UNIVERSIDAD POLITECNICA SALESIANA

UNIDAD DE POSGRADOS

MAESTRÍA EN MÉTODOS MATEMÁTICOS Y SIMULACIÓN NUMÉRICA EN INGENIERÍA

Proyecto de investigación y desarrollo previo a la obtención del Grado de Magister en Métodos Matemáticos y Simulación Numérica en Ingeniería

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO MATEMÁTICO BASADO EN OPTIMIZACIÓN PARA LA GENERACIÓN DE HORARIOS DE CLASES EN LA UNIVERSIDAD POLITÉCNICA SALESIANA

Autor:

Pedro Javier Cornejo Reyes

Director:

PhD. Vladimir Robles Bykbaev, Mst.

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO MATEMÁTICO BASADO EN OPTIMIZACIÓN PARA LA GENERACIÓN DE HORARIOS DE CLASES EN LA UNIVERSIDAD POLITÉCNICA SALESIANA

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO MATEMÁTICO BASADO EN OPTIMIZACIÓN PARA LA GENERACIÓN DE HORARIOS DE CLASES EN LA UNIVERSIDAD POLITÉCNICA SALESIANA

AUTOR:

PEDRO JAVIER CORNEJO REYES

Ingeniero de Sistemas

Maestría en Métodos Matemáticos y Simulación Numérica en Ingeniería.

DIRIGIDO POR:

PhD. VLADIMIR ESPARTACO ROBLES BYKBAEV.

Ingeniero de Sistemas

Master en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Docente de la Universidad Politécnica Salesiana sede Cuenca

CUENCA – ECUADOR

Datos de catalogación bibliográfica

CORNEJO REYES PEDRO JAVIER
"DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO MATEMÁTICO BASADO EN OPTIMIZACIÓN PARA LA GENERACIÓN DE HORARIOS DE CLASES EN LA UNIVERSIDAD POLITÉCNICA SALESIANA"

Universidad Politécnica Salesiana, Cuenca – Ecuador, 2018

MAESTRIA EN MODELOS MATEMÁTICOS Y SIMULACIÓN NUMÉRICA PARA INGENIERÍA

Formato 170 x 240 mm Páginas: 56

Breve reseña de los autores e información de contacto

PEDRO JAVIER CORNEJO REYES
Ingeniero de Sistemas
Colaborador en el grupo de Investigación en Inteligencia Artificial
y tecnologías de asistencia
pjaviercornejo@gmail.com

Dirigido por:
VLADIMIR ESPARTACO ROBLES BYKBAEV
Ingeniero en Sistemas
Master en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital
Docente de la Universidad Politécnica Salesiana sede Cuenca
vrobles@ups.edu.ec

IMPRESO EN ECUADOR – PRINTED IN ECUADOR

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Dedicatoria

A mi amada madre por todo su apoyo, a mi amado padre que, aunque ya no me acompaña siempre está presente en mis pensamientos, y a mi hermano por todo su apoyo y gentileza durante mi desarrollo académico y humano.

Pedro Javier Cornejo Reyes.

PREFACIO

La presente investigación refleja el resultado de la aplicación de las técnicas de programación lineal entera y algoritmos genéticos para el desarrollo e implementación de horarios de clases para la carrera de Ingeniería de Sistemas de la Universidad Politécnica Salesiana.

En el presente trabajo se ha realizado el modelamiento e implementación de un modelo matemático de horarios de clases mediante el levantamiento y aplicación de requerimientos por parte de la institución antes mencionada. Esto último permitió formular restricciones, pues fueron indispensables para delimitación del modelo.

Las conclusiones y resultados de este trabajo pueden ser utilizadas para procesos futuros en los cuales se requiera mejorar e implementar el modelo para asignación de horarios en las distintas carreras de la Universidad Politécnica Salesiana.

PRÓLOGO

La implementación del algoritmo genético y programación lineal entera requirió de procesos previos tales como:

- Reuniones con personal académico y administrativo de la Universidad Politécnica Salesiana para identificar y definir las necesidades referentes a los horarios de clases.
- Diagramación del problema para la correcta abstracción del problema.
- Modelamiento matemático del problema.

El proceso final de este trabajo se traduce en la implementación de horarios de clases mediante los dos algoritmos antes mencionados.

Agradecimientos

Agradezco al Dr. Vladimir Robles Bykbaev quien además de brindarme todo su apoyo científico-técnico, ha sabido ser un gran amigo. A mi esposa Yaroslava por toda su paciencia, apoyo y entrega incondicional para poder culminar mis estudios. Finalmente agradecer al Dr. Fernando Pesántez Avilés por su inmenso respaldo y gentileza durante mis estudios.

Pedro Javier Cornejo Reyes.

CAPÍTULO 1

1. INTRODUCCIÓN

La Universidad Politécnica Salesiana es una institución que se caracteriza por ser innovadora y vanguardista en el campo de investigación dentro del país. Una de sus maneras de contribuir a generar mayor conocimiento y aportes en áreas de investigación para la ayuda mediante tecnología a la sociedad son los distintos grupos de investigación con los que cuenta, uno de ellos es el Grupo de Investigación de Inteligencia Artificial y Tecnologías de Asistencia (GIIATA), que se enfoca al desarrollo de procesos basados en Inteligencia Artificial, minería de datos, etc. Se busca diseñar e implementar nuevas soluciones basadas en técnicas de optimización matemática que se adapten a los requerimientos de sectores más vulnerables de la sociedad.

Por otra parte, es importante destacar que la Universidad Politécnica Salesiana actualmente realiza la asignación de los horarios de clases de manera manual, por lo que los directores de Carrera han sabido manifestar que esta tarea llevada a cabo durante cada ciclo lectivo conlleva problemas. La asignación de horarios en algunas ocasiones se inicia desde cero o basando su organización en ciclos los horarios existentes de ciclos anteriores. Esto último genera inconvenientes al momento de planificar la asignación de horarios a los docentes, puesto que, al ser un proceso complejo, cualquier mala planificación u omisión de datos en la planificación puede generar retrasos o el volver a generar el horario sin que este último se adapte a las necesidades curriculares y académicas.

De igual forma, se debe mencionar que la problemática mencionada antes, se traduce en una realidad que también tiene lugar en otras universidades, y en general en empresas e instituciones de nuestro contexto socio-cultural.

Generar una distribución de carga horaria eficiente supone una tarea compleja para las instituciones, pues optimizar el tiempo laboral de un empleado en la institución y en la medida de lo posible, hacer que el perfil del recurso de la empresa pueda ser aprovechado de mejor manera, es finalmente el objetivo deseado. Las instituciones que tienen este inconveniente que parece de forma recurrente, son las Universidades, pues deben generar cada ciclo un horario de clases diferente que considere lo siguiente:

- Debe adaptarse a la disponibilidad de aulas y los requerimientos de *hardware* y *software* en los distintos laboratorios, considerar las materias que se ofertan, así como los docentes y las competencias de cada uno de ellos.

- De igual forma, esta distribución debe considerar horarios y niveles en los que el perfil del docente encaja, acoplarse a las normativas de educación del Gobierno y a las normativas internas de la Universidad, docentes a tiempo completo y docentes a tiempo parcial, entre otras.
- Por otra parte, si una de las restricciones no se ven resueltas, pues se debe volver a replantear la idea del horario generado y realizar el análisis y planificación nuevamente.

En la actualidad no se cuenta con una alternativa para mejorar este proceso, pues las instituciones no cuentan con el software y los sistemas existentes son altamente costosos o no se adecuan a los requerimientos de gestión de talento humano institucionales.

Actualmente las Universidades, en especial la Universidad Politécnica Salesiana no cuenta con un proceso automatizado para la gestión de carga horaria del personal académico. La propuesta del desarrollo de un algoritmo matemático basado en métodos de optimización permitirá ahorrar tiempo respecto al proceso manual existente, pues se pretende que el docente a cargo de generar el proceso únicamente identifique las restricciones y ya no requiera realizar procesos de validación manual. Todo esto significa mayor tiempo para que los directores de carrera puedan dedicar a otras áreas de planificación, desencadenando en una mejora general de las tareas que deben ser tratadas y tal vez no están recibiendo el tiempo adecuado. Con esta propuesta será más fácil probar diversas alternativas de distribución de carga horaria, pudiendo escoger la que se adapte mejor a las necesidades.

Una vez generado el modelo, este puede ser transferido a otras áreas de la gestión de recursos o personal.

1.1 Programación Lineal y Algoritmos Genéticos: Fundamentos teóricos básicos

1.1.1 Programación Lineal

El problema de la programación lineal (PL) es quizá uno de los problemas de optimización más importantes y más estudiados. Numerosos problemas del mundo real se pueden formular empleando programación lineal. PL se entiende como el proceso que consiste en minimizar o maximizar una función lineal sujeta a un número de restricciones lineales de igualdad y / o desigualdad. Es decir, en un problema de PL se podrían presentar todos los tipos de restricciones [14].

Existen restricciones explícitas que deben constar en la declaración o enunciado del problema. De igual manera, también existen restricciones llamadas implícitas. Estas son restricciones que no se dan explícitamente en la declaración del problema, pero que están presentes de todos modos [4]. Estas restricciones están típicamente asociadas con restricciones naturales en las variables de decisión [18], [28].

En un problema de PL tanto la función objetivo como las restricciones son lineales. Ploskas, Samaras [41] definen:

- Dos problemas PL son equivalentes, si existe una correspondencia uno a uno entre sus puntos o soluciones factibles y sus valores de función objetivo.

- O, de manera equivalente, si un problema PL es óptimo, no factible o ilimitado, entonces su problema PL equivalente también es óptimo, no factible o ilimitado, respectivamente.

1.1.1.1 FORMA GENERAL DE UN PROBLEMA DE PL.

Un problema PL en su forma general se muestra en la ecuación (1).

$$\begin{aligned} \min z = \; & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{s.t.} \quad & A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n \oplus b_1 \\ & A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n \oplus b_2 \\ & \vdots \quad \vdots \quad \vdots \quad \vdots \cdots \vdots \quad \vdots \quad \vdots \quad \vdots \\ & A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{mn}x_n \oplus b_m \end{aligned} \tag{1}$$

Los Coeficientes $A_{ij}, c_j y b_i$ donde $i = 1,2,3,\ldots,m$ y $j = 1,2,3,\ldots,n$, son números reales conocidos y son los datos de entrada del problema. El símbolo $\oplus$ define el tipo de restricciones que pueden ser las siguientes:

- = : restricción de igualdad
- $\geq$: restricción de mayor o igual
- $\leq$: restricción de menor o igual

La palabra min corresponde a un problema de minimización. Las variables X_1, X_2, X_n se llaman variables de decisión y son las variables que desconocemos del problema. Al resolver un problema de PL, queremos encontrar los valores de las variables de decisión mientras minimizamos el valor de la función objetivo sujeta a las restricciones dadas.

1.1.1.2 FORMA MATRICIAL DE UN PROBLEMA DE PL.

Escribiendo la ecuación (1) en notación matricial, se define de la siguiente manera:

$$\begin{matrix} minz = c^T x \\ s.a. Ax \oplus b \end{matrix} \quad (2)$$

donde $c, x \in R^n, b \in R^m, yA \in R^{m_x n}$. x es el vector de las variables de decisión (dimensión $n \times 1$), Aes la matriz de coeficientes de las restricciones (dimensión $m \times n$), c es el vector de coeficientes de la función objetivo (dimensión $n \times 1$), b es el vector del lado derecho de las restricciones (dimensión $m \times 1$), y $\oplus$ es el vector del tipo de restricciones (dimensión $m \times 1$) donde:

- 0: define una restricción de igualdad (=)
- -1: define una restricción de menor o igual ($\leq$)
- 1: define una restricción de mayor o igual ($\geq$)

La mayoría de los problemas PL son problemas aplicados y las variables de decisión generalmente son valores no negativos. Por lo tanto, la no negatividad o restricciones naturales se deben agregar a la ecuación (2).

$$\begin{matrix} minz = c^T x \\ s.a. Ax \oplus b \\ x \geq 0 \end{matrix} \quad (3)$$

El problema de PL general Ecuación (1) puede contener restricciones de igualdad (=) y desigualdad ($\leq, \geq$), rangos en las restricciones y los siguientes tipos de restricciones variables o límites:

- Límite inferior: $x_j \geq l_j$
- Límite superior: $x_j \leq u_j$
- Límite superior e inferior: $l_j \leq x_j \leq u_j$

- Valor fijo $x_j = k, k \in R$
- Variables libres: $-\infty \leq x_j \leq \infty$
- Menos infinito: $x \geq -\infty$
- Más infinito: $x \leq \infty$

1.1.1.3 FORMA CANÓNICA DE UN PROBLEMA PL.

Un problema de PL en su forma canónica tiene todas sus restricciones en la forma 'menor que o igual a' (≤) y todas las variables son no negativas, como se muestra en la ecuación (4):

$$\begin{aligned} \min z = \; & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{s.t.} \quad & A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n \leq b_1 \\ & A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n \leq b_2 \\ & \vdots \quad \vdots \quad \vdots \quad \vdots \cdots \vdots \quad \vdots \quad \vdots \quad \vdots \\ & A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{mn}x_n \leq b_m \\ x_j \geq 0, \quad & (j = 1, 2, \cdots, n) \end{aligned} \tag{4}$$

1.1.1.4 FORMA EN SUMATORIAS DE UN PROBLEMA DE PL.

$$minz = \sum_{j=1}^{n} C_j X_j \tag{5}$$

s. a

$$\sum_{j=1}^{n} a_{ij} X_j \; b_i \forall i = 1,2,3, \ldots, m$$

$$X_j \geq 0 \forall j = 1,2,3, \ldots, n$$

1.1.1.5 PROCEDIMIENTO PARA LA CONSTRUCCIÓN DE MODELOS DE PROGRAMACIÓN LINEAL

Los siguientes pasos nos indican como construir un modelo [6] [28]:

1. **Formulación del modelo:** estructuración de información para determinar la relación de información entre parámetros y variables.

