ISSN-e: 1856-9811

ALGORITMO DE BÚSQUEDA TABÚ: CASO ASIGNACIÓN DE AULAS DE LA UNIVERSIDAD METROPOLITANA

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)

Resumen

El algoritmo de Búsqueda Tabú permite realizar una metodología de optimización cuya característica diferencial es el uso de memoria adaptativa y de estrategias especiales de resolución de problemas, en especial en la distribución de recursos. Dentro de la Universidad Metropolitana se han presentado diversos problemas con la asignación de aulas, debido a la demanda estudiantil, la cual ha tenido un comportamiento errático y difícil de predecir, por lo cual requerían el uso de herramientas o programas de optimización. Para esto se realizó un modelo matemático, donde se plantearon las restricciones fuertes en base a las características de la Universidad Metropolitana y se determinó la función objetivo, para posteriormente emplearla en el algoritmo de búsqueda tabú para evaluar las soluciones generadas. Después de desarrollar el modelo, se procedió a desarrollar el algoritmo de búsqueda tabú, primero se procesó la data de la oferta académica y la de los salones suministrada por la universidad y se desarrolló la solución inicial del problema, la cual se emplea como punto de partida de la búsqueda. Posteriormente, se implementó el modelo en el algoritmo, para que la búsqueda tabú busque reducir la cantidad de secciones donde el salón asignado no tiene los recursos requeridos y optimice los espacios asignados en base a la cantidad de estudiantes inscritos por sección. Posteriormente, se desarrolló un sistema para poder visualizar con mayor facilidad la solución

obtenida y, por último, se validó que la solución obtenida cumpliera con las características de la asignación de aulas de la Universidad Metropolitana en el período académico 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 restrictions 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.

RECIBIDO: 09-03-2024 / ACEPTADO: 11-05-2024 / PUBLICADO: 15-12-2024

Cómo citar: Tagliaferro et al. (2024). Algoritmo de búsqueda Tabú: Caso Asignación de Aulas de la universidad Metropolitana. *Anales*, 40, 1 - 34. https://doi.org/10.58479/acbfn.2024.68

ISSN-e: 1856 - 9811

CONTENIDO

Introducción

La asignación de espacios ha sido un problema presente en todas las instituciones educativas en el mundo. Esto es causado por que cada unidad educativa cuenta con características propias y espacios distintos, causando que cada solución generada debe tomar en consideración distintas variables y restricciones. En la actualidad, la asignación de aulas en la Universidad Metropolitana es realizada manualmente, lo que conlleva a que el proceso tenga una duración de varios días. Anteriormente, se realizaba la asignación de manera automática, pero la misma no tomaba en consideración las distintas características de la universidad, lo que generaba una serie de reclamos que hacían el proceso sumamente largo.

Este proyecto tiene como objetivo general desarrollar un sistema que emplee el algoritmo de Búsqueda Tabú para realizar la asignación de aulas de la Universidad Metropolitana.

El algoritmo de búsqueda tabú, partiendo desde una solución inicial y a través de una serie de estrategias, como la lista tabú, permite realizar una búsqueda eficaz en el espacio de soluciones, con el objetivo de encontrar una solución cercana al óptimo. La Búsqueda Tabú permite realizar malas elecciones durante la exploración, ya que considera que se puede aprender realizando una mala elección y de esta forma dirigirse a zonas más prometedoras.

Diseño del Modelo Matemático

En base a las características de la Universidad Metropolitana, fueron planteados una serie de parámetros para llevar a forma matemática las restricciones duras y establecer la función objetivo.

Parámetros

- S: conjunto de todas las secciones (index i).
- $SINS_i$: número de estudiantes inscritos en la sección i.
- $SHOR_i$: bloques horario requerido por la sección i.
- SV: subconjunto de todas las secciones virtuales.
- SP: subconjunto de todas las sesiones presenciales.

- SSP: subconjunto de todas las secciones semi presenciales.
- $SDOB$: subconjunto de las secciones que tienen doble bloque horario seguido.
- $STRI$: subconjunto de las secciones que tienen tres bloques horarios seguidos.
- $SLAB$: subconjunto de las secciones que requieren un laboratorio.
- $SESP$: subconjunto de las secciones que requieren un aula específica (aula de música, thespis y salón juicio).
- $SLABS_i$: laboratorio solicitado por la sección i.
- $SESPS_i$: aula específica solicitada por la sección i.
- SD_i: días requeridos por la sección i.
- A: conjunto de todas las aulas (index j).
- $ACAP_j$: capacidad del aula j.
- AD_{iw}: aula asignada a la sección i en el día w.
- AS_{ik}: aula asignada a la sección i en el bloque horario k.
- $ALAB$: subconjunto de todos los laboratorios.
- $AESP$: subconjunto de todas las aulas específicas.
- $ALABS_i$: laboratorio asignado a la sección i.
- $AESPS_i$:salón específico asignado a la sección i.
- B: conjunto de todos los bloques horarios disponibles en un dia (index k).
- BSi bloque horario asignado a la sección i.
- D: conjunto de todos los días de la semana que se imparten clases (index w).
- DS_i: días asignados a la sección i.
- C_1: costo asociado a la diferencia entre la cantidad de inscritos en la sección y la capacidad del aula asignada a dicha sección.
- C_2: costo asociado del tipo de aula solicitado por la sección no corresponde al tipo de aula del salón sobre el cual fue asignado.
- NSA: número de secciones donde el tipo de aula que requiere la sección no coincide con el tipo de aula del salón al que fue asignado, en el caso que el tipo de aula sea T (teórico), M (multimedia) y B (laboratorio de sistemas).

Variable

Se planteó una sola variable binaria para controlar la asignación de salones, la variable toma el valor de "1" si la sección tiene asignado un aula en el bloque horario y día solicitado. Pero la misma, toma el valor "0" en el caso contrario, es decir que la sección en el horario no tiene asignado un salón, bloque horario o día. La variable se describe a continuación:

X_{ijkw} = 1 si la sección i se asigna al aula j en el bloque horario k y en el día w; 0 si no

Restricciones

Gracias a la colaboración del personal del área de Control de Estudio fueron obtenidas las restricciones necesarias. Posteriormente, se llevaron a forma matemática empleando la variable y los parámetros determinados anteriormente. El objetivo de estas restricciones duras es impedir que las soluciones obtenidas por la búsqueda tabú contengan asignaciones que no cumplan las características de la Universidad Metropolitana.

Las restricciones duras determinadas fueron las siguientes:

1. La capacidad del aula asignada no puede ser menor a la cantidad de estudiantes inscritos en la sección (R1).
2. Un aula sólo puede tener asignada una sección en el mismo bloque horario y en el mismo día (R2).
3. El bloque horario requerido por la sección debe ser el mismo que el que se le asigna en el aula (R3).
4. Las secciones de tipo virtual no deben tener asignado ningún salón en ningún bloque horario y en ningún día (R4).
5. En las secciones de tipo presencial, el salón asignado debe ser el mismo en ambos días (R5).
6. Todas las secciones de tipo presencial deben ser asignadas a un aula, a un bloque horario y dos días de la semana (R6).
7. Todas las secciones de tipo semi presencial deben ser asignadas a un aula, a un bloque horario y a un día (R7).
8. Los días requeridos por la sección deben ser los mismos días a los que se asigne (R8).
9. Las secciones que requieren dos bloques horarios en un mismo día seguidos deben tener asignado el mismo salón (R9).

10. Las secciones que requieren tres bloques horarios en un mismo día seguidos deben tener asignado el mismo salón (R10).
11. Las secciones que requieran un laboratorio deben tener asignado el laboratorio solicitado (R11).
12. Las secciones que requieran un aula específica deben tener asignado el aula específica solicitada (R12).

La forma matemática de cada una de las restricciones duras se puede observar en la Figura 1.

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

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

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

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

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

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

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

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

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

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

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

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

Figura 1. Restricciones duras de forma matemática.
Fuente: Elaboración propia.

