Universidad Autónoma de Querétaro

Facultad de Ingeniería

Maestría en Ciencias en Inteligencia Artificial

Optimización de procesos de planeación de horarios escolares mediante coloración de grafos

Tesis

Que como parte de los requisitos para obtener el grado de
Maestro en Ciencias en Inteligencia Artificial

Presenta:

Hansel Amadeus Montúffar Otero

Dirigido por:

Dr. Arturo González Gutiérrez

Dr. Arturo González Gutiérrez
Presidente

M. en C. Guillermo Díaz Delgado
Secretario

M. en D. M. Carmen Sosa Garza
Vocal

Dr. Saúl Tovar Arriaga
Suplente

Dr. Roberto Augusto Gómez Loenzo
Suplente

Centro Universitario Querétaro, QRO
Julio 2019
México.

A mi familia y a mis maestros.

Reconocimientos

Se agradece al Consejo Nacional de Ciencia y Tecnología (CONACYT) por proveer los fondos necesarios para realizar esta investigación. Asimismo se agradece a la Facultad de Ingeniería de la Universidad Autónoma de Querétaro por su valioso apoyo y las facilidades prestadas para la realización de mis estudios de posgrado y su culminación con la elaboración de esta tesis. También se reconoce de manera especial al M. en C. Fidel González Gutiérrez por su asesoría en el desarrollo de esta tesis.

Resumen

En esta tesis se estudia el problema de planeación de horarios escolares y se modela mediante el problema de coloración de vértices en un grafo. Un grafo apropiadamente coloreado consiste en una asignación de color a cada uno de sus vértices de modo tal que ningún par de vértices tienen el mismo color si existe una arista que los una.

La modelación consiste en representar cada materia mediante un vértice, y si dos materias no se deben ofrecer en el mismo horario, ya que el plan de estudios requiere que ambas materias se deben cursar durante el mismo período académico, o porque existe al menos un estudiante que deba llevar ambas, entonces se construye una arista uniendo esos dos vértices.

El número de colores necesarios corresponde al número de ranuras de tiempo requeridas para ofertar las materias que un alumno regular puede tomar, donde cada subconjunto de vértices del mismo color representa las materias que pueden ofrecerse al mismo tiempo sin riesgos de conflicto de horario.

Se ha demostrado que el problema de coloración de grafos es NP-completo, por lo que se conjetura que no existe un algoritmo eficiente que produzca soluciones óptimas para cualquier instancia del problema.

En esta tesis se discuten una serie de algoritmos metaheurísticos eficientes basados en técnicas de inteligencia artificial, tales como Búsqueda Local, Búsqueda Tabú y Algoritmos Genéticos. Asimismo se exploran algunas heurísticas basadas en criterios que establecen un orden preferente de coloración de vértices. La eficiencia y resultados de estos algoritmos son medidos utilizando instancias de grafos publicadas por el DIMACS (*Center for Discrete Mathematics*

and Theoretical Computer Science).

Finalmente, se propone un prototipo de sistema en el cual el usuario puede obtener una planeación de horarios de cursos dado el requerimiento de curso a ofertar durante un semestre determinado de acuerdo al plan de estudios en cuestión que establece el orden en que los cursos puedan ser tomados por un estudiante.

(**Palabras Clave:** Optimización, Coloración, Grafos, Planeación.)

Abstract

In this thesis the problem of school schedules planning is studied and it is modeled by the problem of vertex coloring in a graph. An properly colored graph consists of a color assignment to each of its vertices in such a way that no pair of vertices have the same color if there is an edge that unites them.

The modeling consists of representing each subject through a vertex, and if two subjects should not be offered at the same time, since the curriculum requires that both subjects must be taken, or because there is at least one student who could take both, then an edge is constructed by joining those two vertices.

The number of colors required corresponds to the number of time slots required to offer the subjects that a regular student can take, where each subset of vertices of the same color represents the subjects that can be offered at the same time without risks of scheduling conflict.

It has been shown that the problem of graph coloring is NP-complete, so it is conjectured that there is no efficient algorithm that produces optimal solutions for any instance of the problem.

This thesis discusses a series of efficient metaheuristic algorithms based on artificial intelligence techniques such as Local Search, Tabu Search and Genetic Algorithms. Likewise, some heuristics based on a criterion that establish a preferred order of vertex coloring are explored. The efficiency and results of these algorithms are measured using graph instances published by the DIMACS (*Center for Discrete Mathematics and Theoretical Computer Science*) as a comparison.

Finally, a prototype system is proposed in which the user can obtain a planning of course schedules given the course requirement to be offered during a given semester according to the curriculum that establishes the order that the courses could be taken by a student.

Índice general

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

CAPÍTULO 1

Introducción

If P = NP, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in "creative leaps", no fundamental gap between solving a problem and recognizing the solution once it's found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss; everyone who could recognize a good investment strategy would be Warren Buffett.

Scott Aaronson.

1.1. Justificación

El diseño de horarios para una institución de nivel universitario es una tarea difícil debido a que hay diversos factores que hacen complejo el problema. Estos factores son debido a la naturaleza de cómo están diseñados los planes de estudio que se imparten en la facultad. Por ejemplo la situación de los primeros semestres de las carreras de la facultad. Estos semestres tienden a ser denominados como de "tronco común", lo que quiere decir que varios programas requieren que los alumnos cursen ciertas materias en común. El diseño del horario tiene que tener en cuenta no solo las materias que comparten los alumnos

sino también:

- Número de alumnos por materia.
- Número de salones disponibles.
- Limitación de horario.
- Cruzamiento entre materias.

Se entiende por cruzamiento entre materias al hecho de que un estudiante tenga que cursar 2 o más materias que han sido designadas al mismo horario. En la actualidad las instituciones tienden a designar la tarea de crear un horario al personal administrativo. El personal administrativo enfrenta varios desafíos como lo son el hecho de que el número de estudiantes no es el mismo cada semestre, el número de salones puede cambiar, el plan de estudios puede ser actualizado, etc. Por éstas y más razones el personal administrativo tiene que ser altamente experimentado en la creación de horarios, pero aun así es una tarea que consume una gran cantidad de tiempo. Dado que el problema de coloración de vértices es un problema NP-completo, la solución óptima no puede ser encontrada en una cantidad razonable de tiempo. El estudio de los problemas NP-completos es una de las áreas de mayor crecimiento en los últimos años por el reto que nuevos problemas, particularmente del área de la inteligencia artificial, están surgiendo. Existen diversas técnicas para acercarse a la solución óptima; sin embargo, no hay garantía de que la solución dada sea la óptima. Para abordar este problema de coloración de grafos se pueden plantear heurísticas, metaheurísticas y algoritmos de aproximación (Gonzalez, 2007). Con esta investigación se espera proponer una solución práctica al problema de coloración de grafos para el grafo generado con los programas académicos de la Facultad de Ingeniería. Asimis-

mo, se espera poder aportar información que pueda ser de utilidad para desarrollar los horarios para la Facultad de Ingeniería de la Universidad Autónoma de Querétaro

1.2. Hipótesis

Se presume como hipótesis que el problema de planeación de eventos simultáneos se puede modelar mediante el problema de coloración de grafos, para el cual se pueden plantear heurísticas, metaheurísticas en el contexto de la Inteligencia Artificial.

1.3. Objetivo General

Planear de manera apropiada horarios de la Facultad de Ingeniería de la Universidad Autónoma de Querétaro con una modelación basada en grafos de tal manera de que no haya ningún tipo de cruzamientos entre maestros o alumnos.

1.4. Objetivos Específicos

Tener una solución práctica de coloración para este grafo. Es decir, una solución que permita ofrecer las materias sin ningún tipo de cruzamiento o impedimento para los estudiantes en un tiempo razonable.

Crear de un sistema que permita al usuario dar más restricciones al grafo, de modo que el sistema sea flexible y amigable al usuario.

1.5. Marco Teórico

1.5.1. Grafos

La teoría de grafos es un campo de las matemáticas establecido por Leonhard Euler tratando de resolver el problema de los puentes de Königsberg. Un grafo G consiste en un conjunto finito y no vació $V(G)$ de elementos llamados vértices, y de un conjunto finito $E(G)$ de pares de elementos distintos no ordenados. $V(G)$ es llamado el conjunto de vértices y $E(G)$ es llamado el conjunto de aristas. Una arista $\{v, w\}$ indica que une a los vértices v y w y generalmente se abrevia vw (Wilson, 1986).

En la Figura 1.1 se muestra un ejemplo de un grafo G cuyo conjunto de vértices $V(G)$ es $\{u, v, w, z\}$, y cuyo conjunto de aristas $E(G)$ es $\{uv, uw, vw, wz\}$

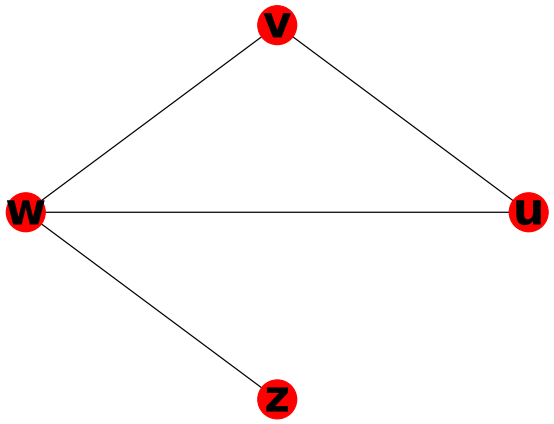


Figura 1.1: Ejemplo de un grafo básico.

1.5.2. Complejidad Computacional

En teoría computacional la solución que se le pueda dar a un problema es medida en su complejidad polinomial de tiempo. Esta es medida en el número de operaciones que el algoritmo deberá de realizar para encontrar una solución. Contar el número de operaciones que un algoritmo realiza de manera exacta es muy complejo para una gran cantidad de algoritmos, por lo que se toma en consideración el término mas alto de operaciones. A esto se le conoce como la notación de gran O (*Big O Notation*).

Big O Notation: Se dice que una función (positiva) $f(n)$ es $O(g(n))$ si existe una constante $c \geq 1$ y $n_0 \geq 1$ de tal manera que $f(n) \leq c \cdot g(n)$ para toda $n \geq n_0$

El tiempo y espacio de complejidad de un algoritmo es expresado en la notación O y describe su razón de crecimiento en términos del tamaño del problema (Gonzalez, 2007).

Problemas NP-Completos

Antes de los años de 1970 los investigadores estaban conscientes de que ciertos problemas podían ser resueltos con algoritmos de baja complejidad polinomial de tiempo $(O(n), O(n)^2, O(n)^3, etc.)$, pero otro tipo de problemas tenían una complejidad exponencial $(O(2^n), O(n!))$. Era claro que incluso con un valor pequeño de n la complejidad de estos problemas era intratable. El término de NP-Completo fue discutido por primera vez por Cook (Cook, 1971). Se introdujo por primera vez la noción de que hay una clase de problemas de decisión que son computacionalmente equivalentes. Esto quiere decir que o todos los problemas en esta categoría tienen una solución en tiempo polinomial o ninguno lo tiene.

Esto es conocido como el problema $P = NP$ el cual es uno de los problemas mas difíciles en teoría computacional.

El problema consiste en determinar si los lenguajes reconocidos por una máquina de Turing determinista son los mismos reconocidos por una máquina de Turing no determinista. Este problema es ilustrado en la Figura 1.2 La conjetura actual es que $P \neq NP$ por lo que estos problemas no tienen algoritmos de tiempo polinomial para resolverse (Garey y cols., 1979; Gonzalez, 2007).