2. **Análisis de la información:** se realiza un análisis minucioso de los parámetros y cómo estos influyen en el modelo. De igual forma, se deberán visualizar los objetivos que pretendemos alcanzar y qué limitantes se tienen para cumplir dichos objetivos.

3. **Definición de variables:** Se debe establecer qué se quiere llegar a conocer para lograr la solución. Este paso es muy importante, pues el definir de forma incorrecta las variables hará que el modelo contenga resultados erróneos.

4. **Establecer la función objetivo:** se define en base a las necesidades planteadas si se desea maximizar la función (por ejemplo, maximizar el rendimiento de la empresa, maximizar el uso efectivo de recursos, etc.). También es posible minimizar la función (si nuestro caso requiere minimizar costos de producción, por ejemplo).

5. **Determinar las restricciones:** se deben evaluar qué restricciones nos impiden lograr un valor máximo o mínimo de la función objetivo, pues hay que tener claro que los recursos con los que se cuenta en cualquier problema no son infinitos para su utilización, lo que nos lleva a tener que restringir nuestro problema.

6. **Solución del modelo matemático:** la función objetivo más las restricciones, conforman el modelo matemático. Este modelo deberá ser resuelto mediante técnicas de optimización que nos permitan hallar los valores de las variables planteadas.

7. **Prueba del modelo y la solución:** se debe evaluar que la respuesta obtenida tenga correspondencia con el estudio planteado. El análisis debe iniciar visualizando que las restricciones se hayan cumplido.

8. **Implementación del modelo:** implementar el modelo.

9. **Controlar y realimentar:** debido a que el modelo permite obtener soluciones óptimas, se deberán realizar correcciones sobre la marcha. Esto último con el fin de mejorar el modelo. Cabe indicar que las modificaciones deberán ser observadas con el fin de controlar que el modelo no presente errores.

1.1.1.6 PROGRAMACIÓN LINEAL ENTERA (PLE)

Un problema de programación lineal entera es un problema de programación lineal con la restricción adicional de que algunas de las variables deben tomar valores enteros. Cuando todas las variables deben tomar valores enteros decimos que se trata de un problema de programación lineal entera puro, en caso contrario decimos que es mixto. Diremos que una variable es binaria si sólo puede tomar los valores 0 y 1 [46].

Una gran variedad de problemas combinatorios del mundo real puede ser planteados como problemas de programación lineal entera [46].

1.1.2 ALGORITMOS GENÉTICOS

1.1.2.1 BREVE DESCRIPCIÓN

Los Algoritmos Genéticos (AG) son métodos de optimización estocástica y están basados en los conceptos de selección natural y procesos evolutivos.

Inicialmente fueron propuestos e investigados por John Holland en la Universidad de Michigan en 1975. En estos sistemas, uno o más cromosomas se combinan para formar la prescripción genética total para la construcción y operación de algunos organismos [16].

Los cromosomas están compuestos de genes que pueden tomar una serie de valores llamados valores de los alelos genéticos. La posición de un gen se identifica por separado de la función del gen. Por lo tanto, podemos hablar de un gen particular, por ejemplo, un gen de color de ojo de un animal, posición 10 y su valor de alelo como ojos azules.

Antes de iniciar la descripción del funcionamiento del AG, se debe indicar cuáles son sus componentes y de qué forma se relacionan sus conceptos con los principios de la evolución [16]. Algunos conceptos análogos entre los AG y la evolución natural son presentados en la Tabla 1.

Tabla 1: *Conceptos básicos en Algoritmos Genéticos.*

Conceptos en la evolución natural	Conceptos en Algoritmos Genéticos
Cromosoma	Cadena (String)
Gen	Características en la cadena
Locus	Posición en la cadena
Genotipo	Estructura de la cadena
Fenotipo	Conjunto de características

1.1.2.2 DEFINICIONES DE UN ALGORITMO GENÉTICO

1.1.2.2.1 FUNCIÓN DE ADAPTACIÓN

Se podría decir que "los AG son mecanismos de búsqueda optimizada que operan sobre un conjunto de códigos posiblemente muy grande, pero finito. Del dominio de búsqueda se toman iteradamente conjuntos de muestras y cada elemento de una muestra es calificado dependiendo de qué tan bien cumpla con los requisitos de aquello que es buscado" [2].

Los elementos, o también denominados individuos, que cumplan mejor con los requisitos planteados en el problema serán seleccionados. Éstos se multiplican en las siguientes iteraciones y son sometidos a ciertos operadores que emulan a los que funcionan en la naturaleza. A cada individuo se le asocia un grado de adaptación que representa una calificación que indica que tan bien cumple la muestra con las condiciones planteadas. Esta calificación es un número mayor o igual a cero que, en conjunto con la selección, se convierte en una medida de la posibilidad de reproducción para cada individuo. Por lo tanto, mientras más grande sea este valor, mayor probabilidad tendrá el individuo de formar parte de la muestra.

- **Gen:** los individuos del problema (posibles soluciones) a quienes llamaremos genes, se pueden representar como un conjunto de características de este último.
- **Alelo:** valor específico que tiene un gen.
- **Cromosoma:** se representa como el conjunto de individuos o genes.

- **Locus:** es la posición específica que un gen ocupa dentro de un cromosoma.

- **Genotipo:** el conjunto de cromosomas que conforman nuestro estudio.

- **Fenotipo:** es la representación final que tendrá nuestro caso de estudio, es decir de qué manera queda conformado nuestro genotipo y de qué manera lo percibe el usuario.

1.1.2.3 OPERADORES GENÉTICOS

SELECCIÓN.

El operador de selección es el encargado de determinar los pobladores que van a tener la posibilidad de ser reproducidos. Tal como lo hace la naturaleza, los individuos que tengan mejores características serán los más aptos para ser elegidos. Lo que cabe destacar es que de alguna manera el algoritmo elegido para este proceso, debe considerar tanto elementos aptos como no aptos para disponer de una población menos homogénea y que por lo tanto pueda variar y evolucionar de mejor manera.

Los algoritmos que se disponen para realizar este proceso son los siguientes [40]:

Selección por ruleta: Asigna a cada cromosoma un porcentaje de la ruleta, el resultado de la suma del porcentaje de cada cromosoma debe dar la unidad. Los individuos que tengan mejor puntuación van a tener un mayor porcentaje de la ruleta, justamente para permitir que tengan mayores posibilidades de reproducirse.

Presenta el inconveniente de que el peor individuo puede ser seleccionado más de una vez.

Selección por torneo: Este algoritmo consiste en realizar comparaciones directas entre los individuos ubicados al azar, y el que tenga mejores características será el seleccionado o ganador. Existen dos versiones:

- Determinística: Se selecciona un elemento denominado p, este es un valor que se genera aleatoriamente. Sin embargo, se suele elegir el valor de $(p = 2)$ que representa el número de cromosomas que ingresarán al torneo para ser comparados.

- Probabilística: el mecanismo es el mismo, lo que se modifica es que no siempre va a ganar el mejor, sino se genera un valor del parámetro k entre [0..1] siendo entonces para $k > 0.5$el candidato más óptimo, mientras que para $k \leq 1$ la selección para el candidato menos apto.

Cabe destacar que para una población p es recomendable que esta no sea muy grande, pues las posibilidades de que un elemento menos apto sea seleccionado son escasas, este fenómeno suele llamarse elitismo global, mientras que con un p más pequeño las posibilidades de que un elemento menos apto se reproduzca son más altas.

CRUCE

Este será nuestro paso en la búsqueda de implementar correctamente el AG, pues en este paso buscaremos cruzar y generar las descendencias. Para ello se tratará de implementar la mejor estrategia con el fin de realizar el cruce.

Su importancia en las transiciones es significativa, pues como Gestal [17] indica: "las tasas de cruce con las que se suele trabajar rondan el 90%".

La idea principal de este método radica en que, si se toman dos individuos adaptados ya al medio, sus genes se compartan y sus mejores características puedan ser propagadas, de tal manera que los individuos hijos tengan mejores bondades que sus padres por separado.

Existen algunos algoritmos que se utilizan para realizar este proceso, a continuación, se describirán brevemente algunos:

- **Cruce de un punto**: es la técnica más utilizada, pues es la más sencilla de todas. Se trata de generar un punto de corte aleatoriamente, a partir de ese valor se realiza un corte en el cromosoma padre 1 y cromosoma padre 2, dividiéndose en dos partes: cabeza y cola. Los hijos entonces pasarán a ser la cabeza del padre 1 con la cola del padre 2 y así mismo la cabeza del padre 2 y la cola del padre 1. Ver figura 1.

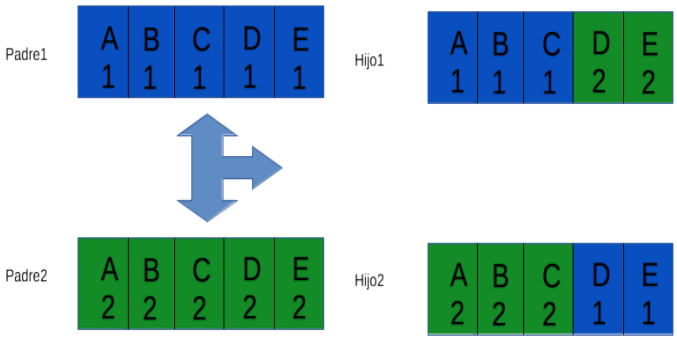

Figura 1: *Cruce de un punto.*

- **Cruce de dos puntos:** La idea es similar a cruce de un punto, pero se realizará el corte en dos segmentos del cromosoma. Generalmente para evitar que se realicen cortes errados, se procede a seleccionar la parte central de uno de los dos padres y los extremos del otro. Ver figura 2.

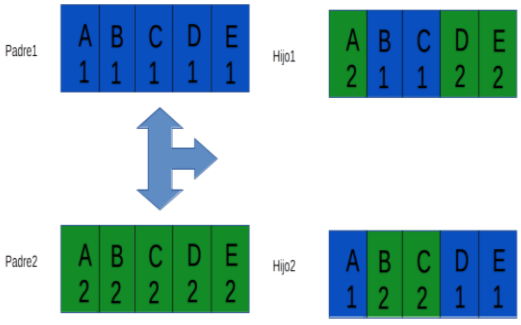

Figura 2: *Cruce de dos puntos.*

- **Cruce uniforme**: Se debe generar una máscara binaria en la que si se encuentra como valor un 1 se seleccionará el gen situado en esa posición de uno de los dos descendientes, mientras que si se obtiene un 0 se seleccionara el gen del otro padre. Para generar el segundo elemento se intercambiará el procedimiento del hijo 1. Ver figura 3.

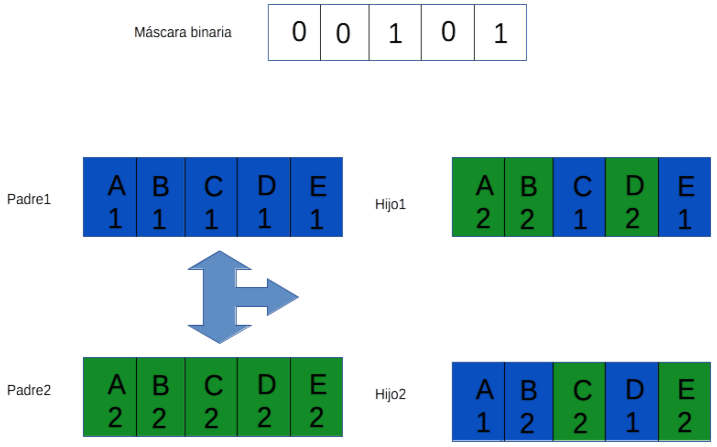

Figura 3: *Cruce uniforme, se puede visualizar que a través de la máscara los elementos si el valor es 1 se seleccionan los elementos del padre 1 y si es 0 se seleccionaran los elementos del padre 2.*

MUTACIÓN

Este proceso va de la mano con el método de cruce, la idea es que uno de los descendientes generados o ambos sean sometidos a este proceso, esto dependerá de un parámetro (Probabilidad de mutación) $Pm \leq 1$. Esto se debe a que los individuos suelen tener un menor ajuste después de mutados.

La mutación de un individuo provoca que alguno de sus genes, generalmente solo uno, varíe su valor de forma aleatoria. Ver figura 4.

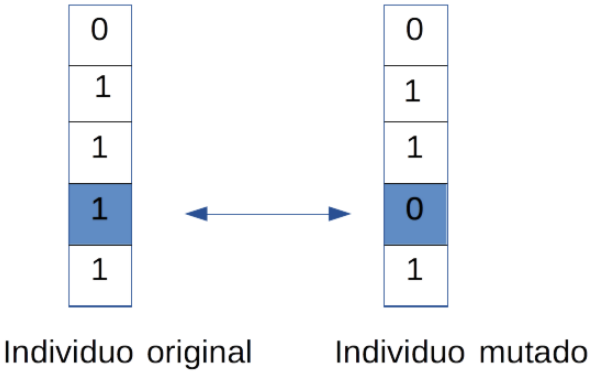

Figura 4: *Mutación aleatoria de un gen del cromosoma.*

CAPÍTULO 2

2. ESTADO DEL ARTE

Este estudio se centra en la utilización de Programación Lineal Entera (PLE) y Algoritmos Genéticos (AG) para realizar una asignación de materias a un distributivo de horarios para las universidades con un menor tiempo computacional y mayor precisión. Ambas constituyen técnicas de optimización.

La PL utilizada para generar los horarios, se basó fundamentalmente en modelos matemáticos con restricciones mediante sistemas de ecuaciones lineales para lo cual utiliza la maximización o minimización de funciones objetivo. Mientras que los AGE se centraron esencialmente en los principios genéticos de la evolución y selección natural, los cuales se programaron mediante restricciones en base a penalizaciones [24]. Es decir, se aplicaron algoritmos genéticos (AG).