Función objetivo

Las restricciones débiles determinadas por el estudio fueron las siguientes:

1. Diferencia entre la cantidad de inscritos en la sección y la capacidad del aula asignada.

2. Tipo de aula asignado a la sección, no coincide con el tipo de aula solicitado. Solo aplica para aulas de tipo T (teórico), M (multimedia) y B (laboratorio de sistemas).

Se seleccionaron estas dos restricciones débiles, debido a que el personal encargado de la asignación de aulas de la universidad indicó que se debe tratar de optimizar el espacio empleado y que no hay suficientes salones multimedia para cumplir con la demanda actual. Dichas restricciones se plantearon en la función objetivo a minimizar y se le asignó un costo o penalidad a cada restricción, el valor de la penalidad se obtiene a través de experimentación.

Es importante destacar que la función objetivo se emplea en el algoritmo de búsqueda tabú para evaluar cada solución, determinar si una solución es mejor que la otra y de esta forma desplazarse a través del espacio de búsqueda. En la Figura 2 se puede observar la función objetivo.

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

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

$$C_2 \, NSA$$

Figura 2. Función objetivo.
Fuente: Elaboración propia.

Procesamiento de la data

Se debe destacar que la data suministrada pertenece a un trimestre del año 2017. La data contiene la oferta académica de dicho trimestre y la asignación de aulas que se realizó.

A continuación, se explicará detalladamente el procesamiento aplicado a la data suministrada por la Universidad Metropolitana.

1. La data obtenida fue separada dos archivos Excel©, uno que contiene la información de las secciones y otro de los salones, cada uno con su información correspondiente.

2. La variable aula/tipo fue separada en dos (aula y tipo de aula), para lograr reconocer con facilidad el tipo de aula correspondiente a cada una de ellas. Además, se eliminaron las aulas que tenían como nombre N/A.
3. Sobre la data de sección, se eliminaron las secciones que no tenían alumnos inscritos, así como las que cerraron.
4. Ambos archivos de Excel se graban como archivos tipo *Comma Separated Value* (.csv).
5. Dentro de Python el equipo creó dos *dataframes* correspondientes a cada uno de los archivos.
6. Luego de la creación del *dataframe* de secciones, el mismo, sufrió los siguientes cambios:
    a. División del tipo de aula requerido (se separaron las secciones que requieren un tipo de aula M, T y B sobre las secciones que requieren de un laboratorio específico).
    b. División del *dataframe* aulas para aulas las cuales serán consideradas en el algoritmo tabú contra aulas que no serán consideradas (tenga en cuenta que aulas específicas como los laboratorios necesitan de una sección específica).
    c. División dependiendo del par de días en los cuales las secciones requieren impartir clases (los *dataframes* fueron divididos en LM = Lunes y Miércoles, MJ = Martes y Jueves, V = Viernes, S= Sábado y D = Domingo.
    d. Posteriormente luego de realizar la asignación inicial de las aulas se procedió a dividir cada grupo de secciones en *dataframes* más pequeños basándose en el bloque horario en el cual iniciaban y terminaban.

Diseño del Algoritmo de Búsqueda Tabú

El diseño del algoritmo tabú fue desarrollado de forma modular. De acuerdo con William (2012) programar de forma modular otorga distintos beneficios, sobre los cuales se pueden mencionar:

- Simplificar el diseño.
- Disminuir la complejidad de los algoritmos.
- Disminuir el tamaño total del programa. Dado que promueve la reusabilidad del código.
- Facilita la depuración y prueba.
- Facilita el mantenimiento.

Además, el comienzo de una solución para luego empezar con la búsqueda de la combinación óptima.

Diseño de la Solución Inicial

La inicial es una debe ser simple, ya que no se requiere que sea compleja para iniciar la búsqueda tabú. En esta solución, se encargó de asignar salones a las secciones intentando cumplir con las restricciones. Principalmente se asignó de tomar en cuenta la restricción 1 (la capacidad del aula asignada no puede ser menor a la cantidad de estudiantes inscritos en la sección) y la restricción 2 (un aula sólo puede tener asignada una sección en el mismo bloque horario y en el mismo día).

Para llevar a cabo la generación de la solución inicial el equipo realizó los siguientes pasos.

1. Ordenar el *dataframe* de secciones (ya dividido en pares de días) por número de estudiantes inscritos de mayor a menor y por la hora en la que inició desde más temprano hasta más tarde.
2. Ordenar el *dataframe* de salones de mayor a menor capacidad.
3. Ejecución del algoritmo inicial para la asignación de salones. Basándose en la lógica mostrada en la Figura 1.

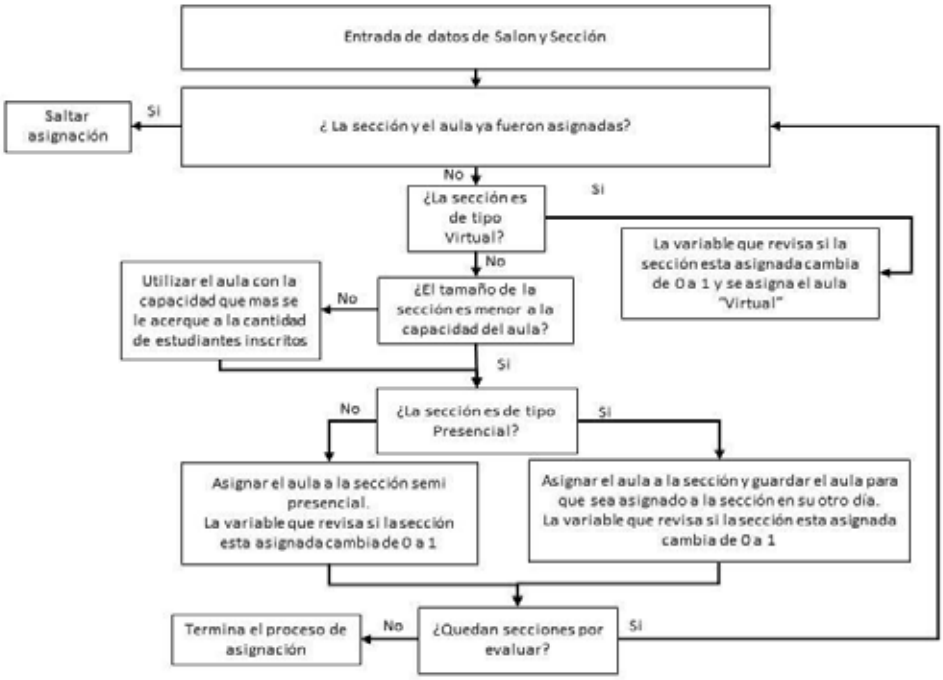


Figura 3. Algoritmo inicial para asignar salones
Fuente: Elaboración propia

Asignación de secciones no Tabú

Debido al hecho de que en la Universidad Metropolitana existen salones específicos para ciertas materias, se observó la necesidad de asignar ciertas secciones a los salones necesarios para llevar a cabo sus actividades. Porque estas secciones requieren de materiales específicos que solo se encuentran disponibles en ciertas aulas.

La razón para asignar estas aulas antes de emplear el algoritmo Tabú, es debido a que al ser aulas que solo se pueden asignar una vez por bloque horario y a ciertas materias, las mismas, no variarán en ningún momento durante la búsqueda tabú, por lo que solo incrementarían el tiempo de búsqueda.

En conclusión, las secciones que requieren el uso de algún laboratorio, aula Thepsis, aula de música y sala de juicio, serán asignadas automáticamente por el sistema a las aulas requeridas dado que no es posible que esas secciones reciban clases en otra aula.

Identificación de secciones con más de 1 bloque horario

Dado que en la Universidad Metropolitana existen secciones con más de 1 bloque horario se tuvo hizo una validación para obtener la lista de cuáles son las secciones con esa cualidad y tomarlas en cuenta.