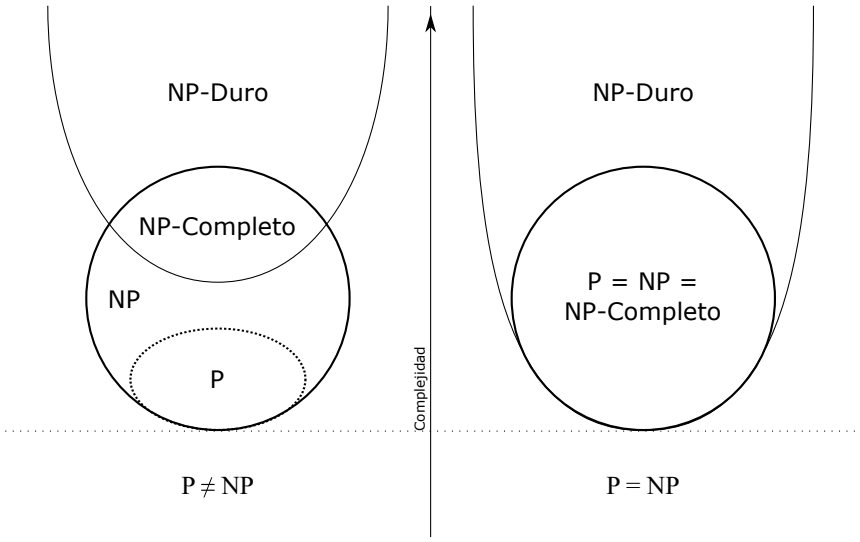


Figura 1.2: Diagrama de Euler para P vs. NP.

1.5.3. Coloración de Grafos

En teoría de grafos, la coloración de grafos es un problema que consiste en asignar un color a cada vértice de manera que ningún vértice tenga el mismo color que su vértice adyacente. (Leighton, 2010; Grimaldi, 1998) Este problema tiene sus inicios en el problema de los 4 colores (Tannenbaum, 1992), y fue planteado por primera vez en 1852 por el matemático Francis Guthrie cuando trataba de colorear el mapa de Inglaterra. En la Figura 1.3 se muestra un mapa

representando 4 territorios, todos los territorios comparten fronteras entre sí.

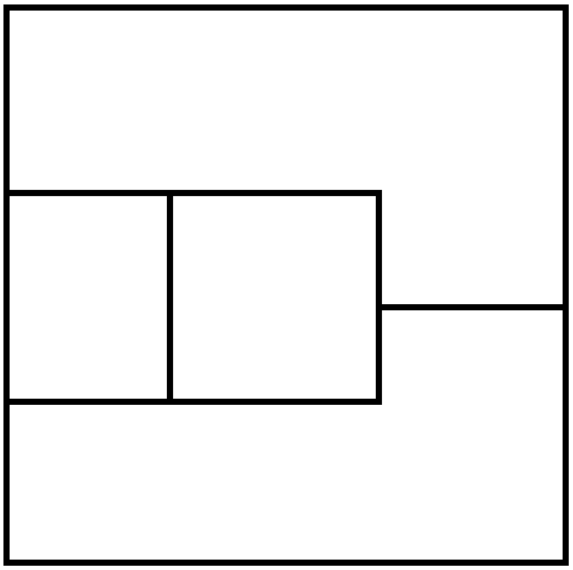


Figura 1.3: Ejemplo de mapa.

El problema plantea la noción de que cualquier mapa puede tener todos los territorios coloreados con solo 4 colores, de manera que cada territorio tenga un color diferente a sus territorios adyacentes. En la Figura 1.4 se muestra la coloración aplicada a este mapa.

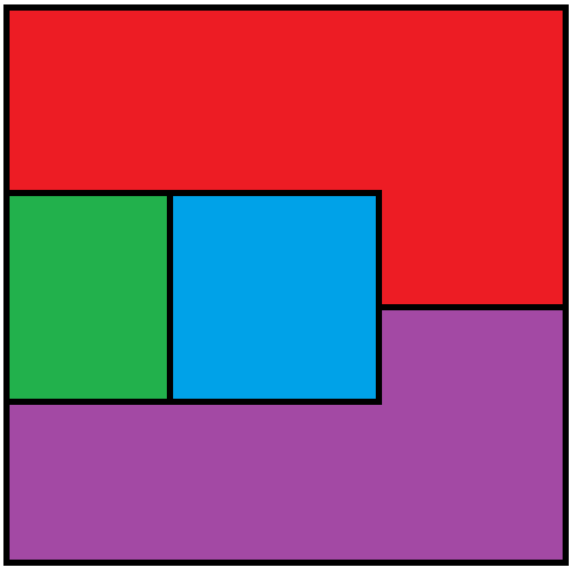


Figura 1.4: Coloración aplicada al mapa.

Mas adelante George David Birkhoff demostró que dicha noción era verdadera en sus estudios de polinomios cromáticos (Birkhoff, 1912). Teniendo un grafo no direccionado G una función $\rho(G, \lambda)$ es una función polinomial que usa λ como el número de colores disponibles, esta función nos dice de cuantas maneras se puede colorear el grafo apropiadamente utilizando a lo más λ colores (Grimaldi, 1998).

El problema está resuelto para mapas, ya que no es posible plantear más de 4 territorios compartiendo fronteras entre sí. La solución del problema muestra que todo grafo planar puede ser coloreado usando a lo más 4 colores (Thomas, 1998). En la Figura 1.5 se puede ver como el mapa puede ser representado como un grafo planar.

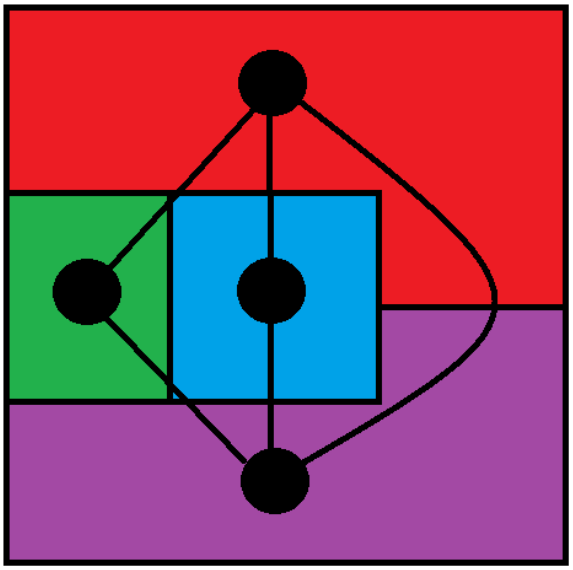


Figura 1.5: Grafo planar generado con el mapa.

Sin embargo, en teoría de grafos el problema se vuelve más interesante, ya que se pueden requerir más de 4 colores. Considere, por ejemplo, el grafo cuyos vértices tiene aristas apuntando a todos los demás vértices, esto es, un grafo completo, siendo $K_n = (V, E)$ con un conjunto de vértices V y un conjunto

de aristas E, donde $|V| = n$ y $|E| = n(n-1)/2$. Para K_n se requieren entonces n colores diferentes para colorear el grafo apropiadamente. (Grimaldi, 1998).

En la Figura 1.6 se muestra el grafo completo de K_6, el cual necesita 6 colores para poder ser coloreado.

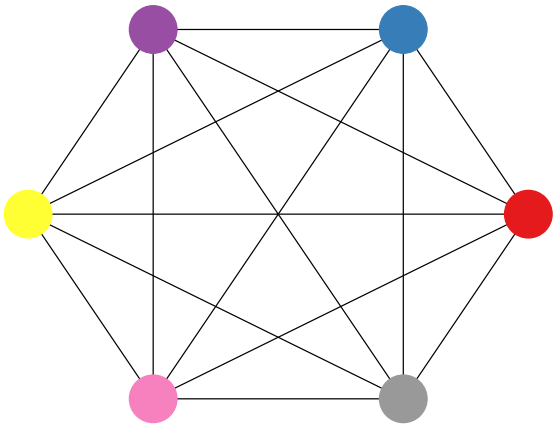


Figura 1.6: Grafo Completo.

Se ha demostrado que el problema de coloración de vértices es un problema NP-completo (Garey y cols., 1979), planteando un gran desafío computacional, ya que para encontrar la solución óptima para toda instancia del problema de coloración de grafos, se requiere grandes cantidades de tiempo de cómputo.

El problema fue establecido como NP-completo en 1972 Karp (1972). Esto se hizo planteándolo como una versión reducida de otro problema NP-completo conocido como 3-SAT, que a su vez es una reducción de SAT. El hecho de que SAT sea NP-Completo fue probado con el teorema de Cook también conocido como el teorema de Cook-Levin (Cook, 1971). En la Figura 1.7 se muestra el diagrama de reducción del problema.

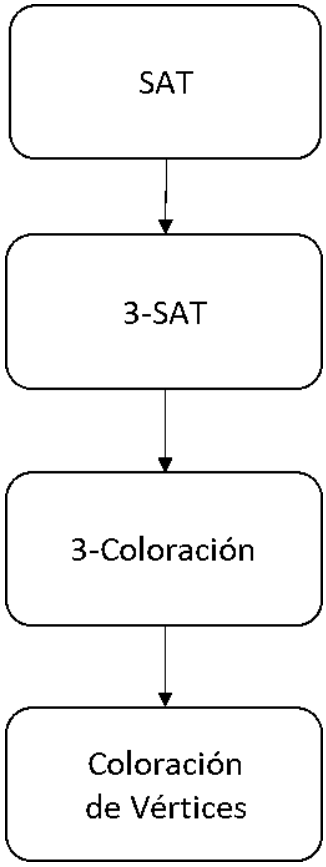


Figura 1.7: Diagrama de reducción.

El número mínimo de colores necesarios para colorear apropiadamente el grafo es referido como número cromático. Este es representado como k_n o como $\chi(G)$. El polinomio cromático puede dar este número $\chi(G) = \min\{k \in \mathbb{N} : p(G,k) > 0\}$.

3-SAT es un problema en el que se tiene una expresión de álgebra booleana. El problema consiste en encontrar una asignación de verdadero o falso a las variables de la expresión de tal manera que la expresión de verdadero como resultado.

La reducción de este problema implica que un problema de 3-SAT puede ser planteado como un caso coloración vértices conocido como 3-Coloración. En

este caso dado un grafo $G(V, E)$ se debe determinar si $\chi(G) \leq 3$. Si este es el caso la coloración implica decir que 3-SAT de verdadero.

Tomemos por ejemplo la expresión 3-SAT: $(X \vee Y \vee \neg W) \wedge (\neg X \vee Y \vee W)$ Para poder modelarla como 3-Coloración primero debemos crear una paleta de 3 nodos T, F, N que representan verdadero, falso y base. En la Figura 1.8 se ilustra esta paleta.

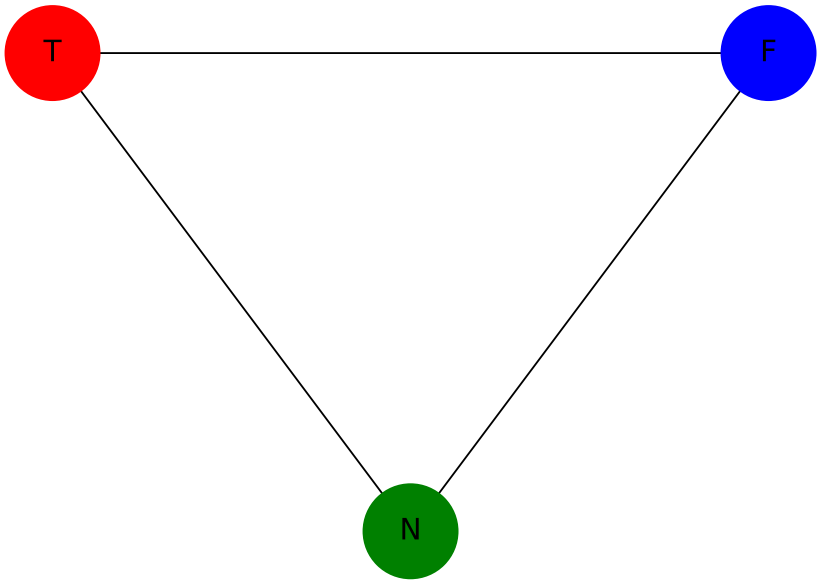


Figura 1.8: Paleta.

El siguiente paso en la modelación es representar las variables que existen en la expresión. Para hacer esto se crean 2 nodos por expresión conectados entre sí para poder representar sus posibles estados de negado o no. Esta modelación puede verse en la Figura 1.9.

