INSTITUCIÓN UNIVERSITARIA POLITÉCNICO GRANCOLOMBIANO

FACULTAD DE INGENIERÍA, DISEÑO, E INNOVACIÓN

INGENIERÍA DE SISTEMAS

GRUPO DE INVESTIGACIÓN E-TIC

ALGORITMO BASADO EN LA COLORACIÓN DE GRAFOS PARA LA PROGRAMACIÓN FLEXIBLE DE HORARIOS Y SALONES EN UNA INSTITUCIÓN UNIVERSITARIA: CASO DE ESTUDIO EN COLOMBIA

PRESENTA:

ANDRÉS DAVID LEAL FIGUEREDO
100267100

ASESOR:

JOSÉ MANUEL CHAUTA TORRES, PhD.

Diciembre, 2024

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La asignación de horarios y salones en instituciones universitarias es un problema complejo clasificado como NP-Completo, debido a la interdependencia de variables como profesores, asignaturas, grupos, salones y restricciones de tiempo. Este trabajo propone un algoritmo basado en la coloración de grafos para programar horarios de manera eficaz y flexible, representando las sesiones de cursos como vértices de un grafo no dirigido, donde las restricciones crean aristas que aseguran la no superposición de eventos conflictivos.

El enfoque incorpora restricciones estrictas y opcionales, garantizando la validez del horario y maximizando la satisfacción de las preferencias. Además, se introdujo un modelo para reservar porcentajes de disponibilidad en los salones, permitiendo flexibilidad en la reubicación de sesiones tras cambios imprevistos.

Los resultados incluyen la caracterización de variables, una complejidad temporal cuadrática en la implementación del algoritmo y un modelo de datos normalizado. Aunque no se lograron realizar experimentos computacionales por la falta de parametrización en los datos iniciales, la solución demuestra su potencial para abordar escenarios reales con un enfoque innovador de flexibilidad.

Las conclusiones resaltan la viabilidad del método para adaptar horarios ante cambios, minimizando traslados entre sesiones consecutivas y optimizando recursos disponibles. Como trabajo futuro, se recomienda considerar la disponibilidad de profesores como restricción flexible, desarrollar una interfaz gráfica para la visualización de horarios y explorar estrategias de ordenamiento para mejorar la calidad del resultado final.

Este estudio contribuye a la literatura sobre programación académica, ofreciendo un modelo adaptable que responde a las necesidades dinámicas de las instituciones universitarias.

INTRODUCCIÓN

La programación de horarios de cursos en las instituciones universitarias es un problema que se enfrenta cada semestre por parte de sus administraciones, e incluso ha sido tema de investigación desde finales del siglo pasado que hasta hoy se sigue tratando desde diferentes enfoques, como la coloración de grafos, algoritmos genéticos, programación lineal y entera, back tracking, y métodos híbridos de estas técnicas, entre otros [1].

Aunque los resultados de las investigaciones han contribuido con modelos rígidos y hay variedad de software comercial que ofrece resolverlo con restricciones simplificadas, no es seguro que la programación obtenida con estos medios sea la mejor; ya que el problema se clasifica de optimización NP-Completo (polinomial no determinista), por lo que no puede resolverse en tiempo polinomial por la complejidad y crecimiento exponencial que implica. Además, pueden surgir cambios en los parámetros dados después que la programación ya haya sido calculada, como un nuevo grupo de curso necesario o la retirada inesperada de un profesor, y corregir estas condiciones en un horario fijo se dificulta por la interdependencia de los cursos.

Las variables y parámetros principales involucradas en el problema son los profesores de diferentes escuelas o departamentos, distintos tipos de asignaturas que deben ser vistas por uno o más programas académicos, múltiples grupos de curso para una misma asignatura pero con diferentes profesores y estudiantes, salones que son requeridos para las sesiones de las asignaturas según el tipo de recursos que ofrecen, planes de estudio de diferentes semestres que comparten asignaturas sugeridas, y una franja de tiempo limitada para que todos los cursos de la institución sean dictados. Esta información es la que requiere la metodología por parte de la administración de la institución para emparejar las sesiones con horarios y salones. También se distinguen dos tipos de restricciones que deben cumplir las asignaciones de los diferentes recursos del horario: duras (de estricto cumplimiento) y suaves (preferenciales pero opcionales).

El método de coloración de grafos (coloración de vértices) es el enfoque principal en el planteamiento de la solución del problema en este trabajo, donde las sesiones de los cursos que deben programarse se representan mediante vértices de un grafo no dirigido, y las restricciones o conflictos que no permiten que dos sesiones de curso puedan ser vistas al mismo tiempo crean una arista entre estos. De esta forma, un color indica un bloque de tiempo específico del horario, y con la ejecución de la coloración del grafo, dos vértices adyacentes son asignados a diferentes colores, de manera que las sesiones de cursos en conflicto sean dictados en diferentes tiempos. Cumpliendo las condiciones fuertemente estrictas (que crean conflictos entre los cursos), e idealmente las restricciones preferenciales pero opcionales (que miden y comparan la satisfacción de los agentes involucrados), mediante la técnica de coloración de grafos en el modelo de las sesiones de los cursos, con una asignación

ordenada de salones de acuerdo con la disponibilidad de su tipología, y reservando un porcentaje de disponibilidad por cada tipo de salón en cada bloque de tiempo, se programa un horario valido y flexible libre de conflictos.

Este documento se estructura así: en la justificación y antecedentes se presenta la importancia del problema tanto para la sociedad como para la comunidad científica, junto con un análisis de investigaciones previas que han empleado enfoques similares para abordarlo. A continuación, se detallan los objetivos y el alcance del proyecto, definiendo las metas a lograr, las restricciones enfrentadas, y el dominio de las variables involucradas. En el marco teórico se desarrollan los conceptos y técnicas necesarias para la comprensión de la metodología, la cual se describe en detalle posteriormente, explicando los pasos seguidos para diseñar la solución propuesta. Los resultados muestran la ejecución y evaluación del algoritmo implementado, mientras que la discusión analiza el modelo, sus ventajas, y la programación obtenida frente a los objetivos planteados, lo que permite, finalmente, formular conclusiones y recomendaciones para futuras investigaciones.

JUSTIFICACIÓN Y ANTECEDENTES

El problema tratado en la investigación no es nuevo, está implícito con el origen de las instituciones educativas y la organización de los horarios de sus cursos ofrecidos, que a través del tiempo y con la extensión de la cobertura educativa, ha sumado factores, relaciones y condiciones que lo han vuelto complejo. La interdependencia entre las variables, como la oferta de asignaturas, disponibilidad de tiempo por cada profesor, las asignaturas en común entre diferentes programas académicos, y los requerimientos de salón por cada asignatura, entre otros; convierten la misión de programar los horarios de los cursos y asignar sus salones, en un proceso arduo que toma tiempo y trabajo colectivo para el acuerdo entre todos los profesores. Además, la poca o nula capacidad de modificar el horario programado cuando ocurren eventos inesperados (como ejemplo, la retirada anticipada de un profesor, o la necesidad de un nuevo grupo de estudiantes, etc.), es un conflicto más que enfrentar debido a la misma naturaleza de la interdependencia entre los cursos.

Aunque la solución actual es obtenida mediante un software comercial, se pueden presentar casos que requieren atención de los coordinadores o directores de escuela, quienes son los responsables de intervenir en la programación con el software. De acuerdo con información de conversaciones con los encargados en la Escuela de las Tecnologías de la Información y las Comunicaciones, tales casos como estudiantes que resultan con un horario diferente al que inscribieron inicialmente, con cursos por ver en un bloque de tiempo común, o con salones que no disponen de los recursos necesarios, o con un profesor diferente al que se planeaba, o cualquier otro imprevisto que perjudica tanto a los estudiantes como a los mismos profesores; son resultado de errores en la inscripción de la información en el proceso de intervención en el software, creando una gran dependencia en el mediador. Desafortunadamente, estos

contratiempos son resultado de la complejidad del problema en sí mismo, cuya solución ofrece restricciones y parámetros demasiado simples y generales que no abordan la naturaleza propia de la universidad, o de la poca flexibilidad que no es capaz de abordar los eventos inesperados.

Los autores Nandal et al. [2] realizan un amplio estudio de la literatura, en la que reflexionan las fortalezas y debilidades de cada artículo revisado, los cuales son los más relevantes en el problema de asignación de la generación de horarios de curso de universidades. En [3] reconocieron la capacidad de generar un horario adecuado libre de conflictos, pero cuya debilidad era la falta de flexibilidad, pues la solución era estricta y lineal en la forma de asignar las sesiones. Otros aportes importantes de su revisión de literatura fue el hecho de ordenar de alguna forma los salones en la etapa de su asignación, o el hecho de ordenar los cursos no por el grado de los vértices que los representan sino por el tipo de salón requerido de los cursos mismos. En su metodología, implementan el enfoque de Welsh Powell, combinando el orden de grado de los vértices con la prioridad según el tipo de salón requerido. La complejidad computacional es de n^2 y espacialmente es de n. Como trabajo futuro, aconsejaron considerar las preferencias de los profesores en la asignación de bloques y tener como restricciones suaves los horarios de actividades de bienestar, como danza o deportes; para una mayor flexibilidad.