Luego, para comprobar si esa cualidad era existente en cada una de las secciones se procedió a validar ese el módulo de validación de restricciones fuertes en el cual si el algoritmo tabú asigna un aula a alguna sección y esa aula está ocupada por una sección con más de un bloque horario esa posible solución es descartada.

Generación del vecindario

Esta función es la encargada de la generación de todos los movimientos posibles para desplazarse a través del vecindario de la solución actual. Para llevar a cabo la realización de este proceso se utiliza el método *combinations* perteneciente al módulo Itertools de Python, la cual genera todas las posibles combinaciones de las secciones y su salón asignado de la solución actual. Además, como es posible que en el bloque horario que se está evaluando no se encuentren asignados todos los salones disponibles, se agrega antes de realizar la generación del vecindario.

Los movimientos generados son de dos tipos, el primer tipo es un movimiento de intercambio, que consiste en intercambiar los salones ya asignados de dos secciones (Figura 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']
```

Figura 4. Ejemplo de movimientos de intercambio generados.
Fuente: elaboración propia.

En la Figura 2 se pueden observar varios movimientos de intercambio, por ejemplo, se apreciar el primer movimiento como la sección BPTMI01-6 tiene asignado el salón A1-216 y la sección BPTMI01-2 tiene asignado el salón A2-001, si se realiza este movimiento se intercambiarán los salones, es decir, la solución generada tendría asignado en la sección BPTMI01-6 el salón A2-001 y la sección BPTMI01-2 el salón A2-001.

El segundo tipo de movimiento se refiera eliminar/agregar que se puede reportar en la Figura 3 que se puede observar que el primer movimiento que el salón A1-104 no se encuentra asignado a ninguna sección en la solución actual, si se realiza el primer movimiento se asignaría a la sección BPTMI01-2 el salón A1-104 y el salón A2-001 no estaría asignado a ninguna sección. Esto ocurre porque el mencionado salón no cumple con los requisitos para la sección de esa materia, por ende, siempre se le dará prioridad en los requerimientos clave para dictar el curso, es decir es las restricciones fuertes para la asignación de aulas.