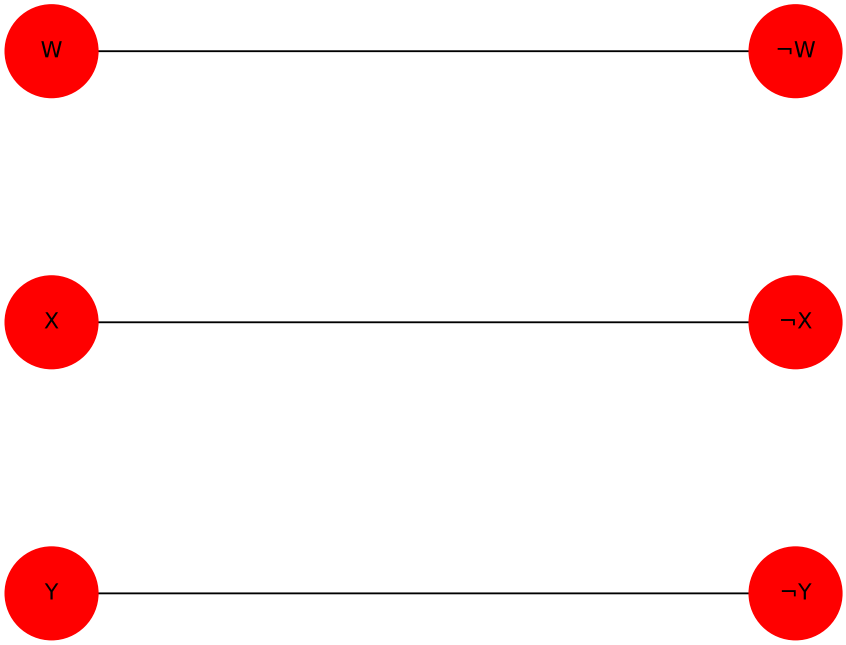


Figura 1.9: Modelación de las variables.

Todos los estados de la variable se conectan por medio de una arista a él nodo N.

El último elemento necesario para la modelación es una compuerta lógica OR la cuál es ilustrada en la Figura 1.10. A esta compuerta se conectan los estados de las variables conforme aparecen en la expresión.

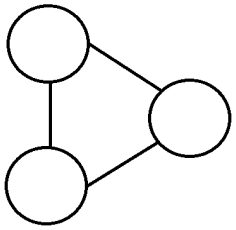


Figura 1.10: Compuerta lógica OR.

La salida de la compuerta es conectada al nodo base N y al nodo de falso F. La modelación de la expresión puede ser vista en la Figura 1.11.

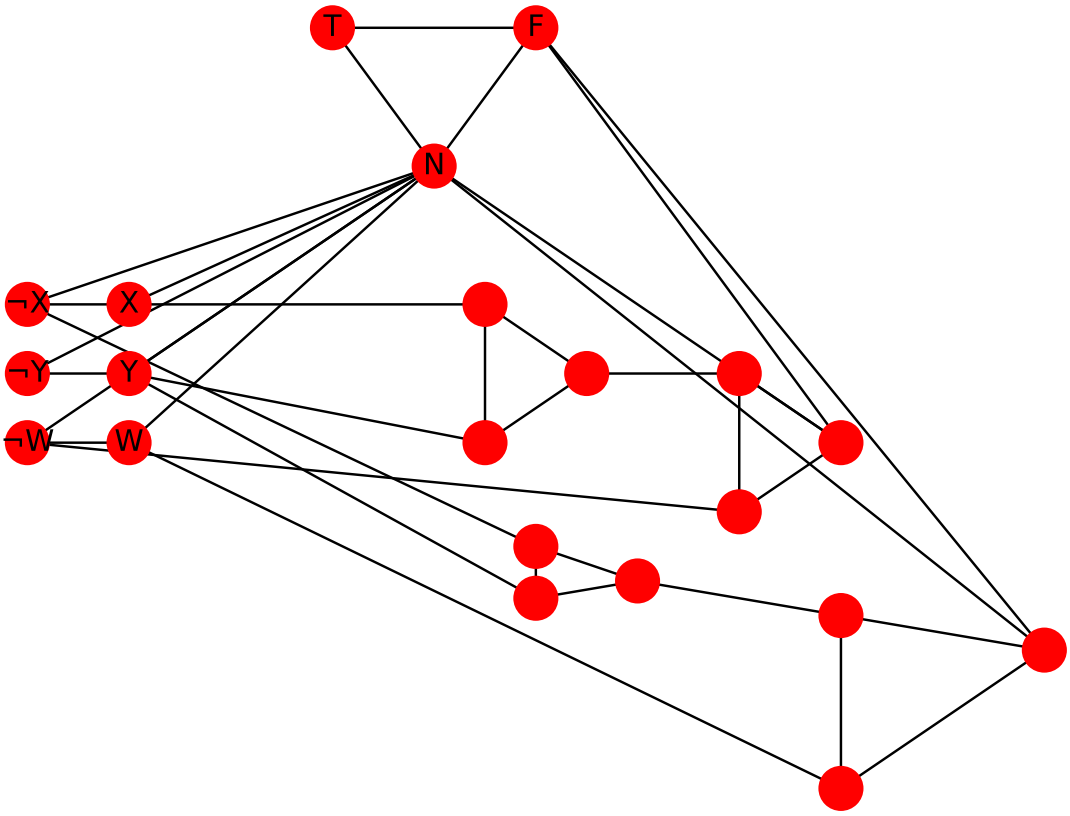


Figura 1.11: Modelación final.

Si este grafo puede ser coloreado con 3 o menos colores ($\chi(G) \leq 3$) esto quiere decir que existe un asignación de verdadero o falso que puede satisfacer la expresión boolena anteriormente establecida.

En este ejemplo, la coloración obtenida es posible con solo 3 colores, la cual se ilustra en la figura 1.12.

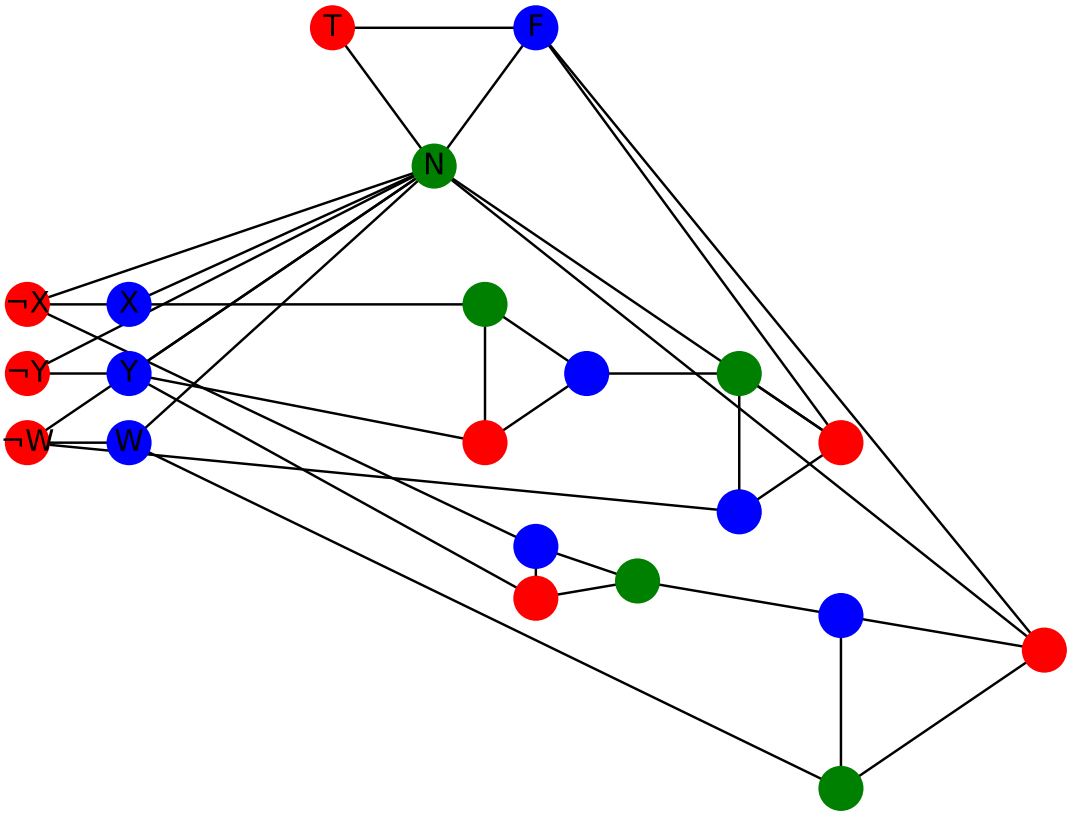


Figura 1.12: Coloración aplicada a la modelación

Como se puede observar, las variables en su estado no negado (X, Y, W) tienen el mismo color que el nodo representativo de falso F esto quiere decir que si a todas las variables se les asigna un estado de falso la expresión 3-SAT tendrá un resultado de verdadero.

En la tabla 1.1 se puede ver la tabla de verdad resultante de la expresión.

X	Y	W	$(X \vee Y \vee \neg W) \wedge (\neg X \vee Y \vee W)$
V	V	V	V
V	V	F	V
V	F	V	V
V	F	F	F
F	V	V	V
F	V	F	F
F	F	V	V
F	F	F	V

Tabla 1.1: Tabla de verdad de la expresión 3-SAT

Como podemos ver en ella esto es cierto y dado que la coloración y la tabla de verdad dan el mismo resultado se pude concluir que 3-Coloración es NP-Completo. 3-Coloración es un problema de decisión si ($\chi(G) \leq 3$) o no. Dado esto coloración de vértices se reduce a saber con cuantos colores se pude colorear apropiadamente un grafo.

Dada la naturaleza de los problemas NP-completos rara vez se trata de encontrar la solución óptima, por lo que el problema se reduce a encontrar una solución viable utilizando heurísticas o algoritmos de aproximación, los cuales producen resultados cercanos a la solución óptima, en un marco de tiempo razonable, de tal manera que la solución sea viable.

Por ejemplo, suponga que un grafo tiene un número cromático óptimo de 9 ($n_k = 9$), esto es, se necesitan como mínimo 9 colores diferentes para colorearlo. Un algoritmo de aproximación o heurística podría dar como resultado 10 ($n_k = 10$), el cual es un resultado aceptable ya que se puede producir en tiempos de cómputo muy razonables.

Existen diversas técnicas que tratan de solucionar el problema de coloración de grafos. Ninguna de estas soluciones garantiza soluciones óptimas, sin embargo son mucho más rápidas que tratar de encontrar soluciones óptimas.

1.6. Esquema de la tesis

En esta tesis se presenta el trabajo realizado, primero presentando en el proceso de metodología la modelación necesaria para tener la demanda de horarios como un grafo. Después se presentan los algoritmos usados para poder aplicar una coloración a este grafo, posteriormente se discute el proceso de asignación de salones que se pueda realizar, y finalmente se discuten los resultados obtenidos.

CAPÍTULO 2

Metodología

> *The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; premature optimization is the root of all evil (or at least most of it) in programming.*
>
> *Donald Knuth*

2.1. Modelación

La coloración de grafos para el desarrollo de horarios se propuso por primera vez en 1979 (Thomson Leighton, 1979). Para poder aplicar la coloración de vértices a un sistema de planeación es necesario definir los eventos como vértices. Si un vértice tiene una arista que lo conecta con otro vértice esto quiere decir que ambos vértices son eventos que no pueden ocurrir al mismo tiempo. Para modelar el horario necesario para la facultad de ingeniería en ambos campus (Centro Universitario y Campus Aeropuerto) se define cada materia como un vértice.

Sí, de acuerdo al plan de estudios, es requerido de un alumno regular llevar más de una materia el mismo semestre, se entiende que estas materias

deberán de ser ofertadas en un horario diferente. Dado que estas materias no pueden ser enseñadas en el mismo horario se les conecta con una arista. Esta modelación fue propuesta con anterioridad (Montuffar-Otero, 2017).

En la tabla 2.1 se listan las materias que se requieren para el primer semestre de la carrera de Ingeniería Biomédica, con el plan de estudios establecido en el 2013 (IBM13).

Tabla 2.1: Plan de primer semestre de Ingeniería Biomédica.

Materia	**Nombre**	**Semestre**
10	BIOLOGÍA	1
203	ÁLGEBRA LINEAL	1
204	QUÍMICA	1
206	UNIVERSIDAD Y SOCIEDAD	1
214	PROBABILIDAD Y ESTADÍSTICA	1
811	CÁLCULO DIFERENCIAL	1

Un alumno inscrito en primer semestre deberá de llevar todas las materias dentro del mismo semestre. Por lo tanto, cuando se modele el grafo basado en ese semestre, todas las materias deberán de estar conectadas entre si por medio de aristas.