Una vez aplicadas las restricciones de la PLE y la AG, se procedió a realizar la generación de horarios mediante la asignación de horas, docente, ciclo, asignatura, día y período, o, en otras palabras, este proceso de generación de horarios se conoce como timetabling (tablas de horarios).

La aplicación de AG se utiliza para poder optimizar el timetabling para la asignación de recursos humanos en función de horarios laborales, escolares, asignación de conferencias [15][8]. Tal como indica [15] en su propuesta permite minimizar el número de casos en los que se asigna más de una sesión con el mismo tema al período de tiempo dentro de un día. Así mismo, permite minimizar el número de casos en los que el número de presentaciones asignadas a cada sesión paralela es diferente entre sí.

La PLE funciona como un localizador de vecinos [29], dado que mide distancias entre los elementos que se requieren ubicar (docentes, materias, horas, días de la semana, entre otros) para minimizar el número de errores en la asignación horaria. Por lo tanto, podemos resolver instancias muy grandes y recuperar soluciones dentro de límites de tiempo aceptables.

Para generar horarios para clases universitarias más cómodos se puede incluir en la función objetivo, la preferencia de horarios tanto para docentes como para estudiantes [9], [38].

2.1 Programación Lineal Entera (PLE)

Se entiende a la Programación Lineal (LP) como un subconjunto de la programación matemática, donde el objetivo se centra en la asignación eficiente de recursos limitados respecto a la minimización de costos [8].

La formulación de un problema se puede resolver mediante la PLE gracias a que "es una herramienta de modelado muy poderosa para la identificación formal y precisa de decisiones y restricciones relevantes" [26].

Las técnicas de programación entera lineal pueden resolver de forma más satisfactoria el problema del timetabling ([9]; [19]; [31]). En otras palabras, podemos decir, que la PLE se refiere a una "técnica de optimización matemática que maximiza o minimiza una función sujeta a restricciones lineales de igualdad y desigualdad, donde todas las variables están restringidas a valores enteros" [19]. Además, la "PLE permite incluir un mayor número de variables para reducir el tamaño del problema" [39], con lo cual el problema tiene una mayor delimitación.

La PLE permite la creación de "validadores de soluciones lineales, por tanto, determinar si la solución obtenida mediante dicha técnica es factible y, simultáneamente, producen los valores para las diversas penalizaciones" [29]. Al tiempo que permite crear dichos validadores, la PLE permite obtener un "tiempo de resolución notablemente corto para llegar a la solución óptima, ya que puede minimizar los coeficientes lineales" ([39]; [43]). Sin embargo hemos de indicar "que la PLE es determinística con respecto a los AG, ya que las soluciones de optimización obtenidos por PLE son exactos, pero no evita la aparición de espacios vacíos entre los valores asignados" [42]

2.2 Algoritmos Genéticos (AG)

Por su parte, la aplicación de AG permite "reducir la búsqueda espacial y el tiempo necesario para encontrar buenas soluciones" [13], ya que "imita el comportamiento de la evolución biológica para resolver problemas particulares mediante programación" [36]. Los AG se consideran una técnica tan popular "utilizada para resolver el problema de programación. Se inventó por primera vez en la Universidad de Michigan por el Prof. John Holland en 1975" [2].

El AG es un método que se puede generar a “partir del cálculo evolutivo en la técnica de optimización. Se clasifica como algoritmo heurístico (no solo la solución exacta pero cercana también es aceptable) y como método estocástico (utilizando un proceso aleatorio y un criterio de selección)” [35], [6].

Es importante indicar que los AG son “considerados como una buena herramienta general de optimización merced a su flexibilidad y la simplicidad conceptual. Dichos AG son escogidos como metodología básica para la resolución de problemas de timetabling” [32], [2].

Los AG se han utilizado ampliamente “en muchas áreas de difícil manejo para el ser humano dadas las complicaciones en los procesos de programación manual. Con los AG las soluciones son más creíbles, eficientes, más rápidas y de mayor complejidad. En muchos casos, los AG producen soluciones que son mucho mejores que el programador que escribió el algoritmo” [36]. De hecho, se ha evidenciado que mediante “la aplicación de AG se alcanza una solución óptima de horarios” [35].

Los AG tienen varias ventajas en comparación con otras técnicas de optimización, por ejemplo, “pueden realizar una búsqueda multidireccional utilizando un conjunto de soluciones candidatas” [49]. En nuestro caso particular la “asignación de horarios se considera un problema de optimización porque busca soluciones óptimas en el espacio de búsqueda de posibles cronogramas” [36], para lo cual se consideró la aplicación del AG para optimizar el problema de los cronogramas de horarios de docentes de la UPS.

Principalmente los AG se fundamentan en la metaheurística, dado que mejoran la calidad de la solución aplicando operadores genéticos [22], [13], esto último permite que el timetabling sea un proceso automático. A su vez, el AG “realiza una búsqueda del dominio para que las soluciones candidatas muten, de tal manera que el cronograma inicial se respalde en operadores de mutación específicos. Esto último permite que se cumplan la restricciones duras y suaves” [36], [5].

Dentro de los AG “la mutación se aplica para mejorar el rendimiento de la operación cruzada al permitir explorar un amplio espacio de búsqueda y mantener la diversidad genética de una generación a otra” [1]. También es importante destacar que, para mantener la diversidad genética, el AGE se basa en la probabilidad de cruce, por lo tanto, “el rendimiento en el algoritmo genético es muy sensible al parámetro de probabilidad de cruce” [23].

En los algoritmos genéticos “cada caso de programación viene en forma de un cromosoma que incluye tareas asignadas y procesadores. En este método, cada gen del cromosoma cubre un par de tareas y su procesador determinado. Esta representación

permite una administración sencilla y al pasar la fase de priorización de tareas de la asignación del procesador" [3].

CAPÍTULO 3

3. EXPERIMENTACIÓN Y RESULTADOS

El presente estudio se centró en la creación de distributivo de horarios de clases para nivel de estudios universitarios mediante la utilización de dos técnicas de optimización. Por un lado, se aplicó un modelo matemático basado en Programación Lineal Entera (PLE), y posteriormente se aplicó la técnica computacional denominada algoritmos genéticos.

Si bien la PLE y la técnica de AG son similares, inicialmente se trabajó con PLE, ya que es conocido que los modelos matemáticos basados en programación lineal tienen un bajo costo computacional. Posteriormente se continuó el desarrollo de los distributivos mediante la aplicación de AG, dado que es una técnica actual y altamente eficaz para la resolución de problemas de optimización complejos. La aplicación de PLE y AG permitieron llevar a cabo el modelo previamente planteado e implementarlo.

Este estudio también pretendió evaluar cuál de las dos técnicas tuvo mayor precisión para alcanzar los resultados esperados en la creación del distributivo de horarios de clases.

3.1 Programación Lineal Entera

3.1.1 Modelo Propuesto

Se realizó la implementación de un algoritmo matemático mediante Álgebra Lineal y Programación Lineal Entera para la generación de un modelo para asignación automática con restricciones. Debido a que esta técnica requiere definir restricciones para delimitar el problema y llegar a los resultados esperados se realizaron varias reuniones con personal de la Universidad Politécnica Salesiana. Esto permitió llevar a cabo un análisis de los criterios que son tomados en cuenta al momento de diagramar un horario de clases y tener claro los criterios que intervienen en el proceso de generación de un distributivo de materias.

El modelo planteado se evaluó y diseñó de acuerdo a la información recolectada, llegando así a determinar los comportamientos necesarios para traducirlos en restricciones y la función objetivo que resuelva el problema.

Las restricciones identificadas son las siguientes:

- Un período no puede repetirse en un mismo día para una materia, de esta manera no se asignarán varias materias de un mismo ciclo a un mismo periodo de clases. Por ejemplo, no puede haber dos materias del ciclo 1 que se asignen de 7:00 a 9:00 am.

- Un docente no puede dar la misma materia en un mismo día.

- Garantizar que todas las materias ofertadas sean asignadas al horario de clases.

- Garantizar que todos los docentes disponibles sean asignados al horario de clases.

- Evitar cruces de varias materias que de un docente en un mismo periodo de clases.

- Garantizar que una materia no exceda el número de créditos en una semana de clases.

- Adecuar las horas de clases de los docentes al calendario entregado por cada uno de los docentes a la directora de carrera.

- Garantizar que los mejores perfiles de los docentes sean asignados a la materia que más se adapte al perfil.

Alcances de este modelo.

- El modelo *timetabling* propuesto está desarrollado para los tres primeros ciclos, puesto que en estos niveles es donde más alumnos se tiene y por lo tanto se ofertan más grupos de materias, por lo tanto, se genera la mayor cantidad de conflictos y se hace complejo evitar cruces entre los docentes y las materias.

- El modelo propuesto se adapta a la creación de ***n*** grupos para los distintos semestres.

- Este modelo no abarca la asignación de aulas para dictar clases.

- Los períodos de clases han sido divididos en segmentos de dos horas.

Nuestro modelo de programación lineal entera está compuesto por los conjuntos que se muestran en la Tabla 2:

Tabla 2: *Conjunto de datos definidos para el modelo.*

Conjuntos	**Rangos**
Conjunto de docentes.	$i \in E = \{D_1, D_2, \ldots\ldots, D_n\}$
Conjunto de días	$j \in D = \{1,2, \ldots\ldots, 6\}$
Conjunto de materias	$k \in M = \{M_1, M_2, \ldots\ldots, M_n\}$
Subconjunto de materias por ciclo, el cual está delimitado por el conjunto de materias a hasta b que conforman un ciclo.	$(a, b) \in M_b = \{M_1, M_2, \ldots\ldots, M_n\}$
Conjunto de horas de clases	$l \in P = \{1,2, \ldots\ldots, 7\}$
Conjunto de semestres de clases.	$m \in S = \{1,2, \ldots\ldots, 10\}$
Número de créditos de materias	N_k
Conjunto de grupos de semestres	$G = \{1,2, \ldots\ldots, n\}$
Conjunto CG que relaciona ciclos con grupos	$CG = \{\text{Ciclo1:Grupo 1}, Ciclo1: Grupo2.., Ciclon: Grupon\}$

La función objetivo definida para nuestro modelo matemático es una función de maximización, que pretende maximizar la ocupación de espacios utilizados por los docentes. La función diseñada se encuentra descrita en la ecuación 6.

$$max\left(\sum_i \sum_j \sum_k \sum_l \sum_m X_{ijklm} \bullet D_{ijlm} \bullet \alpha_{ik}\right) \quad (6)$$

Donde:

X_{ijklm}: Variable de decisión binaria que indica si el docente (i) fue ubicado en un día (j) de la semana, en un período de clases (l) en una materia (k) perteneciente a un ciclo (m).

$$X_{ijklm} = \begin{cases} 1, Si\ el\ docente\ i\ es\ asignado\ en\ un\ día\ j\ a\ una\ materia \\ k\ en\ una\ hora\ l \\ 0, casocontrario \end{cases}$$

D_{ijlm}: Es necesario que los docentes tengan identificado el horario en el que pueden desempeñar sus labores, pues en base a esto los directores de carrera planifican los distributivos. Ha sido necesario diferenciar el calendario de cada docente de acuerdo a los semestres de clases, ya que en los ciclos iniciales existen materias comunes entre varias carreras por lo que se necesita reservar a los docentes que están a cargo de estas materias de modo que no se inhabilite el periodo por completo en todos los semestres sino exclusivamente al periodo que se ve afectado directamente. Este es un parámetro de penalización, pues contiene la disponibilidad de un docente (i) en un periodo (l) en un día (j) para un semestre (m). Por lo que en caso de asignar un docente en un periodo que no le corresponda, se obtendrá un valor de penalidad muy negativo (-10) para que la función objetivo que tiene por objeto el maximizar, pueda determinar que no debe volver a asignar en un espacio equivocado. Si por el contrario la asignación es exitosa entonces el valor obtenido será 1. Un ejemplo de distribución de horarios para un docente se puede ver en la Tabla 3.

$$D_{ijlm} = \begin{cases} 1, Si\ el\ docente\ i\ puede\ ser\ asignado\ en\ un\ día\ j\ en \\ el\ período\ l\ del\ semestre\ m. \\ -1, si\ se\ desea\ que\ el\ docente\ sea\ tomado\ en\ cuenta \\ para\ ser \\ asignado\ en\ este\ espacio\ pero\ \text{sin}\ tener\ prioridad. \\ \\ -10, para\ que\ no\ sea\ tomado\ en\ cuenta. \end{cases}$$

Tabla 3: *Ejemplo de un horario de períodos de clase disponibles para un docente en un semestre particular de clases. Estos datos se utilizan para el parámetro de penalización D_ijlm*

DÍA / **Docente**	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
Docente 1	1	-10	-10	-10	-10	1
	1	-10	-10	-10	-10	1
	-10	-10	1	-10	1	-10
	-10	-10	1	-10	1	-10
	1	-10	1	1	-10	-10

α_{ik} Este factor relaciona la pertenencia de un docente hacia una materia. Esta ponderación se obtiene de la API (Application Programming Interface por sus siglas en inglés) de Perfiles de Docentes descrita en [1]. En la tabla 4 se presenta la relación-docente.

Tabla 4: *Ejemplo de ponderaciones α_ik. Este factor relaciona al docente con la materia. Mientras más alto sea su valor, mayor probabilidad existe de que el docente dicte esa materia.*

DOCENTE	**PONDERACIÓN**	**MATERIA**
DOCENTE 1	1	Materia 1
DOCENTE 1	0.1	Materia 2

Restricciones

Las restricciones utilizadas para nuestro estudio fueron analizadas previamente con personal que labora en la Universidad Politécnica Salesiana. Estas restricciones fueron llevadas a su forma matemática para que el modelo propuesto pueda utilizarlas y llegar a la respuesta deseada.

