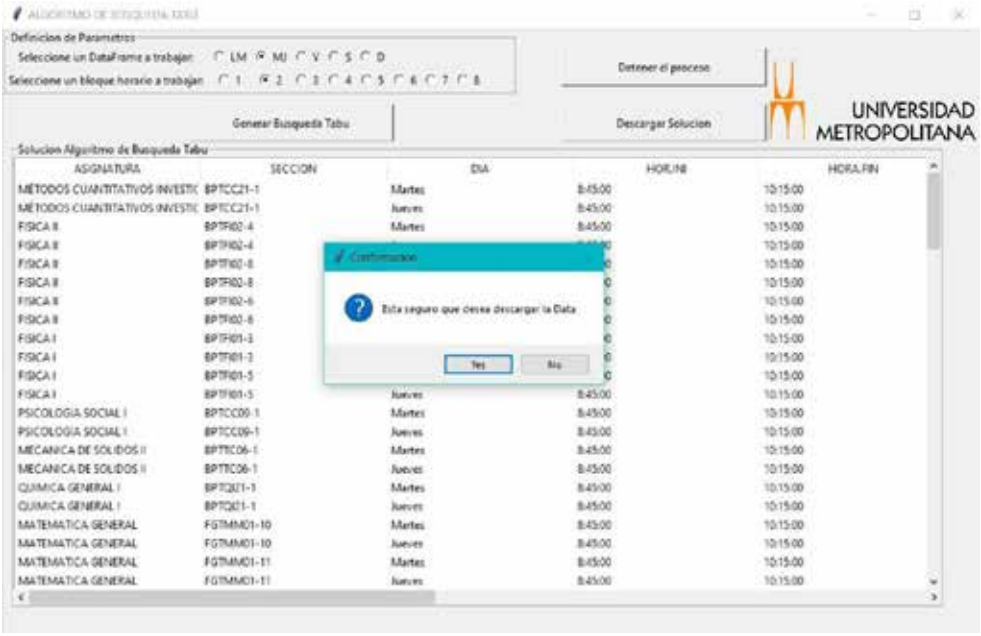


Figure 9. Solution download.
Source: own elaboration.

IV.4 System Validation

Once the results of the classroom allocation through the Tabu Search Algorithm were obtained, the team proceeded to verify if the solution complied with the conditions of the Metropolitan University classroom allocation. To verify this, one must check if all the hard constraints posed in the mathematical model are met. Most of the hard constraints are met automatically due to the way the system was designed, only the following constraints must be evaluated during the search algorithm (listed according to the mathematical model):

- Restriction 1: The assigned classroom capacity cannot be less than the number of students enrolled in the section.
- Restriction 2: A classroom can only have one section assigned to it in the same time block and on the same day.

- Restriction 9: Sections that require two time blocks on the same day in a row must be assigned the same room.
- Restriction 10: Sections that require three time blocks on the same day in a row must be assigned the same room.

To evaluate whether these constraints are met in the solution generated by the tabu algorithm for an hourly block, the hard constraints function was used, see Appendix A for the function code, which yields true if the generated solution meets the four constraints named above.

In Table 4, it can be observed that the function check with the solutions generated by the tabu search algorithm for blocks LM1, LM2, MJ2, MJ3 and D3. As can be seen, all four constraints are met, so the generated solutions meet the conditions of the MU assignment.

Table 4. Validation of hard constraints by Dataframe

Validación restricciones duras				
Dataframe	Restricción 1	Restricción 2	Restricción 9	Restricción 10
LM1	Se cumple	Se cumple	Se cumple	Se cumple
LM2	Se cumple	Se cumple	Se cumple	Se cumple
MJ2	Se cumple	Se cumple	Se cumple	Se cumple
MJ3	Se cumple	Se cumple	Se cumple	Se cumple
D3	Se cumple	Se cumple	Se cumple	Se cumple

Source: own elaboration.

IV.5 Solutions generated

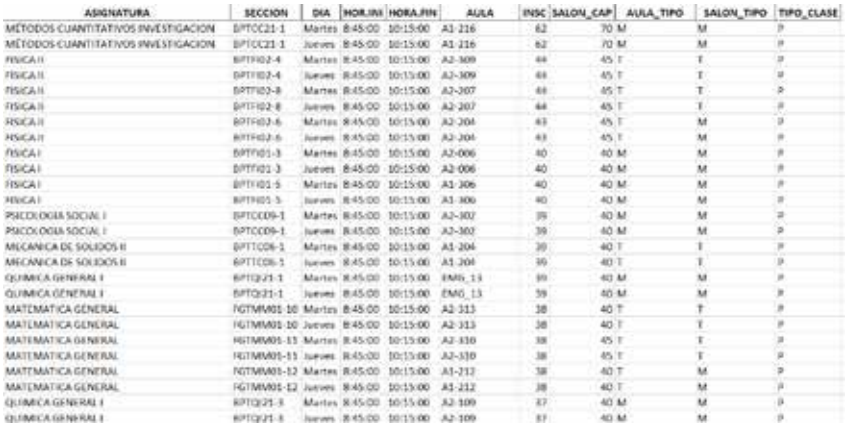
In the following figures you can see a fragment of the solution generated for the hourly blocks LM1, LM2, MJ2 and MJ3 after being extracted from the system, i.e. in Excel file (see Figure 10, Figure 11, Figure 12 and Figure 13).