En la Figura 2.1 se puede ver como sería la modelación de este semestre. Este es un grafo completo, por lo que todas las materias deben de ser ofertadas en un horario diferente.

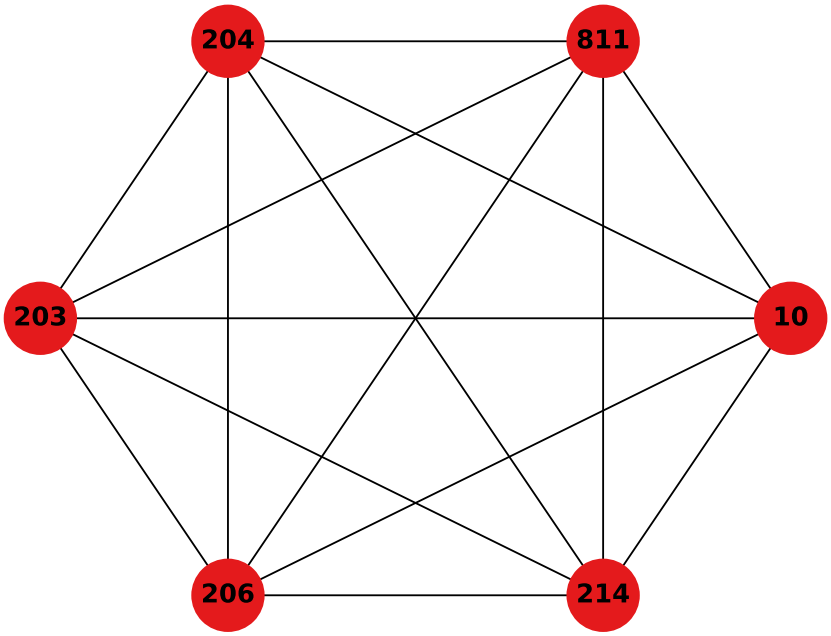


Figura 2.1: Modelado del primer semestre de Ingeniería Biomédica.

En la tabla 2.2 se listan las materias que se requieren para el primer semestre de la carrera de Ingeniería Física, con el plan de estudios establecido en el 2014 (IFI14).

Tabla 2.2: Plan de primer semestre de Ingeniería Física.

Materia	Nombre	Semestre
203	ÁLGEBRA LINEAL	1
204	QUÍMICA	1
206	UNIVERSIDAD Y SOCIEDAD	1
811	CÁLCULO DIFERENCIAL	1
1291	FÍSICA I	1
1292	LABORATORIO DE QUÍMICA	1
1293	TALLER OPTATIVO I	1

El mismo proceso de modelación de grafos se puede aplicar de la misma manera como se puede ver en la Figura 2.2.

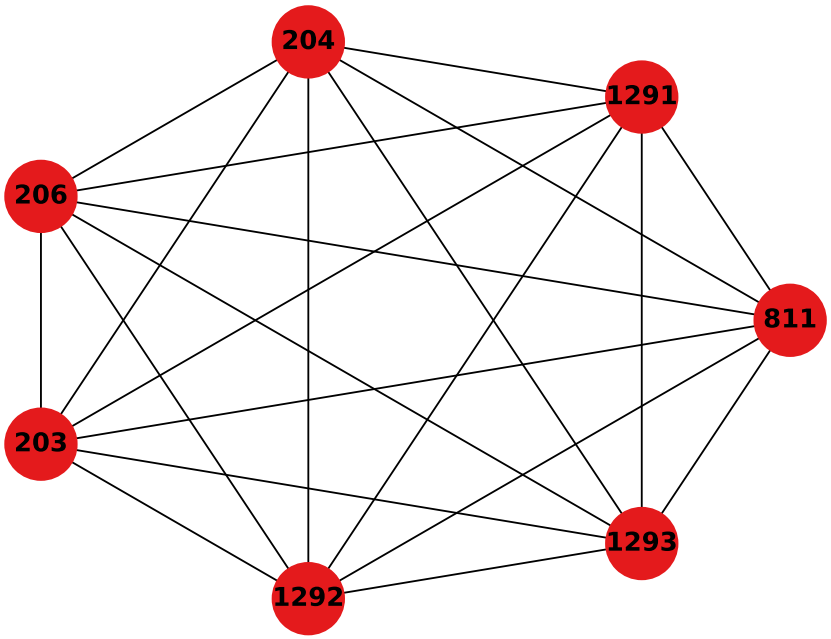


Figura 2.2: Modelado del primer semestre de Ingeniería Física.

Sin embargo, debido a que diferentes carreras tienen algunas materias en común, los grafos de cada carrera pueden resultar interconectados. Tomando como ejemplo las Figuras 2.1 y 2.2 se puede ver que el primer semestre de Ingeniería Física y el primer semestre de Ingeniería Biomédica comparten las materias con claves 203, 204, 206 y 811. Dado que se trata de materias comunes no hay problema de que ambas carreras la oferten al mismo tiempo. En la Figura 2.3 se muestra el grafo resultante al combinar ambos grafos mostrado en las Figuras 2.1 y 2.2.

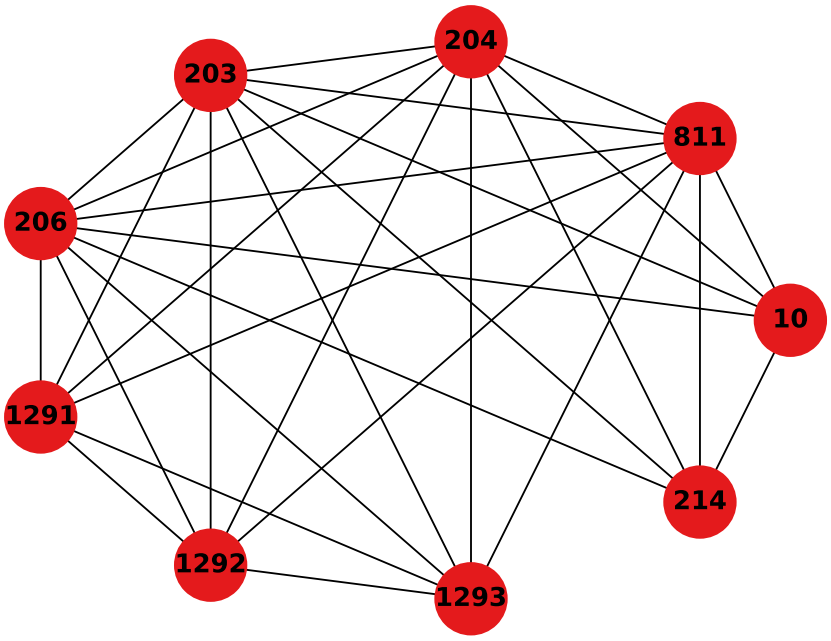


Figura 2.3: Modelado del primer semestre de Ingeniería Física junto con el primer semestre de Ingeniería Biomédica.

Es importante mencionar que en el modelo propuesto no se consideran las materias que en los planes de estudio se indican como Servicio Social y Prácticas Profesionales, ya que no son tomadas en un aula. La misma situación se da con las materias de Deportes y Cultura Deportiva ya que, aunque se dan en la escuela, no requieren aula por que se ofrecen diversas actividades deportivas en diferentes horarios.

Finalmente, no se consideraron las materias de Inglés y de Segundo Idioma, puesto que el horario y aula para estas, es determinado en un espacio específico por el personal administrativo de cada facultad.

El proceso de modelación se aplica para todos los semestres de las carreras ofertadas en cada campus. En el campus aeropuerto estas son Ingeniería Física, Ingeniería Biomédica e Ingeniería en Nanotecnología.

Ingeniería Física e Ingeniería Biomédica tienen 8 semestres mientras que

Ingeniería en Nanotecnología tiene 9. Las 3 carreras tienen varias materias en común, especialmente en los primeros 3 semestres. Esto es debido a que las materias en común tienden a ser materias básicas para las ingenierías, como lo son cálculo integral, programación, universidad y sociedad, etc. Conforme se avanza en los semestres cada vez son menos las materias en común entre carreras o a veces es solo una entre 2 de las 3 carreras. En los últimos semestres ya no se tienen materias en común debido a como se especializan las carreras.

En la Figura 2.4 se puede ver la modelación de todas las carreras impartidas en el campus aeropuerto.

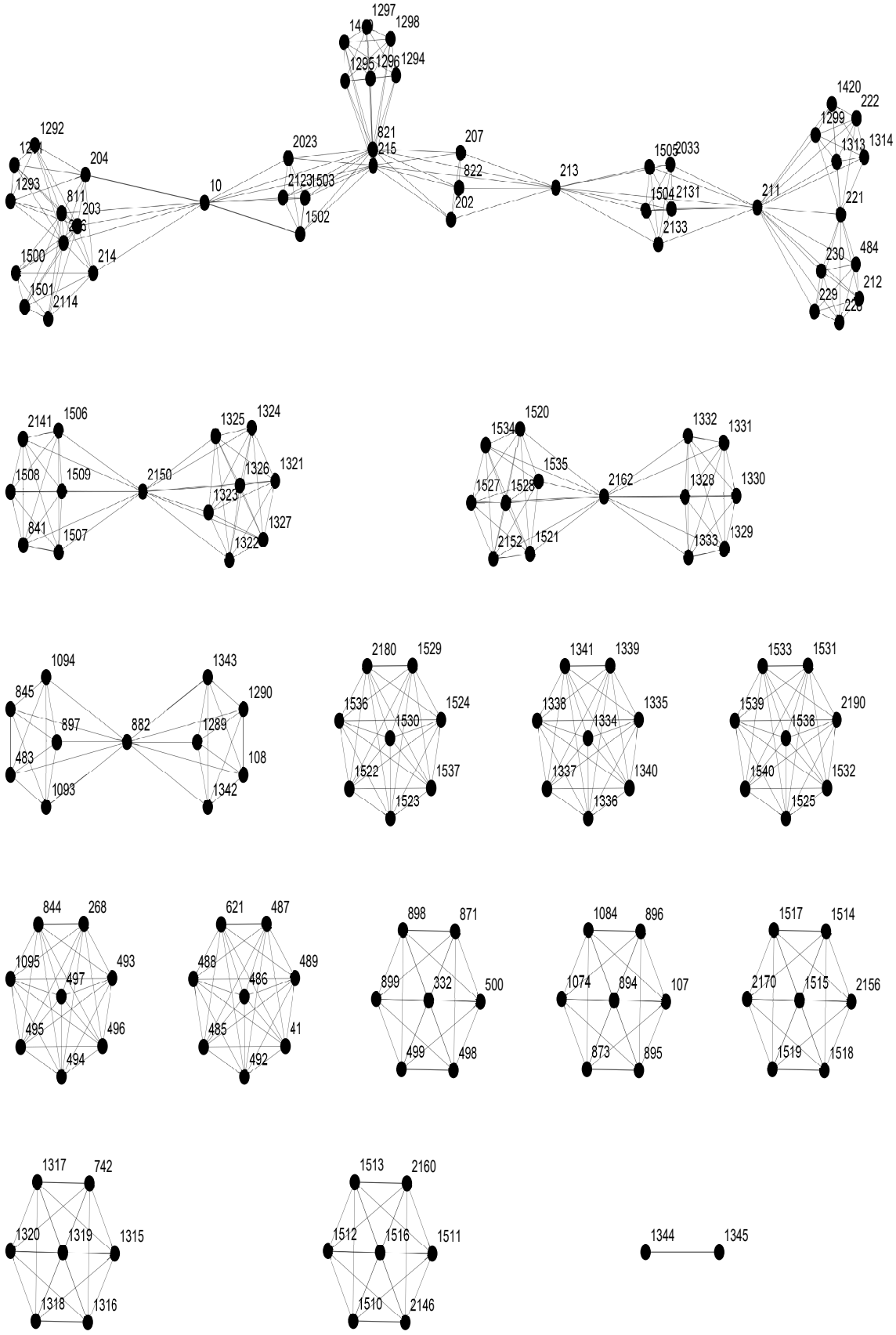


Figura 2.4: Modelado de las carreras de Campus Aeropuerto.