```
['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']
```

Figura 5. Ejemplos de movimientos de eliminar agregar generados.
Fuente: elaboración propia.

Intercambio de secciones y salones (Swap)

El módulo de intercambiar de secciones y salones es el encargado de realizar la función de cambio entre 2 secciones y 2 salones. La cual es clave en el proceso de asignación de aulas utilizando un algoritmo de búsqueda tabú. Debido a que el mismo evalúa cada posible combinación y obtener la menor función objetivo. Para llevar a cabo este proceso, el módulo recibe la solución actual que se está trabajando junto al par de secciones y de salones a las cuales se realizará el cambio. Una vez el cambio es realizado se retorna la solución actual actualizada con el cambio nuevo. Este módulo también puede asignar un salón no asignado en la solución a una sección, para ello solo intercambia el salón de la primera sección que recibe.

Función Objetivo

Para realizar el cálculo del valor de la función objetivo de la solución a evaluar, se basó en el modelo matemático previamente explicado. Para llevar a cabo este módulo, el equipo tomó la solución actual y se dividió en dos *dataframes* nuevos uno correspondiente para las secciones presenciales y el otro para las secciones semi-presenciales. Una vez obtenido ambos *dataframes* se calcula en cada uno, la suma de la diferencia de la cantidad de inscritos en la sección y la capacidad del aula asignada. Así como, la suma de los salones que no coincide el tipo de aula. Ambas sumatorias son realizadas para calcular el costo final de la función objetivo para luego ser multiplicada por el costo asignado en la función de búsqueda tabú y así obtener el costo total de alguna solución.

Validación de Restricciones Fuertes

Para llevar a cabo la validación de las restricciones fuertes, el equipo diseñó el algoritmo de búsqueda tabú considerando las mismas. Además, se diseñaron validaciones que se cumplen dentro del propio sistema, así como hay otras restricciones las cuales se validan igualmente con código para verificar que la solución cumpla con las restricciones.

A continuación, se explica detalladamente el procedimiento llevado a cabo para realizar la validación de cada una de las restricciones mencionadas en el modelo matemático.

1. La capacidad del aula asignada no puede ser menor a la cantidad de estudiantes inscritos en la sección. Se diseñó un módulo el cual revisa que el número de estudiantes inscritos sea menor a la capacidad de estudiantes en ese salón.
2. Un aula sólo puede tener asignada una sección en el mismo bloque horario y en el mismo día. Se diseñó un módulo el cual revisa que solo exista un aula asignada a una sección en un bloque horario.
3. El bloque horario requerido por la sección debe ser el mismo que el que se le asigna en el aula. Se diseñó el sistema de asignación de aulas de modo que solo existan cambios de aula y secciones al sistema no se le otorgó la capacidad de realizar cambios entre bloques horarios.
4. Las secciones de tipo virtual no deben tener asignado ningún salón en ningún bloque horario y en ningún día. Se diseñó el sistema de asignación de aulas de modo que solo exista asignación a secciones tipo “P” (Presencial) o tipo “SP” (Semi-Presencial) si el sistema consigue alguna sección con algún otro tipo la descarta automáticamente.
5. En las secciones de tipo presencial, el salón asignado debe ser el mismo en ambos días. Se diseñando el módulo Swap incluyó en el código que una vez se realice la asignación a alguna sección de algún salón, el movimiento sea replicado para el otro día.
6. Todas las secciones de tipo semi-presencial deben ser asignadas a un aula, a un bloque horario y a un día. El sistema de asignación de aulas se encarga de asignarle a cada sección un aula el día correspondiente y el bloque horario está definido desde un inicio de parte de la universidad en la data entregada.
7. Las secciones que requieren dos bloques horarios en un mismo día seguidos deben tener asignado el mismo salón. Se diseñó un módulo el cual identifica estas secciones con más de un bloque horario y al momento de evaluar cada posible solución en el módulo de validación de restricciones fuertes se revisa que cada solución cumpla con esto sino es descartada.
8. Las secciones que requieren tres bloques horarios en un mismo día seguidos deben tener asignado el mismo salón. Se diseñó un módulo el cual identifica estas secciones con más de un bloque horario y al momento de evaluar cada posible solución en el módulo de validación de restricciones fuertes se revisa que cada solución cumpla con esto sino es descartada.

9. Las secciones que requieran un laboratorio deben tener asignado un laboratorio del tipo requerido. Se encargó de asignar las secciones que requieren un laboratorio específico, para ello se emplea el módulo asignación de secciones no tabú.
10. Las secciones que requieran un aula específica (música, Thespis) deben tener asignado el aula específica solicitada. S se encargó de asignar las secciones que requieren un aula específica (salón juicio, música y Thespis), para ello se emplea el módulo asignación de secciones no tabú.

Búsqueda Tabú

A continuación, se presentará el procedimiento para llevar a cabo la búsqueda tabú. Inicialmente, se calcula el valor de la función objetivo de la solución inicial. Luego, se inicia la generación del vecindario. Para este procedimiento, se obtiene una muestra aleatoria de un 25% del tamaño total del vecindario generado por el módulo de generación de vecindario, luego de obtener este espacio de soluciones se procede a verificar si cumple con el módulo de restricciones fuertes. Si el movimiento cumple con las restricciones se guarda ese movimiento sino es descartado. Para que un vecindario sea admisible se necesita que el tamaño del vecindario que cumple con las restricciones sea por lo menos de un 50% del generado, de no ser así se busca un vecindario nuevo.

Se decidió seleccionar aleatoriamente un 25% de los movimientos del vecindario de la solución que se está evaluando, debido a que revisar el vecindario completo demoraría demasiado tiempo y no resultaba efectivo, ya que pasaba muchas iteraciones sin obtener una mejora en el valor de la función objetivo de la mejor solución encontrada.

Una vez obtenido el nuevo vecindario, se procede a seleccionar el mejor movimiento, el que tenga un menor valor objetivo. Una vez obtenido este movimiento se procede a revisar si al realizar este cambio se continuará cumpliendo con las restricciones duras. Si se procede satisfactoriamente, se evalúa si el movimiento pertenece o no a la lista tabú. Si no pertenece, se agrega a la lista y se actualiza la solución con el nuevo cambio. De pertenecer, se evalúa igualmente si ese movimiento pudiera mejorar la mejor solución actual. Si el movimiento mejora la solución, se aplica criterio de aspiración el cual se encarga de saltar la restricción tabú y realizar el movimiento. Si el movimiento no mejora la mejor solución actual es descartado.

Adicionalmente, se puede mencionar que cada cierto número de iteraciones que ocurren en la búsqueda tabú el vecindario es renovado de la misma forma que fue explicado anteriormente. Esto es debido a que, se consiguió que se mejora significativamente el valor de la función objetivo al renovar cada cierta iteración el vecindario y no cada interacción.

El número de iteraciones necesarias para renovar el vecindario varía según el tamaño del vecindario de la solución actual, debido a que se selecciona como valor el 10% de la longitud del vecindario dividido entre 10 si es mayor a 100 dicho valor, en caso contrario no se divide. Este método se obtuvo después de probar distintos valores para seleccionar cada iteración, la cual se debe renovar en el vecindario y se optó que variará según el vecindario debido a que

en ciertos periodos de los días domingo o viernes el vecindario puede ser muy reducido. Esto se debe, porque la universidad tiene muy pocos cursos en estos días mencionados.

También se puede mencionar que, existe una condición de parada del algoritmo el cual ocurre si el valor de la función objetivo no mejora luego de 200 iteraciones. Finalmente, se obtiene una solución la cual cumple con todas las restricciones y además la misma disminuye significativamente el valor de la función objetivo. Para una explicación detallada véase la Figura 4 la cual representa gráficamente un diagrama de la lógica.

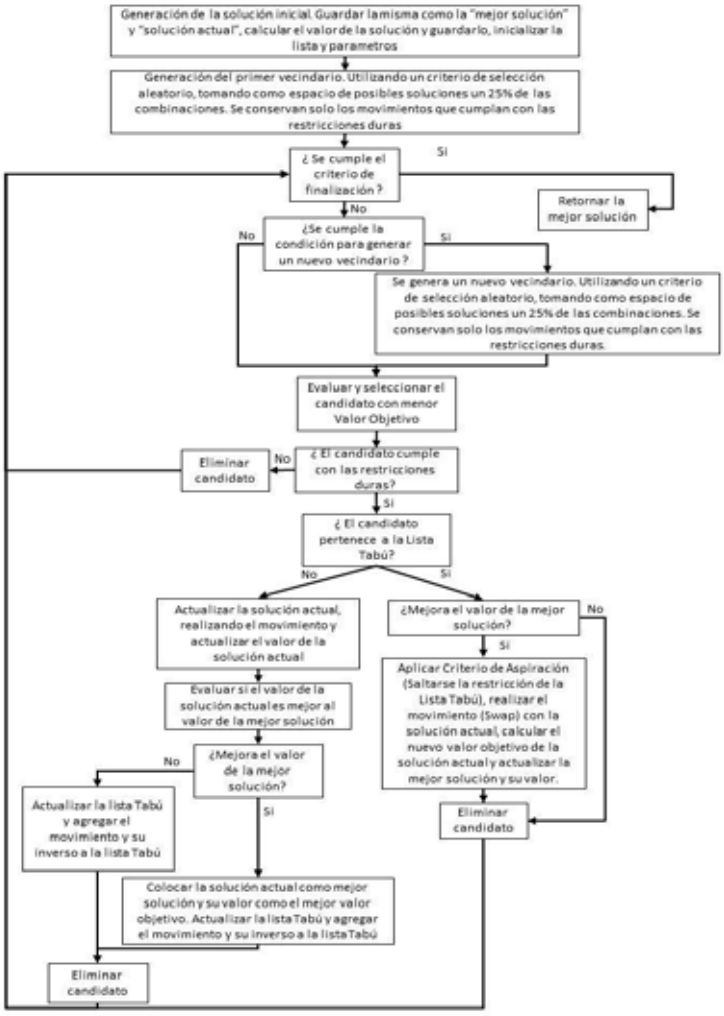


Figura 6. Diagrama lógica algoritmo de búsqueda tabú.

Fuente: Elaboración propia.

Para ver el código del algoritmo de búsqueda tabú se puede observar en el Apéndice A.

Diseño del Sistema de Asignación de Aulas

Para poder visualizar con mayor facilidad la asignación realizada con el algoritmo de búsqueda tabú, se desarrolló un sistema empleando la librería *tkinder* de Python para realizar la interfaz gráfica.

Se definieron los siguientes requisitos de usuario para poder conocer qué se necesita que haga el sistema:

- El sistema debe realizar de forma automática la asignación de aulas.
- Se debe poder visualizar la asignación de aulas.
- El sistema debe permitir extraer la asignación de aulas del sistema para poder guardarla.
- El sistema debe tener una interfaz intuitiva.

Para que el sistema funcione es requerida la siguiente data:

- Oferta académica en un archivo csv.
- Información de las aulas de la universidad, se debe emplear tres archivos csv separados de la siguiente manera:
    - Primer archivo con la información de las aulas de tipo T (teórico), M (multimedia) y B (laboratorio de sistema)
    - Segundo archivo con la información de los laboratorios y aulas con usos específicos.
    - Tercer archivo solo con los nombres de las aulas de tipo T (teórico), M (multimedia) y B (laboratorio de sistema).

Es importante destacar que, al reemplazar los archivos ya colocados en el sistema con los nuevos, estos se deben adaptar a la estructura de los anteriores.

El sistema manipula la data a través del algoritmo de búsqueda tabú para generar la asignación de aulas y la salida del sistema sería:

- Asignación de aulas en un archivo Excel.
- Asignación de aulas del bloque horario y días seleccionados en un archivo Excel.

Análisis e interpretación de los resultados

Con la ejecución de la Búsqueda Tabú fue realizado tomando en consideración los requerimientos de la universidad, para cumplir exitosamente se realizó una serie de pruebas

al algoritmo para evaluar su efectividad sobre los siguientes parámetros: el valor de la función objetivo y el tiempo de ejecución.

Inicialmente, se presentarán los resultados correspondientes al cálculo del periodo tabú. Así como, el cálculo del mejor costo asociado del tipo de aula solicitado por la sección no corresponde al tipo de aula del salón sobre el cual fue asignado (C2), los cuales fueron obtenidos, mediante experimentación. Además, con respecto al costo asociado a la diferencia entre la cantidad de inscritos en la sección y la capacidad del aula asignada a dicha sección (C1) se estableció que será fijado en 1, esto se debe al valor que a la diferencia entre la capacidad del aula con el número de estudiantes inscritos en la sección.

Dichos resultados se presentan tabulados, tomando en cuenta cada uno de los parámetros fijados anteriormente y así comprender el comportamiento del algoritmo de Búsqueda Tabú.

Es importante destacar que la asignación suministrada pertenece a un trimestre del año 2017 y se empleó la oferta académica de ese trimestre en el algoritmo de búsqueda tabú y la solución inicial.

A continuación, se presenta una tabla 1 la cual contiene una serie de abreviaciones utilizadas en este capítulo para comprender mejor los resultados obtenidos.

Tabla 1. Abreviaciones utilizadas.

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

Fuente: Elaboración propia.

En la Tabla 2 se presentarán los resultados obtenidos correspondientes al cálculo del periodo Tabú. Bajo el periodo Tabú obtenido se realizaron los cálculos restantes.