ASIGNATURA	SECCION	DIA	HOR.INI	HORA.FIN	AULA	INSC	SALON_CAP	AULA_TIPO	SALON_TIPO	TIPO_CLASE
MATEMATICAS I	BPTMI01-5	Lunes	8:45:00	10:15:00	A1-204	39	40 T		T	P
MATEMATICAS I	BPTMI01-6	Miercoles	8:45:00	10:15:00	A1-204	39	40 T		T	P
MATEMATICAS I	BPTMI01-2	Lunes	8:45:00	10:15:00	A2-207	38	45 T		T	P
MATEMATICAS I	BPTMI01-2	Miercoles	8:45:00	10:15:00	A2-207	38	45 T		T	P
MATEMATICAS I	BPTMI01-4	Lunes	8:45:00	10:15:00	A1-303	38	40 M		M	P
MATEMATICAS I	BPTMI01-4	Miercoles	8:45:00	10:15:00	A1-303	38	40 M		M	P
MATEMATICA GENERAL	FGTMM01-1	Lunes	8:45:00	10:15:00	A1-205	38	40 T		T	P
MATEMATICA GENERAL	FGTMM01-1	Miercoles	8:45:00	10:15:00	A1-205	38	40 T		T	P
MATEMATICA GENERAL	FGTMM01-2	Lunes	8:45:00	10:15:00	A2-310	38	45 T		T	P
MATEMATICA GENERAL	FGTMM01-2	Miercoles	8:45:00	10:15:00	A2-310	38	45 T		T	P
MATEMATICAS II	BPTMI02-2	Lunes	8:45:00	10:15:00	EMG_19	37	45 M		M	P
MATEMATICAS II	BPTMI02-2	Miercoles	8:45:00	10:15:00	EMG_19	37	45 M		M	P
MATEMATICAS II	BPTMI02-4	Lunes	8:45:00	10:15:00	A2-309	37	45 T		T	P
MATEMATICAS II	BPTMI02-4	Miercoles	8:45:00	10:15:00	A2-309	37	45 T		T	P
COMPETENCIAS EN ACCION	FGTIE01-5	Lunes	8:45:00	10:15:00	A2-312	35	40 M		M	P
COMPETENCIAS EN ACCION	FGTIE01-5	Miercoles	8:45:00	10:15:00	A2-312	35	40 M		M	P
ECUACIONES DIFERENCIALES	BPTMI11-1	Lunes	8:45:00	10:15:00	A1-300	35	35 M		M	P
ECUACIONES DIFERENCIALES	BPTMI11-1	Miercoles	8:45:00	10:15:00	A1-300	35	35 M		M	P
ECUACIONES DIFERENCIALES	BPTMI11-3	Lunes	8:45:00	10:15:00	A1-207	35	35 M		M	P
ECUACIONES DIFERENCIALES	BPTMI11-3	Miercoles	8:45:00	10:15:00	A1-207	35	35 M		M	P
OPTIMIZACION I	FPTGT04-1	Lunes	8:45:00	10:15:00	A1-102	35	40 M		M	P
OPTIMIZACION I	FPTGT04-1	Miercoles	8:45:00	10:15:00	A1-102	35	40 M		M	P
OPTIMIZACION I	FPTGT04-2	Lunes	8:45:00	10:15:00	A2-004	34	40 M		M	P
OPTIMIZACION I	FPTGT04-2	Miercoles	8:45:00	10:15:00	A2-004	34	40 M		M	P
OPTIMIZACION I	FPTGT04-3	Lunes	8:45:00	10:15:00	A1-202	34	40 M		M	P
OPTIMIZACION I	FPTGT04-3	Miercoles	8:45:00	10:15:00	A1-202	34	40 M		M	P

Figure 10. Fragment of the solution generated for LM1.
Source: own elaboration.