Las restricciones han sido dividas en dos grupos: restricciones fuertes y restricciones suaves.

Restricciones Fuertes

Las restricciones fuertes deben cumplirse de forma obligatoria, pues son aquellas que la Universidad considera a fin de evitar que se generen horarios de clases sin sentido o imposible de llevar a la práctica laboral.

(1) En la ecuación 7 se detalla la restricción que evita que un periodo se repita en un mismo día para una materia, de tal manera que no se asignen varias materias de un mismo ciclo a un mismo periodo de clases.

$$\sum_{i}\sum_{j}\sum_{m=a}^{b} X_{ijklm} \leq 1, \forall_{k,l} \ (7)$$

(2) La ecuación 8 se detalla la restricción que asegura que las materias de los diferentes semestres sean asignadas al menos una por día. Aplicando esta restricción evitamos que haya ciclos más o menos cargados en un día de la semana.

$$\sum_{j}\sum_{m} X_{ijklm} = 1, \forall_{i,k,l} \ (8)$$

(3) La ecuación 9 evita cruces de varias materias para un docente en un mismo periodo de clases.

$$\sum_{i}\sum_{j}\sum_{l} X_{ijklm} \leq 1, \forall_{k,m} \ (9)$$

(4) La ecuación 10 asegura que todos los días de la semana que están disponibles sean asignados con al menos una materia.

$$\sum_{j} X_{ijklm} \leq 1, \forall_{i,k,l,m} (10)$$

(5) La ecuación 11 permite que una materia no sea asignada más de una vez en un mismo día de la semana.

$$\sum_{j} \sum_{k} X_{ijklm} \leq 1, \forall_{i,l,m} (11)$$

(6) La ecuación 12 garantiza que una materia no exceda el número de créditos en una semana de clases.

$$\sum_{m} X_{ijklm} = N_k, \forall_{i,j,k,l} \ (12)$$

Restricciones Suaves

Las restricciones suaves nos sirven para dar lineamientos al sistema de ciertos criterios que pueden o no cumplirse, pero que en la medida de lo posible requerimos que se satisfagan.

Para esto se ha definido las siguientes restricciones suaves:

(1) Los docentes tienen horarios de preferencia de acuerdo a planificaciones que tienen que ver con sus actividades diarias dentro de la Universidad, por lo cual el modelo deberá satisfacer esta condición pudiendo o no cumplirla. Por ello, todo dependerá de que tanto afecte esto a la generación total de la solución, pues si los horarios propuestos chocan con las restricciones fuertes, entonces difícilmente pueda satisfacer este criterio.

Para esta restricción se utilizó el modelo de matriz de la Tabla 3.

(2) Los docentes tienen un parámetro de pertinencia con respecto a las materias ofertadas, pudiendo de acuerdo al valor de dicho parámetro ser más propicios a ser seleccionados para dictar una u otra materia. De igual manera que la restricción (1), dependerá de cómo esto afecte a las restricciones fuertes y su valor de ajuste sea modificado. Para esta restricción se utilizó el modelo de matriz de la Tabla 4.

3.1.2 Experimentación mediante Programación Lineal

El modelo propuesto se implementó en el lenguaje de programación Python versión 3.6, lenguaje en el que fue factible realizar varias pruebas con distintas librerías para resolución matemática para modelos con Programación Lineal Entera con la finalidad de identificar la que mejor se adaptará para resolver nuestro modelo.

El hardware utilizado se muestra en la Tabla 5.

Tabla 5: *Descripción de componentes de Hardware utilizados para resolver computacionalmente el modelo matemático propuesto.*

COMPONENTE	DESCRIPCIÓN
Memoria RAM	16 GB DDR3
Disco duro	SSD 512 GB
Procesador	Core I7 2.1 GHz hasta 3.1GHz turboboost
Marca	Apple MacBook Pro Modelo 2013
Sistema Operativo	MacOS High Sierra

Tabla 6: *Conjunto de software utilizado para implementar el modelo matemático en Python.*

Software	Descripción
Python 3.6	Lenguaje de programación en el que se implementó el modelo.
PyCharm	IDE para programación en Python
Pip 9.0.1	Gestor de paquetes de Python
PuLP 1.6.8	Librería open source de Python utilizada para describir problemas de optimización como modelos matemáticos. PuLP se integra con varios solver externos.

GLPK 4.65	Paquete open source orientado a la resolución de problemas de gran escala en programación lineal, programación entera mixta, etc.
Pandas 0.22.0	Es una librería de Python de alto desempeño, provee facilidad de uso de estructuras de datos y herramientas de análisis de datos.
SCIP 5.0.1	SCIP Optimization Suite es un paquete de herramientas para generar y resolver programas enteros mixtos no lineales, en particular, programas lineales enteros mixtos y programas enteros de restricción.

El software que se utilizó para implementar los Algoritmos se muestra en la Tabla 6.

En lo que se refiere al proceso de implementación, se detallarán algunas consideraciones que se tomaron para manejar la información y los conjuntos de datos necesarios para trabajar.

- Los conjuntos de datos definidos en la Tabla 2: docentes, días, materias, períodos y ciclos fueron definidos con la estructura de datos de Python denominada "diccionario". Esta estrategia se tomó para poder trabajar con datos numéricos (las claves) que referencia a los valores (los nombres). Por ejemplo: un diccionario de días estaría definido de la siguiente manera.

Dias={0: "Lunes", 1: "Martes", 2: "Miércoles", 3: "Jueves", 4: "Viernes", 5: "Sábado"}

De esta manera el índice que corresponde a la matriz está relacionado con las llaves de los diccionarios, y el momento de mostrar los resultados se accede al índice para tener el nombre y que puedan ser interpretados rápidamente.

- La matrizD_{ijlm} fue definida de dimensiones ($numero_{docentes} x numero_{dias} x numero_{períodos} x numero_{ciclos}$).

Esta matriz carga los datos de un archivo Excel que contiene el horario de los docentes.

- La matriz α_{ik} posee las dimensiones ($numero_{docentes} \cdot numero_{materias}$).

Los datos de la matriz α_{ik} son consumidos de la API descrita en [12].

La definición del problema de maximización implementado en PuLP se muestra en la Fig. 5.

```
prob = pulp.LpProblem("Asignacion Horarios", pulp.LpMaximize)
x = pulp.LpVariable.dicts("x", itertools.product(range(numero_de_docentes),
                                                 range(numero_de_dias), range(numero_de_periodos),
                                                 range(numero_de_ciclos)), cat=pulp.LpBinary)
```

Figura 5: *Ejemplo de cómo definir un problema de maximización con la librería PuLP.*

Parámetros:

- Se definió un nombre de función y si llevará a cabo maximización o minimización de la función objetivo. En nuestro caso lo hemos denominado Asignación de Horarios y es una función de maximización.

- A la función debemos indicarle que variable va a ser con la que trabajamos en nuestro caso es la variable “x”.

- Existen varias estrategias para determinar la dimensión de nuestra matriz $X_{i,j,k,l,m}$, nosotros hemos optado por realizar el producto de ($docentes \cdot días \cdot períodos \cdot ciclos$).

- Finalmente se indica que se está trabajando con valores binarios.

Un ejemplo de restricción implementada mediante PuLP se muestra en la Fig.6.

```
for ciclo in range(numero_de_ciclos):
    prob += sum(x[(docente, dia, periodo, ciclo)] for dia in range(numero_de_dias)
                for periodo in range(numero_de_periodos) for docente in range(numero_de_docentes)
                ) == Creditos[ciclo]
```

Figura 6: *Ejemplo de definición de restricciones mediante librería PuLP. Esta restricción corresponde a la restricción fuerte 6.*

El diagrama de flujo utilizado para modelar el problema se puede observar en la Figura 7.

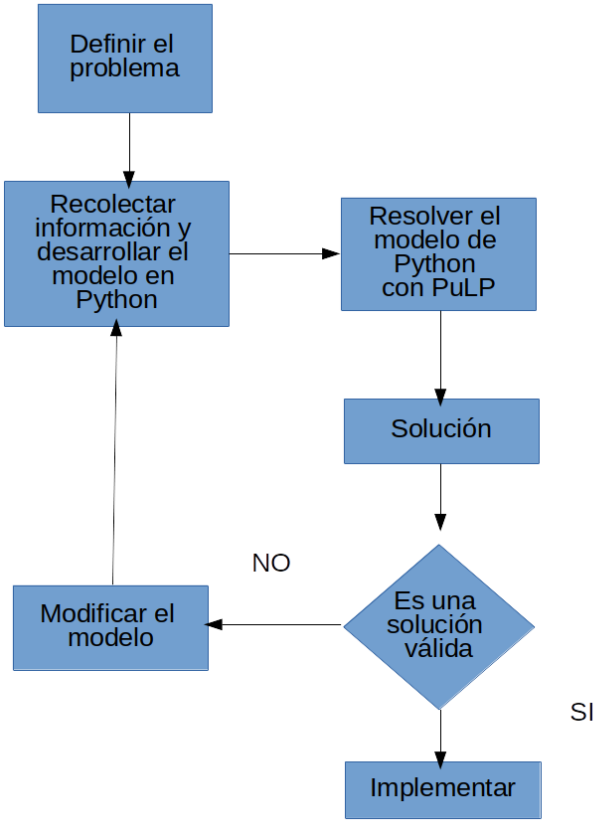

Figura 7: *Diagrama de flujo del proceso de modelamiento con la librería PuLP para desarrollar e implementar el problema de Timetabling. Modificado de Stuart Mitchell. et al 2009.*

3.1.3 Resultados Programación Lineal.

Una vez definido e implementado nuestro modelo matemático en Python, se procedió a resolver mediante el solver propio de PuLP y las librerías GLPK y SCIP. Se realizaron las siguientes comparaciones: Tiempo de resolución, consumo de memoria e iteraciones requeridas para determinar cuál de las tres librerías es más eficiente en el ámbito computacional (ver Figuras. 9, 10 y 11).

En cuanto al tiempo computacional se observa que si bien los tiempos computacionales generados por las librerías son cortos, la librería SCIP requiere mayor tiempo para generar una solución óptima (Ver Fig. 8).

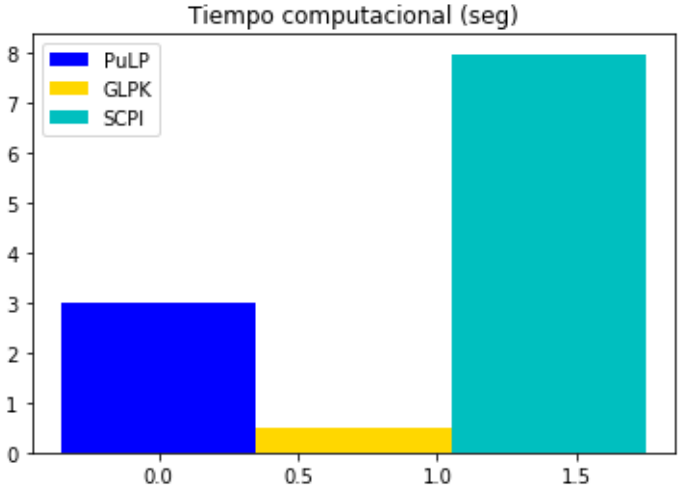

Figura 8: *Gráfica comparativa del tiempo computacional que se toman las librerías PuLP, GLPK y SCPI para resolver el Algoritmo Matemático para Timetabling.*

En la Fig. 9 se puede apreciar el gasto computacional, donde es mayor al utilizar la librería SCIP.

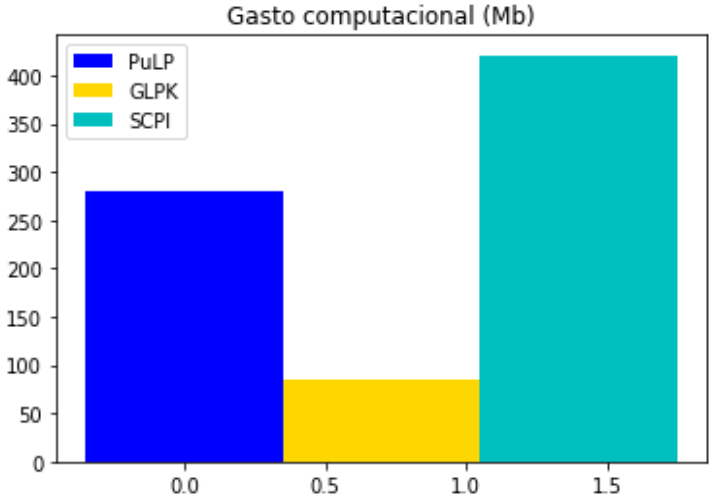

Figura 9: *Comparación del gasto computacional causado por las librerías PuLP, GLPK y SCPI para resolver el Algoritmo Matemático para Timetabling.*

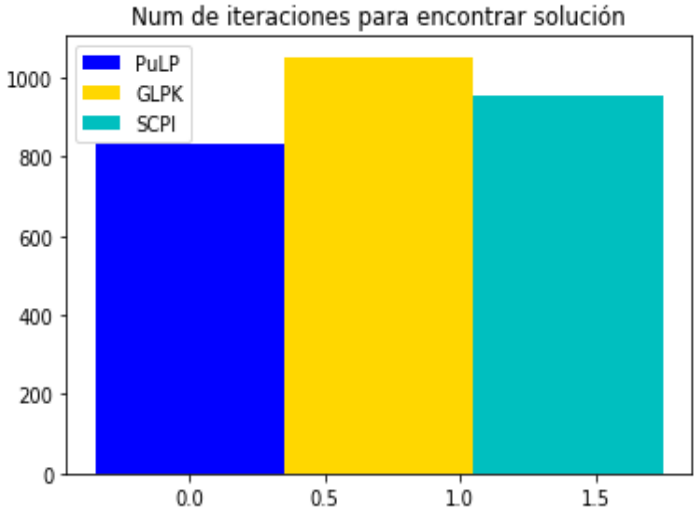

Figura 10: *Comparación sobre el número de iteraciones que son necesarias en cada una de las librerías PuLP, GLPK y SCPI para resolver el Algoritmo Matemático para Timetabling.*

En la Fig. 10 se aprecia que la librería GLPK requiere mayor número de iteraciones para encontrar una solución óptima.