Tabla 2. Cálculo del mejor periodo tabú.

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	v(N)	567	271,9696901

LM2 Posee un Valor funcion objetivo inicial de: 951

Fuente: Elaboración propia.

IV.2 Resultados

IV.2.3 De acuerdo con el tiempo necesario para obtener cada solución

Para calcular el tiempo necesario para obtener cada solución, el equipo utilizó una librería llamada *timeit* de Python la cual mide el tiempo desde que se inicia la ejecución de cierta función hasta su fin. En la Tabla 3 se muestran los resultados obtenidos al calcular las soluciones del Algoritmo de Búsqueda Tabú.

Tabla 3. Tiempo necesario para obtener la solución del Algoritmo de Búsqueda Tabú.

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

Fuente: elaboración propia.

De acuerdo con los resultados reflejados en la tabla 13, se puede observar que a partir del cálculo de cada uno de los tiempos de los *dataframes* se logró generar una solución factible en 95,04 minutos.

Por otra parte, según lo comunicado por E. Oberto, " La Universidad Metropolitana actualmente realiza la asignación de aulas de manera manual respetando los requerimientos de cada asignatura y de los profesores, logrando minimizar la cantidad de reclamos generados por el sistema anterior. También mencionaron que, el tiempo dedicado para realizar este proceso es de dos a tres días generalmente los fines de semana". (comunicación personal 2 de junio de 2021).

IV.3 Sistema de Asignación de Aulas

En la Figura 7, se muestra la pantalla inicial del sistema desarrollado por el equipo. En donde, se selecciona el *dataframe* a trabajar, así como el bloque horario.

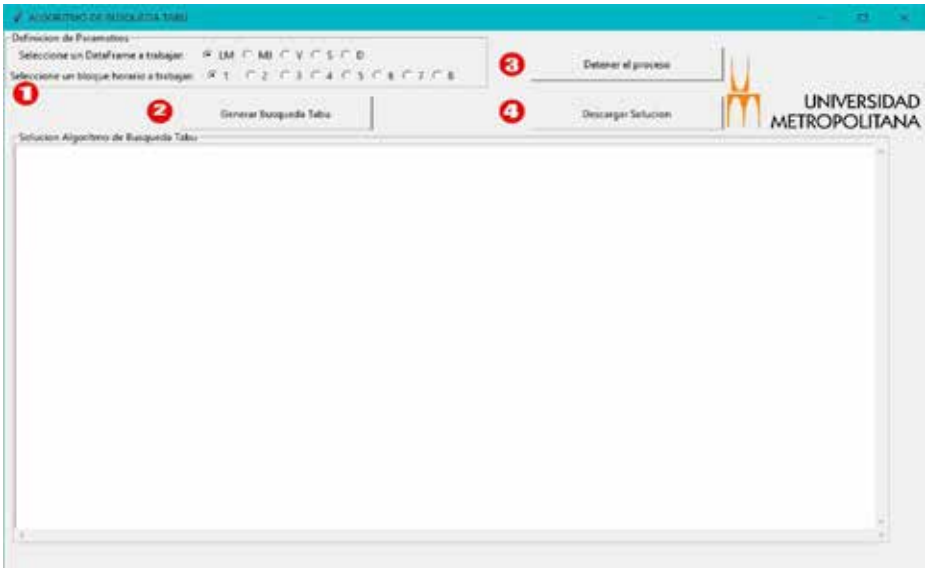


Figura 7. Pantalla inicial del sistema.
Fuente: elaboración propia.

La enumeración siguiente, son los pasos para seguir del inicio del sistema y su forma correcta de ejecutar la búsqueda tabú.

Se selecciona el dataframe y el bloque horario a trabajar.

1. El botón de “Generar Búsqueda Tabú” es el encargado de interactuar con el Algoritmo de Búsqueda Tabú y realizar la búsqueda de acuerdo con los parámetros seleccionados en el punto 1. Tenga en cuenta que, al darle clic el proceso inicia, pero el mismo puede tomar varios minutos en generar una solución.
2. El botón de “Detener el proceso” es el encargado de detener y cerrar el programa si por alguna necesidad es necesario detener el sistema.
3. El botón “Descargar Solución” es el encargado de descargar la solución generada por el Algoritmo de Búsqueda Tabú en un archivo Excel. El mismo posee validaciones para que no se pueda descargar el archivo si no se ha generado una solución.

En la Figura 8, se presenta la pantalla del sistema con una solución generada utilizando el *dataframe* MJ y el periodo 2. Utilizando las barras de desplazamiento se puede apreciar con totalidad la solución generada. En este momento, el botón de “Descargar Solución” le permitirá descargar la solución generada en un archivo Excel el cual se almacenará en la carpeta donde tenga el sistema guardado.

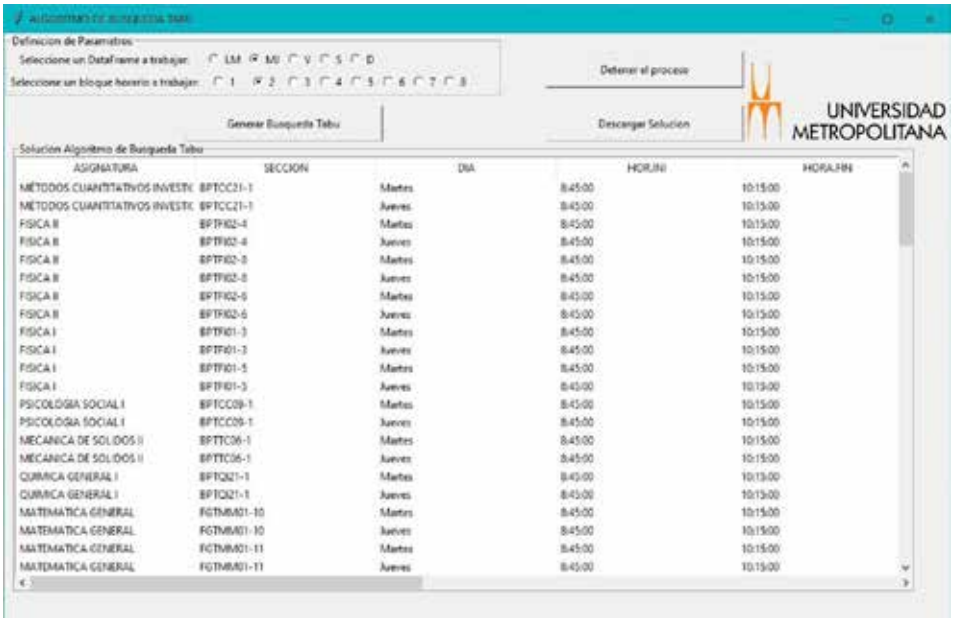


Figura 8 Muestra de solución del sistema.
Fuente: elaboración propia.

En la Figura 9, se presenta la pantalla del sistema una vez se le da clic al botón de "Descargar solución". El mismo le generará al usuario una ventana la cual confirma si el mismo desea descargar la solución.

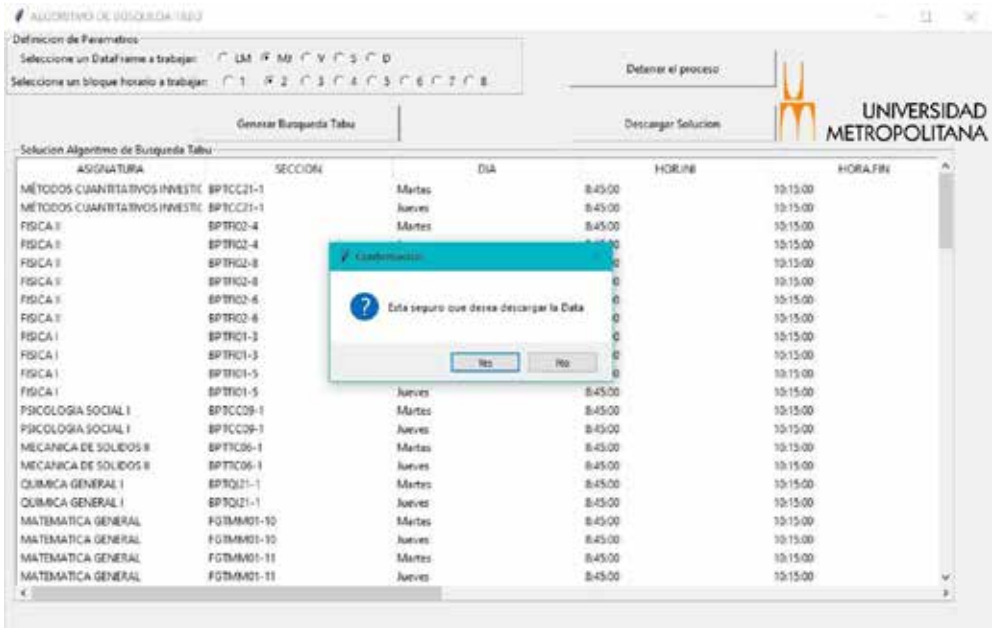


Figura 9. Descarga de la solución.
Fuente: elaboración propia.

IV.4 Validación del Sistema

Una vez obtenidos los resultados de la asignación de aulas a través del Algoritmo de Búsqueda Tabú, el equipo procedió a verificar si la solución cumplía con las condiciones de la asignación de aulas de la Universidad Metropolitana. Para verificar esto, se debe comprobar si se cumplen todas las restricciones duras planteadas en el modelo matemático. La mayoría de las restricciones duras se cumplen automáticamente debido a la forma que se diseñó el sistema, solo las siguientes restricciones se deben evaluar durante el algoritmo de búsqueda (enumeradas según el modelo matemático):

- Restricción 1: La capacidad del aula asignada no puede ser menor a la cantidad de estudiantes inscritos en la sección.

- Restricción 2: Un aula sólo puede tener asignada una sección en el mismo bloque horario y en el mismo día.
- Restricción 9: Las secciones que requieren dos bloques horarios en un mismo día seguidos, deben tener asignado el mismo salón.
- Restricción 10: Las secciones que requieren tres bloques horarios en un mismo día seguidos, deben tener asignado el mismo salón.

Para evaluar si estas restricciones se cumplen en la solución generada por el algoritmo tabú para un bloque horario, se empleó la función restricciones duras, véase Apéndice A para ver el código de la función, la cual arroja verdadero si la solución generada cumple las cuatro restricciones nombradas anteriormente.

En la Tabla 4, se puede observar que la comprobación de la función con las soluciones generadas por el algoritmo de búsqueda tabú para los bloques LM1, LM2, MJ2, MJ3 y D3. Como se observa, se cumplen las cuatro restricciones, por lo que las soluciones generadas cumplen las condiciones de la asignación de la Universidad Metropolitana.

Tabla 4. Validación de las restricciones duras por 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

Fuente: elaboración propia.

IV.5 Soluciones generadas

En las siguientes figuras se puede ver un fragmento de la solución generada para los bloques horarios LM1, LM2, MJ2 y MJ3 después de haber sido extraída del sistema, es decir, en archivo Excel (véase Figura 10, Figura 11, Figura 12 y Figura 13).

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

Figura 10. Fragmento de la solución generada para LM1.
Fuente: elaboración propia.

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

Figura 11. Fragmento de la solución generada para LM2.
Fuente: elaboración propia.

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	BPTF02-4	Martes	8:45:00	10:15:00	A2-309	44	45	T	T	P
FISICA II	BPTF02-4	Jueves	8:45:00	10:15:00	A2-309	44	45	T	T	P
FISICA II	BPTF02-8	Martes	8:45:00	10:15:00	A2-207	44	45	T	T	P
FISICA II	BPTF02-8	Jueves	8:45:00	10:15:00	A2-207	44	45	T	T	P
FISICA II	BPTF02-6	Martes	8:45:00	10:15:00	A2-204	43	45	T	M	P
FISICA II	BPTF02-6	Jueves	8:45:00	10:15:00	A2-204	43	45	T	M	P
FISICA I	BPTF01-3	Martes	8:45:00	10:15:00	A2-008	40	40	M	M	P
FISICA I	BPTF01-3	Jueves	8:45:00	10:15:00	A2-008	40	40	M	M	P
FISICA I	BPTF01-5	Martes	8:45:00	10:15:00	A1-306	40	40	M	M	P
FISICA I	BPTF01-5	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	BPTQ21-1	Martes	8:45:00	10:15:00	EMG_13	39	40	M	M	P
QUIMICA GENERAL I	BPTQ21-1	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	BPTQ21-8	Martes	8:45:00	10:15:00	A2-109	37	40	M	M	P
QUIMICA GENERAL I	BPTQ21-8	Jueves	8:45:00	10:15:00	A2-109	37	40	M	M	P

Figura 12. Fragmento de la solución generada para MJ2.
Fuente: elaboración propia.

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	BPTQ23-1	Martes	10:30:00	12:00:00	A2-309	36	45	T	T	P
QUIMICA GENERAL II	BPTQ23-1	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	BPTBC21-4	Martes	10:30:00	12:00:00	A1-302	35	40	M	M	P
CONTABILIDAD I	BPTBC21-4	Jueves	10:30:00	12:00:00	A1-302	35	40	M	M	P
COMPETENCIAS EN ACCION	FGEH05-2	Martes	10:30:00	12:00:00	A1-303	35	40	M	M	P
COMPETENCIAS EN ACCION	FGEH05-2	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	FPTMI21-2	Martes	10:30:00	12:00:00	A2-314	35	40	M	M	P
MODELOS ESTOCASTICOS	FPTMI21-2	Jueves	10:30:00	12:00:00	A2-314	35	40	M	M	P
QUIMICA GENERAL I	BPTQ21-2	Martes	10:30:00	12:00:00	A1-210	35	35	M	M	P
QUIMICA GENERAL I	BPTQ21-2	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

Figura 13. Fragmento de la solución generada para MJ3.
Fuente: elaboración propia.

Las anteriores figuras son soluciones generadas por la búsqueda tabú que cumplen con las necesidades y restricciones de la universidad, además se obtuvieron varias soluciones viables en menos de 200 microsegundos.

Con estas pruebas se puede demostrar que el algoritmo de Búsqueda Tabú es efectivo y cumple con las necesidades de la universidad.

Conclusiones

- En primer lugar, se identificaron las restricciones fuertes y débiles del problema, con la ayuda del personal de la universidad y la asignación suministrada. Esto permitió asegurar la veracidad de los resultados obtenidos.

- En segundo lugar, se desarrolló un modelo matemático para resolver el problema, teniendo en cuenta las restricciones identificadas.

- En tercer lugar, se diseñó un algoritmo de búsqueda tabú basado en el modelo matemático con. Este algoritmo se centró en optimizar los espacios asignados y disminuir la cantidad de salones en los que no coincide el tipo de aula.

- Finalmente, se desarrolló un sistema en el lenguaje de programación Phyton para evaluar y visualizar los resultados obtenidos. Este sistema se validó verificando que cumple las restricciones duras de la asignación de aulas de la universidad Metropolitana.

Recomendaciones

- Se sugiere descargar la data de las asignaciones, ya que, si se realiza por bloque horario, se perderá la asignación previa. Además, la data se descarga en un archivo excel que el usuario puede modificar si así lo desea.

- Al momento de recibir la data por parte de la universidad, es recomendable realizar una validación a la data de los salones. Debido a que pueden existir disparidades en cuanto a la data colocada en el sistema y la actual.

- Además, es recomendable que al momento de insertar la data en el sistema se compruebe que los archivos se encuentren estructurados de la misma manera a los preestablecidos en el sistema. También, si es posible, se sugiere reemplazar la información en los archivos ya existentes.

- Finalmente, es recomendable que al momento de volver a la presencialidad se visite cada uno de los departamentos de la universidad para integrar restricciones al sistema que sean útiles para toda la comunidad universitaria.

Referencias

Bullet Solutions. (s.f.). [en línea]. España. Recuperado de: https://www.bulletsolutions.com/es/

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

Glover, F. (1989). “Tabu Search – Part I” 190-206.

Glover, F. y Batista, B. (2006). Introducción a la búsqueda Tabú. España: Autor.

Glover, F. y Kochenberger, G. (2003). *Handbook of metaheuristic*. Nueva York, Boston, Dordrecht, Londres y Moscú: Kluwer academic publishers

Landeau, R. (2007). *Elaboración de trabajos de investigación.* Venezuela: Alfa

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

Pacheco, J. (2021). *Variable Cualitativa (definición, tipos, ejemplos y características)*, [en línea]. Recuperado de: https://www.webyempresas.com/variable-cualitativa/

Pacheco, J. (2021). *Variables cuantitativas (definición, características y clasificación)*, [en línea]. Recuperado de: https://www.webyempresas.com/variables-cuantitativas/

Paneque, R. (1998). Metodología de la Investigación: *Elementos básicos para la investigación clínica.* Recuperado de: http://www.sld.cu/galerias/pdf/sitios/bioestadistica/metodologia_de_la_investigacion_1998.pdf

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

Riojas, A. (2005) Conceptos, algoritmo y aplicación al problema de las N–reinas. Búsqueda de tabú. Lima, Perú.

Roldán, P. (2019). *Modelo matemático*, [en línea]. Recuperado de: https://economipedia.com/definiciones/modelo-matematico.html

Suárez, J. G., Manchego, F. A., Nole, A. A., Nicho, G. B., & Anticona, M. T. (2009). *Generación Inteligente de Horarios empleando heurísticas GRASP con Búsqueda Tabú para la Pontificia Universidad Católica del Perú*. https://drive.google.com/file/d/1_oMzD_Cnur6E7ViCk-4keOvboVcvUXdL/view?usp=sharing

William, R. (2012). *Ventajas de la Programación Modular*. Recuperado de: http://progmodular.blogspot.com/2012/08/ventajas.html

Yepes, V. (2014). *Optimización y programación matemática,* [en línea]. Universidad Politécnica de Valencia. Recuperado de: https://victoryepes.blogs.upv.es/2014/06/05/optimizacion-programacion-matematica/

APÉNDICES

Apéndice A. Código de las funciones principales del algoritmo tabú