Según Avinash, Jain y Kumar [3], después de comparar los tiempos computacionales de las diferentes heurísticas propuestas para resolver la coloración de grafos grandes, el algoritmo NGC (Novel scheme for Graph Coloring) es el mejor entre los demás (Greedy based Coloring, Welsh Powell, Graph Coloring using Backtracking, the Largest Saturation Degree (DSATUR), the Recursive Largest First (RLF)) en cuanto a tiempo y calidad de soluciones obtenidas. La metodología que propusieron fue tener un grafo por cada año (semestre de una carrera especifica en nuestro caso), aplicar la coloración NGC para asignarle un bloque de horario a cada curso (donde los bloques se conciben únicamente por rango de horas. y no consideran la combinación de estos con los días), después agrupa los cursos por el bloque asignado, y en la iteración de cada grupo asigna un salón común (no se distingue tipos de salones). La diferencia entre esta metodología y la nuestra, es que nosotros tenemos tipos de salones que, según los recursos que ofrecen, las sesiones de los cursos requieren un tipo de salón especifico. Además, identificamos un bloque como la combinación de día y rango de horas, mientras que ellos solo los ven como horas. Finalmente, ellos tienen un grafo por cada año (aunque dicen que, si dos años tienen un docente en común, combinan los grafos de esos años en ese caso), mientras que nosotros tenemos un solo grafo para todos los cursos ofrecidos por la institución.

Donderia y Jana [4] proponen un método de coloración de grafo novedoso (NGC) (que fue analizado en el artículo pasado de Avinash, Jain y Kumar), cuyos resultados de colores usados en las soluciones fueron comparables a la cantidad de colores usados

por DSATUR, pero NGC resulto usando menos tiempo de ejecución. La única restricción de la metodología en relación con el contexto de nuestro problema es que la cantidad de colores disponibles para ellos fue igual al número de vértices del grafo, mientras que la cantidad de colores disponibles en nuestro caso es la cantidad de bloques de horario disponibles, que es mucho menor que la cantidad de vértices del grafo del modelo de sesiones de cursos. Siendo así, la metodología no puede ser implementada en nuestro caso.

En su artículo, Brélaz [5] repasa sobre los métodos de coloración de vértices que ya existían en 1979. Comparando las soluciones obtenidas por todos los métodos nombrados (Welsh Powell, Matula, Dunstan, Tehrani, SLI (Smallest Last with Interchanges), DSATUR, Matula-DSATUR, DSI (DSATUR with Interchanges)), en aspectos como el margen de error y el tiempo de ejecución, de manera general; comparten la demostración de superioridad de DSI, Matula-DSATUR y DSATUR, resultando estos en una menor cantidad de colores usados y en menor tiempo. Algo importante es que también destacan la inferioridad del método Welsh Powell en las diferentes comparativas de resultados.

Los autores Poddar y Mondal [6] desarrollan un arduo estudio de la literatura, haciendo un viaje cronológico por cada hito o evento donde hubo un avance importante, fuera en el problema de la coloración de grafos, o en el de la programación de los cursos universitarios. Destacan artículos donde representan el modelo mediante grafos pesados para encontrar una coloración del grafo con costo mínimo, o la incorporación de la asignación de salones durante el proceso de coloración. Este último fue una gran base para la metodología propia de este trabajo, pues se aplica el mismo concepto de asegurar la asignación de salones para las sesiones durante el proceso de coloración. También destacan la maximización de satisfacción de restricciones suaves en el proceso de optimización de la solución usando el mínimo número de colores posible. Sin embargo, la metodología propuesta fue básica para los resultados que presentaron, demostrando la satisfacción de las restricciones duras únicamente, con un caso de prueba pequeño.

En el artículo de Assi, Halawi y Haraty [7], implementan un enfoque genético para calcular el cumplimiento de las restricciones, tanto duras como suaves, donde se les da un ‘penalty’ cuando incumplen una de las restricciones. Sin embargo, este enfoque es diferente a la coloración de grafos que implementa nuestro trabajo, y si bien es una propuesta valida e interesante, para esta investigación nos enfocamos en la aplicación de la teoría de grafos.

Ekanayake et al. [8] comparan la implementación de algoritmos genéticos, de iteración de búsqueda local, de coloración de grafos, y heurísticos, en la solución del problema de programación de horarios de exámenes, concluyendo que entre todas las implementaciones, la del algoritmo genético dio mejores resultados para asignar recursos a los exámenes, y destacaron que la coloración de grafos es apropiada para

identificar los conflictos de manera más clara, asegurando que ninguna asignación de recursos duplicada ocurra. Sin embargo, como el problema de programación de exámenes es naturalmente diferente al problema de programación de horarios, no tiene demasiada relevancia la conclusión sobre el algoritmo genético, para nuestro caso particular de la institución.

En la exhaustiva revisión de literatura que hicieron Zabidee y Adnan [1], encontraron que la mayoría de los estudios usaron información real de instituciones para las pruebas, además de observar que la mayoría de los métodos empleados para el problema de programación de horarios universitarios usaron enfoques híbridos. Finalmente, recomiendan como trabajo futuro, usar modelos de datos estandarizados, para facilitar los enfoques de optimización, haciendo comparaciones más efectivas de las preferencias y requerimientos propias de cada institución.

En la implementación de la coloración Welsh Powell hecha por Bendi, Sunarni y Alfian [9], proponen como pasos metodológicos: recolectar la información del modelo y codificarla, proceso de coloración del grafo, asignar bloque de tiempo, y hacer un análisis de optimización. Sin embargo, no es claro como hacen cada paso, ya que la descripción de cada uno las hace desde la especificación de los resultados, haciendo confuso el desarrollo de cada etapa.

Muklason et al. [10] propusieron una metodología hibrida, donde inicialmente genera una solución factible con Coloración de Grafos, y en una segunda etapa minimizan las restricciones suaves usando el Algoritmo Late Acceptance Hill Climbing. Al final, concluyeron que este último es más optimo en tiempo y en minimización de penalty en comparación con la Coloración de Grafos combinado con el Algoritmo de Hill Climbing. Sin embargo, como el enfoque principal de nuestra investigación es basar todo el algoritmo en la Teoría de Grafos, el algoritmo propuesto en el artículo no resulta relevante para el modelo.

En el artículo clásico de 1967, publicado por Welsh y Powell [11], se propone un algoritmo en el que se ordenan los vértices a colorear de mayor a menor grado, luego se asigna un primer color al vértice de mayor grado, y ese mismo color lo asigna a los demás vértices de mayor grado que no sean adyacentes al primero coloreado; esto lo repite hasta que todos los vértices tengan asignado un color. La idea de ordenar los vértices según el grado de forma descendente propuesta en este artículo fue una base importante para el proceso de coloración en nuestra metodología. De acuerdo con el contexto de nuestro trabajo, modificamos la asignación de colores, siguiendo el mismo orden que el Algoritmo Welsh Powell, pero iterando cada vértice de mayor a menor grado y asignarle el primer color adecuado para él; es decir, no hacemos las asignaciones de un color a los vértices no adyacentes del primero con ese color.

Laguna y Martí [12] proponen un Algoritmo GRASP (Greedy Randomized Adaptive Search Procedure), orientado a la coloración de grafos dispersos (que es el tipo de

grafo del contexto de nuestra problemática). Ellos hacen una revisión literaria sobre algunas técnicas de coloración (entre ellas, la de Welsh Powell), diciendo que esas metodologías revisadas son fáciles y rápido de implementar, pero que a menudo producen coloraciones que están lejos de ser optimas, en cuanto a número de colores usados se refiere. Sin embargo, el objetivo de la coloración requerida en nuestra metodología no es usar la menor cantidad de salones, es decir, menor cantidad de bloques de tiempo, sino por el contrario, usar todos los colores, pero cuya cantidad de asignaciones sea distribuida, por ello convenimos en continuar con el enfoque de ordenamiento por grado usado en el Algoritmo Welsh Powell.

Finalmente, en el año 1996, Baker y Coffman [13] plantean una metodología para hacer una coloración de grafos pesados de asignación de colores limitada, que inicialmente fue pertinente incluir para el proceso de distribuir la asignación de los colores en nuestro caso de estudio. Sin embargo, esta metodología no se encontró tan flexible, ya que busca minimizar la cantidad de asignaciones de cada color teniendo un límite; y aunque tenemos una limitación en el número de asignación de colores, el objetivo no es minimizarlo.

Existe una literatura en el abordaje de la programación de horarios y la asignación de salones en instituciones de educación superior que, mediante diferentes enfoques, han contribuido con metodologías generales y simples que llegan a ofrecer soluciones factibles al problema. Sin embargo, la mayoría se quedan cortas debido a la simpleza de las variables, pues no se tiene en cuenta factores como la tipología de los salones, la presencia de asignaturas transversales en múltiples programas académicos, la distribución de los bloques de tiempo para una repetición semanal de contenido efectiva, y, sobre todo, un carácter flexible que permita modificar el horario fácilmente, después de haber sido calculado.

A través del desarrollo de una metodología que aplica la teoría de coloración de grafos, esta investigación no solo se apoya en estudios previos de gran relevancia, sino que también propone un enfoque específico y adecuado a la estructura administrativa de la Institución Universitaria Politécnico Grancolombiano. Esto garantiza que el algoritmo diseñado refleje las características únicas de la institución, logrando una solución que, además de ser factible, incorpora la flexibilidad como principio central. De esta manera, se busca mejorar la programación y permitir adaptaciones según sea necesario, con un enfoque en la satisfacción de restricciones suaves que faciliten su ajuste posterior sin comprometer la integridad de los horarios. En resumen, este proyecto es clave para mejorar la eficiencia administrativa, responder a las demandas de la comunidad universitaria y contribuir al avance de metodologías flexibles y personalizables en el ámbito de la programación de horarios.

OBJETIVO GENERAL

Diseñar un algoritmo aplicando la coloración de grafos para la asignación eficaz y flexible de horarios y salones para cursos en la Institución Universitaria Politécnico Grancolombiano.

OBJETIVOS ESPECÍFICOS

- Identificar y caracterizar las relaciones entre las variables y parámetros asociados a la programación y ubicación de los cursos.
- Analizar la complejidad computacional y flexibilidad de los resultados de la metodología propuesta.
- Diseñar, ejecutar y analizar experimentos computacionales con datos reales anonimizados de casos pasados en la Institución Universitaria Politécnico Grancolombiano.