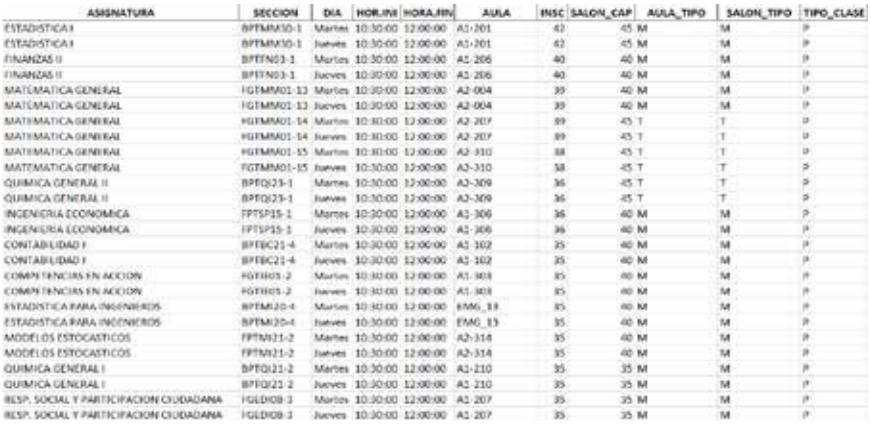
ASIGNATURA	SECCION	DIA	HOR.INI	HORA.FIN	AULA	INSC	SALON_CAP	AULA_TIPO	SALON_TIPO	TIPO_CLASE
MATEMATICAS I	BPTMI01-6	Lunes	8:45:00	10:15:00	A1-204	39	40 T		T	P
MATEMATICAS I	BPTMI01-6	Miercoles	8:45:00	10:15:00	A1-204	39	40 T		T	P
MATEMATICAS I	BPTMI01-2	Lunes	8:45:00	10:15:00	A2-207	38	45 T		T	P
MATEMATICAS I	BPTMI01-2	Miercoles	8:45:00	10:15:00	A2-207	38	45 T		T	P
MATEMATICAS I	BPTMI01-4	Lunes	8:45:00	10:15:00	A1-303	38	40 M		M	P
MATEMATICAS I	BPTMI01-4	Miercoles	8:45:00	10:15:00	A1-303	38	40 M		M	P
MATEMATICA GENERAL	FGTMM01-1	Lunes	8:45:00	10:15:00	A1-205	38	40 T		T	P
MATEMATICA GENERAL	FGTMM01-1	Miercoles	8:45:00	10:15:00	A1-205	38	40 T		T	P
MATEMATICA GENERAL	FGTMM01-2	Lunes	8:45:00	10:15:00	A2-310	38	45 T		T	P
MATEMATICA GENERAL	FGTMM01-2	Miercoles	8:45:00	10:15:00	A2-310	38	45 T		T	P
MATEMATICAS II	BPTMI02-2	Lunes	8:45:00	10:15:00	EMG_19	37	45 M		M	P
MATEMATICAS II	BPTMI02-2	Miercoles	8:45:00	10:15:00	EMG_19	37	45 M		M	P
MATEMATICAS II	BPTMI02-4	Lunes	8:45:00	10:15:00	A2-309	37	45 T		T	P
MATEMATICAS II	BPTMI02-4	Miercoles	8:45:00	10:15:00	A2-309	37	45 T		T	P
COMPETENCIAS EN ACCION	FGTIE01-5	Lunes	8:45:00	10:15:00	A2-312	35	40 M		M	P
COMPETENCIAS EN ACCION	FGTIE01-5	Miercoles	8:45:00	10:15:00	A2-312	35	40 M		M	P
ECUACIONES DIFERENCIALES	BPTMI11-1	Lunes	8:45:00	10:15:00	A1-300	35	35 M		M	P
ECUACIONES DIFERENCIALES	BPTMI11-1	Miercoles	8:45:00	10:15:00	A1-300	35	35 M		M	P
ECUACIONES DIFERENCIALES	BPTMI11-3	Lunes	8:45:00	10:15:00	A1-207	35	35 M		M	P
ECUACIONES DIFERENCIALES	BPTMI11-3	Miercoles	8:45:00	10:15:00	A1-207	35	35 M		M	P
OPTIMIZACION I	FPTGT04-1	Lunes	8:45:00	10:15:00	A1-102	35	40 M		M	P
OPTIMIZACION I	FPTGT04-1	Miercoles	8:45:00	10:15:00	A1-102	35	40 M		M	P
OPTIMIZACION I	FPTGT04-2	Lunes	8:45:00	10:15:00	A2-004	34	40 M		M	P
OPTIMIZACION I	FPTGT04-2	Miercoles	8:45:00	10:15:00	A2-004	34	40 M		M	P
OPTIMIZACION I	FPTGT04-3	Lunes	8:45:00	10:15:00	A1-202	34	40 M		M	P
OPTIMIZACION I	FPTGT04-3	Miercoles	8:45:00	10:15:00	A1-202	34	40 M		M	P

Figure 11. Fragment of the solution generated for LM2.
Source: own elaboration.



ASIGNATURA	SECCION	DIA	HOR.INI	HORA.FIN	AULA	INSC	SALON_CAP	AULA_TIPO	SALON_TIPO	TIPO_CLASE
MÉTODOS CUANTITATIVOS INVESTIGACION	BPTCC21-1	Martes	8:45:00	10:15:00	A1-216	62	70	M	M	P
MÉTODOS CUANTITATIVOS INVESTIGACION	BPTCC21-1	Jueves	8:45:00	10:15:00	A1-216	62	70	M	M	P
FISICA II	BPTFI02-4	Martes	8:45:00	10:15:00	A2-309	44	45	T	T	P
FISICA II	BPTFI02-4	Jueves	8:45:00	10:15:00	A2-309	44	45	T	T	P
FISICA II	BPTFI02-8	Martes	8:45:00	10:15:00	A2-207	44	45	T	T	P
FISICA II	BPTFI02-8	Jueves	8:45:00	10:15:00	A2-207	44	45	T	T	P
FISICA II	BPTFI02-6	Martes	8:45:00	10:15:00	A2-204	43	45	T	M	P
FISICA II	BPTFI02-6	Jueves	8:45:00	10:15:00	A2-204	43	45	T	M	P
FISICA I	BPTFI01-3	Martes	8:45:00	10:15:00	A2-006	40	40	M	M	P
FISICA I	BPTFI01-3	Jueves	8:45:00	10:15:00	A2-006	40	40	M	M	P
FISICA I	BPTFI01-5	Martes	8:45:00	10:15:00	A1-306	40	40	M	M	P
FISICA I	BPTFI01-5	Jueves	8:45:00	10:15:00	A1-306	40	40	M	M	P
PSICOLOGIA SOCIAL I	BPTCC09-1	Martes	8:45:00	10:15:00	A2-302	39	40	M	M	P
PSICOLOGIA SOCIAL I	BPTCC09-1	Jueves	8:45:00	10:15:00	A2-302	39	40	M	M	P
MECANICA DE SOLIDOS II	BPTTC06-1	Martes	8:45:00	10:15:00	A1-204	39	40	T	T	P
MECANICA DE SOLIDOS II	BPTTC06-1	Jueves	8:45:00	10:15:00	A1-204	39	40	T	T	P
QUIMICA GENERAL I	BPTQI21-1	Martes	8:45:00	10:15:00	EMG_13	39	40	M	M	P
QUIMICA GENERAL I	BPTQI21-1	Jueves	8:45:00	10:15:00	EMG_13	39	40	M	M	P
MATEMATICA GENERAL	FGTMM01-10	Martes	8:45:00	10:15:00	A2-313	38	40	T	T	P
MATEMATICA GENERAL	FGTMM01-10	Jueves	8:45:00	10:15:00	A2-313	38	40	T	T	P
MATEMATICA GENERAL	FGTMM01-11	Martes	8:45:00	10:15:00	A2-310	38	45	T	T	P
MATEMATICA GENERAL	FGTMM01-11	Jueves	8:45:00	10:15:00	A2-310	38	45	T	T	P
MATEMATICA GENERAL	FGTMM01-12	Martes	8:45:00	10:15:00	A1-212	38	40	T	M	P
MATEMATICA GENERAL	FGTMM01-12	Jueves	8:45:00	10:15:00	A1-212	38	40	T	M	P
QUIMICA GENERAL I	BPTQI21-3	Martes	8:45:00	10:15:00	A2-109	37	40	M	M	P
QUIMICA GENERAL I	BPTQI21-3	Jueves	8:45:00	10:15:00	A2-109	37	40	M	M	P