```
#función para calcular el valor objetivo de la solución
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 días a la semana pero cuenta como una
    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_funcion_objetivo = acumulado_salones + secciones_erradas

    return valor_funcion_objetivo
```

Figura 14. Función para calcular el valor objetivo.
Fuente: elaboración propia.

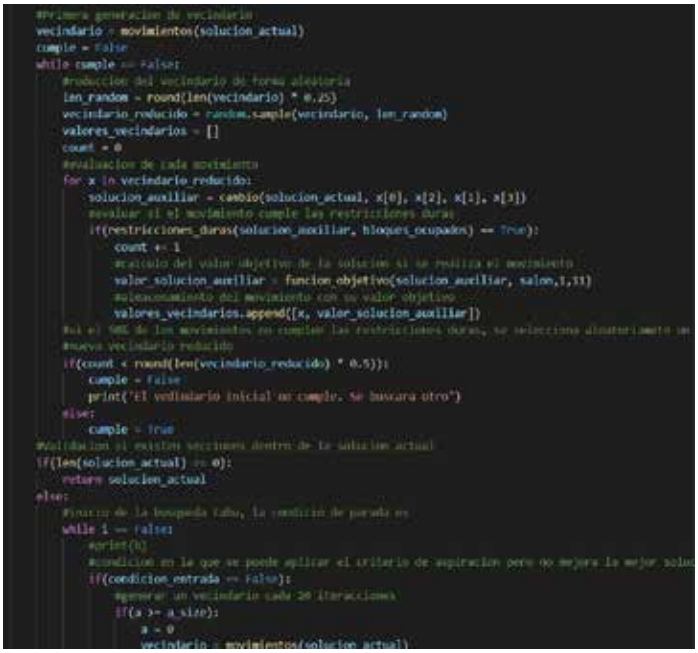


Figura 15. Función para realizar el Algoritmo de Búsqueda Tabú (Parte 1).
Fuente: elaboración propia.

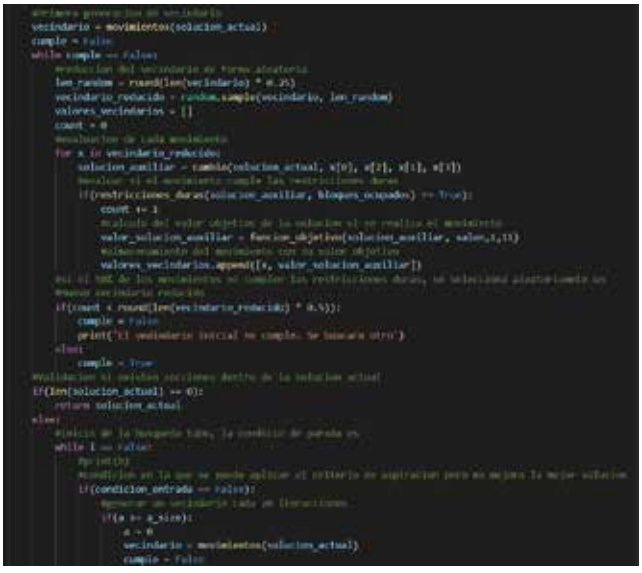


Figura 16. Función para realizar el Algoritmo de Búsqueda Tabú (Parte 2).
Fuente: elaboración propia.

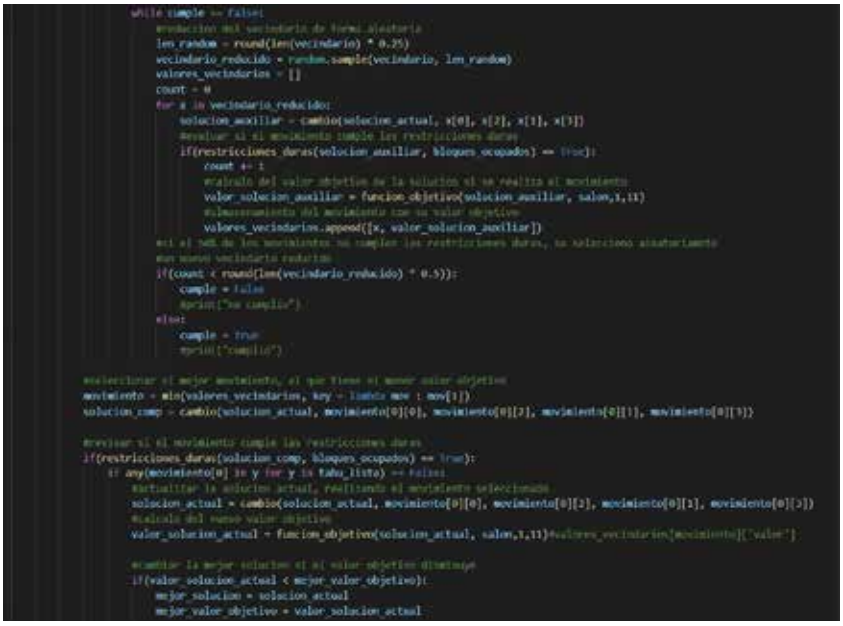


Figura 17. Función para realizar el Algoritmo de Búsqueda Tabú (Parte 3).
Fuente: elaboración propia.

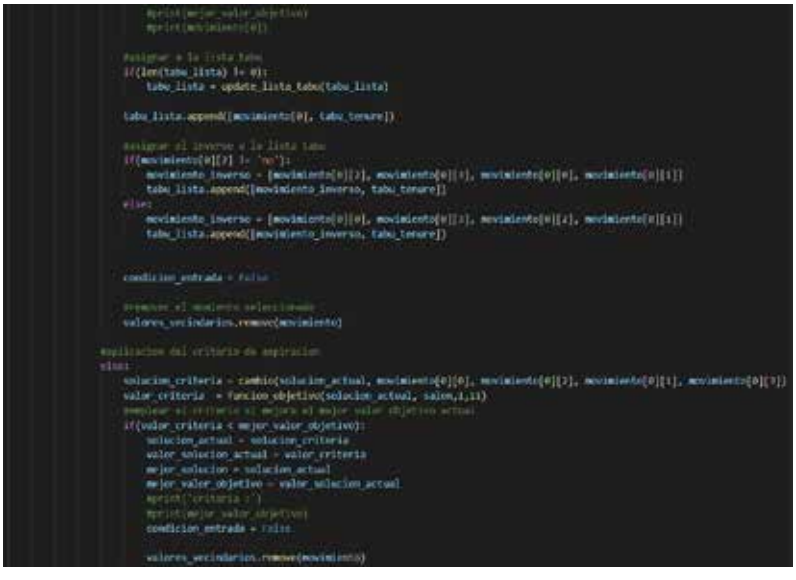
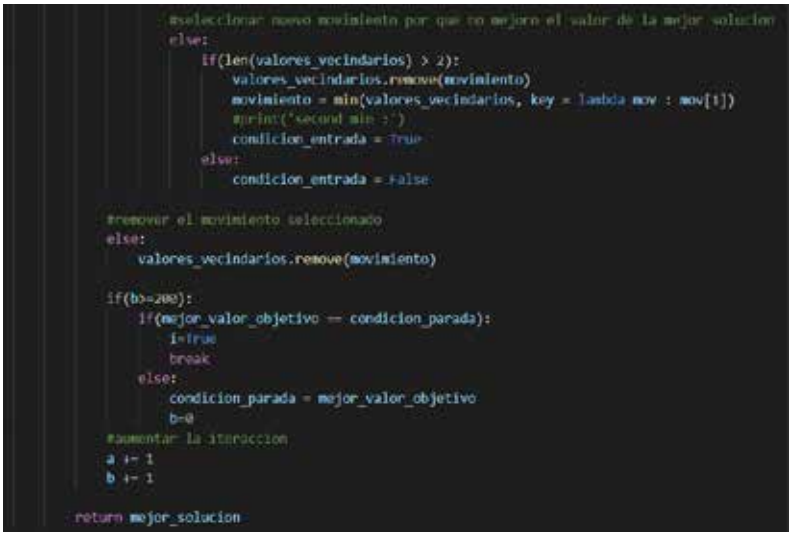
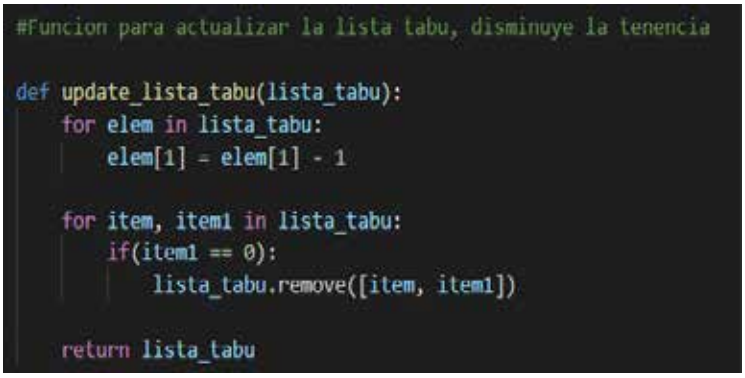


Figura 18. Función para realizar el Algoritmo de Búsqueda Tabú (Parte 4).
Fuente: elaboración propia.