ALCANCE

La investigación ofrecerá una metodología que aborde el problema de la programación de horarios de los cursos de las instituciones universitarias, contemplando el aporte de las principales investigaciones pasadas que enfrentaron el problema desde el enfoque de la coloración de grafos, y relacionadas con la tipología y caracterización propia del grafo a construir, basado en la estructura administrativa de la Institución Universitaria Politécnico Grancolombiano. Dicho de otro modo, en base a la literatura se considerarán las técnicas que han demostrado el mejor desempeño para obtener una solución factible, además de métodos que son posibles adaptar al diseño del algoritmo para tratar cada una de las variables y parámetros como sea necesario. Todo lo anterior tendrá como criterio principal la flexibilidad para modificar el horario después de haber sido calculado, y por ende la satisfacción de restricciones suaves.

Los escenarios e información de entrada de los experimentos computacionales serán tomados de la programación oficial de horarios y salones junto con la oferta de las sesiones de los cursos de la Facultad de Ingeniería, Diseño e Innovación de la Institución Universitaria Politécnico Grancolombiano en el periodo 2023-10, donde los datos sensibles que pueden contener información privada o personal de algún sujeto serán anonimizados por el asesor del proyecto quien también dirige el Semillero de Teoría de Grafos, y por lo tanto tendría la autorización y el acceso a la documentación de esta información.

Ya que las técnicas obtenidas de la literatura plantean diferentes ordenes en el proceso de la coloración de grafos, sumado a la naturaleza de la categorización del problema NP-Completo, se considerará el algoritmo propuesto no determinista debido al enfoque heurístico implícito, pudiendo generar diferentes resultados para una misma entrada en la que, sin importar el criterio de orden planteado en la metodología, al presentarse el caso de encontrar más de un elemento en el mismo nivel de ordenamiento que

determina el siguiente vértice a colorear, procesar alguno primero que otro altera el orden de manera distinta que cualquier otro del mismo nivel.

El criterio para evaluar la factibilidad del algoritmo se basará en que el tiempo de ejecución no sobrepase el espacio que existe entre el momento en el que se obtiene toda la información requerida por el algoritmo, dada por la administración, y el plazo final previo a la inscripción de asignaturas, obviando el hecho de que la programación resultante fuera lógicamente factible.

Por otro lado, para evaluar la flexibilidad del resultado, se tendrá en cuenta la medida de satisfacción de las restricciones suaves que cubre la metodología propuesta, tales como la disponibilidad de salones libres por cada bloque del horario y la asignación de salones en el mismo campus para pares de sesiones consecutivas que tengan conflictos.

Finalmente, cabe recalcar que la función del algoritmo será programar bloques del horario y asignar salones a un conjunto de sesiones de cursos inicialmente documentados por la administración de la institución; la organización y asignación de profesores y bloques de tiempo requerido por cada sesión de curso no entran en el alcance del proyecto.

MARCO TEÓRICO

En este apartado se explica la definición de los conceptos esenciales de Teoría de Grafos y el problema teórico de la Coloración de Grafos, que fueron necesarios para el desarrollo del trabajo y la solución del problema, haciendo énfasis en su relevancia y aplicación dentro de la investigación.

GRAFO

Un grafo puede definirse como una estructura compuesta por vértices y aristas, utilizada para modelar relaciones entre objetos. Se denota mediante $G = (V, E)$, donde se denomina V el conjunto finito no vacío de vértices de G, y se denomina E el conjunto de aristas dirigidas de G [14, p. 514]. Esta estructura se acopla al modelo de las sesiones de los cursos, pues cada sesión se puede representar mediante un vértice, y el conflicto entre dos sesiones que evite que estas no puedan ser vistas al mismo tiempo o espacio, se puede representar mediante una arista entre ambos vértices.

DIRECCIÓN DE ARISTAS

Un grafo es dirigido si las aristas pueden ser atravesadas únicamente en una dirección, en cuyo caso, E es un conjunto de pares ordenados de elementos tomados de V. Cuando las aristas pueden ser atravesadas en ambos sentidos, E es un conjunto de pares desordenados de elementos tomados de V, y el grafo se denomina no dirigido [15, p. 110]. Para el modelo del grafo de sesiones de cursos, dado que las aristas simplemente representan conflictos, donde el orden entre los vértices en conflicto no lo altera, no es necesario que las aristas tengan dirección, por lo que el grafo del modelo es no dirigido.

TRAYECTORIA (WALK)

Sea x, y vértices (no necesariamente distintos) en un grafo G. Una trayectoria $x - y$ es una secuencia finita

$$x = x_0, e_1, x_1, e_2, \dots, e_{n-1}, x_{n-1}, e_n, x_n = y$$

de vértices y aristas de G, empezando en el vértice x y terminando en él y, involucrando las n aristas $e_i = \{x_{i-1}, x_i\}$, donde $1 \leq i \leq n$ [14, p. 515].

La longitud de la trayectoria es n, el número de aristas en ella.

Cualquier trayectoria $x - y$ donde $(n > 1) \wedge (x = y)$ se denomina trayectoria cerrada, de otro modo se denomina trayectoria abierta.

Este concepto es fundamental para la comprensión de los demás conceptos que continúan y que están involucrados en la teoría de la coloración de grafos.

CAMINO (PATH)

Si ningún vértice del camino $x - y$ aparece más de una vez, entonces el camino se denomina un camino $x - y$. Cuando $x = y$, el concepto ciclo es usado para describir dicho camino como cerrado [14, p. 516].

CONECTIVIDAD DE GRAFO

Sea $G = (V, E)$ un grafo, dirigido o no, se denomina conectado si hay un camino entre cualquier par de vértices distintos de G [14, p. 517]. Para el modelo de sesiones de cursos, el grafo representativo no es necesariamente conectado, ya que existe la posibilidad de haber una sesión que no tenga conflictos con ninguna otra (como es el caso de los cursos electivos) y entonces los vértices que representan esos cursos no tendrían aristas con ningún otro vértice.

GRADO Y ADYACENCIA DE VÉRTICES

Sea G un grafo no dirigido. Por cada vértice v de G, el grado de v es el número de aristas en G que inciden con v [14, p. 530]. Cada vértice que está conectado a v mediante una arista, se denomina adyacente de v. En otras palabras, podemos definir el grado de v como la cantidad de vértices adyacentes a él [15, p. 111]. Estos dos conceptos fueron fundamentales en el desarrollo de la metodología, ya que se propuso un ordenamiento de los vértices de forma descendente según el grado de estos en el proceso de coloración; y la evaluación de la mayoría de las condiciones relacionadas a los conflictos entre los vértices, consistió en la iteración de la búsqueda de adyacencia en otras estructuras de datos que los contenían.

REPRESENTACIÓN COMPUTACIONAL

En la implementación de algoritmos de grafos se pueden usar principalmente dos estructuras diferentes, llamadas matrices de adyacencia y listas de adyacencia, que son definidas a continuación.

MATRIZ DE ADYACENCIA

Dado un grafo $G = (V, E)$, una matriz de adyacencia es una matriz $A_{n\mathrm{X}n}$ donde $A_{ij} = 1$ sí y solo si los vértices v_i y v_j son adyacentes, y $A_{ij} = 0$ en el caso contrario [16, p. 24]. Aunque esta estructura es eficiente para revisar si dos vértices son adyacentes, para los grafos que son dispersos (que son aquellos en los que el número de aristas es lejano al número máximo de aristas posibles), la mayoría de los valores en la matriz serian igual a cero, al costo espacial de n^2 elementos, y donde la búsqueda de vértices adyacentes costaría n iteraciones. Teniendo en cuenta que el grafo del modelo de sesiones de cursos es disperso, esta estructura no es adecuada.

LISTA DE ADYACENCIA

Dado un grafo $G = (V, E)$, una lista de adyacencia es un vector de tamaño n, donde cada elemento adj_v corresponde a la lista que contiene todos los vértices adyacentes del vértice v, para todo $v \in V$ [16, p. 23]. Ya que la operación de buscar los vértices adyacentes de un vértice especifico es frecuente en la metodología propuesta, esta estructura es la adecuada para ello.

PROBLEMA DE COLORACIÓN

De acuerdo con Lewis [16, p. 9], sea $G = (V, E)$ un grafo, que consiste en un conjunto V de n vértices, y un conjunto E de m aristas. Dado dicho grafo, el problema de coloración de grafos buscar asignar a cada vértice $v \in V$ un entero $c(v) \in \{1,2, \dots, k\}$, donde cada número de 1 a k representa un color, tal que:

- $c(v) \neq c(u) \forall \{v, u\} \in E$, es decir, que los vértices adyacentes en G deben ser asignados a colores diferentes; y
- k es el mínimo, es decir, que se use la menor cantidad de colores posibles.

Este último término k se denomina el número cromático de G y se escribe $\chi(G)$ [14, p. 565].

Teniendo en cuenta que en el modelo de sesiones de cursos los vértices adyacentes son las sesiones que tienen conflictos, y por ende deben ser asignadas a diferentes bloques de horario; el problema de coloración es aplicable al interpretar los colores como esos bloques de horario, evitar que cualquier par de vértices adyacentes sean asignados al mismo color, evita que cualquier par de sesiones con conflictos sean asignadas en el mismo bloque de horario.

PROPIEDADES DE COLORACIÓN

Lewis define las siguientes definiciones útiles que describen una solución de coloración de grafos y sus propiedades [16, p. 10].