Figure 12. Fragment of the solution generated for MJ2.
Source: own elaboration.



ASIGNATURA	SECCION	DIA	HOR.INI	HORA.FIN	AULA	INSC	SALON_CAP	AULA_TIPO	SALON_TIPO	TIPO_CLASE
ESTADISTICA I	BPTMM30-1	Martes	10:30:00	12:00:00	A1-201	42	45	M	M	P
ESTADISTICA I	BPTMM30-1	Jueves	10:30:00	12:00:00	A1-201	42	45	M	M	P
FINANZAS II	BPTFN03-1	Martes	10:30:00	12:00:00	A1-206	40	40	M	M	P
FINANZAS II	BPTFN03-1	Jueves	10:30:00	12:00:00	A1-206	40	40	M	M	P
MATEMATICA GENERAL	FGTMM01-13	Martes	10:30:00	12:00:00	A2-004	39	40	M	M	P
MATEMATICA GENERAL	FGTMM01-13	Jueves	10:30:00	12:00:00	A2-004	39	40	M	M	P
MATEMATICA GENERAL	FGTMM01-14	Martes	10:30:00	12:00:00	A2-207	39	45	T	T	P
MATEMATICA GENERAL	FGTMM01-14	Jueves	10:30:00	12:00:00	A2-207	39	45	T	T	P
MATEMATICA GENERAL	FGTMM01-15	Martes	10:30:00	12:00:00	A2-310	38	45	T	T	P
MATEMATICA GENERAL	FGTMM01-15	Jueves	10:30:00	12:00:00	A2-310	38	45	T	T	P
QUIMICA GENERAL II	BPTQI23-1	Martes	10:30:00	12:00:00	A2-309	36	45	T	T	P
QUIMICA GENERAL II	BPTQI23-1	Jueves	10:30:00	12:00:00	A2-309	36	45	T	T	P
INGENIERIA ECONOMICA	FPTSP15-1	Martes	10:30:00	12:00:00	A1-306	36	40	M	M	P
INGENIERIA ECONOMICA	FPTSP15-1	Jueves	10:30:00	12:00:00	A1-306	36	40	M	M	P
CONTABILIDAD I	BPTEC21-4	Martes	10:30:00	12:00:00	A1-302	35	40	M	M	P
CONTABILIDAD I	BPTEC21-4	Jueves	10:30:00	12:00:00	A1-302	35	40	M	M	P
COMPETENCIAS EN ACCION	FGTEH05-2	Martes	10:30:00	12:00:00	A1-303	35	40	M	M	P
COMPETENCIAS EN ACCION	FGTEH05-2	Jueves	10:30:00	12:00:00	A1-303	35	40	M	M	P
ESTADISTICA PARA INGENIEROS	BPTMI20-4	Martes	10:30:00	12:00:00	EMG_13	35	40	M	M	P
ESTADISTICA PARA INGENIEROS	BPTMI20-4	Jueves	10:30:00	12:00:00	EMG_13	35	40	M	M	P
MODELOS ESTOCASTICOS	FPTNI21-2	Martes	10:30:00	12:00:00	A2-314	35	40	M	M	P
MODELOS ESTOCASTICOS	FPTNI21-2	Jueves	10:30:00	12:00:00	A2-314	35	40	M	M	P
QUIMICA GENERAL I	BPTQI21-2	Martes	10:30:00	12:00:00	A1-210	35	35	M	M	P
QUIMICA GENERAL I	BPTQI21-2	Jueves	10:30:00	12:00:00	A1-210	35	35	M	M	P
RESP. SOCIAL Y PARTICIPACION CIUDADANA	FGEDI08-3	Martes	10:30:00	12:00:00	A1-207	35	35	M	M	P
RESP. SOCIAL Y PARTICIPACION CIUDADANA	FGEDI08-3	Jueves	10:30:00	12:00:00	A1-207	35	35	M	M	P

Figure 13. Fragment of the solution generated for MJ3.
Source: own elaboration.

The above figures are solutions generated by the tabu search that meet the needs and constraints of the university, and several viable solutions were obtained in less than 200 microseconds.

With these tests it can be demonstrated that the Tabu Search algorithm is effective and meets the needs of the university.