```
                #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
```

Figura 19. Función para realizar el Algoritmo de Búsqueda Tabú (Parte 5).
Fuente: elaboración propia.

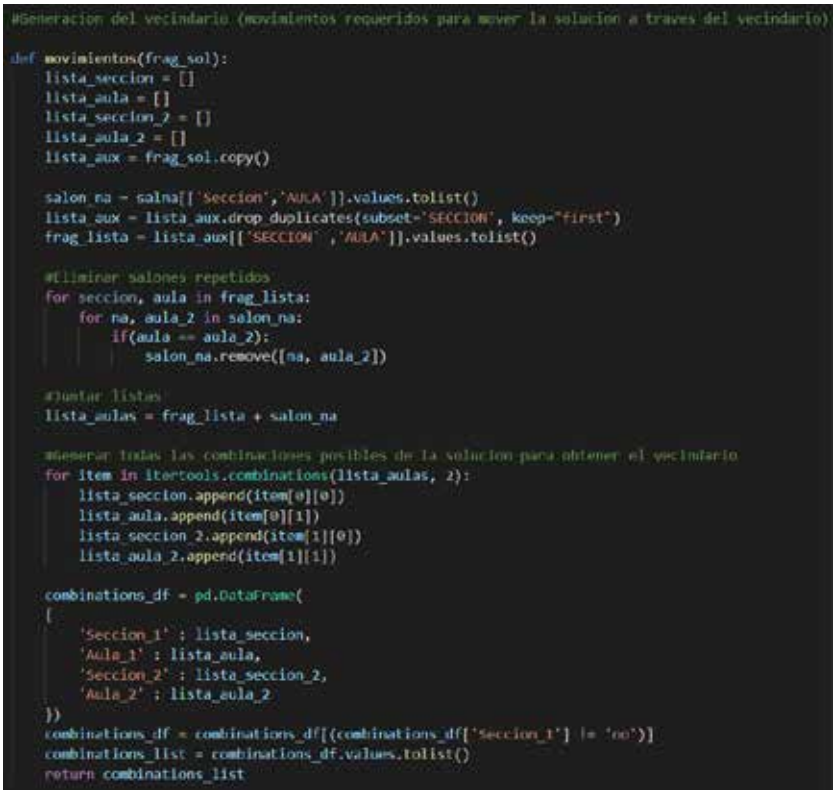