- Una coloración de grafo es denominada completa si todos los vértices $v \in V$ son asignados a un color $c(v) \in \{1, \dots, k\}$; en caso contrario la coloración es considerada parcial. Los resultados de la metodología deben dar una coloración completa del grafo de sesiones, ya que eso significa que todas las sesiones de curso fueron asignadas con un horario.
- Un choque (o Clash en inglés) describe una situación donde un par de vértices adyacentes $u, v \in V$ son asignados al mismo color (es decir, $\{u, v\} \in E$ y $c(u) = c(v)$). Si una coloración no presenta choques, entonces se considera correcta; en caso contrario es considerada incorrecta. Evidentemente, la coloración resultante de la metodología debe ser correcta para que no haya cursos en conflicto asignados al mismo bloque de tiempo.
- Una coloración es factible si y solo si es tanto completa como correcta.

- El número cromático de un grafo G, denotado por $\chi(G)$, es el mínimo número de colores requerido en una coloración factible de G. Una coloración factible de G que use exactamente $\chi(G)$ colores se considera optima. Como la cantidad de colores en el modelo de sesiones equivale a la cantidad de bloques de horario disponibles en el contexto de la institución, el objetivo de la metodología no es obtener una coloración optima (lo cual significaría usar menos bloques de horario); sino por el contrario, usar todos los bloques de horario disponibles, pero de forma distribuida, tal que las cantidades de vértices asignados a cada color sean cercanas.
- Una clase de color es un conjunto que contiene todos los vértices de un grafo que son asignados a un color particular en una solución. Es decir, dado un color particular $i \in \{i, \dots, k\}$, una clase de color es definida como el conjunto $\{v \in V : c(v) = i\}$. En el modelo de sesiones de curso, el concepto de clase de color representa las sesiones que son asignadas a un bloque de horario específico.
- Un conjunto independiente es un subconjunto de vértices $I \subseteq V$ que son mutuamente no adyacentes. Es decir, $\forall u, v \in I, \{u, v\} \notin E$. En otras palabras, un conjunto independiente en el modelo de sesiones de cursos puede contener clases que no tienen conflictos entre sí, como, por ejemplo, una clase de color, que representaría las sesiones que fueron asignadas a un bloque de tiempo en común.
- Un clique es un subconjunto de vértices $C \subseteq V$ que son mutualmente adyacentes: $\forall u, v \in C, \{u, v\} \in E$. En otras palabras, un clique en el modelo de sesiones de cursos puede contener sesiones que tienen conflictos entre sí.
- Todos los vértices deberían ser asignados exactamente a una clase de color cada uno, de manera que la intersección entre dos clases de colores sea el conjunto vacío [16, p. 11]. Sin embargo, dado que, en el modelo de sesiones de cursos, cada sesión puede requerir diferente cantidad de bloques de tiempo, cada vértice puede requerir diferente cantidad de clases de colores asignadas, por lo que fue pertinente considerar la asignación múltiple de clases de colores a un mismo vértice. Respetando que los vértices adyacentes no tengan clases de colores asignadas en común, y que todos los vértices tengan al menos una clase de color asignada, la coloración resultante de la metodología sigue considerándose factible.

COMPLEJIDAD COMPUTACIONAL (NP COMPLETO)

Considerando el sentido estricto de resolver el problema de coloración como el hecho de tomar cualquier grafo (de cualquier tamaño y tipología) y retornar una solución óptima para todos los casos (es decir, de manera determinista); hallar la coloración más optima entre el conjunto de todas las posibles coloraciones ocuparía n^n soluciones candidatas, teniendo en cuenta que el número de vértices es n, y que el número

máximo de colores a asignar es igual al número de vértices, es decir, que habría n opciones de color por cada n vértices. Aunque este número de diferentes soluciones podría reducirse al representar las soluciones como particiones, los números de Stirling (con el cual se calcula la tasa de crecimiento de las particiones respecto a n) exhiben tasas de crecimiento exponencial para la mayoría de los valores. La razón por la que el problema de coloración de grafo se considera intratable, se debe a que se considera un problema de satisfactibilidad, que, de acuerdo con Stephen Cook, es un problema NP-completo, lo cual implica que no se conoce ningún algoritmo polinomialmente acotado para lograr esta tarea [16, p. 15]. Por lo que no sería correcto resolver el problema en el sentido estricto de encontrar su solución óptima, sino de aproximarse mediante algoritmos o métodos heurísticos a soluciones aceptables o factibles.

METODOLOGÍA

En este apartado se propone cada una de las etapas para el desarrollo de la solución, iniciando por la caracterización de los parámetros, donde se identifican cada una de las entidades planteadas, sus relaciones e importancia dentro del sistema del modelo; luego se define cada una de las restricciones que cubre el modelo según su tipo; y en base a la caracterización de los parámetros, se nombra cada uno de los atributos que el modelo necesita obtener para generar las estructuras de datos necesarias; y así ejecutar el algoritmo donde se hace la coloración del grafo creado (que representa la asignación de bloques de horario a las sesiones de cursos), distinguiendo las sesiones según la cantidad de bloques de tiempo que requieren y el orden basado en el grado del vértice; para finalizar con la asignación de salones, priorizando aquellas sesiones cuyo tipo de salón requerido solo están disponibles en un campus, y luego procesar los que no requieren de un campus especifico, en base a la asignación de vértices adyacentes de bloques en un mismo día. De esta forma, se programan los horarios y salones del modelo de las sesiones de cursos de forma factible y flexible, satisfaciendo todas las restricciones duras, y aproximando la satisfacción de las suaves.

CARACTERIZACIÓN DE PARÁMETROS

El sistema de información sobre el que fue basada la asignación de horarios y salones de sesión para los cursos ofrecidos por la Institución Universitaria Politécnico Grancolombiano fue concebido mediante un modelo de base de datos relacional con once tablas, cuyo significado y trascendencia se explica a continuación.

PROGRAM (PROGRAMA ACADÉMICO)

Representa el nombre de cada programa académico (técnico, tecnólogo, profesional, especialización, maestría, o doctorado) especifico que ofrece la institución. Por ejemplo, Ingeniería de Sistemas.

CAMPUS

Representa cada campus que contiene los salones disponibles para ser asignados a la población de estudio. Como la población de estudio son las sesiones ofrecidas en la sede de Bogotá, los campus de la institución en esta ciudad son el City Campus y el Campus Principal.

BUILDING (EDIFICIO)

Representa cada edificio que hay en cada uno de los campus. Si uno de los campus es un edificio en sí mismo (como el caso de City Campus), entonces se nombran igual, pero es obligatorio que cada edificio sea parte de un campus, ya que este último es esencial para evaluar las restricciones espaciales. Un ejemplo podría ser el edificio H que está en el Campus Principal.

ROOM TYPE (TIPO DE SALÓN)

Representa cada tipo de salón que dispone la institución. Cada uno se diferencia según el tipo de recursos, infraestructura, o modalidad que tenga, pero esencialmente se distinguen por un nombre, y algo muy importante, el valor decimal del porcentaje de ocupación máxima. Por ejemplo, dos tipos de salones podrían ser ‘Laboratorio de Física’ (con porcentaje de ocupación máxima igual a 0.5, equivalente al 50%) y ‘Dibujo’ (con porcentaje de ocupación máxima igual a 0.85, equivalente al 85%).

ROOM (SALÓN)

Representa cada salón especifico que dispone la universidad para la población de estudio. Son aquellos que, en base a su tipo y aforo, son asignados a las sesiones en la organización del horario.

Además, teniendo en cuenta el edificio y, por ende, campus al que pertenece, se restringe la asignación de salones para bloques de tiempo continuos, en casos donde no se deben dar dos sesiones consecutivas en diferentes campus. Un ejemplo de salón podría ser el 102 del edificio C del Campus Principal, que es de tipo ‘Computadores’ y su aforo es de 40 personas.

SUBJECT (ASIGNATURA)

Representa cada asignatura especifica que puede ser vista por alguno de los programas académicos en algún semestre. Para la metodología, solo se caracterizó por su nombre y un código que la identifica, por ejemplo ‘Opción de Grado [ISI]’ con código ‘TIC50000’.

CURRICULUM (PLAN DE ESTUDIOS)

Representa la sugerencia de una asignatura para ser vista para un programa académico específico en un semestre específico. Como el plan de estudios de cada programa específico sugiere un orden para inscribir sus asignaturas; entonces, por ejemplo, si se sugiere ver las asignaturas de ‘Matemáticas’, ‘Introducción a las Ingenierías’, ‘Create Camps’, y ‘Cultura Ambiental’ en el primer semestre de Ingeniería de Sistemas, entonces habría un registro por cada una de las cuatro asignaturas mencionadas relacionadas a ese programa académico y en ese semestre específico. Si bien, el orden planteado en el plan de estudios no es fijo ya que el estudiante es libre de elegir que asignaturas cursa desde el segundo semestre, las relaciones de las asignaturas con el programa académico y el semestre suponen los conflictos necesarios para generar las aristas en la creación del grafo.

SESSION (SESIÓN)

Representa los requerimientos de cada una de las sesiones semanales que hacen parte de una asignatura. Estos requerimientos básicamente son el tipo de salón en el

que se debe dar la sesión, y la cantidad preferencial de bloques de tiempo seguidos que debe durar la sesión. Cabe aclarar varias cosas: primero, la cantidad de bloques de tiempo seguidos requerido puede ser mínimo 1 y máximo 2; segundo, esta cantidad de bloques de tiempo seguidos se dice preferencial porque si no se logra asignar la sesión en dos bloques de tiempo seguidos, el registro de la sesión se convertiría en dos sesiones con los mismos valores, pero cuya cantidad de bloques de tiempo requeridos seria 1, es decir, la sesión se daría en bloques de tiempo no consecutivos; tercero, los registros de esta tabla son las sesiones a dictar semanalmente en una asignatura, y como cada una de las sesiones puede requerir diferentes recursos, se justifica cada uno de los requerimientos solicitados. Entonces, por ejemplo, un registro de sesión podría ser de la asignatura de 'Física Mecánica', que requeriría 2 bloques de tiempo consecutivos idealmente, y se debería ver en un salón de tipo 'Común'; mientras que otro registro de sesión podría ser de la misma asignatura, pero que requeriría 1 bloque de tiempo, y el tipo de salón a usar seria 'Laboratorio de Física'.