El siguiente paso en la modelación es considerar cuantos grupos hay por materia. El número de grupos cambia de semestre a semestre dependiendo solamente de la demanda de alumnos por materia. Para modelar esto se hace un proceso que en adelante referirá como clonación de vértices. Este proceso implica el crear otro vértice, este vértice tendrá como valor la misma clave que el vértice a clonar mas el valor de 0.01 cada vez que se clone. Por ejemplo si se clona el vértice 203 dos veces estos vértices tendrán los valores 203.01 y 203.02. Las aristas de estos nuevos vértices serán las mismas que las del vértice clonado, esto es debido a que sigue siendo la misma materia con las mismas restricciones. En la Figura 2.5 se puede ver el grafo resultante cuando se aplica el proceso de clonación al grafo del primer Semestre de Biomédica. El proceso de clonación se aplicó una vez al vértice 203 que representa la materia de Biología.

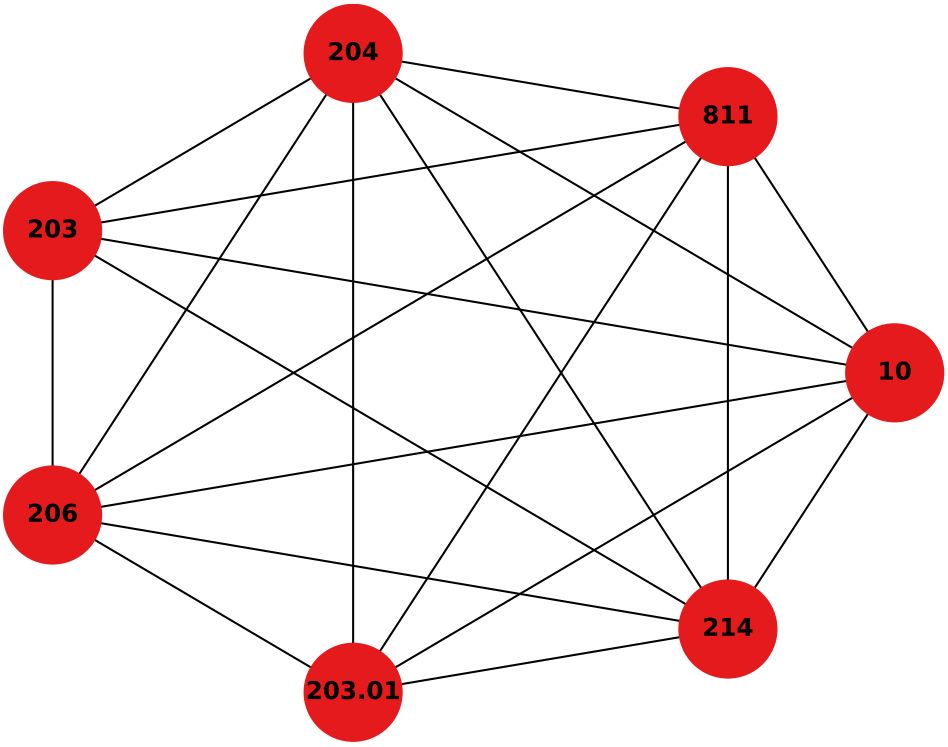


Figura 2.5: Modelado con clonación al vértice 203

La última variable a considerar en el proceso de modelación son los maestros. Si un maestro imparte más de una materia, dentro del mismo campus, se considera que no es posible que ese maestro imparta ambas materias al mismo tiempo. Para modelar esta variable se crea una arista entre todos los vértices que representen materias impartidas por el mismo maestro. En la Figura 2.6 se representa este proceso asumiendo que un maestro imparte las materias 1291 (Física I) y 214 (Probabilidad y Estadística) dentro del mismo campus.

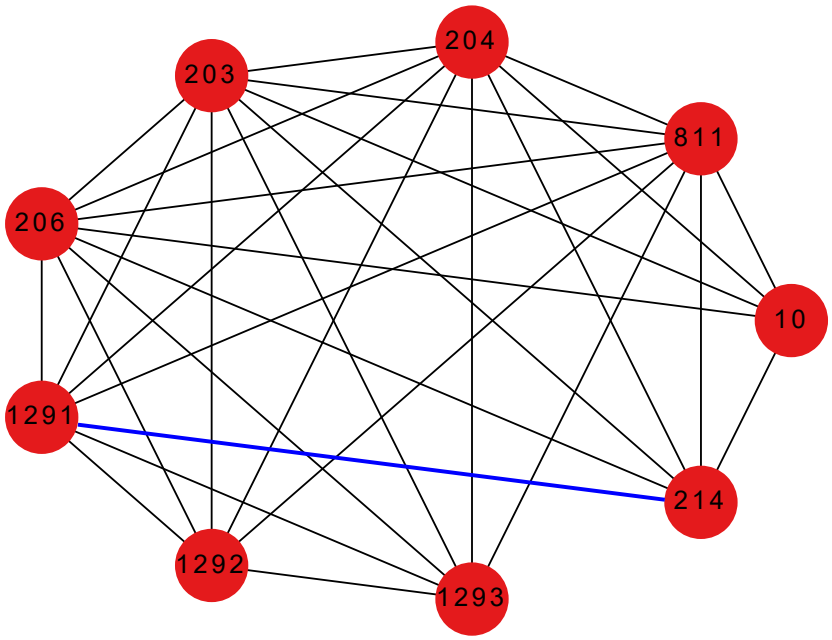


Figura 2.6: Modelado de un grafo con una arista para maestros.

Una vez que se haya modelado el grafo se procede al proceso de coloración.

2.2. Coloración

Como se mencionó, en la introducción, el problema de coloración de grafos es un problema NP-Completo. La definición formal del problema es:

Coloración de Grafos:

Entrada: Dado un grafo $G = (V, E)$, con el conjunto V de vértices y E de aristas donde $|V| = n$, y un conjunto de colores $C = c_1, c_2, ..., c_m$, para $m \leq n$.

Salida: Producir una asignación $f : C \rightarrow V$ de colores C al conjunto de V de vértices del grado de tal modo que existe una arista $e = v_i, v_k \in E$ donde los vértices, v_i, v_k deben de ser coloreados con diferente color.

La naturaleza de este tipo de problemas implica que encontrar la solución óptima toma mucho tiempo computacional. Debido a esta problemática se utilizan técnicas de aproximación para tratar de encontrar una respuesta factible. A continuación se describen los algoritmos usado para encontrar esta solución tratando de minimizar el número de colores necesarios.

La eficiencia de estos algoritmos es probada utilizando el desafío de DIMACS (*The Center for Discrete Mathematics and Theoretical Computer Science*). La información completa de estas instancias se encuentra en el apéndice A. Este desafío cuenta con 125 grafos de los cuales se conoce el número cromático óptimo de 69 de ellos. El resultado se mide utilizando 3 métricas:

- La solución aproximada que proporciona el algoritmo $\hat{f}$.
- La razón de aproximación entre la solución aproximada $\hat{f}$ del algoritmo y la solución óptima f^* o $\chi(G)$ $\frac{\hat{f}}{\chi(G)}$.
- El tiempo de ejecución del algoritmo, medido en segundos.

2.2.1. Heurísticas

Los algoritmos heurísticos son algoritmos diseñados para tratar de aproximar una solución cercana a la óptima. Esto es necesario cuando no es posible encontrar la solución óptima por medios tradicionales, o simplemente no es práctico como lo es en el caso de los problemas NP-Completos. El propósito de una heurística es el de encontrar una solución factible en un marco de tiempo razonable (Pearl, 1984).

En el caso de coloración de grafos una de las técnicas mas utilizadas para aproximar la respuesta es la de los algoritmos voraces (*Greedy Algorithms*) también conocido como coloración secuencial. Estos algoritmos siguen la heurística de recorrer los vértices del grafo de manera secuencial y colorear el vértice con el primer color disponible (Mitchem, 1976).

A pesar de que estos algoritmos pueden ser ejecutados en tiempo lineal no garantizan el encontrar la solución óptima. En la Figura 2.7 se puede ver el proceso de coloración voraz aplicado a un grafo con $\chi(G) = 2$ siguiendo el orden alfabético de los vértices (Tannenbaum, 1992). En esta instancia el orden

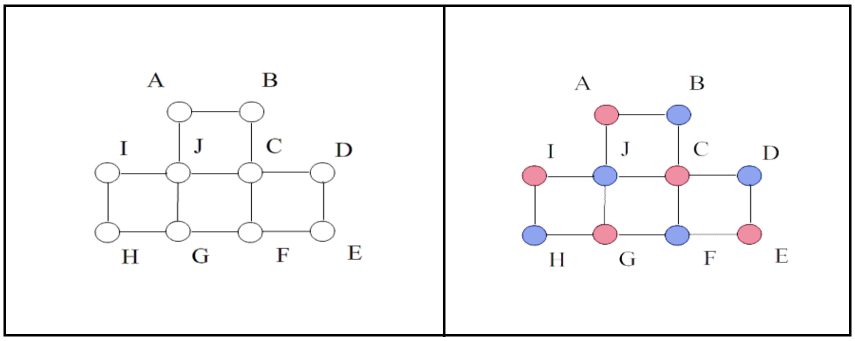


Figura 2.7: Coloración voraz con resultado óptimo.

alfabético produjo un resultado óptimo de $n_k = 2$ sin embargo el resultado puede cambiar completamente dependiendo de el orden en el que se etiqueten los

vértices. Esto se ve en la Figura 2.8.

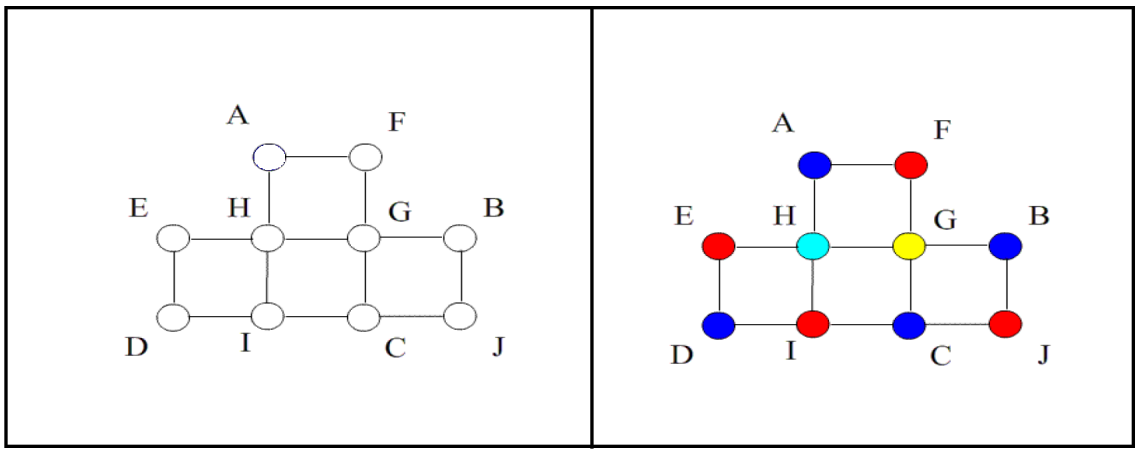


Figura 2.8: Coloración voraz con resultado $n_k = 4$.

En esta instancia a pesar de que se usa el mismo grafo el orden de etiquetamiento que se le da a los vértices hace que el resultado se $n_k = 4$, este resultado es el doble del óptimo. Este resultado es sub-óptimo especialmente considerando la aplicación que se le quiere dar al problema. En planeación de horarios esto quiere decir que se usaría el doble de espacios temporales de los que se requieran. El algoritmo usado por esta heurística es el siguiente:

Entrada: Grafo G, vértices en orden v.

Salida: Número cromático n.

$i = 1, j = 1$.

foreach $v_i \in V$ **do**

  if *Para toda arista* $v_i, v_k \in E$*, ningún vértice* v_k *esta coloreado con* c_j

  then

    Asignar $c_j \rightarrow v_i$.

  else

    Elegir el mínimo j para asignar $c_j \rightarrow v_i$.

  end

end

El número cromático es j.

Algoritmo 1: Algoritmo heurístico basado en ordenamiento.

Para probar esta heurística se realizaron las pruebas del desafío de DIMACS siguiendo 3 ordenes diferentes.

Por Ordenamiento

En esta ejecución del algoritmo, la heurística procede coloreando los vértices conforme al orden establecido por el etiquetamiento de los vértices v_1 a v_n del vértice 1 al vértice n. Este orden tuvo resultados mixtos, encontrando el resultado óptimo varias veces pero fallando, en ciertos grafos, con una razón de aproximación de más de 3 varias veces.