```
#Funcion para actualizar la lista tabu, disminuye la tenencia

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
```

Figura 20. Función para actualizar la lista tabú.
Fuente: elaboración propia.

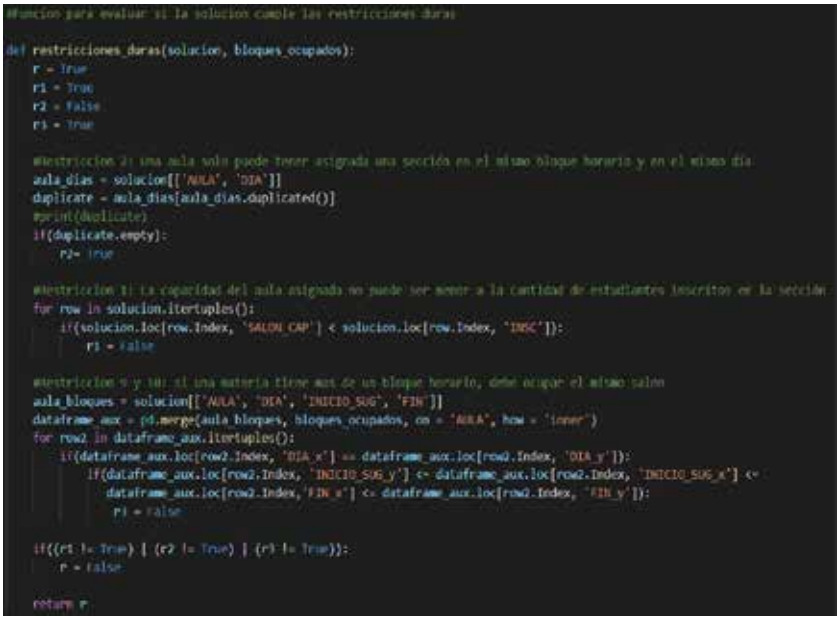
```python
#Funcion para intercambiar los salones entre dos secciones o asignar un salon (no asignado) a una seccion

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
```

Figura 21. Función para realizar el movimiento (Swap).
Fuente: elaboración propia.



```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
```

Figura 22. Función para generar los movimientos (Generar vecindario).
Fuente: elaboración propia.



```
#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 dia
    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 3 y 10: si una materia tiene mas de un bloque horario, debe ocupar el mismo salon
    aula_bloques = solucion[['AULA', 'DIA', 'INICIO_SUG', '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_SUG_y'] <= dataframe_aux.loc[row2.Index, 'INICIO_SUG_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
```

Figura 23. Función para validar las restricciones fuertes.
Fuente: elaboración propia.