Conclusions

- First, the strong and weak constraints of the problem were identified, with the help of the university staff and the assignment provided. This made it possible to ensure the veracity of the results obtained.
- Second, a mathematical model was developed to solve the problem, taking into account the identified constraints.
- Third, a tabu search algorithm was designed based on the mathematical model with. This algorithm focused on optimizing the allocated spaces and decreasing the number of classrooms where the classroom type does not match.
- Finally, a system was developed in the Phyton programming language to evaluate and visualize the results obtained. This system was validated by verifying that it complies with the hard constraints of the Metropolitan University classroom allocation.

Recommendations

- It is suggested to download the assignment data, since, if it is done by time block, the previous assignment will be lost. In addition, the data is downloaded in an excel file that the user can modify if desired.
- At the moment of receiving the data from the university, it is advisable to validate the data from the classrooms. Because there may be disparities between the data placed in the system and the actual data.
- In addition, it is recommended that at the moment of inserting the data into the system, it should be verified that the files are structured in the same way as those pre-established in the system. Also, if possible, it is suggested to replace the information in the existing files.
- Finally, it is recommended that at the moment of returning to face-to-face attendance, each of the university's departments be visited in order to integrate restrictions to the system that are useful for the entire university community.

REFERENCES

Bullet Solutions (n.d.). [online]. Spain. Retrieved from: https://www.bulletsolutions.com/es/

Dréo, J., Pétrowski, A., Siarry, P. and Taillard, P. (2003). *Metaheuristics for Hard Optimization*. Germany: Springer.

Glover, F. (1989). "Tabu Search - Part I" 190-206.

Glover, F. and Batista, B. (2006). Introduction to Taboo search. Spain: Autor.

Glover, F. and Kochenberger, G. (2003). *Handbook of metaheuristic*. New York, Boston, Dordrecht, London and Moscow: Kluwer academic publishers.

Landeau, R. (2007). *Elaboration of research papers.* Venezuela: Alfa

Osman, I. and Kelly, J. (1996). Meta-Heuristics: Theory and Applications, Boston USA Ed. Kluwer.

Pacheco, J. (2021). *Qualitative variable (definition, types, examples and characteristics)*, [online]. Retrieved from: https://www.webyempresas.com/variable-cualitativa/

Pacheco, J. (2021). *Quantitative variables (definition, characteristics and classification)*, [online]. Retrieved from: https://www.webyempresas.com/variables-cuantitativas/

Paneque, R. (1998). Research Methodology: *Basic elements for clinical research.* Retrieved from: http://www.sld.cu/galerias/pdf/sitios/bioestadistica/metodologia_de_la_investigacion_1998.pdf

Pirim, H., Bayraktar, E. and Eksioglu, B. (2008). *Tabu Search: A Comparative Study*. United States: I-Tech Education and Publishing.

Riojas, A. (2005) Concepts, algorithm and application to the N-queen problem. Tabu search. Lima, Perú.

Roldán, P. (2019). *Mathematical model*, [online]. Retrieved from: https://economipedia.com/definiciones/modelo-matematico.html

Suárez, J. G., Manchego, F. A., Nole, A. A., Nicho, G. B., & Anticona, M. T. (2009). *Intelligent Generation of Schedules employing GRASP heuristics with Tabu Search for the Pontificia Universidad Católica del Perú.* https://drive.google.com/file/d/1_oMzD_Cnur6E7ViCk-4keOvboVcvUXdL/view?usp=sharing

William, R. (2012). *Advantages of Modular Programming*. Retrieved from: http://progmodular.blogspot.com/2012/08/ventajas.html

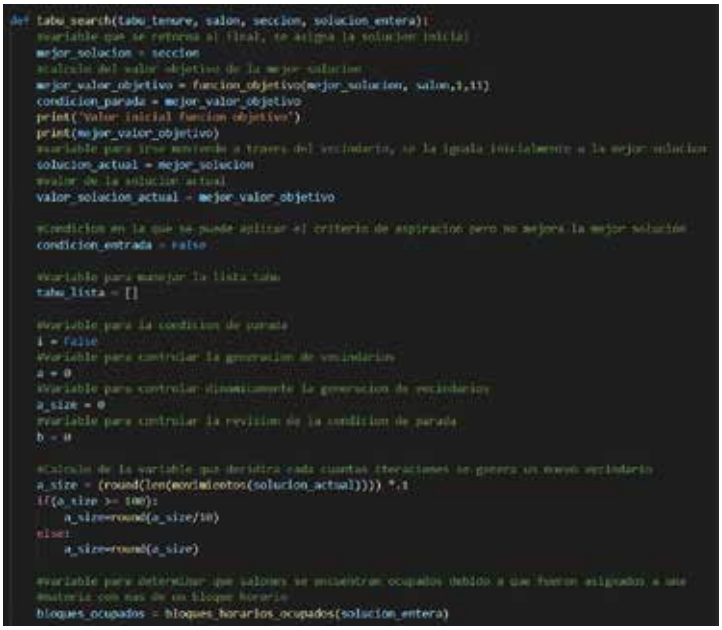
Yepes, V. (2014). *Optimization and mathematical programming,* [online]. Polytechnic University of Valencia. Retrieved from: https://victoryepes.blogs.upv.es/2014/06/05/optimizacion-programacion-matematica/

APPENDICES

Appendix A. Code of the main functions of the tabu algorithm.