COURSE (GRUPO DE CURSO)

Representa los diferentes grupos de curso que pueden ofrecerse por cada asignatura. Sin embargo, de forma general, un curso no necesariamente está relacionado a una sola asignatura, sino que, en el contexto de la institución del caso de estudio, se crea un grupo para cada curso de las asignaturas ofrecidas. En otras palabras, esto no restringe el caso de las instituciones en las que hay grupos fijos de estudiantes que deben ver las mismas asignaturas obligatoriamente. Cada grupo de curso está constituido por un código identificador, un nombre, y un tamaño que significa la cantidad máxima de estudiantes que podrían entrar al grupo. Un ejemplo de grupo de curso podría tener '11011' como código, con nombre '108', y un tamaño de máximo 36 estudiantes.

TEACHER (PROFESOR/A)

Representa cada uno de las y los profesores que pueden dictar los diferentes cursos ofrecidos por la institución. Como solo se caracterizan por el nombre, 'Manuel Chauta' podría ser un ejemplo.

COURSE – SESSION – TEACHER (GRUPO DE CURSO – SESIÓN – PROFESOR/A)

Si la tabla Sesión representaba los requerimientos de las sesiones de una asignatura, esta tabla representa la combinación de lo anterior con un profesor y con un curso especifico, es decir, los registros de esta tabla representan cada una de las sesiones que deben ser asignadas a un horario y salón. Es en base a los registros de esta tabla que se crea el grafo, ya que cada una de las combinaciones grupo de curso – sesión – profesor representan un vértice, porque contiene toda la información necesaria de las sesiones para evaluar las restricciones y condiciones que generan las aristas, programan los bloques de tiempo, y asignan los salones.

RESTRICCIONES

Hay dos tipos de restricciones que necesitan ser cumplidas por la programación de horarios: restricciones duras y restricciones suaves. Las restricciones que son obligatorias para que el algoritmo las satisfaga se denominan restricciones duras, mientras que las restricciones que no son obligatorias pero que hacen que el cronograma sea óptimo y factible se denominan restricciones blandas [2].

RESTRICCIONES DURAS

Temporales

- Una sesión solo puede requerir uno o dos bloques de tiempo.
- Dos sesiones no pueden verse en el mismo bloque de tiempo si:
    - Son dictadas por el mismo profesor.
    - Sus asignaturas son del mismo semestre sugerido en el mismo programa académico.
    - Son del mismo grupo de curso.

Espaciales

- Un salón no puede ser asignado a más de una sesión en un bloque de tiempo específico.

RESTRICCIONES SUAVES

Temporales

- Una sesión que requiere dos bloques de tiempo debería estar en dos bloques del horario consecutivos en un mismo día.

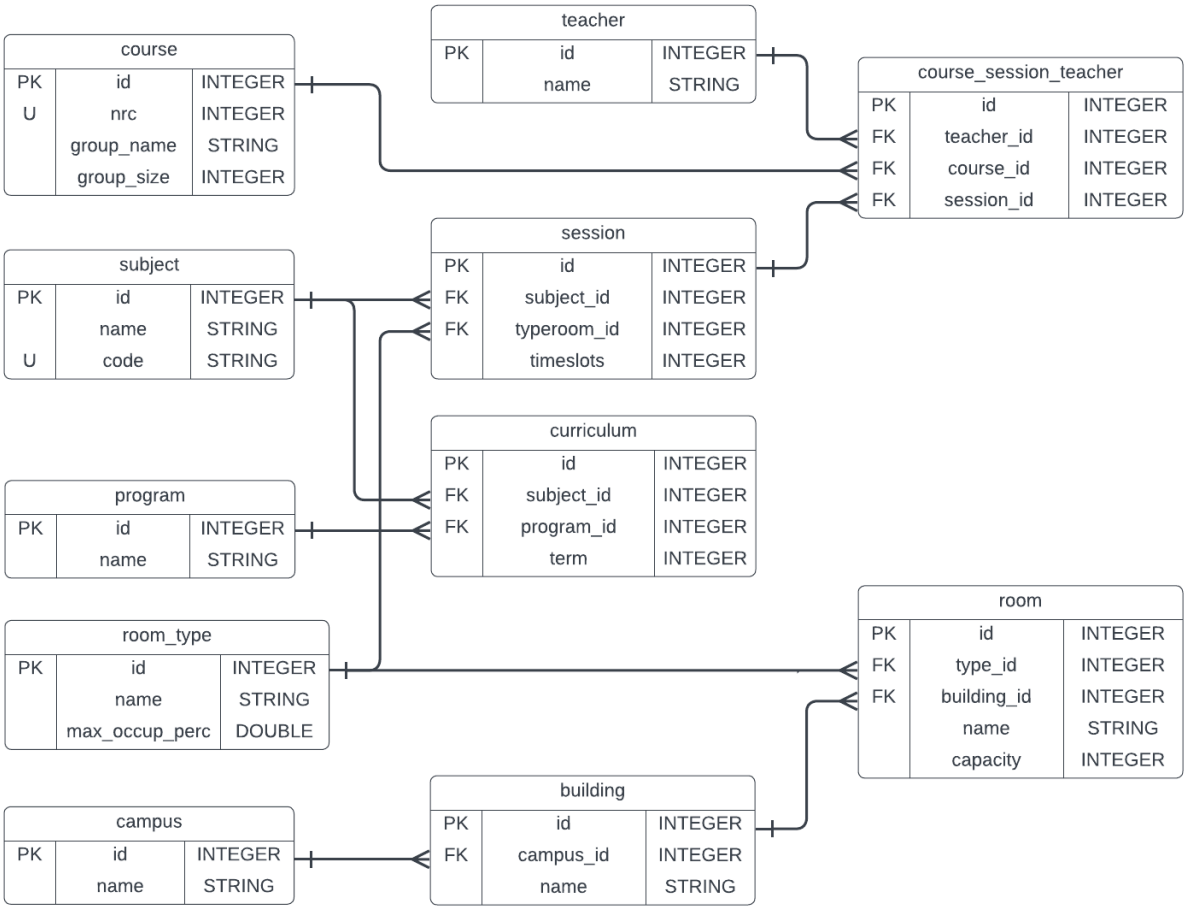
Espaciales

- Dos sesiones asignadas a bloques consecutivos no deberían ser asignadas a salones cuyos edificios sean de diferentes campus, si:
    - Son dictadas por el mismo profesor.
    - Sus asignaturas son del mismo semestre sugerido en el mismo programa académico.
    - Son del mismo grupo de curso.
- Una sesión que requiere dos bloques de tiempo debería ser asignada al mismo salón.

OBTENCIÓN DE LA INFORMACIÓN

Todos los atributos requeridos son importados de forma masiva para llenar la base de datos desde la cual se crearán las estructuras de datos necesarias para la implementación del algoritmo. Las entidades de la base de datos se muestran en la siguiente imagen.

Figura 1. Diagrama de Entidades



Fuente: Elaboración Propia.

SESIONES POR CURSOS Y PROFESORES

El director o representante de cada facultad proporciona la información de las sesiones de los cursos de su escuela. Esta información consta de los siguientes datos:

- Código de la asignatura de la sesión.
- Nombre del tipo de salón requerido por la sesión.
- Numero de bloques de tiempo de la sesión.
- Nombre del/a profesor/a que dicta la sesión del curso.
- Email del/a profesor/a que dicta la sesión del curso.
- Nombre del grupo del curso.
- Tamaño del grupo del curso.
- NRC del grupo de curso.

SALONES DISPONIBLES

Se debe dar la información de los salones que dispone la universidad. Teniendo en cuenta que cada salón es diferente, los datos que componen cada salón son los siguientes:

- Tipo de salón.
- Porcentaje de ocupación máxima del tipo de salón.
- Nombre del edificio del salón.
- Campus del edificio del salón.
- Nombre del salón.
- Aforo del salón.

PLAN DE ESTUDIOS DE PROGRAMAS ACADÉMICOS

Se debe dar la información del plan de estudios sugerido de cada programa académico, que consta de las asignaturas planeadas por inscribir en cada periodo de cada programa, es decir, de:

- Nombre del programa académico del plan de estudios.
- Periodo especifico del programa académico del plan de estudios.
- Asignaturas sugeridas en el periodo especifico del programa académico del plan de estudios.

OTRAS VARIABLES

La cantidad de días hábiles para el horario y la cantidad de bloques de tiempo por día deben ser inicializados para ordenar la asignación de colores del algoritmo.

Cada tipo de salón deberá tener un porcentaje de ocupación máxima dada en forma decimal entre 0 y 1 ($0 \leq \max\ \_occup\_perc \leq 1$). El objetivo de estos porcentajes es no usar la disponibilidad máxima de cada tipo de salón en todos los bloques de tiempo, sino que, de acuerdo con ese porcentaje de ocupación máxima, se calculan la cantidad entera de salones que se pueden ocupar, reservando un porcentaje restante de espacios, para que puedan ser usados de forma flexible después de la ejecución del algoritmo. En otras palabras, el margen entre la disponibilidad total y la ocupación máxima de salones, permite guardar “salones de reserva” en cada bloque de tiempo, que pueden ser usados para reasignar sesiones cuyo tipo de salón haya ocupado su máxima disponibilidad en el bloque de tiempo que fue programado inicialmente, o incluso para asignar nuevas sesiones de cursos que pueden surgir después de haber calculado el horario final (o cualquier otra condición posterior a la obtención de la programación), ofreciendo flexibilidad a la solución. Esto también soporta la distribución uniforme de las sesiones a través de los bloques de todo el horario.