Es importante indicar que de las tres librerías: “SCIP”, “PuLP” y “GLPK” se logró determinar, que la primera requiere mayor tiempo y esfuerzo computacional para generar una solución óptima (horario de clases) (ver Fig. 8, 9 y 10). Esto último, debido a que requiere un mayor número de número de iteraciones para llegar a dicha solución.

Posterior al análisis respecto al rendimiento de cada librería empleada para generar una solución óptima, se realizó la comparación sobre la calidad de los resultados obtenidos mediante cada una de las tres librerías antes mencionadas. Se evidenció que, respecto a esto último, la librería “SCIP” presenta una mejor distribución de materias durante la semana de clase, gracias a que está estructurada en base a un algoritmo de optimización robusto (ver Figuras 11, 12 y 13).

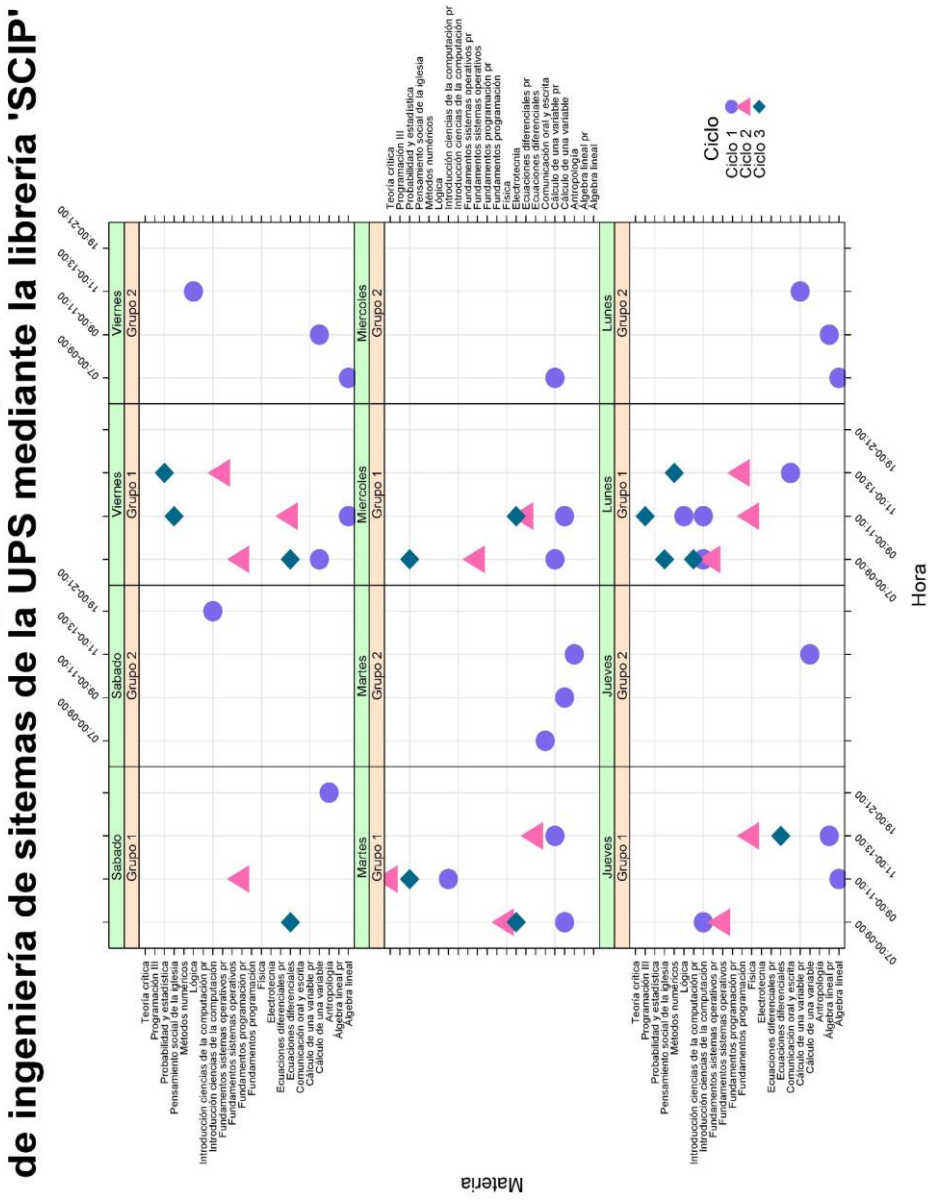

Figura 11: *En esta gráfica se puede apreciar que en los ciclos las horas están distribuidas equitativamente en función de la materia y grupo. El algoritmo fue más eficiente al aplicar esta librería, ya que logró una menor concentración de materias para los días sábado*

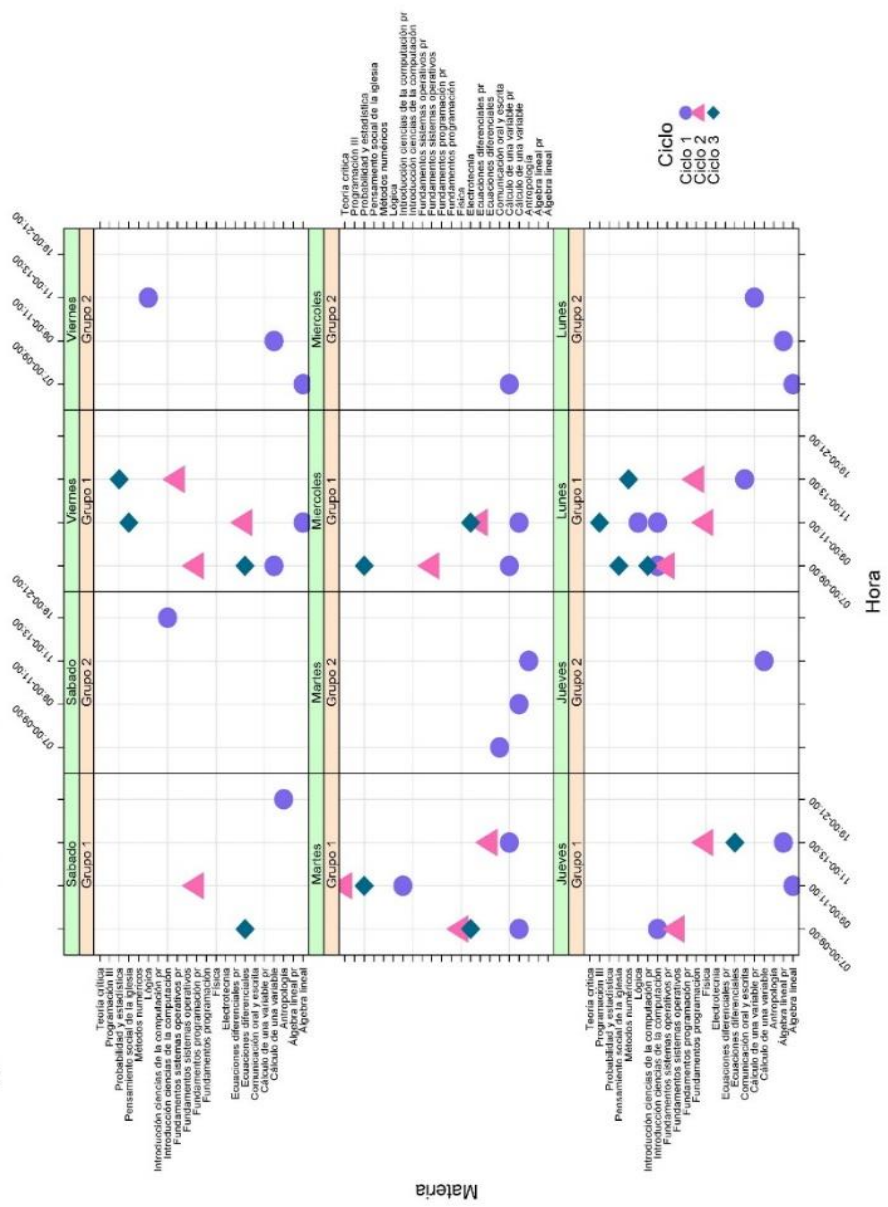

Figura 12: *Si bien en la gráfica sbserva en la semana una distribución de materias por ciclo equitativa, el día lunes y viernes contienen una mayor concentración de materias.*

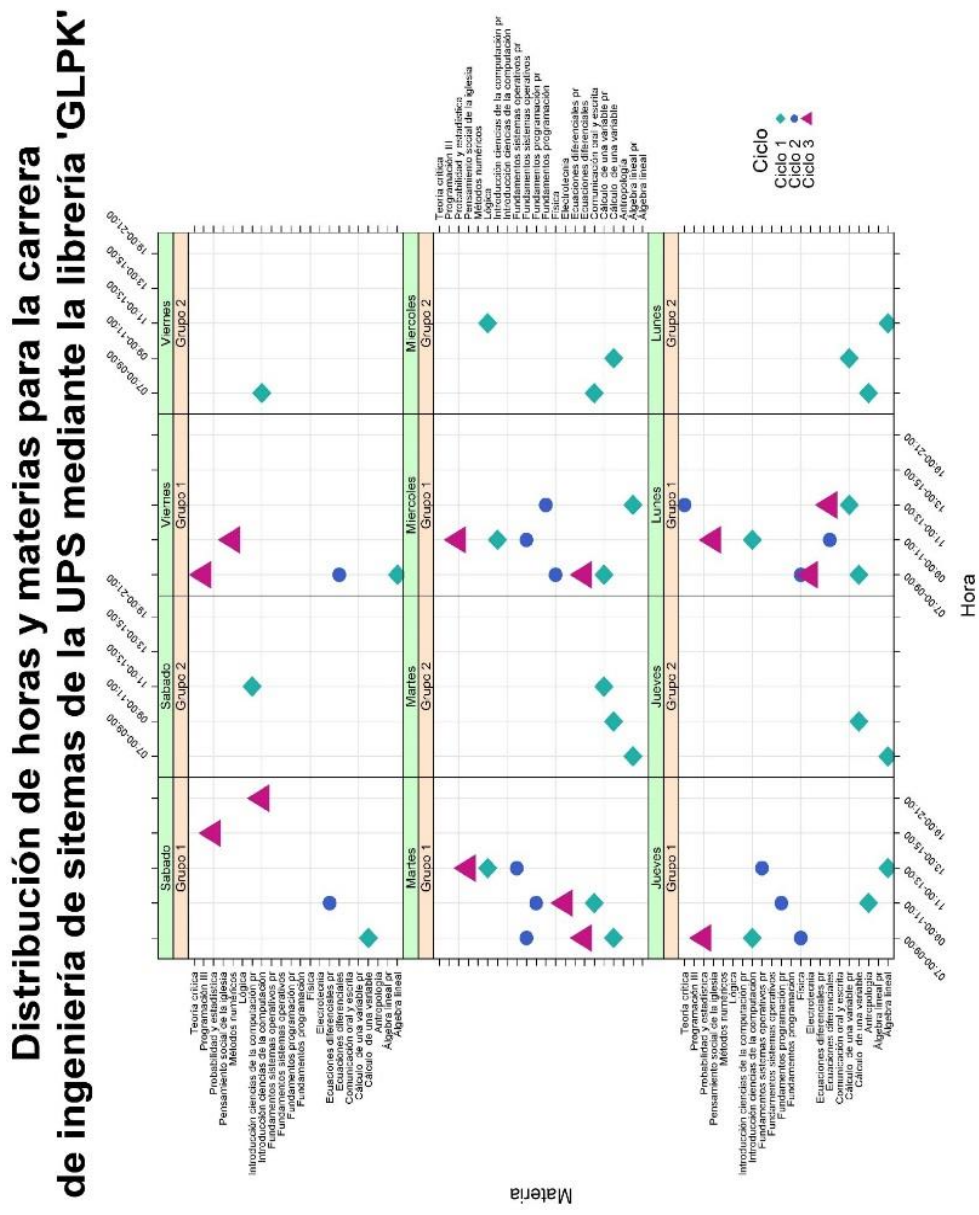

Figura 13: *El día viernes y sábado se parecían menos sobrecargados de materias.*

3.2 Algoritmos Genéticos

3.2.1 Modelo Propuesto con Algoritmos Genéticos.

De igual manera que en el caso de PLE se realizó la implementación de una solución mediante Algoritmos Genéticos para la generación de un modelo para asignación automática con restricciones. Debido a que esta técnica requiere definir restricciones para delimitar el problema y llegar a los resultados esperados se realizaron varias reuniones con personal de la Universidad Politécnica Salesiana y se llevó a cabo un análisis de los criterios que son tomados en cuenta al momento de diseñar un horario de clases y tener claro los criterios que intervienen en el proceso de generación de un distributivo de materias. Adicionalmente, se debe destacar que la implementación previa del modelo mediante PLE permitió que las restricciones e ideas de cómo realizar el AG sean más fácilmente acogidas.

Las restricciones con las que trabajamos este modelo son las siguientes:

- Un período no puede repetirse en un mismo día para una materia, de esta manera no se asignarán varias materias de un mismo ciclo a un mismo periodo de clases. Por ejemplo, no puede haber dos materias del ciclo 1 que se asignen de 7:00 a 9:00 am.

- Un docente no puede dar la misma materia en un mismo día.

- Garantizar que todas las materias ofertadas sean asignadas al horario de clases.

- Garantizar que todos los docentes disponibles sean asignados al horario de clases.