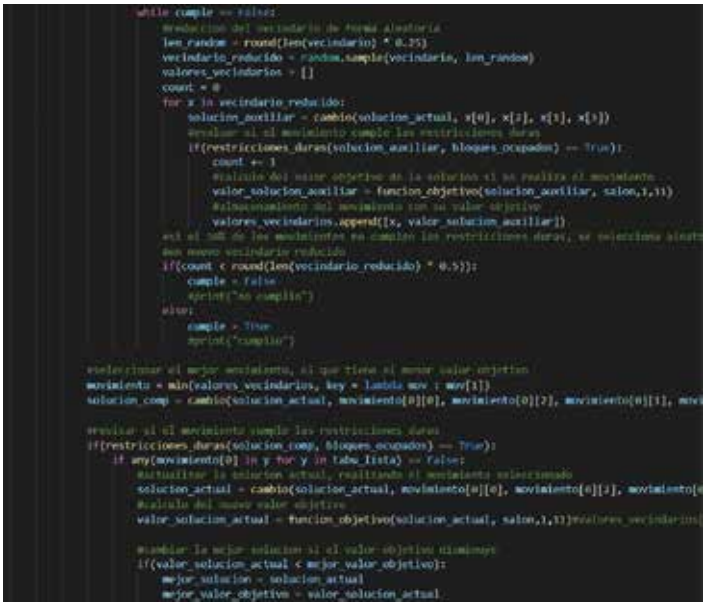
```
#funcion para calcular el valor objetivo de la solucion
def funcion_objetivo(solucion, salon, costo_capacidad, costo_salon_errado):
    acumulado_salones = 0
    secciones_erradas = 0
    valor_funcion_objetivo = 0
    p_solucion = solucion[(solucion["TIPO_CLASE"] == "P")].copy()
    sp_solucion = solucion[(solucion["TIPO_CLASE"] == "SP")].copy()

    #calculo de las secciones presenciales, se emplea un for porque es una sumatoria
    for fila in p_solucion.itertuples():
        #suma de la diferencia de la cantidad de inscritos en la seccion y la capacidad del aula asignada
        acumulado_salones += abs(p_solucion.loc[fila.Index, "SALON_CAP"] - p_solucion.loc[fila.Index, "INSC"])
        #suma de los salones que no coincide el tipo de aula
        if(p_solucion.loc[fila.Index, "AULA_TIPO"] != p_solucion.loc[fila.Index, "SALON_TIPO"]):
            secciones_erradas += 1

    #dividir entre dos, debido a que son dos dias a la semana pero cuenta como uno
    secciones_erradas /= 2
    acumulado_salones /= 2

    #Calculo de las secciones semi presenciales, se emplea un for porque es una sumatoria
    for fila2 in sp_solucion.itertuples():
        #suma de la diferencia de la cantidad de inscritos en la seccion y la capacidad del aula asignada
        acumulado_salones += abs(sp_solucion.loc[fila2.Index, "SALON_CAP"] - sp_solucion.loc[fila2.Index, "INSC"])
        #suma de los salones que no coincide el tipo de aula
        if(sp_solucion.loc[fila2.Index, "AULA_TIPO"] != sp_solucion.loc[fila2.Index, "SALON_TIPO"]):
            secciones_erradas += 1

    #variable que acumula el valor de la diferencia entre los inscritos en la seccion por la capacidad del aula
    #asignada, se multiplica por el costo asociado
    acumulado_salones *= costo_capacidad

    #variable que acumula la cantidad de secciones que no coincide el tipo de aula del salon con el solicitado,
    #se multiplica por el costo asociado
    secciones_erradas *= costo_salon_errado

    #calculo del valor objetivo de la solucion
    valor_fucion_objetivo = acumulado_salones + secciones_erradas

    return valor_fucion_objetivo
```

Figure 14. Function for calculating the target value.
Source: own elaboration.



```
def tabu_search(tabu_tenure, salon, seccion, solucion_entera):
    #variable que se retorna al final, se asigna la solucion inicial
    mejor_solucion = seccion
    #calculo del valor objetivo de la mejor solucion
    mejor_valor_objetivo = funcion_objetivo(mejor_solucion, salon,1,11)
    condicion_parada = mejor_valor_objetivo
    print('Valor inicial funcion objetivo')
    print(mejor_valor_objetivo)
    #variable para ir moviendo a traves del vecindario, se la iguala inicialmente a la mejor solucion
    solucion_actual = mejor_solucion
    #valor de la solucion actual
    valor_solucion_actual = mejor_valor_objetivo

    #condicion en la que se puede aplicar el criterio de aspiracion pero no mejora la mejor solucion
    condicion_entrada = False

    #Variable para manejar la lista tabu
    tabu_lista = []

    #variable para la condicion de parada
    i = False
    #Variable para controlar la generacion de vecindarios
    a = 0
    #Variable para controlar dinamicamente la generacion de vecindarios
    a_size = 0
    #variable para controlar la revision de la condicion de parada
    b = 0

    #Calculo de la variable que decidira cada cuantas iteraciones se genera un nuevo vecindario
    a_size = (round(len(movimientos(solucion_actual)))) *.1
    if(a_size >= 100):
        a_size=round(a_size/10)
    else:
        a_size=round(a_size)

    #variable para determinar que salones se encuentran ocupados debido a que fueron asignados a una
    #materia con mas de un bloque horario
    bloques_ocupados = bloques_horarios_ocupados(solucion_entera)
```