GENERACIÓN DE LAS ESTRUCTURAS DE DATOS

A continuación, se nombran las estructuras de datos que se usaran para el algoritmo, que se llenaran con la información almacenada en la base de datos.

VARIABLES DE CANTIDAD

El número total de vértices (sesiones de curso) se denomina n.

El número total de tipos de salón se denomina nrt.

El número total de colores (bloques de horario) se denomina nc.

El número total de salones se denomina nr.

LISTA DE ADYACENCIA DE LAS SESIONES

Cada vértice se nombra con el id de los registros de la tabla ‘course_session_teacher’, que representa la combinación de las sesiones de curso dadas por un profesor especifico. Con este id, se pueden obtener todos los demás atributos necesarios para evaluar las restricciones para construir las aristas entre los vértices. Para el proceso de la creación de las aristas:

1. Se crea una lista de adyacencia de tamaño N, donde N es la cantidad de vértices.
2. Se iteran todos los profesores, y se hace una consulta de los registros de la tabla ‘course_session_teacher’ cuyo profesor asignado sea el mismo profesor iterado.
    a. Itera cada registro obtenido, y por cada otro, lo agrega a la lista de adyacencia en el índice de ese registro obtenido.
3. Se iteran todas las asignaturas sugeridas para un mismo programa académico y semestre especifico, y se hace una consulta de los registros de la tabla ‘course_session_teacher’ cuyo programa académico y semestre especifico sugerido sea el mismo que los de la asignatura iterada.
    a. Itera cada registro obtenido, y por cada otro, lo agrega a la lista de adyacencia en el índice de ese registro obtenido.
4. Se iteran todos los grupos de curso, y se hace una consulta de los registros de la tabla ‘course_session_teacher’ cuyo grupo de curso sea el mismo grupo de curso iterado.
    a. Itera cada registro obtenido, y por cada otro, lo agrega a la lista de adyacencia en el índice de ese registro obtenido.

Habiendo evaluado las tres posibles restricciones duras que generan conflictos entre las sesiones, todas las aristas del grafo habrán quedado representadas en la lista de adyacencia, que será representada como un arreglo de conjuntos de enteros de tamaño n.

ARREGLO DE ÚLTIMO COLOR CON DISPONIBILIDAD POR TIPO DE SALÓN

Este arreglo de enteros será del tamaño nrt, contendrá el color actual con disponibilidad por cada tipo de salón. Inicialmente, todos los valores serán igual a 1, representando que todos los tipos de salón tienen disponibilidad en el primer color.

MATRIZ DE OCUPACIÓN ACTUAL DE TIPOS DE SALÓN POR COLOR

Este matriz de enteros será del tamaño nc por nrt, contendrá la cantidad de asignaciones que tiene un tipo de salón en un color. Inicialmente, todos los valores serán igual a 0 por defecto, representando que no se ha hecho ninguna asignación a ningún tipo de salón en ningún color.

ARREGLO DE PORCENTAJES DE OCUPACIÓN MAXIMA POR TIPO DE SALÓN

Este arreglo de decimales de tamaño nrt representara el campo de 'max_occup_perc' que tienen los tipos de salón en la base de datos.

ARREGLO DE DISPONIBILIDAD POR TIPO DE SALÓN

Este arreglo de enteros de tamaño nrt contendrá la cantidad total de salones que dispone cada tipo de salón.

ARREGLO DE OCUPACIÓN MAXIMA POR TIPO DE SALÓN

Este arreglo de enteros de tamaño nrt tendrá la máxima cantidad de salones que pueden ser ocupados por cada tipo de salón, lo cual equivale a la cantidad de asignaciones que puede ser asignado un color para un tipo de salón. Sea tr_i un tipo de salón cualquiera, se calcula como el número techo del producto entre el porcentaje de ocupación máxima de tr_i y la disponibilidad de tr_i.

ARREGLO DE CLASES DE COLORES

Mediante un arreglo de vectores de enteros de tamaño nc, cada índice del arreglo representa un color, y su vector contendrá los vértices que fueron asignados a ese color.

MATRIZ DE COLORES ASIGNADOS A VÉRTICES

Mediante una matriz de enteros tamaño n por 2, se contendrá un arreglo de tamaño 2 por cada vértice, y los valores de ese arreglo serán los colores que tiene asignados, teniendo en cuenta que la asignación múltiple de colores a vértices será efectuada por el algoritmo.

ARREGLO DE CAMPUS OBLIGATORIOS POR VÉRTICE

Este arreglo de enteros de tamaño n contendrá el id del único campus que dispone el tipo de salón que requiere cada vértice. Inicialmente, los valores de este arreglo serán 0 por defecto, representando que cualquier campus (teniendo en cuenta que solo hay

dos campus, y ambos son físicos) dispone del tipo de salón que requiere el vértice. En la etapa de asignación de salones, este arreglo será completado con los id correspondientes.

ARREGLO DE SALONES POR TIPO DE SALÓN

Mediante un arreglo de vectores de enteros de tamaño nrt, cada índice del arreglo representará un tipo de salón, y su valor será un vector con todos los salones que son de ese tipo.

ORDEN DE LOS BLOQUES DE TIEMPO

Los bloques de tiempo del horario se ordenarán de forma intercalada por día y luego por bloque. Es decir, se recorre los bloques del horario por un bloque de tiempo específico, cambiando cada día intermedio, y después la iteración continua desde el siguiente bloque de tiempo intermedio, para volver a recorrer por día intermedio. Por ejemplo, para una franja horaria de 6 bloques y 5 días a la semana, este sería el orden en que se iterarían los bloques de tiempo.

Tabla 1. Distribución de Bloques de Horario

	D1	D2	D3	D4	D5
B1	1	10	2	11	3
B2	16	25	17	26	18
B3	4	12	5	13	6
B4	19	27	20	28	21
B5	7	14	8	15	9
B6	22	29	23	30	24

Fuente: Elaboración Propia.

Este orden permite iniciar la asignación de sesiones cuya cantidad de bloques de tiempo requerida es 2, pues cuando se le asigna el bloque actual con disponibilidad para su tipo de salón requerido, también se le asigna el siguiente bloque, pero en el mismo día, que equivale a sumarle 15. Los bloques de 1 a 15; ya que desde el bloque 16, no quedan bloques consecutivos con disponibilidad asegurada en el mismo día.

ORDEN EN LA ASIGNACIÓN DE SALONES

Para la asignación de los salones, se obtienen primero los id de los campus para las sesiones cuyo tipo de salón solo está disponible en un campus. Esas sesiones serán las primeras en ser asignadas, ya que no tienen más opción de campus. Cuando se vayan a asignar las sesiones cuyo tipo de salón requerido están disponibles en más de

un campus, se revisa si tienen vértices adyacentes en los bloques de tiempo consecutivos (antes y después), y en cuyo caso tengan un solo campus, se les asigna un salón de ese mismo campus. De esta manera, se aproxima la satisfacción de la asignación de salones en el mismo campus para sesiones que tengan mismo profesor, sugerencia de plan de estudios, o grupo de curso.

ALGORITMO

CREACIÓN DE VARIABLES

La definición de que significa cada variable y como se completa o asigna, está en el apartado de generación de estructuras de datos. Aquí se muestra el código de cada estructura para tener una idea de cómo podría programarse en los diferentes lenguajes de programación.

Figura 2. Variables Usadas en el Algoritmo

1: $n : int \leftarrow$ numero total de vertices
2: $nrt : int \leftarrow$ numero de tipos de salon
3: $nc : int \leftarrow$ numero de colores
4: $nr : int \leftarrow$ numero de salones
5: $g : set < int > [n] \leftarrow$ lista de adyacencia
6: $last\_color\_ts : int[nrt] \leftarrow$ color actual con disponibilidad por tipo de salon
7: $occup : int[nc][nrt] \leftarrow$ ocupacion actual de tipos de salon por cada color
8: $mop : double[nrt] \leftarrow$ porcentajes de maxima ocupacion por tipo de salon
9: $total\_disp : int[nrt] \leftarrow$ cantidad de salones por tipo de salon
10: $max\_occup : int[nrt] \leftarrow$ cantidad maxima de ocupacion por tipo de salon
11: $c\_vertices : vector < int > [nc] \leftarrow$ clases de colores
12: $v\_colors : int[n][2] \leftarrow$ colores de cada vertiece
13: $v\_campus : int[n] \leftarrow$ campus obligatorio de cada vertice

Fuente: Elaboración Propia.