En la tabla 2.3 se muestran resultados parciales de este algoritmo

A pesar de los resultados mixtos esta aproximación probo tener un tiempo de ejecución muy pequeño incluso para grafos con 1000 vértices. Los resultados del algoritmo aplicado a todas las instancias pueden verse en el apéndice B.

Tabla 2.3: Resultados parciales del coloración por Ordenamiento.

Nombre	$\hat{f}$	$\chi(G)$	**Tiempo**	$\frac{\hat{f}}{\chi(G)}$
1-Insertions_4	6	4	0.004	4.5
anna	12	11	0.012	1.091
flat1000_50_0	126	50	0.7	2.52
fpsol2.i.1	65	65	0.122	1.0
inithx.i.1	54	54	0.444	1.0
le450_5c	18	5	0.136	3.6
miles500	23	20	0.014	1.15
myciel3	5	4	0.0	1.25
zeroin.i.1	49	49	0.036	1.0

Por Mayor Grado

En teoría de grafos el grado de un vértice se refiere al número de aristas en un grafo G que son incidentes con un vértice v. Esto se representa como $deg(v)$ ó como $\triangle(v)$ (Grimaldi, 1998). Entre mayor sea el grado de un vértice más grande es el número de vecinos que tiene este. En este orden la heurística procede coloreando los vértices conforme al orden establecido por que vértice tiene mayor grado. Es decir el vértice con más vecinos es el primero en ser coloreado y se recorren los vértices de manera descendente conforme a su grado, hasta que el último en ser coloreado es el vértice con menor grado. En la tabla 2.4 se pueden ver resultados parciales de este algoritmo.

Tabla 2.4: Resultados parciales del coloración por Mayor Grado.

Nombre	$\hat{f}$	**Óptimo**	**Tiempo**	$\frac{\hat{f}}{\chi(G)}$
1-Insertions_4	5	4	0.004	1.25
anna	11	11	0.008	1.0
flat1000_50_0	125	50	0.859	2.5
fpsol2.i.1	65	65	0.13	1.0
inithx.i.1	54	54	0.447	1.0
le450_5c	13	5	0.148	2.6
miles500	20	20	0.008	1.0
myciel3	4	4	0.0001	1.0
zeroin.i.1	49	49	0.032	1.0

El algoritmo fue capaz de obtener el resultado óptimo una mayor cantidad de veces que el algoritmo por ordenamiento. El algoritmo mostró un tiempo de ejecución similar. Los resultados del algoritmo aplicado a todas las instancias pueden verse en el apéndice C.

Por Menor Grado

Como última prueba experimental de este algoritmo se hizo una ejecución basada en el vértice con el menor grado. Esta ejecución toma el orden inverso de la ejecución por mayor grado. Es decir el vértice con menor grado es el primero en ser coloreado y se recorren los vértices de manera ascendente conforme a su grado. En la tabla se 2.5 pueden ver resultados parciales de este algoritmo.

Tabla 2.5: Resultados parciales del coloración por Menor Grado.

Nombre	$\hat{f}$	$\chi(G)$	**Tiempo**	$\frac{\hat{f}}{\chi(G)}$
1-Insertions_4	5	4	0.002	1.25
anna	13	11	0.009	1.182
flat1000_50_0	125	50	0.686	1.64
fpsol2.i.1	66	65	0.156	1.015
inithx.i.1	55	54	0.396	1.019
le450_5c	19	5	0.116	3.8
miles500	28	20	0.015	1.4
myciel3	4	4	0.0001	1.0
zeroin.i.1	50	49	0.027	1.02

Los resultados de la ejecución de este algoritmo no fueron satisfactorios, puesto de que tuvo un tiempo de ejecución similar al coloración por mayor grado, pero falló en encontrar el óptimo en instancias que los algoritmos anteriores si encontraron. Lo que determina su ineficiencia. Los resultados del algoritmo aplicado a todas las instancias pueden verse en el apéndice D.

2.2.2. Metaheurísticas

Las metaheurísticas son procedimientos de alto nivel diseñados para para desarrollar algoritmos de optimización heurísticos. Las metaheurísticas se desarrollan específicamente para encontrar una solución que sea ”lo suficientemente buena” en un tiempo de computación que sea ”lo suficientemente pequeño” (Gonzalez, 2007).

El termino ”metaheurísticas” fue propuesto por Glover en 1986 (Glover, 1986). En general significa ”buscar en un nivel más alto”. Mientras que las heurísticas tienden a ser específicas al problema que tratan de resolver, mientras que, la técnica descrita por la metaheurística, puede ser aplicada a varios problemas. A continuación se describen las metaheurísticas usadas en esta tesis.

Búsqueda Local

Búsqueda local (en inglés *Local Search*) es una metaheurística basada en el espacio de búsqueda. El espacio de búsqueda son todas las soluciones posibles que el problema puede tener. En el caso de coloración de grafos serían todas las coloraciones validas que puede tener el grafo.

La idea de la búsqueda local es muy simple: Dada una solución s para un problema de optimización, se trata de mejorar la solución haciendo cambios locales (Gonzalez, 2007). Sí de esta manera se mejora la solución, entonces una nueva solución s' es obtenida. Este proceso se repite hasta que no se pueda mejorar la respuesta actual.

Formalmente, un problema de optimización $\prod$ tiene una colección de instancias $(\mathcal{S}, c)$. $\mathcal{S}$ es el conjunto de todas las soluciones factibles. El segundo com-

ponente es c, en una instancia es la función objetivo $c : \mathcal{S} \to \mathbf{R}$. El objetivo de búsqueda local es el de encontrar una solución $s* \in \mathcal{S}$ con un valor mínimo o máximo. Es decir:

$$c(s^*) = \underset{s \in \mathcal{S}}{óptimo}\, c(s)$$

Una función de vecindario (*neighborhood function*) $\mathcal{N} : \mathcal{S} \to 2^{\mathcal{S}}$ específica, por cada solución $s \in \mathcal{S}$ hay un subconjunto $\mathcal{N}(s)$ de vecinos de s, o soluciones cercanas a s. Este algoritmo también es conocido como ”algoritmo de mejoramiento iterativo”. Esta algoritmo tiende a encontrar un mínimo local.

Entrada: Un conjunto $\mathcal{S}$ de soluciones factibles, una función de vecindario $\mathcal{N}$, y una función objetivo c.

Salida: Una solución óptima local $s \in \mathcal{S}$ con respecto a $\mathcal{N}$ y c.

Computar una solución inicial factible $s \in \mathcal{S}$.

while $\mathcal{N}(s)$ *contiene una solución local mejor que* s **do**

  Escoger una solución $s' \in \mathcal{N}(s)$ con mejor valor $c(s')$ que $c(s)$.

  Determinar $s \leftarrow s'$.

end

La solución es s.

Algoritmo 2: Algoritmo de Búsqueda Local.

Para aplicar este proceso a coloración de grafos es necesario definir un coloración factible como la solución inicial. $\mathcal{N}$ se define como todas las instancias de coloración en donde se cambio el color de los vértices que tengan un mismo color seleccionado al azar. $\mathcal{S}$ es definido como todas las coloraciones factibles dentro de $\mathcal{N}$.

El algoritmo sigue repitiéndose hasta que se eliminó la mayor cantidad de colores posibles. En la tabla se pueden ver resultados parciales de la experimen-

tación de este algoritmo.

Tabla 2.6: Resultados parciales del coloración por Búsqueda Local.

Nombre	$\hat{f}$	$\chi(G)$	**Tiempo**	$\frac{\hat{f}}{\chi(G)}$
1-Insertions_4	5	4	0.006	1.25
anna	11	11	0.39	1.0
flat1000_50_0	122	50	2.266	2.44
fpsol2.i.1	65	65	0.399	1.0
inithx.i.1	54	54	1.114	1.0
le450_5c	15	5	0.291	3.0
miles500	21	20	0.022	1.05
myciel3	4	4	0.0001	1.0
zeroin.i.1	49	49	0.075	1.0

El algoritmo probo ser mejor que la heurística. Pero cabe recalcar que su tiempo de ejecución fue mayor. Los resultados del algoritmo aplicado a todas las instancias pueden verse en el apéndice D.

Búsqueda Tabú

Como se menciono anteriormente los algoritmos de búsqueda local tienden a encontrar el máximo o mínimo local. Para poder definir estos conceptos primero se observa la gráfica en la Figura 2.9.

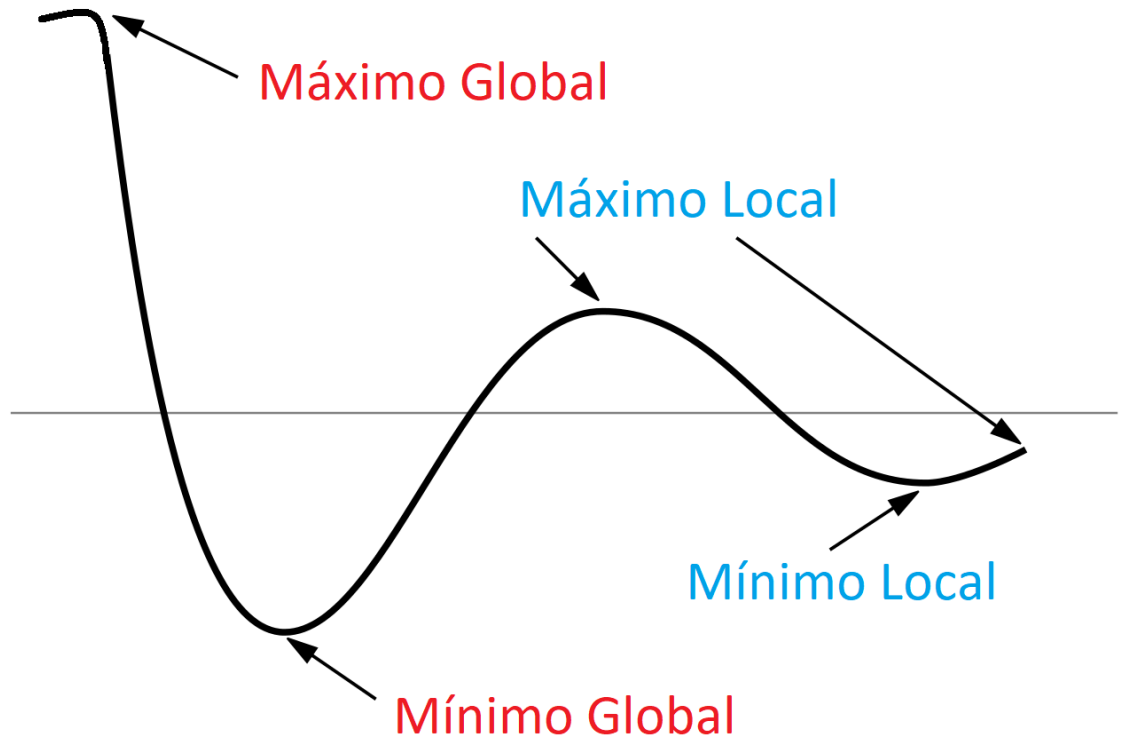


Figura 2.9: Gráfica de Máximo y Mínimo Local.

En esta gráfica se ilustran todas las soluciones factibles de algún problema de optimización. En azul se resaltan el máximo y el mínimo local. Estas serían los mejores soluciones que un algoritmo pudiera encontrar dependiendo de si se esta minimizando o maximizando. En rojo se resaltan el máximo y mínimo local. Un algoritmo que trata de optimizar soluciones puede dar como solución óptima el mínimo o máximo local. Esto se debe a que los vecinos de estas soluciones dan un resultado peor, por lo que el algoritmo entiende que la solución en donde se encuentra no pude ser mejorada.

Para evitar que el algoritmo se quede en estos valores locales se puede implementar búsqueda tabú (*Tabu Search*). Búsqueda tabú es una metaheurística que guía el proceso de búsqueda de una heurística local. El termino fue defi-

nido en el mismo artículo que definió metaheurísitica (Glover, 1986).