- Evitar cruces de varias materias que de un docente en un mismo periodo de clases.

- Garantizar que una materia no exceda el número de créditos en una semana de clases.

- Adecuar las horas de clases de los docentes al calendario entregado por cada uno de los docentes a la directora de carrera.

- Garantizar que los mejores perfiles de los docentes sean asignados a la materia que más se adapte al perfil.

Alcances de este modelo.

El modelo timetabling propuesto está desarrollado para los tres primeros ciclos, puesto que en estos niveles es donde más alumnos se tiene y por lo tanto se ofertan más grupos de materias, por lo tanto, se genera la mayor cantidad de conflictos y se dificulta el evitar cruces entre los docentes y las materias.

- El modelo propuesto se adapta a la creación de n grupos para los distintos semestres.
- Este modelo no abarca la asignación de aulas para dictar clases.
- Los días de clases considerados son de lunes a sábado. Se debe en la medida de lo posible ubicar asignaturas en los días sábado.
- Los períodos de clases se han sido dividido en segmentos de una hora cada uno dándonos un total de 14 períodos.
- Las asignaturas que se manejaron están distribuidas de la siguiente manera:

a. 6 asignaturas de primer ciclo grupo 1. Existen tres asignaturas que se dividen en horas de docencia y horas práctica, para lo cual hemos tomado la estrategia de dividir cada una de estas materias en dos, una materia con la carga horaria y otra materia con la carga práctica.

b. 6 asignaturas de primer ciclo grupo 2. De igual manera, se consideró el criterio anterior con materias con horas de docencia y horas prácticas.

c. 5 asignaturas de segundo ciclo grupo 1. Se tomó en cuenta el criterio de horas de docencia y horas prácticas.

d. 6 asignaturas de tercer ciclo grupo 1. No hizo falta considerar horas de docencia y horas prácticas.

e. En total las asignaturas que tuvo que ubicar el algoritmo es de 33 entre horas de docencia y prácticas.

3.2.2 Codificación del Algoritmo Genético

El Algoritmo Genético propuesto consiste en un modelo proyectado a 5 dimensiones, tres dimensiones a lo largo y ancho de los ejes cartesianos, y dos dimensiones inmersas en cada casillero.

Las dimensiones situadas externamente en los ejes son: día de la semana, hora de clases, y ciclo al que pertenece la materia. Mientras que en la parte interna se encuentran el docente y la materia a ser ubicados. Este conjunto de información codificado representa un horario de clases.

Dentro del proceso de ejecución del algoritmo los padres deben ser generados y seleccionados para la reproducción (operador cruce), a continuación, los cromosomas seleccionados se cruzarán generando dos hijos sobre cada uno de los cuales actuará un operador de mutación. El resultado de la combinación de las anteriores funciones será un conjunto de individuos (posibles soluciones al problema), mismas que en la evolución del Algoritmo Genético formarán parte de la siguiente población.

Los individuos del problema (posibles soluciones) a quienes llamaremos genes, se pueden representar como un conjunto de características de este último. Los genes al estar agrupados se denominan cromosomas.

El conjunto de parámetros que representan un cromosoma se denominan genotipos. Finalmente podemos decir que el fenotipo pasará a ser la representación de nuestro problema.

Para nuestro caso, el problema se ha definido con dos fenotipos, un fenotipo local y uno global. La propuesta consiste en evaluar inicialmente la función de ajuste local para determinar los cruces en cada uno de los ciclos y posteriormente evaluar como interactúan los ciclos en conjunto obteniendo un valor de función fitness global.

Como cromosoma local se ha definido a cada registro de la matriz que corresponde a cada ciclo, es decir, cada periodo de clases; si contamos con 14 períodos, entonces tendremos 14 cromosomas. El cromosoma global está definido como cada una de las capas o ciclos que contiene el horario de clases. Por ejemplo, si nuestro estudio está compuesto por tres ciclos, tendremos entonces, tres cromosomas contenidos uno por cada ciclo.

El propósito de representar el Algoritmo Genético de esta manera es debido a que por la cantidad de restricciones que se tiene y la representación en capas que se ha elegido

resulta más óptimo el evaluar ciclo a ciclo, así se puede mejorar tanto el rendimiento como la calidad de las soluciones, pues se pueden intercambiar ciclos que tenga menor valor de ajuste (fitness) y mejorar significativamente la población para así poder generar soluciones óptimas. (Ver figuras 14 y 15).

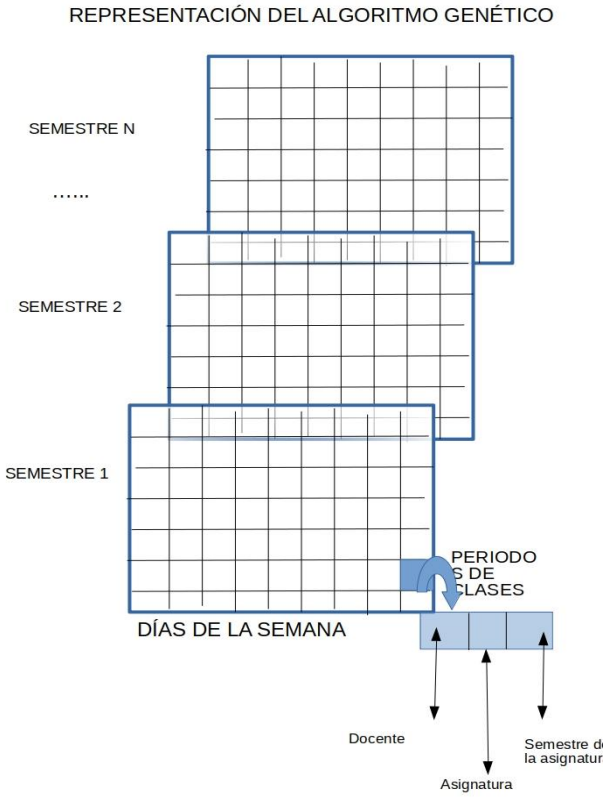

Figura 14: *Representación de la estructura del Algoritmo Genético.*

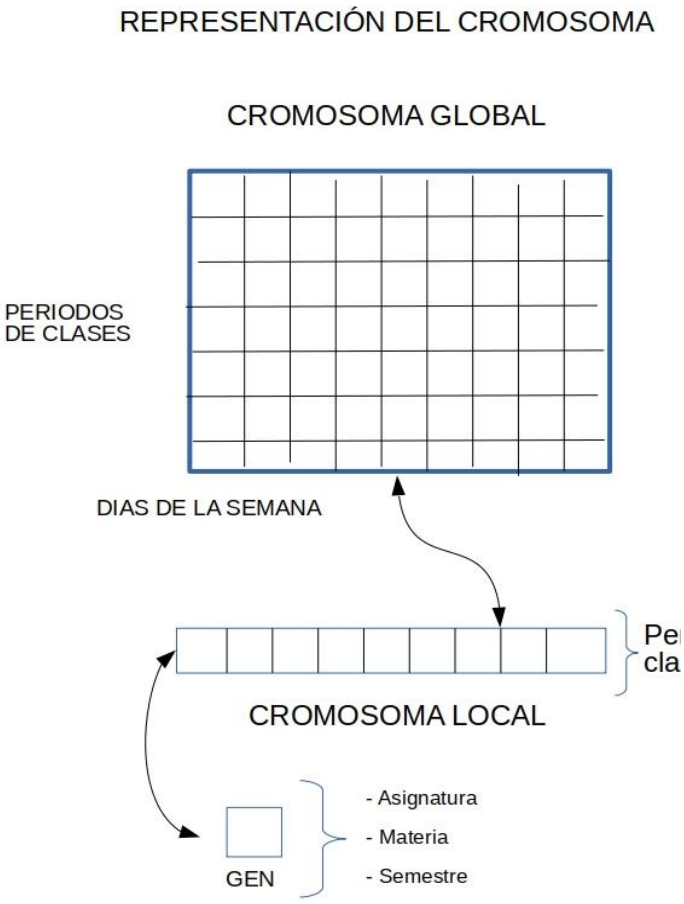

Figura 15: *Representación del cromosoma en el Algoritmo Genético propuesto.*

Parámetros de inicialización del Algoritmo Genético.

Para nuestro estudio debimos considerar algunas variables y parámetros que nos permitan configurar inicialmente nuestro AG.

Se consideró una población inicial de 8 elementos, una probabilidad de mutación P_m de 2%, se realizaron pruebas con 20000 iteraciones, cada una de las cuales generará nuevos pobladores y se eliminarán los pobladores con el valor más bajo de fitness. Se

ha considerado también que nos quedaremos para cada iteración con una población no mayor a 150 elementos

3.2.3 Proceso del Algoritmo Genético

Nuestro algoritmo se generó bajo el modelo generacional, ya que en cada iteración se genera nueva población, reemplazando de esta manera la población antigua por una nueva. Esta población estará dada por la combinación de los elementos generados luego de aplicar los operadores genéticos más elementos que son generados de una población aleatoria completamente nueva. Para el funcionamiento del algoritmo necesitamos generar una población inicial con la que se va a trabajar. Con ello, este algoritmo ubica las materias con su duración respectiva, pero sin tomar en cuenta las restricciones impuestas, de tal modo que no corresponderán a soluciones del problema. Mediante la técnica de AG se pretende ajustar y cumplir con las restricciones fuertes y suaves de modo que se pueda llegar a la solución prevista.

A continuación, se realiza un método de selección basado en la calidad, o valor del fitness calculado, los candidatos seleccionados son sometidos a torneo y pasarán a formar parte de nuestro proceso de cruce de padres, generando así hijos que mutarán y pasarán a formar parte de la nueva población. A continuación, se ordena la población para quedar solamente con los mejores elementos de la población.

3.2.3.1 Operador de Cruce.

El método implementado es una combinación entre en la selección aleatoria y la selección por torneo. El proceso inicia ordenando la población de mejor a peor fitness, a continuación, se genera un valor aleatorio que determina el rango de búsqueda de los pobladores a ser seleccionados, finalmente son seleccionados dos elementos aleatoriamente dentro del rango establecido y se someterán a torneo para competir entre ambos elementos y se seleccionará el que tenga mejor valor de función de ajuste.

El método de cruce propuesto ha sido pensado para intercambiar únicamente la dimensión que pertenece a los ciclos. Se define aleatoriamente un valor de corte que nos indicará cuantos ciclos del cromosoma padre y del cromosoma madre deberán ser intercambiados, el algoritmo deberá cruzar los ciclos correspondientes del denominado cromosoma padre y del cromosoma madre, generando así dos hijos nuevos que se agregarán a la población. Se utilizó esta metodología ya que al intercambiar entre ciclos no se afecta la planificación de las materias, evitando que por ejemplo se pierdan materias o se afecte la duración de las mismas por lo tanto la estructura del horario no perderá sentido (Ver figura 16).

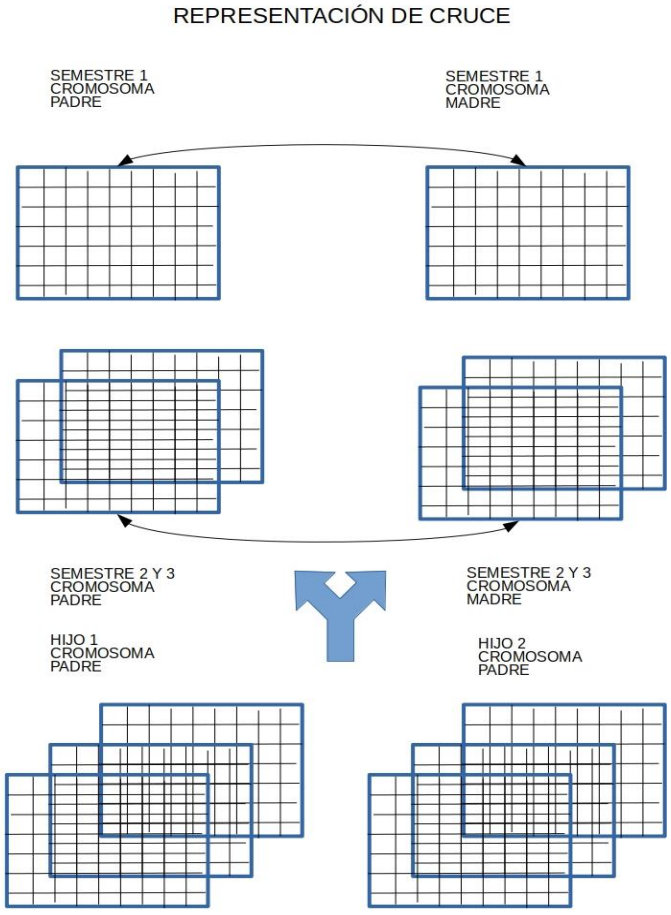

Figura 16: *Método de cruce implementado para el Algoritmo Genético.*

3.2.3.2 Operador de Mutación.

El operador de mutación propuesto tiene como objetivo realizar movimientos en los cromosomas de los días y de los períodos, para ello se ha utilizado el método de la ruleta rusa para determinar en qué sentido va a mutar el cromosoma. Pudiendo realizarse la mutación en el eje "x" que pertenece a los días de la semana si el valor aleatorio obtenido es menor o igual a 0.35. En el eje "y" que corresponde a los períodos u horas de clase si el valor está entre 0.35 y 0.7, o en ambos ejes para valores mayores a 0.7 y

menores a 1. De tal manera que el AG procederá a intercambiar los valores de las horas de clases de la materia elegida aleatoriamente. La Probabilidad de que el individuo ingrese a mutar viene dada por el parámetro P_m que tiene un valor es de 2%. P_m fue definido luego de realizar algunas pruebas en las que se determinó que ese valor es el que mejor se adapta a las condiciones del modelo.

Para implementar este operador se tomaron en cuenta algunas consideraciones para que al momento de ejecutar la mutación los horarios de clases no se vean afectados.