COLORACIÓN DE VÉRTICES DE UN MISMO NÚMERO DE BLOQUES DE TIEMPO

Esta función recibe como parámetros una cola de enteros, que representan vértices sin colorear denominada Q, y la cantidad de bloques de tiempo requeridos por esos vértices, denominada nb. En la primera línea se crea una cola de vértices huérfanos denominada qh, que contendrá aquellos vértices que no pudieron ser coloreados por el algoritmo. Mientras que la cola Q tenga elementos, se obtiene el siguiente vértice de esta denominado vi. Luego se obtiene el tipo de salón requerido por ese vértice denominado tsr. Para conocer que color empezar a usar para ese vértice vi, se obtiene del arreglo $last\_color\_ts$ en el índice vi, denominado uc. Se crea una bandera que será marcada como verdadera cuando no haya colores disponibles para vi, denominada $b1$. En el bucle que inicia en la línea 7, se crea otra bandera que será marcada como

verdadera en caso de encontrar un vértice adyacente de vi asignado a uc, denominada $b2$. Se revisa si nb es 2 (es decir, si Q contiene los vértices que requieren dos bloques de tiempo consecutivos), y si uc es mayor que 15 (pues en el orden de los bloques de horario, 15 es el último bloque desde el que se puede asignar dos bloques de tiempo); de ser así, se crean dos copias de vi, denominadas $vb1$ y $vb2$, pero se les cambia el número de bloques de tiempo requeridas a 1 (para intentar se coloreados en bloques de tiempo no consecutivos en una próxima ejecución de coloración con la cola de vértices que requieren solo un bloque de tiempo), y se agregan a qh. En caso de que nb no sea 2 (es decir, no se están coloreando vértices que requieren dos bloques de tiempo consecutivos), o que uc no sea 15; itera los vértices coloreados por uc, denominado cada uno como vcu, y en caso tal que vcu sea adyacente a vi, se marca $b2$ como verdadera y se deja de iterar los vértices coloreados por uc. Después de esto, si $b2$ está marcada como verdadera (es decir, en caso de que haya un vértice adyacente de vi con el color uc), se revisa si uc es mayor o igual que nc (es decir, si el color con el que se está intentando colorear vi es el último), entonces se marca $b1$ con true (representando que no hay colores disponibles para vi), y se agrega vi a qh, para finalmente romper el bucle. Pero si, uc es menor que nc, entonces uc incrementa (lo cual significa que cambia al siguiente color), y se continua al inicio del bucle (para volver a probar la coloración, pero con el siguiente color). En caso de que $b2$ fuese false (es decir, no se encontraron adyacentes de vi en uc), se agrega vi a la clase de color uc, se agrega uc a los colores de vi (en v_colors), y se incrementa la ocupación de tsr en uc. La línea 35 revisa si la coloración es para los vértices que requieren dos bloques de tiempo consecutivos, en cuyo caso obtiene el color consecutivo de uc en el mismo día (sumándole 15), denominado suc, y: agrega vi a la clase de color suc, se agrega suc a los colores de vi (en v_colors), y se incrementa la ocupación de tsr en suc. Finalmente, revisa si la ocupación actual de tsr en uc alcanzo su máxima ocupación, en cuyo caso incrementa el color actual de disponibilidad de tsr. Al finalizar el bucle después de la línea 46, se elimina vi de Q, y después de haber iterado todos los vértices de Q para intentar colorearlos, retorna qh, que cuando:

- Está vacía, significa que coloreo todos los vértices de la cola Q.
- Tiene elementos y nb es 2, significa que se dividieron al menos un vértice que requería dos bloques de tiempo consecutivos.
- Tiene elementos y nb es 1, significa que no pudo colorear todos los veritces del grafo con los porcentajes de ocupacion maxima definidas, por lo que los tipos de salones de los vértices en qh deberan aumentar su porcentaje de ocupacion maxima para volver a intentar hacer la asignacion de horarios.

Figura 3. Coloración de vértices de un mismo número de bloques de tiempo

Algorithm 1 colorear_cola

Require: Cola de vertices sin colorear Q, numero de bloques nb
Ensure: Cola de vertices huerfanos qh

```
1:  qh : queue {cola de vertices huerfanos}
2:  while !Q.empty() do
3:      vi ← Q.front()
4:      tsr ← get_room_type(vi)
5:      uc ← last_color_ts[tsr]
6:      b1 : boolean ← false
7:      loop
8:          b2 : boolean ← false
9:          if (nb == 2) and (uc > 15) then
10:             vb1, vb2 ← copias por valor de vi
11:             change_timeslots(vb1, 1)
12:             change_timeslots(vb2, 1)
13:             qh.push(vb1)
14:             qh.push(vb2)
15:         else
16:             for vcu ∈ c_vertices[uc] do
17:                 if g[vi].contains(vcu) then
18:                     b2 ← true
19:                     break
20:                 end if
21:             end for
22:             if b2 then
23:                 if uc >= nc then
24:                     b1 ← true
25:                     qh.push(vi)
26:                     break loop
27:                 else
28:                     uc ← uc + 1
29:                     continue loop
30:                 end if
31:             end if
32:             c_vertices[uc].append(vi)
33:             v_colors[vi][0] = uc
34:             occup[uc][tsr] ← occup[uc][tsr] + 1
```

```
35:         if nb == 2 then
36:           suc ← uc + 15
37:           c_vertices[suc].append(vi)
38:           v_colors[vi][1] = suc
39:           occup[suc][tsr] ← occup[suc][tsr] + 1
40:         end if
41:         if occup[uc][tsr] >= max_occup[tsr] then
42:           last_color_ts[tsr] ← last_color_ts[tsr] + 1
43:         end if
44:       end if
45:       break loop
46:     end loop
47:     Q.pop()
48: end while
49: return qh
```

1

Fuente: Elaboración Propia.

COLORACIÓN DE GRAFO

Esta función se encarga de colorear todos los vértices del grafo, donde inicia obteniendo aquellos vértices que requieren dos bloques de tiempo consecutivos, denominados $vs2$. Luego crea una cola denominada $q2$, donde agregar los vértices $vs2$ ordenados de mayor a menor grado. Al colorear esa cola $q2$ con la función $colorear\_cola$, pasando como parámetro de bloques de tiempo igual a 2, se obtiene una cola denominada $qh1$, la cual representa los vértices huérfanos que requerían dos bloques de tiempo consecutivos, pero tuvieron que ser divididos en dos vértices de un bloque de tiempo requerido. Si $qh1$ no está vacía, intenta volver a colorear sus vértices con la función $colorear\_cola$, pasando como parámetro de bloques de tiempo igual a 1, obteniendo una nueva cola de huérfanos denominada $qh2$, que, en caso de tener elementos, significa que los colores no tuvieron disponibilidad suficiente para colorearlos, y entonces se lanza un error informando esto. En caso de haber coloreado todos los vértices que requerían dos bloques de tiempo consecutivos, sea cumpliendo esta restricción suave, o dividiéndolos en vértices de un bloque de tiempo requerido; continua después de la línea 9, con la obtención de aquellos vértices que requieren un bloque de tiempo, denominados $vs1$. Luego crea una cola denominada $q1$, donde agregar los vértices $vs1$, ordenados de mayor a menor grado. Al colorear esa cola $q1$ con la función $colorear\_cola$, pasando como parámetro de bloques de tiempo igual a 1, se obtiene una cola denominada $qh3$, la cual representa los vértices que no pudieron ser coloreados, significando que los colores no tuvieron disponibilidad suficiente, y entonces se lanza un error informando esto. Si no se lanza ningún error, significa que todos los vértices fueron coloreados exitosamente.

Figura 4. Algoritmo de coloración de todos los vértices del grafo

Algorithm 2 colorear_grafo

Ensure: Coloracion de todos los vertices del grafo

```
1: vs2 ← vertices que requieren 2 bloques de tiempo consecutivos
2: q2 ← cola con vs2 ordenados de mayor a menor grado
3: qh1 ← colorear_cola(q2, 2)
4: if !qh1.empty() then
5:    qh2 ← colorear_cola(qh1, 1)
6:    if !qh2.empty() then
7:       error (disponibilidad de colores insuficientes)
8:    end if
9: end if
10: vs1 ← vertices que requieren 1 bloque de tiempo
11: q1 ← cola con vs1 ordenados de mayor a menor grado
12: qh3 ← colorear_cola(q1, 1)
13: if !qh3.empty() then
14:    error (disponibilidad de colores insuficientes)
15: end if
```

Fuente: Elaboración Propia.

ASIGNACION DE SALON DE CAMPUS A UNA SESION

Esta función recibe como parámetros un tipo de salón tsr, un color i_c, un campus cmp_i, y un vertice i, para asignar un salón de un campus a una sesión. Básicamente, itera los salones de tipo tsr disponibles en el campus cmp_i, revisa si el salón iterado no está ocupado en ese color i_c, y si la capacidad del salón es mayor o igual al tamaño del grupo del vértice i. Si ambas condiciones se cumplen, se cambiar el estado de ocupación de ese salón en ese color, se asigna el salón iterado al vértice i, y se retorna true, indicando que la sesión fue asignada a un salón.

Figura 5. Asignación de salón de campus a una sesión

Algorithm 3 salon_por_campus

Require: tipo de salon tsr, color i_c, campus cmp_i, vertice i
Ensure: Asignacion de salon de un campus i_c al vertice i

```
1: for i_r ∈ rooms[cmp_i][tsr] do
2:     if !c_rooms[i_c][i_r] and r_capacity[i_r] ≥ v_gsize[i] then
3:         c_rooms[i_c][i_r] ← true
4:         v_rooms[i] ← i_r
5:         return true
6:     end if
7: end for
8: return false
```

Fuente: Elaboración Propia.

CALCULAR CAMPUS UNICO DE ADYACENTE EN BLOQUE CONSECUTIVO

Esta función recibe como parámetros un color i_c, y un vertice i, para obtener el campus único requerido por un vértice adyacente de i en el bloque de horario consecutivo de i_c en el mismo dia. Lo que hace es revisar si i_c no es el ultimo bloque del dia. Si no lo es, entonces retorna 0, representando que no hubo un campus requerido; pero en caso afirmativo, obtiene el bloque consecutivo en el mismo día de i_c, denominado cc, itera los vértices coloreados con la clase de color cc, y si encuentra que alguno de ellos es adyacente a i, y ademas tiene un campus único requerido, entonces retorna ese campus. Esto permite asignar salones del mismo campus a sesiones consecutivas en conflicto.

Figura 6. Algoritmo de calcular campus único de adyacente en bloque consecutivo

Algorithm 4 campus_adyacente

Require: color i_c, vertice i
Ensure: campus de vertice adyacente de i en i_c, 0 si no hay adyacencia

```
1: if !is_last(i_c) then
2:     cc ← consecutive_color(i_c)
3:     for va ∈ c_vertices[cc] do
4:         if g[i].contains(va) and v_campus[va] ≠ 0 then
5:             returns v_campus[va]
6:         end if
7:     end for
8: end if
9: return 0
```

Fuente: Elaboración Propia.