En búsqueda tabú se emplea lo que se conoce como una lista tabú (*Tabu List*). Esta es una lista de un tamaño determinado en el que se ponen ciertas soluciones factibles del problema de optimización. Estas soluciones no pueden ser determinadas como la nueva solución por un cierto número de iteraciones de búsqueda. Este proceso es descrito en el siguiente algoritmo:

Entrada: Un conjunto $\mathcal{S}$ de soluciones factibles, una función de vecindario $\mathcal{N}$, una función objetivo c, una lista tabú T con tamaño máximo tam.

Salida: Una solución óptima local $s \in \mathcal{S}$ con respecto a $\mathcal{N}$ y c.

Computar una solución inicial factible $s \in \mathcal{S}$.

while $\mathcal{N}(s)$ *contiene una solución local mejor que* s **do**
  Escoger una solución $s' \in \mathcal{N}(s)$ con mejor valor $c(s')$ que $c(s)$.
  if $s' \notin T$ **then**
    Determinar $s \leftarrow s'$.
    if $|T| > tam$ **then**
      Borrar primer elemento de T.
    end
    Agregar s a T.
  end
end

La solución es s.

Algoritmo 3: Algoritmo de Búsqueda Tabú.

En coloración de grafos el algoritmo más usado, basado en búsqueda tabú, es conocido como *TABUCOL*. Este algoritmo fue propuesto en 1987 y es conocido como uno de los mejores algoritmos para coloración de grafos. (Hertz, 1987).

Este algoritmo trata de reducir el número de colores usados con una solución inicial dada. Para hacer esto genera un vecindario de búsqueda de manera aleatoria. Después se busca una solución factible mejor que la solución actual. Esta solución es puesta en una lista tabú y no es seleccionada otra vez hasta cierto número de iteraciones.

En la implementación experimental se definió un número de iteraciones de 1000, con un vecindario de búsqueda de 50 elementos y una lista tabú de 50 elementos. En la tabla 2.7 se pueden ver resultados parciales de este algoritmo.

Tabla 2.7: Resultados parciales del coloración por Búsqueda Tabú.

Nombre	$\hat{f}$	$\chi(G)$	**Tiempo**	$\frac{\hat{f}}{\chi(G)}$
1-Insertions_4	5	4	24.413	1.25
anna	11	11	48.551	1.0
flat1000_50_0	123	50	11129.372	2.46
fpsol2.i.1	65	65	563.535	1.0
inithx.i.1	54	54	941.245	1.0
le450_5c	16	5	508.016	3.2
miles500	20	20	73.343	1.0
myciel3	4	4	8.933	1.0
zeroin.i.1	49	49	199.313	1.0

Este algoritmo probó tener una eficiencia satisfactoria a pesar de que el tiempo de ejecución es mucho mayor.Los resultados del algoritmo aplicado a todas las instancias pueden verse en el apéndice F.

Algoritmos Genéticos

Los algoritmos genéticos fueron propuestos por primera vez en 1992 (Holland, 1992). Este tipo de algoritmos emula una versión simplificada del proceso evolutivo de los seres vivos. Este comportamiento es emulado usando los elementos de:

- Población.
- Aptitud.
- Cruzamiento.
- Mutación

La población es definida como un número dado de instancias que un problema puede tener. A cada individuo de la población se le conoce como cromosoma. Y cada elemento de este individuo se conoce como gen. Esto se puede ver ilustrado en la Figura 2.10.

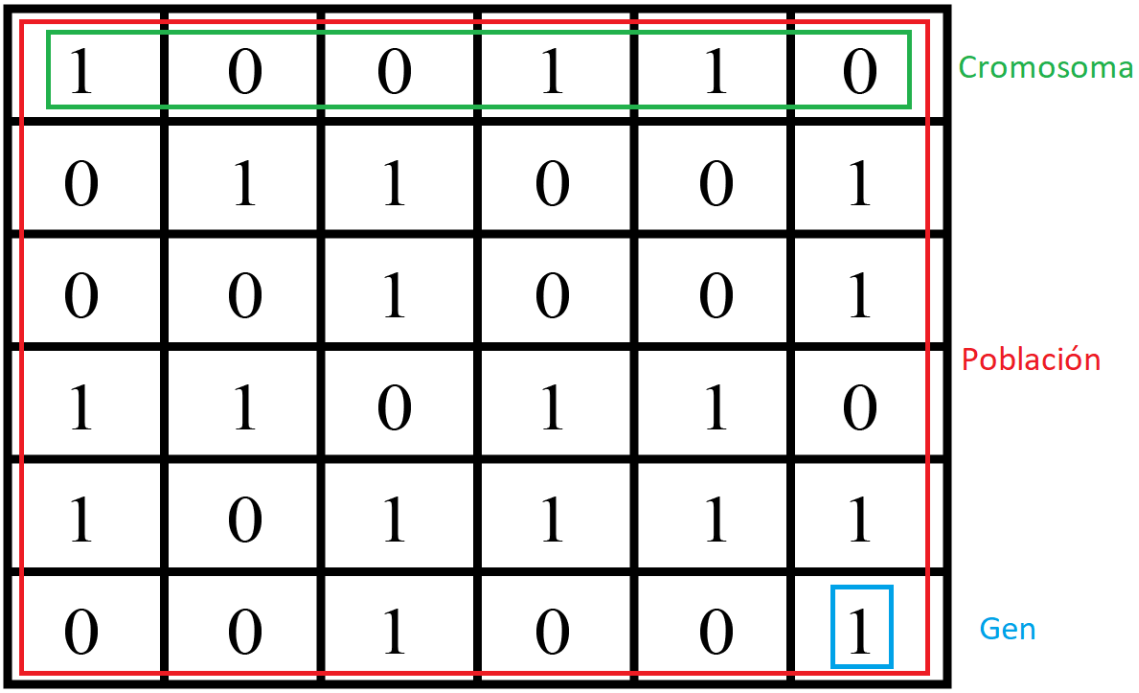


Figura 2.10: Conceptos de Algoritmo Genético.

La aptitud es una función que determina un valor para poder medir que tan ”buena” es la instancia que se ha dado. Esta aptitud es conforme a una función objetivo.

Ya que se tiene la población se define un número de generaciones. Este número es cuantas veces se ejecutarán los procesos de cruzamiento y mutación. Por cruzamiento se entiende como cuando 2 cromosomas de una población intercambian genes. A los cromosomas originales de la población se les conoce como padres (padre 1 y padre 2) y a los resultados del intercambio de genes como hijos (hijo 1 e hijo 2). Este proceso se puede ver ilustrado en la Figura 2.11.

1	1	1	1	1	1
0	0	0	0	0	0

Padre 1 y Padre 2

1	1	1	0	0	0
0	0	0	1	1	1

Hijo 1 e Hijo 2

Figura 2.11: Ejemplo de cruzamiento.

El proceso de selección de padres es definido previamente. En este caso se uso el proceso de selección por torneo. Este proceso selecciona como padre 1 al elemento más apto y a padre 2 como el elemento menos apto. A la siguiente población se pasan los 2 elementos mas aptos de los padres e hijos. Se recorre de esta manera toda la población hasta que todos los elementos hayan sido cruzados.

Finalmente en cada generación se define un sector de la población a mutar. Este sector es normalmente el que peor aptitud tenga. El proceso de mutación es cuando a un cromosoma se le altera de manera aleatoria sus genes. El propósito de este proceso es el de evitar mínimos locales. Es proceso se puede ver ilustrado en la Figura 2.12.

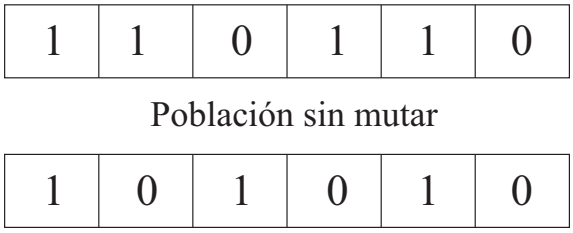


Figura 2.12: Ejemplo de mutación.

En coloración de grafos este proceso puede ser emulado (Musa M. Hindi, 2010). Para poder hacer esto primero se obtiene un coloración inicial válido. De este coloración se extrae el número cromático n_k. Con este número se genera una población de manera aleatoria.

Esta población la conforman un número de coloraciones aleatorias usando $n_k - 1$ colores. La función de aptitud es cuantos conflictos tiene este coloración, por conflictos se entiende como el número de aristas cuyos vértices tienen el mismo color. En la experimentación se definió una población de tamaño, una probabilidad de mutación de 50 % y se ejecutó por 1000 generaciones. En la tabla 2.8 se pueden ver resultados parciales de este algoritmo.

Tabla 2.8: Resultados parciales del coloración por Algoritmo Genético.

Nombre	$\hat{f}$	$\chi(G)$	**Tiempo**	$\frac{\hat{f}}{\chi(G)}$
1-Insertions_4	6	4	106.469	1.5
anna	12	11	213.154	1.091
flat1000_50_0	126	50	50288.52	2.52
fpsol2.i.1	65	65	2611.324	1.0
inithx.i.1	54	54	4295.946	1.0
le450_5c	18	5	2324.405	3.6
miles500	23	20	327.753	1.15
myciel3	4	4	17.047	1.0
zeroin.i.1	49	49	902.598	1.0

Este algoritmo probó ser por mucho el más lento. Los resultados del algo-

ritmo aplicado a todas las instancias pueden verse en el apéndice G.

2.2.3. Post-procesamiento de balanceo

Independientemente del algoritmo de coloración usado se hace un proceso después de la coloración. Este proceso se define como Balanceo del Grafo. En este proceso se trata de que cada color usado tenga el mismo número de vértices. Esto se hace debido a que se quiere usar el mismo número de salones en cada espacio temporal con el propósito de que se dejen vacíos el menor número de salones posibles.

En la tabla 2.9 se ve un coloración sin balancear.

Tabla 2.9: Coloración sin balancear.

Ranura de Tiempo	Cursos/Ranura de Tiempo	Tamaño del Cluster
0	204, 204.01, 204.02, 213, 213.01, 229, 229.01	7
1	221, 221.01, 221.02, 811, 821	5
2	203, 211, 211.01, 211.02, 211.03, 215, 215.01, 215.02	8
3	206, 212, 821.01, 821.02, 1420	5
4	203.01, 203.02, 203.03, 203.04, 228, 822, 1299, 1419	8
5	202, 206.01, 206.02, 206.03, 206.04, 230, 1294, 1313	8
6	207, 212.01, 811.01, 811.02, 811.03, 811.04, 1295, 1314, 1418	9
7	10, 205, 222, 228.01, 1291, 1296	6
8	214, 230.01, 1292, 1297	4
9	10.01, 10.02, 484, 484.01, 1277, 1293, 1298	7
10	214.01, 214.02, 214.03	3
	Total	70
	Promedio	6
	Desviacion estandar	2

Como se puede ver en la tabla el algoritmo de coloración fue capaz de asignar las clases en un horario determinado sin cruzamientos, sin embargo, algunos horarios tendrían más clases que otros. En una organización ideal el número de clases totales estaría distribuido de manera equitativa entre los ho-

rarios. En este caso en particular se tienen 70 clases y 11 horarios. En una asignación ideal el número de clases estaría los más cerca posible del promedio. Por lo que es necesario aplicar el proceso de balanceo.

Esto se hace recorriendo el color que más vértices tenga y checando si los vértices pueden ser cambiados al color con menos vértices. Este proceso se repite hasta que cada color tenga el número más cercano posible al promedio del vértices respecto al número de colores. El proceso es descrito en el siguiente algoritmo:

Entrada: Colores con sus clases.

Salida: Nueva asignación de clases a colores.

Determinar $prom$ como el promedio de clases dividiendo el número de clases entre el número de colores usados.

foreach $color \in colores$ **do**
  Determinar tam como el número de colores en la clase.
  foreach $clase \in color$ **do**
    if $tam > prom$ **then**
      Tratar de mover clase a otro color.
      if *clase no tiene vecinos en el otro color* **then**
        clase se mueve a otro color˙ $tam = tam - 1$.
      end
    end
  end
end

Algoritmo 4: Algoritmo de Balanceo.

Este algoritmo solo mueve la clase a otro color si está no tiene vecinos en el nuevo color. Esto hace que no todos los colores sean movidos aunque no se haya alcanzado el promedio. Si este es el caso dichos colores permanecen con

su mismo número de clases ó hasta lo que pudo ser reducido.

En la tabla 2.10 se ve un coloración al que se le ha aplicado el proceso de balanceo.

Tabla 2.10: Coloración balanceada.