Se sabe que el cambio de cromosomas no puede ser realizado entre cromosomas que pertenezcan a diferentes ciclos para así mantener la estructura y por el contrario no degradar la calidad de los horarios.

Por otra parte, se debe tomar en cuenta que el intercambio de materias debe ser realizado entre materias con la misma duración, pues al momento de intercambiar las horas podría sobre montarse sobre otra materia, lo cual nos dejaría con horarios inconsistentes y se lograría al igual que con la consideración anterior degradar la calidad de la población. Para esto nosotros hemos implementado un algoritmo de búsqueda que permite identificar y localizar la materia que será intercambiada, de modo que al comparar sus longitudes estas deberán ser iguales para realizar el cambio, caso contrario, se continúa realizando iteraciones hasta encontrar una materia que cumpla las condiciones antes mencionadas.

Se debe tomar en cuenta también que existen horarios con espacios vacíos entre materias, por lo cual se debe tomar una estrategia frente a esto, pues si las restricciones vienen dadas por ejemplo para materias que se pueden dictar únicamente en la mañana, se podría caer en un bucle del cual no se lograría converger hacia una solución deseada (Ver figuras 17 y 18).

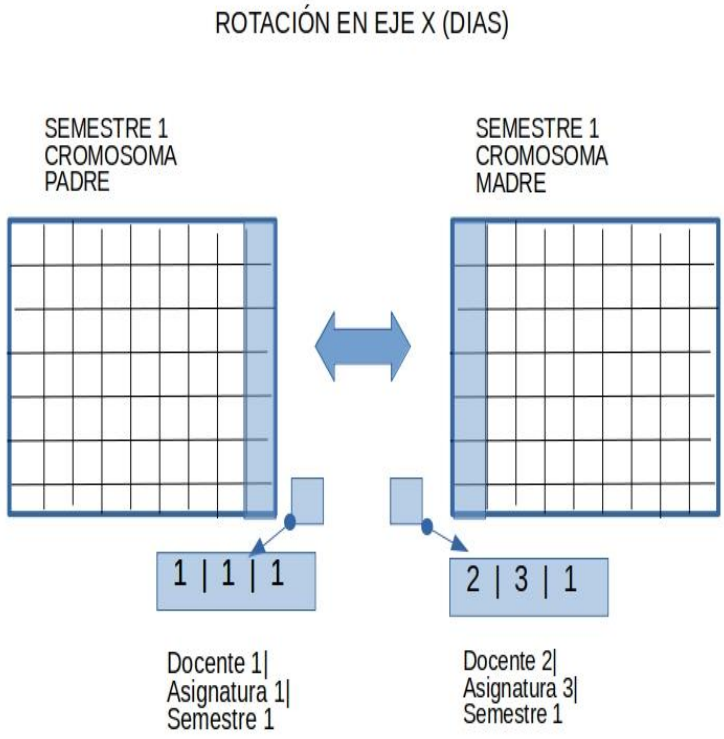

Figura 17: *Representación del método de mutación cuando se produce en el eje de los días para el Algoritmo Genético. Como se ve en la figura se busca un día de la semana para ser intercambiado.*

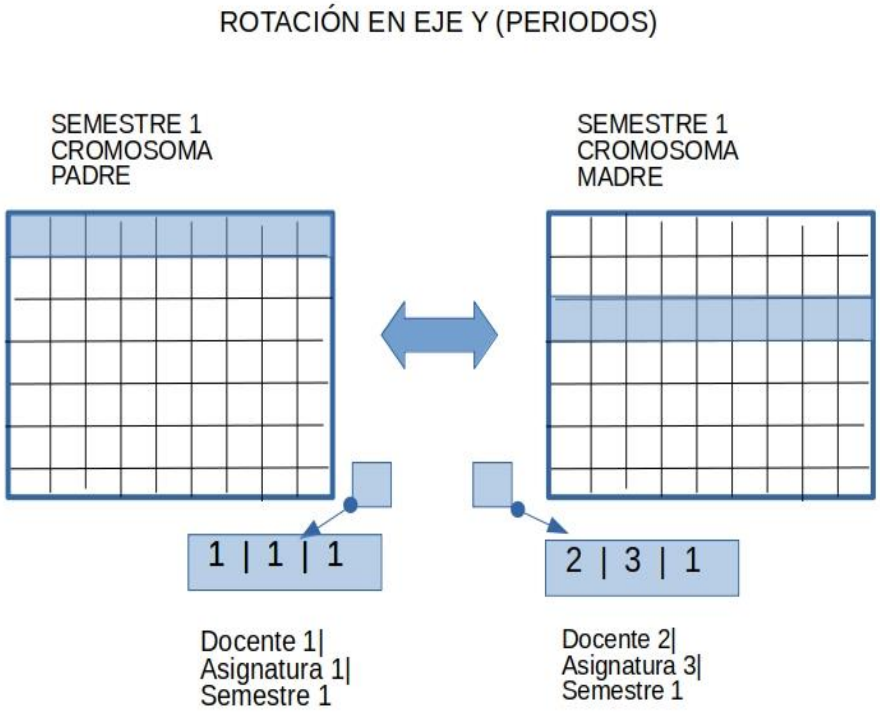

Figura 18: *Representación del método de mutación cuando se produce en el eje de los períodos para el Algoritmo Genético. En la figura se observa que se realiza el intercambio entre las filas de los períodos de clases.*

3.2.3.3 Función Objetivo

La función objetivo se encuentra definida en la ecuación 13 es una parte clave en la definición del AG, ya que de esta dependerá el poder determinar la convergencia óptima del modelo. La función objetivo está conformada por las variables n_T,n_c. Donde n_T representa el número total de casos para la asignación en los horarios. n_c es el número de cruces o numero de choques que se produce en en las restricciones. El valor óptimo será de 100, ya que en caso de que el AG detecte que se han eliminado todos los cruces el valor de n_c será cero. Esta función deberá ser evaluada de manera local, es decir en cada capa de ciclo. Y de igual manera se deberá evaluar de manera global, la cual será la que determine la calidad del poblador como tal.

$$FO = 100 * \left(\frac{n_T - n_c}{n_T}\right) \quad (13)$$

3.2.4 Resultados Algoritmos Genéticos (AG)

Luego de realizar varias experimentaciones, es importante indicar que no se consiguió realizar un estudio que se ajuste a todas las restricciones planteadas, pues los tiempos del AG son demasiado elevados. Sin embargo, simplificando el estudio en relación a las restricciones se lograron obtener resultados que permitieron validar que el AG está correctamente implementado a pesar de que se requieren ajustes para que se puedan agilitar los tiempos de evolución del mismo.

Con respecto a la aplicación de PLE no se aplicó en el AG la restricción de asignación de asignaturas respecto a la jornada diurna para todos los docentes de la carrera de ingeniería de sistemas. Con respecto a esto último, se aprecia en la Fig. 19 el ajuste del AG lento y una gran cantidad de iteraciones para alcanzar una óptima evolución.

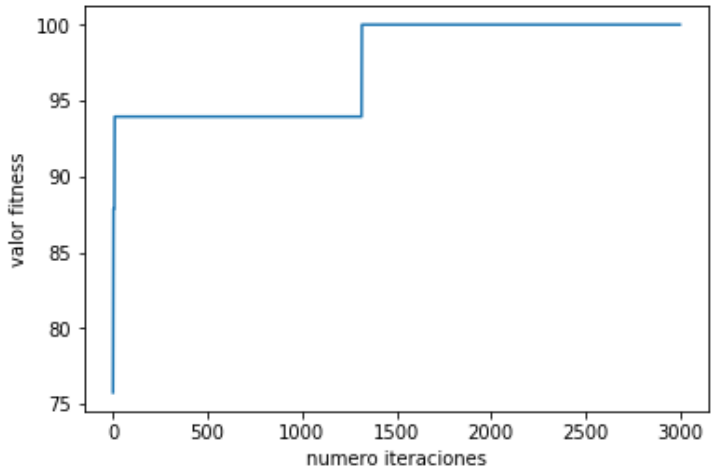

Figura 19: *En esta gráfica se aprecia el ajuste del algoritmo genético luego de cumplir 3.000 iteraciones.*

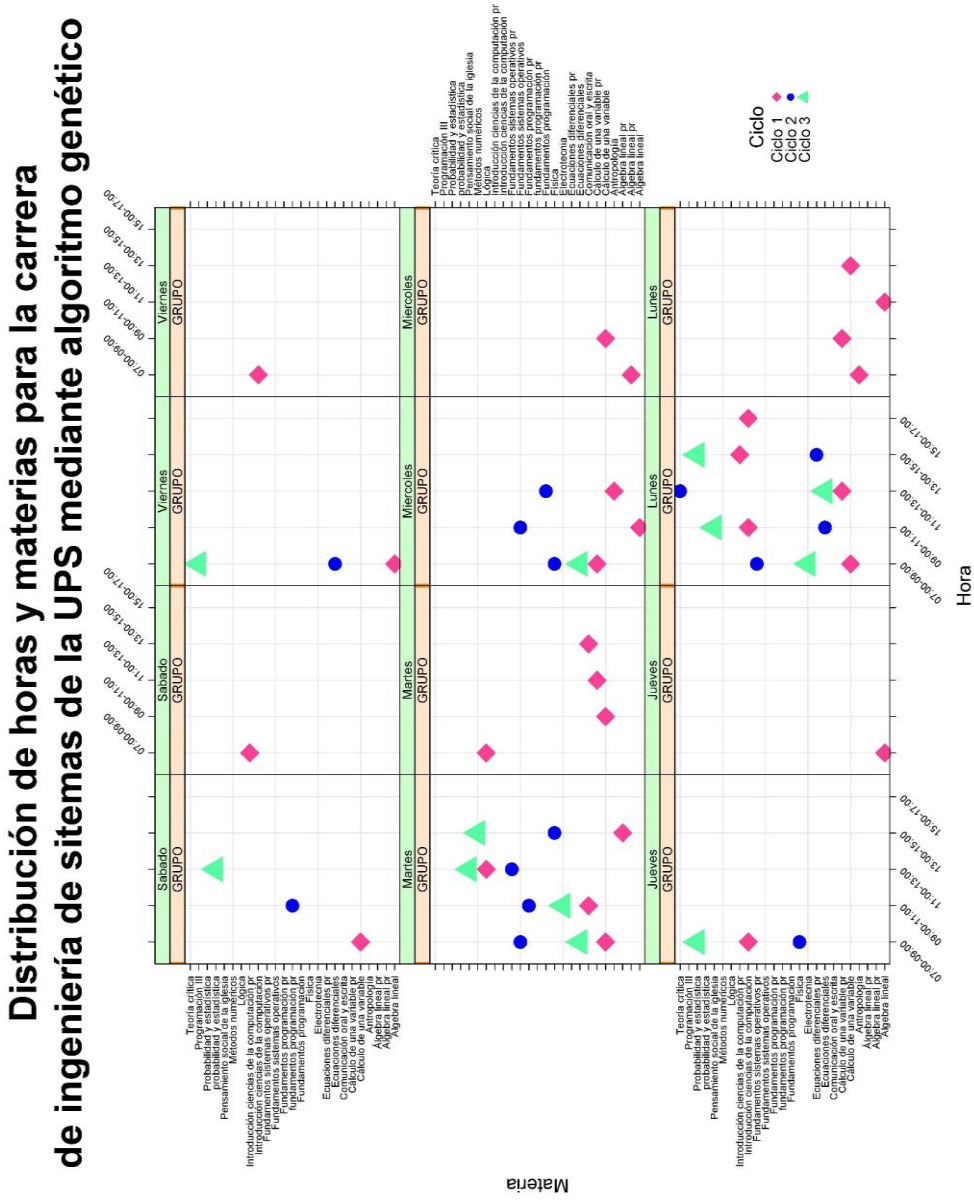

Figura 20: *Se aprecia que los días jueves, viernes y sábado presentan una menor concentración de materias en contraste con los demás días de la semana.*

A continuación, en la **Tabla 6** se destacan las principales características de la PLE y de los AG, así como también su respectivo funcionamiento dentro de la programación.

Tabla 6: *Se aprecian las diferencias acerca de las características y funcionamiento entre PLE y AG.*

Característica	Programación Lineal Entera (PLE)	Algoritmos Genéticos
Tipo de algoritmo	Determinístico	Estocástico
Numero de soluciones	Una sola solución	Puede llegar a generar n soluciones
Numero de iteraciones	≤ 1000	> 100000
Requerimiento computacional	Bajo	Alto
Tiempo de solución	≤ 1 min	> 1 hora
Capacidad de implementar reglas complejas	No permite	Si permite

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Programación lineal entera (PLE)

Se ha observado que la aplicación de PLE genera resultados en tiempos ciertamente menores al minuto (60 segundos). Sin embargo, su implementación conlleva dificultades. Esto último dado que es preciso manejar con claridad los conceptos de PLE, aunado a ello se deben identificar minuciosamente las restricciones del problema bajo estudio.

Respecto a las restricciones anteriormente indicadas, es indispensable indicar con exactitud qué función cumplirán dentro del algoritmo.

Es necesario indicar que, para la aplicación de la PLE correspondiente al problema específico de este estudio, no fue necesario contar con un hardware especializado, o con características fuera de las que tiene computador personal común, ya que el esfuerzo computacional es mínimo.

Con respecto a los horarios generados mediante PLE, se observó que el modelo logró ajustarse generosamente a las restricciones impuestas para timetabling deseado.

El modelo generado a través de PLE es de naturaleza determinística, dado que entrega una única solución para el timetabling.

Las iteraciones necesarias para llegar a una solución óptima a través del PLE son escasas, ya que se encuentran dentro de un rango inferior a las 1.000.

También se puede indicar que el modelo PLE requiere generar un alto número de variables y restricciones para poder delimitar y especificar el problema al tiempo que permitir resolverlo de manera más eficiente.