Figure 15. Function to perform the Tabu Search Algorithm (Part 1).
Source: own elaboration.



```
            while cumple == False:
                #reduccion del vecindario de forma aleatoria
                len_random = round(len(vecindario) * 0.25)
                vecindario_reducido = random.sample(vecindario, len_random)
                valores_vecindarios = []
                count = 0
                for x in vecindario_reducido:
                    solucion_auxiliar = cambio(solucion_actual, x[0], x[1], x[2], x[3])
                    #evaluar si el movimiento cumple las restricciones duras
                    if(restricciones_duras(solucion_auxiliar, bloques_ocupados) == True):
                        count += 1
                        #calculo del valor objetivo de la solucion si se realiza el movimiento
                        valor_solucion_auxiliar = funcion_objetivo(solucion_auxiliar, salon,1,11)
                        #almacenamiento del movimiento con su valor objetivo
                        valores_vecindarios.append([x, valor_solucion_auxiliar])
                #si 50% de los movimientos no cumplen las restricciones duras, se selecciona aleato
                #un nuevo vecindario reducido
                if(count < round(len(vecindario_reducido) * 0.5)):
                    cumple = False
                    #print("no cumple")
                else:
                    cumple = True
                    #print("cumple")

        #seleccionar el mejor movimiento, el que tiene el menor valor objetivo
        movimiento = min(valores_vecindarios, key = lambda mov : mov[1])
        solucion_comp = cambio(solucion_actual, movimiento[0][0], movimiento[0][2], movimiento[0][1], mov

        #revisar si el movimiento cumple las restricciones duras
        if(restricciones_duras(solucion_comp, bloques_ocupados) == True):
            if any(movimiento[0] in y for y in tabu_lista) == False:
                #actualizar la solucion actual, realizando el movimiento seleccionado
                solucion_actual = cambio(solucion_actual, movimiento[0][0], movimiento[0][2], movimiento[0
                #calculo del nuevo valor objetivo
                valor_solucion_actual = funcion_objetivo(solucion_actual, salon,1,11)#valores_vecindarios

                #cambiar la mejor solucion si el valor objetivo disminuye
                if(valor_solucion_actual < mejor_valor_objetivo):
                    mejor_solucion = solucion_actual
                    mejor_valor_objetivo = valor_solucion_actual
```

Figure 16. Function to perform the Tabu Search Algorithm (Part 2).
Source: own elaboration.

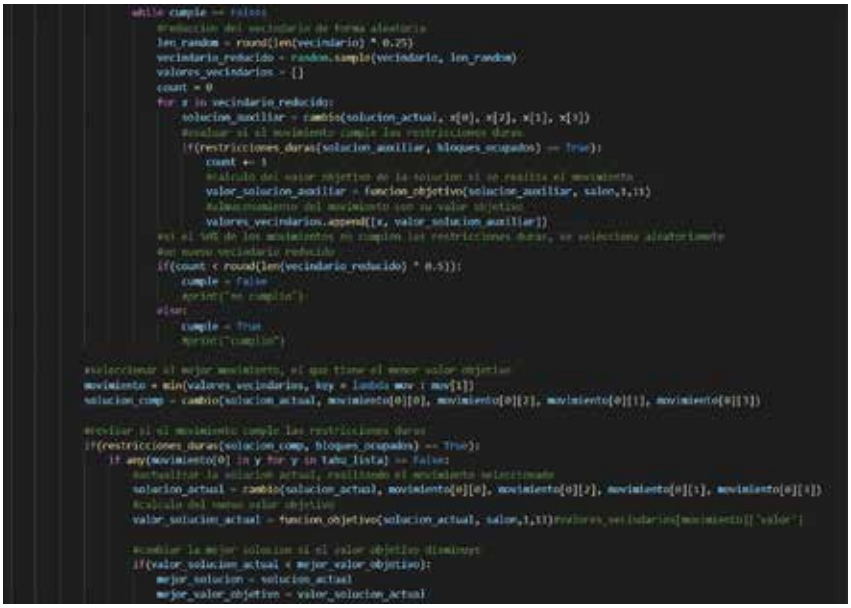


Figure 17. Function to perform the Tabu Search Algorithm (Part 3).
Source: own elaboration.

Figure 18. Function to perform the Tabu Search Algorithm (Part 4).
Source: own elaboration.

```
            #seleccionar nuevo movimiento por que no mejoro el valor de la mejor solucion
            else:
                if(len(valores_vecindarios) > 2):
                    valores_vecindarios.remove(movimiento)
                    movimiento = min(valores_vecindarios, key = lambda mov : mov[1])
                    #print('second min :')
                    condicion_entrada = True
                else:
                    condicion_entrada = False

        #remover el movimiento seleccionado
        else:
            valores_vecindarios.remove(movimiento)

        if(b>=200):
            if(mejor_valor_objetivo == condicion_parada):
                i=True
                break
            else:
                condicion_parada = mejor_valor_objetivo
                b=0
        #aumentar la iteraccion
        a += 1
        b += 1

    return mejor_solucion
```

Figure 19. Function to perform the Tabu Search Algorithm (Part 5).
Source: own elaboration.

```
#Funcion para actualizar la lista tabu, disminuye la tenencia

def update_lista_tabu(lista_tabu):
    for elem in lista_tabu:
        elem[1] = elem[1] - 1

    for item, item1 in lista_tabu:
        if(item1 == 0):
            lista_tabu.remove([item, item1])

    return lista_tabu
```