ASIGNAR SALONES

Esta función se encarga de asignar los salones a todos los vértices del grafo. Primero, itera todos los vértices, y en caso tal que su tipo de salón requerido solo está disponible en un campus, invoca la función $salon\_por\_campus$, pasando como argumentos el tipo de salon requerido por el vertice, el color del vertice, el vertice mismo, y el campus unico requerido. Esta funcion le asignara un salon del campus unico requerido a ese vertice, y si no hubo campus unico requerido, entonces agrega el vertice a una cola. Despues de haber asignado salones a todos los vertices que requerían de un campus unico, itera los vertices de la cola, y busca si ese vertice tenia adyacentes en el bloque consecutivo del mismo dia (invocando la function $campus\_adyacente$ y pasando como argumentos el color del vertice de la cola y el vertice en si mismo), y si ademas, el adyacente requeria de un campus unico, le asignar un salon de ese campus al vertice iterado de la cola. Si el adyacente no requeria de un campus unico, entonces el vertice iterado de la cola asigna un salon de cualquier campus.

Figura 7. Algoritmo de asignación de salones a todos los vértices del grafo

Algorithm 5 asignar_salones

Ensure: Asignacion de salones para todos los vertices

```
1:  q : queue < int > ← cola de vertices sin campus unico requerido
2:  for i ∈ {1, ..., n} do
3:      tsr ← get_room_type(i)
4:      i_c ← v_colors[i][0]
5:      cmp_i ← get_campus_id(tsr)
6:      if cmp_i ≠ 0 then
7:          salon_por_campus(tsr, i_c, cmp_i, i)
8:      else
9:          q.push(i)
10:     end if
11: end for
12: for i ∈ q do
13:     i_c ← v_colors[i][0]
14:     b5 ← campus_adyacente(i_c, i)
15:     tsr ← get_room_type(i)
16:     for cmp_i ∈ {b5, ..., ncampus} do
17:         salon_por_campus(tsr, i_c, cmp_i, i)
18:         if c_rooms[i_c][i_r] then
19:             break
20:         end if
21:     end for
22: end for
```

Fuente: Elaboración Propia.

RESULTADOS Y DISCUSIÓN

En este apartado se describe la especificación de los parámetros, que, de acuerdo con el contexto del caso de estudio, se definen ciertas constantes. Finalmente, se proponen ideas de trabajo futuro, resultantes de reflexionar sobre posibles puntos de mejora que tiene la metodología propuesta, que por cuestiones de tiempo no pudieron realizarse.

ESPECIFICACIÓN DE PARÁMETROS

Cada bloque de tiempo dura 90 minutos, y hay un tiempo de descanso o “break” de 20 minutos entre dos bloques. Teniendo en cuenta que la jornada académica consta de 6 bloques de tiempo (desde las 7:00 hasta las 17:40), y se consideran 5 días hábiles por semana, entonces se pueden diferenciar 30 combinaciones de (día, bloque). Este número de combinaciones es la cantidad de colores disponibles para implementar la coloración de grafos.

La cantidad de bloques de tiempo consecutivos requerida por una sesión solo puede ser 1 o 2. Cuando es 2, significa que es deseable que la sesión sea programada a dos bloques de tiempo consecutivos. Si una sesión requiere dos bloques de tiempo, pero no son necesariamente consecutivos, se pueden crear dos sesiones con los mismos atributos, pero de un solo bloque de tiempo requerido.

La cantidad de sesiones obtenidas equivale a todas las sesiones requeridas por los grupos de curso sugeridos para los programas académicos pertenecientes a la Facultad de Diseño, Ingeniería e Innovación de la Institución Universitaria Politécnico Grancolombiano en el primer semestre del año 2023.

RESULTADOS

…

Sobre la complejidad computacional de la metodología, el algoritmo de coloración de cola itera cada uno de los vértices, y por cada uno, la operación más costosa que realiza es la iteración de los vértices asignados al color actual que tiene disponibilidad con el tipo de salón requerido, para confirmar la adyacencia de alguno de ellos. La revisión de esto implica una complejidad logarítmica al usar conjuntos en lugar de listas en la representación del grafo como lista de adyacencia, pero teniendo en cuenta que el grado resultante del modelo es disperso, entonces el costo asintótico de la búsqueda de adyacencia es despreciable en ese sentido. Como la cantidad de vértices asignados a cada color siempre será menor a la cantidad de salones (denominada nr), y el número total de sesiones de curso es poco más de n (esta cantidad puede aumentar con la división de las sesiones de curso que requieren dos bloques de tiempo consecutivos), entonces se considera que la complejidad computacional del algoritmo de coloración de cola es aproximadamente $\theta(n) = n * nr$.

En cuanto a la complejidad del algoritmo de coloración del grafo, la consulta de vértices según la cantidad de bloques de tiempo consecutivas que requieren se realiza con filtros lógicos a la base de datos. Pero la operación de ordenamiento de cada una de las consultas por el grado de los vértices tiene un costo cuadrático, pues esa es la complejidad más eficiente entre los algoritmos de ordenamiento óptimos.

CONCLUSIONES

El modelo garantiza que se cumplen todas las restricciones duras. Se busca que se garantice la satisfacción de las restricciones suaves en la mayor proporción posible. Además, aquellos casos de restricciones suaves que no fueron cumplidos pueden serlo mediante modificaciones manuales, gracias a la flexibilidad modular de la solución, que reserva salones libres de todos los tipos en todos los bloques del horario, permitiendo la asignación de estos después de haber calculado la programación de horarios y salones a todas las sesiones de los cursos ofrecidos por la institución.

Todas las características de los objetos implicados en el problema de la programación de horarios y salones, que son relevantes en la evaluación de conflictos duros entre las sesiones de los cursos ofrecidos en la Institución Universitaria Politécnico Grancolombiano fueron identificados, organizados por entidades y normalizados de forma que, se asegura la estandarización del modelo para evitar la duplicidad de información, y promover la aplicación de la metodología propuesta en instituciones cuya estructura administrativa se asemeja a la del caso de estudio.

TRABAJO FUTURO

Se identifican los siguientes puntos (sin algún orden especifico) a realizar para continuar con el desarrollo de la solución propuesta:

- Considerar el horario particular de disponibilidad que tiene cada profesor como una nueva restricción suave, ya que naturalmente los tiempos de todos los profesores son diferentes por el contexto de sus vidas personales.
- Desarrollar una interfaz gráfica que permita a cualquier persona diligenciar la información requerida por el sistema, de manera manual y masiva; para evitar la dependencia del uso del sistema por algún mediador

REFERENCIAS

[1] F. Zabidee y M. H. M. Adnan, «Optimization in University Student Timetables: A Comprehensive Literature Review,» *Journal of Advanced Research in Applied Sciences and Engineering Technology,* p. 14 – 43, 2024.

[2] P. Nandal, A. Satyawali, D. Sachdeva y A. S. Tomar, «Graph Coloring based Scheduling Algorithm to automatically generate College Course Timetable,» *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence),* pp. 210-214, 2021.

[3] Avinash, R. Jain y R. Kumar, «University Time Table Scheduling Using Graph Coloring Technique,» *ResearchGate,* 2018.

[4] V. Donderia y P. K. Jana, «A novel scheme for graph coloring,» *Procedia Technology,* vol. 4, pp. 261-266, 1 2012.

[5] D. Brélaz, «New methods to color the vertices of a graph,» *Communications of the ACM,* vol. 22, nº 4, pp. 251-256, 4 1979.

[6] N. Poddar y B. Mondal, «AN INSTRUCTION ON COURSE TIMETABLE SCHEDULING APPLYING GRAPH COLORING APPROACH,» *International Journal of Recent Scientific Research,* vol. 9, nº 2, pp. 23939-23945, 2 2018.

[7] M. Assi, B. Halawi y R. A. Haraty, «Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem,» *Procedia Computer Science,* p. 899 – 906, 2018.

[8] T. W. Ekanayake, P. Subasinghe, S. Ragel, A. Gamage y S. Attanayaka, «Intelligent Timetable Scheduler: A Comparison of Genetic, Graph Coloring, Heuristic and Iterated Local Search Algorithms,» *2019 International Conference on Advancements in Computing, ICAC 2019,* p. 85 – 90, 2019.

[9] R. K. J. Bendi, T. Sunarni y A. Alfian, «Using Graph Coloring For University Timetable Problem,» *International Journal of Science and Research,* vol. 7, nº 11, pp. 1692-1697, 2018.

[10] A. Muklason, B. A. Nugroho, E. Riksakomara, R. Tyasnurita, F. Mahananto, R. A. Vinarti y M. A. Nuriman, «Flexible Automated Course Timetabling System with Lecturer Preferences Using Hyper-heuristic Algorithm,» *ACM International Conference Proceeding Series,* p. 258 – 262, 2022.

[11] D. J. A. Welsh y M. B. Powell, «An upper bound for the chromatic number of a graph and its application to timetabling problems,» *The Computer Journal,* vol. 10, nº 1, pp. 85-86, 1 1967.

[12] M. Laguna y R. Martí, «A GRASP for Coloring Sparse Graphs,» *Computational Optimization and Applications,* vol. 19, nº 2, pp. 165-178, 1 2001.

[13] B. S. Baker y E. G. Coffman, «Mutual exclusion scheduling,» *{Theoretical Computer Science,* vol. 162, nº 2, pp. 225-243, 1996.

[14] R. P. Grimaldi, Discrete and combinatorial mathematics, 5 ed., Rose-Hulman Institute of Technology: Addison Wesley, 2004.

[15] A. Laaksonen, Competitive Programmer's Handbook, Helsinki: CSES, 2018.

[16] R. Lewis, A Guide to Graph Colouring, UK: Springer, 2015.

[17] E. Bampis, A. Kononov, G. Lucarelli y I. Milis, «Bounded max-colorings of graphs,» *Journal of Discrete Algorithms,* p. 56 – 68, 2014.