Una vez que se logra implementar el modelo de PLE no requiere una supervisión para el consiguiente ajuste paramétrico. Esto último en vista de que, al ser un modelo determinístico, las variables y restricciones son exactas.

Algoritmos genéticos (AG)

En cuanto a los AG, se pudo determinar que, si bien la programación del algoritmo como tal es más flexible, su implementación requiere del conocimiento de conceptos acerca de programación avanzada. La programación avanzada es indispensable para implementar estrategias encaminadas a reducir el número de iteraciones, y así se logre alcanzar un menor gasto computacional y permitir una evolución más rápida del algoritmo.

Es de vital importancia simplificar la representación del algoritmo genético. Ya que como se mencionó entre líneas anteriores, la propia implementación es compleja, y la representación inadecuada podría lugar a complicaciones durante la implementación de los operadores genéticos.

Se observó que es de suma importancia la definición de una función objetivo que permita describir y entender el comportamiento evolutivo del algoritmo genético.

Cuando se aplicó el AG para el timetabling, se pudo evidenciar que por la representación compleja que conlleva dicho proceso, el AG requiere de un elevado número de iteraciones para llegar una posible solución óptima.

Respecto al problema antes descrito, la aplicación del AG conlleva un alto gasto en cuanto al tiempo computacional, lo cual conduce a pensar que se requiere un equipo computacional con características elevadas. Esto último permitiría la implementación de estrategias de programación de mayor eficacia, tal y como lo representan los hilos computacionales.

También se logró determinar que con la aplicación del AG para el timetabling deseado no se alcanzó a completar con el número de restricciones impuestas. El número de dichas restricciones no se consiguió, ya que tanto el número de iteraciones, como el gasto computacional requeridos no estuvieron al alcance del proceso bajo estudio.

Para finalizar se indicará que la implementación de las reglas en los AG versus la PLE resultó más factible, debido a que la programación como tal, resulta mayormente intuitiva.

En cuanto a los tiempos computacionales se evidenció la gran ventaja que representa la implementación de timetabling mediante PLE versus AG.

La implementación de AGE resulta ventajoso frente a la PLE en cuanto al número de soluciones óptimas, ya que la PLE es un modelo determinístico que nos permite generar una única solución, mientras que AG al constituir un modelo estocástico, da lugar a varias posibilidades. Esto último dependerá de la calidad de implementación del AG.

RECOMENDACIONES

Se sugiere que para mejorar el desempeño del AG la relación de las materias con las jornadas educativas debe ser un factor clave a tomar en cuenta, ya que esto podría reducir de forma significativa el gasto computacional que se realiza en el operador de mutación. Esto último debido a que las búsquedas estarían mejor focalizadas en ubicar períodos que se adapten a la jornada requerida.

En cuanto al desarrollo y evaluación de AG y PLE se sugiere buscar en lo posible simplificar el modelo propuesto para que las operaciones puedan ser desarrolladas sobre vectores y no sobre matrices.

BIBLIOGRAFÍA

[1] Abdullah, S., & Turabieh, H. (2012). On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems. *Information Sciences*, *191*, 146-168.

[2] Ahmad, I. R., Sufahani, S., Ali, M., & Razali, S. N. (2018, April). A Heuristics Approach for Classroom Scheduling Using Genetic Algorithm Technique. In *Journal of Physics: Conference Series* (Vol. 995, No. 1, p. 012050). IOP Publishing.

[3] Akbari, M. (2018). An Efficient Genetic Algorithm for Task Scheduling on Heterogeneous Computing Systems Based on TRIZ. *Journal of Advances in Computer Research*, *9*(3), 103-132.

[4] Akkan, C., & Gülcü, A. (2018). A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem. Computers & Operations Research, 90, 22-32.

[5] AL-MILLI, N. R. (2011). HYBRID GENETIC ALGORITHMS WITH SIMULATING ANNEALING FOR UNIVERSITY COURSE TIMETABLING PROBLEMS. *Journal of Theoretical & Applied Information Technology*, *29*(2).

[6] Alsmadi, O. M. K., Za'er, S., Abu-Al-Nadi, D. I., & Algsoon, A. (2011, May). A novel genetic algorithm technique for solving university course timetabling

problems. In *Systems, Signal Processing and their Applications (WOSSPA), 2011 7th International Workshop on* (pp. 195-198). IEEE.

[7] Azadeh, A., Gholizadeh, H., & Jeihoonian, M. (2013). A multi-objective optimisation model for university course timetabling problem using a mixed integer dynamic non-linear programming. International Journal of Services and Operations Management, 15(4), 467-481.

[8] Babaei, H., Karimpour, J., & Hadidi, A. (2015). A survey of approaches for university course timetabling problem. Computers & Industrial Engineering, 86, 43-59.

[9] Boland, N., Hughes, B. D., Merlot, L. T., & Stuckey, P. J. (2008). New integer linear programming approaches for course timetabling. Computers & Operations Research, 35(7), 2209-2233.

[10] Bucco, G. B., Bornia Poulsen, C. J., & Bandeira, D. L. (2017). Desenvolvimento de um modelo de programação linear para o Problema da Construção de Grades Horárias em Universidades. Gestão e produção. São Carlos. Vol. 24, n. 1 (jan./abr. 2017), p. 40-49.

[11] Burke, E. K., Mareček, J., Parkes, A. J., & Rudová, H. (2012). A branch-and-cut procedure for the Udine course timetabling problem. Annals of Operations Research, 194(1), 71-87.

[12] Calle-López, D., Cornejo-Reyes, J., Pesántez-Avilés, F., Rodas-Tobar, M., Vásquez-Vásquez, C., & Robles-Bykbaev, V. (2018). Un sistema experto basado en minería de datos y programación entera lineal para soporte en la

asignación de materias y diseño de horarios en educación superior. *Enfoque UTE*, *9*(1), 102-117.

[13] Chiarandini, M., Birattari, M., Socha, K., & Rossi-Doria, O. (2006). An effective hybrid algorithm for university course timetabling. *Journal of Scheduling*, *9*(5), 403-432.

[14] Diaz-Gomez, P. A., & Hougen, D. F. (2007). Initial Population for Genetic Algorithms: A Metric Approach. In *GEM* (pp. 43-49).

[15] Emrah, B. E., & Edis, R. S. (2013). An integer programming model for the conference timetabling problem. CBU J Sci, 9(2), 55-62.

[16] Flores-Bazán, F. (2009). Optimización lineal: una mirada introductoria.

[17] Gestal Pose, M. (2006). Introducción a los algoritmos genéticos. Universidad de la Coruña, Tecnologías de la información y las comunicaciones.

[18] Guerrero, S. H. (2009). Programación lineal aplicada.

[19] Gunnarsdóttir, Þ. (2018). *An Integer Programming Formulation for the Music School Timetabling Problem* (Doctoral dissertation).

[20] Gunawan, A., Ng, K. M., & Poh, K. L. (2012). A hybridized Lagrangian relaxation and simulated annealing method for the course timetabling problem. Computers & Operations Research, 39(12), 3074-3088.

[21] Havås, J., Olsson, A., Persson, J., & Schierscher, M. S. (2013). Modeling and optimization of university timetabling-a case study in integer programming.

[22] Huynh, N. T., & Chien, C. F. (2018). A hybrid multi-subpopulation genetic algorithm for textile batch dyeing scheduling and an empirical study. Computers & Industrial Engineering.

[23] Kohshori, M. S., & Abadeh, M. S. (2012). Hybrid genetic algorithms for university course timetabling. *International Journal of Computer Science Issues (IJCSI)*, *9*(2), 446.

[24] Kong, M. (2010). Investigación de operaciones: programación lineal. problemas de transporte. análisis de redes.

[25] Kuri, Á., & Galaviz, J. (2009). Algoritmos genéticos.

[26] Lach, G., & Lübbecke, M. E. (2012). Curriculum based course timetabling: new solutions to Udine benchmark instances. Annals of Operations Research, 194(1), 255-272.

[27] Lindahl, M., Sørensen, M., & Stidsen, T. R. (2018). A fix-and-optimize matheuristic for university timetabling. Journal of Heuristics, 1-21.

[28] Martínez, S. I. A., & Vértiz, C. G. (2014). Investigaciones de operaciones.

[29] McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., & Qu, R. (2012). A new model for automated examination timetabling. Annals of Operations Research, 194(1), 291-315.

[30] Méndez, F. M. C., Robles-Bykbaev, V., & Vanegas-Peralta, P. (2017, March). THERAPIST-TRAINER: UNA APLICACIÓN EDUCATIVA BASADA EN ALGORITMOS GENÉTICOS Y ENTORNOS DE APRENDIZAJE

VIRTUAL PARA LA PREPARACIÓN DE ESTUDIANTES DE LOGOPEDIA. In CIIEE 2017.

[31] MirHassani, S. A., & Habibi, F. (2013). Solution approaches to the course timetabling problem. Artificial Intelligence Review, 39(2), 133-149.

[32] Mohammed, M. A. (2017). A Review of Genetic Algorithm Application in Examination Timetabling Problem*" Mazin Abed Mohammed," Mohd Khanapi Abd Ghani," Omar Ibrahim Obaid," Salama A. Mostafa,"Mohd Sharifuddin Ahmad," Dheyaa Ahmed Ibrahim and" MA Burhanuddin" Biomedical Computing and Engineering Technologies (BIOCORE) Applied Research Group. *Journal of Engineering and Applied Sciences*, *12*(20), 5166-5181.

[33] Nagata, Y. (2018). Random partial neighborhood search for the post-enrollment course timetabling problem. Computers & Operations Research, 90, 84-96.

[34] Nissen V. (2018) Applications of Evolutionary Algorithms to Management Problems. In: Moutinho L., Sokele M. (eds) Innovative Research Methodologies in Management. Palgrave Macmillan, Cham

[35] Nugraha, T. A., Putra, K. T., & Hayati, N. (2017). University Course Timetabling with Genetic Algorithm: A Case Study. *Journal of Electrical Technology UMY*, *1*(2), 100-105.

[36] Obaid, O. I., Ahmad, M., Mostafa, S. A., & Mohammed, M. A. (2012). Comparing performance of genetic algorithm with varying crossover in solving

examination timetabling problem. *J. Emerg. Trends Comput. Inf. Sci*, *3*(10), 1427-1434.

[37] Peña, D., Tchernykh, A., Nesmachnow, S., Massobrio, R., Feoktistov, A., Bychkov, I., ... & Garichev, S. N. (2018). Operating cost and quality of service optimization for multi-vehicle-type timetabling for urban bus systems. Journal of Parallel and Distributed Computing.

[38] Pereira, V., & Gomes Costa, H. (2016). Linear integer model for the course timetabling problem of a faculty in Rio de Janeiro. Advances in Operations Research, 2016.

[39] Phillips, A. E., Waterer, H., Ehrgott, M., & Ryan, D. M. (2015). Integer programming methods for large-scale practical classroom assignment problems. *Computers & Operations Research*, *53*, 42-53.

[40] Pillay, N. (2014). A survey of school timetabling research. Annals of Operations Research, 218(1), 261-293.

[41] Ploskas, N., & Samaras, N. (2017). Linear Programming Using MATLAB® (Vol. 127). Springer.

[42] Ribeiro Filho, G., & Lorena, L. A. N. (2006). An integer programming model for the school timetabling problem. In *XIII Congresso Latino-Ibero Americano de Investigaci {ó} n Operativa (CLAIO)* (Vol. 27, p. 6).

[43] Sánchez-Partida, D., Martínez-Flores, J. L., & Olivares-Benítez, E. (2014). An integer linear programming model for a university timetabling problem

considering time windows and consecutive periods. Journal of Applied Operational Research, 6(3), 158-173.

[44] Saviniec, L., Santos, M. O., & Costa, A. M. (2018). Parallel local search algorithms for high school timetabling problems. European Journal of Operational Research, 265(1), 81-98.

[45] Song, T., Liu, S., Tang, X., Peng, X., & Chen, M. (2018). An iterated local search algorithm for the University Course Timetabling Problem. Applied Soft Computing.

[46] Vanderbei, R. J. (2015). Linear programming. Heidelberg: Springer.

[47] Weise, T. (2009). Global optimization algorithms-theory and application. Self-published, 2.

[48] Xiao, J., Pachl, J., Lin, B., & Wang, J. (2018). Solving the block-to-train assignment problem using the heuristic approach based on the genetic algorithm and tabu search. Transportation Research Part B: Methodological, 108, 148-171.

[49] Yang, S., & Jat, S. N. (2011). Genetic algorithms with guided and local search strategies for university course timetabling. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 41(1), 93-106.

UNIVERSIDAD POLITECNICA SALESIANA

UNIDAD DE POSGRADOS

MAESTRÍA EN MÉTODOS MATEMÁTICOS Y SIMULACIÓN NUMÉRICA EN INGENIERÍA

Autor: Pedro Javier Cornejo Reyes

Director: PhD. Vladimir Robles Bykbaev

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO MATEMÁTICO BASADO EN OPTIMIZACIÓN PARA LA GENERACIÓN DE HORARIOS DE CLASES EN LA UNIVERSIDAD POLITÉCNICA SALESIANA

La presente investigación refleja el resultado de la aplicación de las técnicas de programación lineal entera y algoritmos genéticos para el desarrollo e implementación de horarios de clases para la carrera de Ingeniería de Sistemas de la Universidad Politécnica Salesiana.

En el presente trabajo se ha realizado el modelamiento e implementación de un modelo matemático de horarios de clases mediante el levantamiento y aplicación de requerimientos por parte de la institución antes mencionada. Esto último permitió formular restricciones, pues fueron indispensables para delimitación del modelo.

Las conclusiones y resultados de este trabajo pueden ser utilizadas para procesos futuros en los cuales se requiera mejorar e implementar el modelo para asignación de horarios en las distintas carreras de la Universidad Politécnica Salesiana.