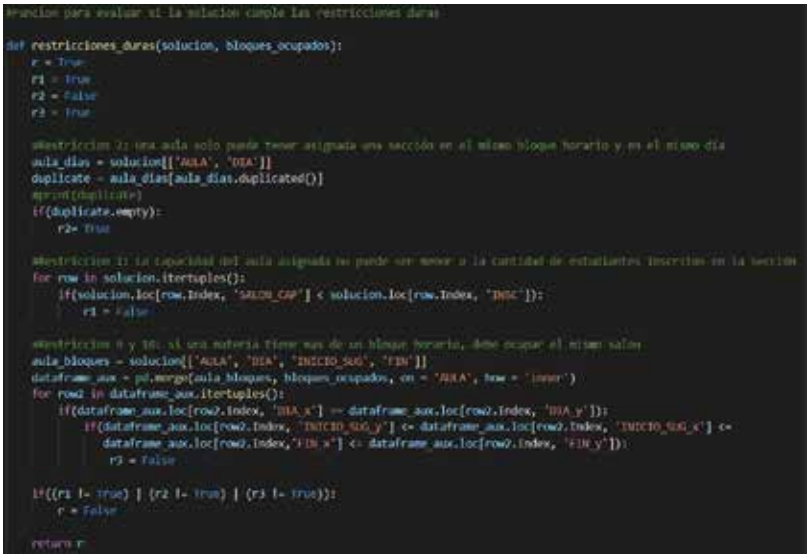
Figure 20. Function for updating the tabu list.
Source: own elaboration.

```python
#funcion para intercambiar los salones entre dos secciones o asignar un salon (no asignado) a una seccion

def cambio(solucion_actual, seccion_1, seccion_2, aula_1, aula_2):
    salon = dict_salones(sal)
    solucion_auxiliar = solucion_actual.copy()

    for row_aux in solucion_auxiliar.itertuples():
        if(solucion_auxiliar.loc[row_aux.Index, 'SECCION'] == seccion_1):
            solucion_auxiliar.at[row_aux.Index, 'AULA'] = aula_2
            solucion_auxiliar.at[row_aux.Index, 'SALON_TIPO'] = salon[aula_2]['tipo']
            solucion_auxiliar.at[row_aux.Index, 'SALON_CAP'] = salon[aula_2]['capacidad']

        elif(solucion_auxiliar.loc[row_aux.Index, 'SECCION'] == seccion_2):
            solucion_auxiliar.at[row_aux.Index, 'AULA'] = aula_1
            solucion_auxiliar.at[row_aux.Index, 'SALON_TIPO'] = salon[aula_1]['tipo']
            solucion_auxiliar.at[row_aux.Index, 'SALON_CAP'] = salon[aula_1]['capacidad']

    return solucion_auxiliar
```

Figure 21. Function to perform the movement (Swap).
Source: own elaboration.

```python
#Generacion del vecindario (movimientos requeridos para mover la solucion a traves del vecindario)

def movimientos(frag_sol):
    lista_seccion = []
    lista_aula = []
    lista_seccion_2 = []
    lista_aula_2 = []
    lista_aux = frag_sol.copy()

    salon_na = salna[['seccion','AULA']].values.tolist()
    lista_aux = lista_aux.drop_duplicates(subset='SECCION', keep="first")
    frag_lista = lista_aux[['SECCION' ,'AULA']].values.tolist()

    #Eliminar salones repetidos
    for seccion, aula in frag_lista:
        for na, aula_2 in salon_na:
            if(aula == aula_2):
                salon_na.remove([na, aula_2])

    #Juntar listas
    lista_aulas = frag_lista + salon_na

    #Generar todas las combinaciones posibles de la solucion para obtener el vecindario
    for item in itertools.combinations(lista_aulas, 2):
        lista_seccion.append(item[0][0])
        lista_aula.append(item[0][1])
        lista_seccion_2.append(item[1][0])
        lista_aula_2.append(item[1][1])

    combinations_df = pd.DataFrame(
    {
        'Seccion_1' : lista_seccion,
        'Aula_1' : lista_aula,
        'Seccion_2' : lista_seccion_2,
        'Aula_2' : lista_aula_2
    })
    combinations_df = combinations_df[(combinations_df['Seccion_1'] != 'no')]
    combinations_list = combinations_df.values.tolist()
    return combinations_list
```

Figure 22. Function to generate the movements (Generate neighborhood).
Source: own elaboration.



```
#funcion para evaluar si la solucion cumple las restricciones duras

def restricciones_duras(solucion, bloques_ocupados):
    r = True
    r1 = True
    r2 = False
    r3 = True

    #Restriccion 2: una aula solo puede tener asignada una sección en el mismo bloque horario y en el mismo día
    aula_dias = solucion[['AULA', 'DIA']]
    duplicate = aula_dias[aula_dias.duplicated()]
    #print(duplicate)
    if(duplicate.empty):
        r2= True

    #Restriccion 1: la capacidad del aula asignada no puede ser menor a la cantidad de estudiantes inscritos en la sección
    for row in solucion.itertuples():
        if(solucion.loc[row.Index, 'SALON_CAP'] < solucion.loc[row.Index, 'INSC']):
            r1 = False

    #Restriccion 9 y 10: si una materia tiene mas de un bloque horario, debe ocupar el mismo salon
    aula_bloques = solucion[['AULA', 'DIA', 'INICIO_SLG', 'FIN']]
    dataframe_aux = pd.merge(aula_bloques, bloques_ocupados, on = 'AULA', how = 'inner')
    for row2 in dataframe_aux.itertuples():
        if(dataframe_aux.loc[row2.Index, 'DIA_x'] == dataframe_aux.loc[row2.Index, 'DIA_y']):
            if(dataframe_aux.loc[row2.Index, 'INICIO_SLG_y'] <= dataframe_aux.loc[row2.Index, 'INICIO_SLG_x'] <=
               dataframe_aux.loc[row2.Index,'FIN_x'] <= dataframe_aux.loc[row2.Index, 'FIN_y']):
                r3 = False

    if((r1 != True) | (r2 != True) | (r3 != True)):
        r = False

    return r
```

Figure 23. Function to validate strong constraints.
Source: own elaboration.