Ranura de Tiempo	Cursos/Ranura de Tiempo	Tamaño del Cluster
0	203.01, 203.02, 203.03, 203.04, 228, 822, 1299, 1419	8
1	203, 211, 211.01, 211.02, 211.03, 215, 215.01, 215.02	8
2	10.01, 10.02, 484, 484.01, 1277, 1293, 1298	7
3	204, 204.01, 204.02, 213, 213.01, 229, 229.01	7
4	202, 206.01, 206.02, 206.03, 206.04, 1294	6
5	207, 811.01, 811.02, 811.03, 811.04, 1418	6
6	10, 205, 222, 228.01, 1291, 1296	6
7	212.01, 214.01, 214.02, 214.03, 1295, 1314	6
8	214, 230, 230.01, 1292, 1297, 1313	6
9	206, 212, 821.01, 821.02, 1420	5
10	221, 221.01, 221.02, 811, 821	5
	Total	70
	Promedio	6
	Desviacion estandar	1

2.3. Asignación

Ya que se tiene una coloración válida y balanceada se trata de asignar salones a cada una de las clases en cada uno de los colores. Esta asignación se basa en el ”problema de asignación”. El problema de asignación es un problema fundamental en optimización combinatoria (Cela, 1963). El problema consiste en asignar a un número de agentes un número de tareas. Cualquier agente puede ejecutar cualquier tarea, sin embargo diferentes agentes tienen costos diferentes para las tareas. Para desarrollar todas las tareas un solo agente tiene que desarrollar una sola tarea. Puede ser el caso de que hay más agentes que tareas.

Debido a su naturaleza combinatoria se han desarrollado varios algoritmos para la solución de este problema. Entre ellos están la solución húngara (Munkres, 1957). Así como soluciones a partir de programación lineal.(Burkard y cols., 2009). En esta investigación se utilizo una solución basada en el problema de flujo de costo mínimo (*Minimum-cost flow problem*). El problema es modelado como un grafo bipartito. En este grafo los nodos de un lado representan los agentes o trabajadores y los de la lado opuesto las tareas, mientras que, las aristas representan que trabajador puede ser asignado a que tarea. En la Figura 2.13 se ve un ejemplo de la modelación del problema como un grafo bipartito.

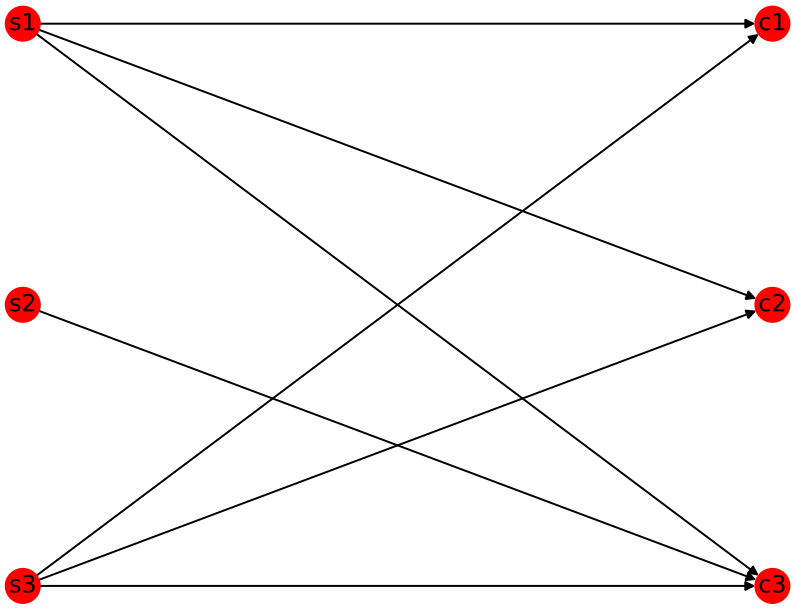


Figura 2.13: Ejemplo de grafo de asignación.

El problema consiste en encontrar la manera menos costosa de mandar una cierta cantidad de flujo a través de una red de flujo (Ahuja, 2017). Una red de flujo es un grafo dirigido $G = (V, E)$ con un vértice de origen $s \in V$ y un vértice de destino $t \in V$, donde cada arista $(u, v) \in E$ tiene una capacidad $(u, v) > 0$, flujo $f(u, v) > 0$ y un costo $a(u, v)$. El costo de mandar un flujo a través de la

arista (u, v) es $f(u, v) \cdot a(u, v)$. El problema requiere mandar una cantidad de flujo d desde inicio s a destino t.

La definición formal del problema es como minimizar el costo total del flujo en todas las aristas:

$$\sum_{(u,v)\in E} a(u, v) \cdot f(u, v)$$

con las restricciones:

- Restricción de capacidad: $f(u, v) \leq c(u, v)$
- Simetría de sesgo: $f(u, v) = -f(u, v)$
- Conservación de flujo: $\sum_{w\in V} f(u, w) = 0$ para todo $u \neq s, t$
- Flujo requerido: $\sum_{w\in V} f(s, w) = d$ y $\sum_{w\in V} f(w, t) = d$

Para plantear el problema en la asignación de salones es necesario tener una lista de salones con su capacidad de estudiantes. También es necesario el contar con la demanda que existe por cada clase en cada uno de los colores. Ya con esta información se calcula una matriz de costos.

Esta matriz de costos es calculada usando la relación entre la demanda de estudiantes y la capacidad del aula. Si un aula tiene una capacidad de 40 alumnos y una clase tiene una demanda de 30 estudiantes el costo en esa relación se obtendrá restando la capacidad menos la demanda. Es decir en esta instancia tendrá un costo de 10. Es posible que haya instancias donde la demanda sea mayor a la capacidad del aula, en este caso el cálculo del costo dará un resultado negativo. En estos casos no indica el costo si no se pone un valor 'N' indicando que no es posible. En este caso no se construirá una arista entre el

nodo del aula y el nodo de la clase.

En la tabla 2.11 se puede ver un ejemplo de la matriz de costos generada.

Tabla 2.11: Ejemplo de matriz de costos.

	Clase 1	**Clase 2**	**Clase 3**
Aula 1	47	37	57
Aula 2	N	N	10
Aula 3	48	38	58

Los costos ilustrados en esta matriz representan el costo de cada arista.

Finalmente se tiene que modificar el grafo agregando el nodo de inicio I y el nodo de fin F. Estos nodos son agregados para poder determinar el flujo que tiene que salir del nodo I y llegar a F. En el caso de la asignación de horarios la cantidad de flujo es el número de clases a las que se les quiere asignar un salón. Esto es debido a que todas las clases tienen que tener un salón pero puede haber salones vacíos. En la Figura 2.14 se puede ver el grafo resultante de esta modificación.

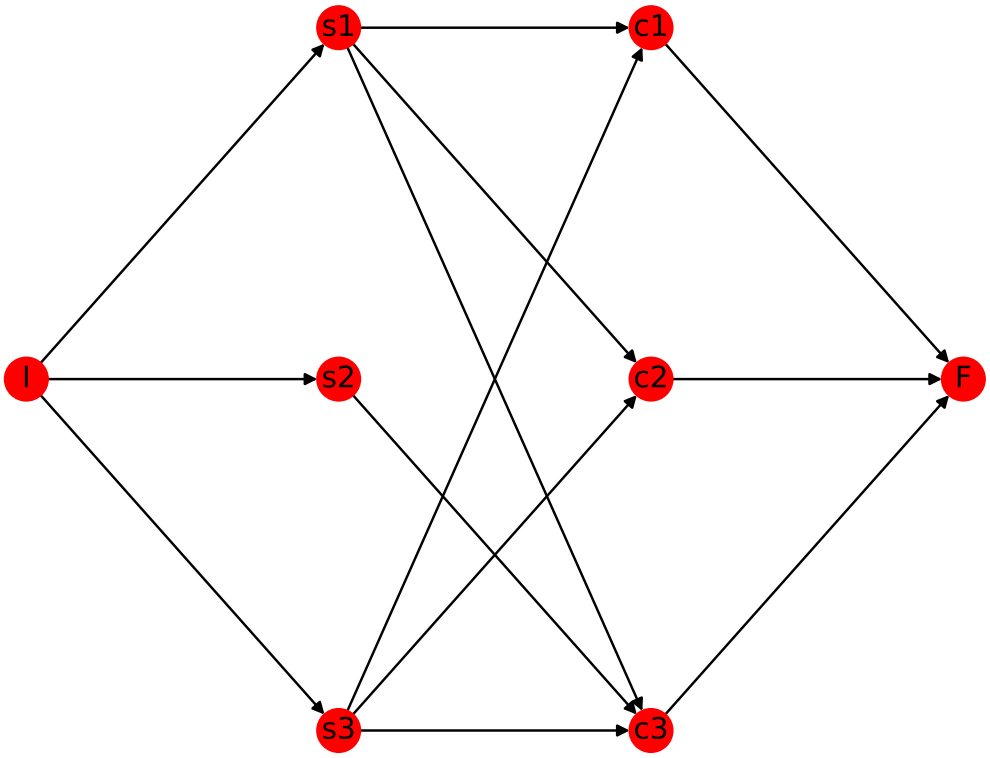


Figura 2.14: Grafo de asignación modificado.

Este grafo tiene aristas con capacidad de 1 para asegurar de que solo un salón puede ser asignado a cada clase, mientras que el costo de las aristas que salen de I y llegan a F es de 0. En la Figura 2.15 se puede ver las características finales del grafo.

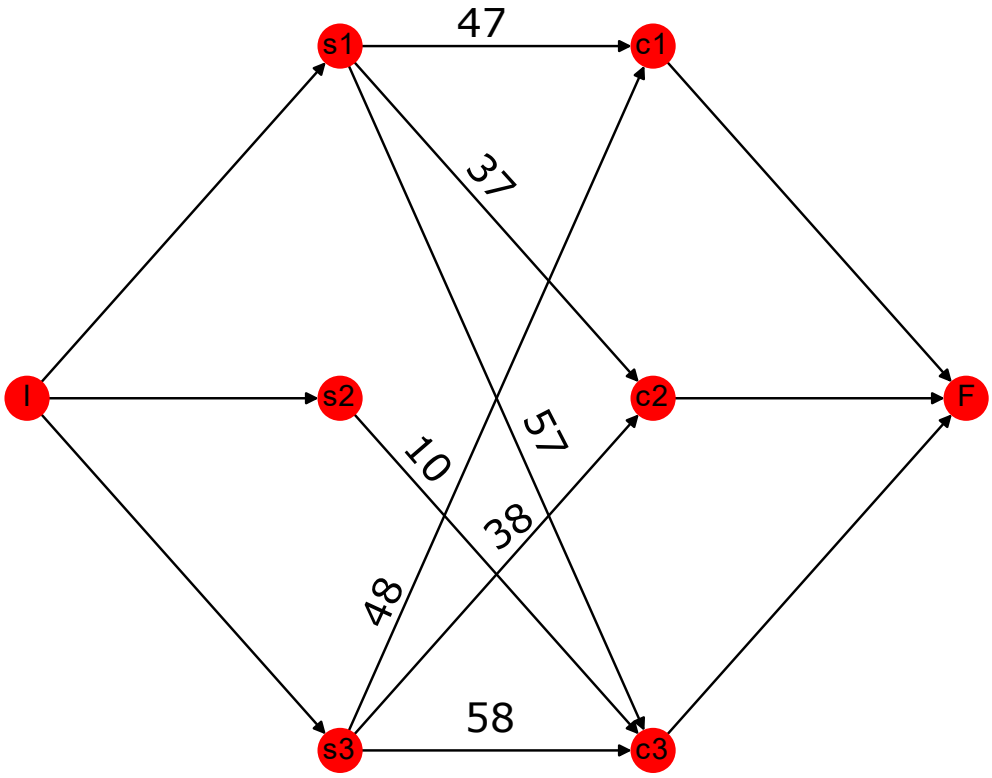


Figura 2.15: Descripción de las características del grafo.

Este grafo es resuelto con las restricciones establecidas previamente. De esta manera se asegura de que el salón asignado a cada clase tendrá el menor número de asientos libres disponibles.

CAPÍTULO 3

Resultados y Discusión

If I have seen further it is by standing on the shoulders of Giants.

Issac Newton

3.1. Resultados Experimentales

Los algoritmos fueron probados en todas las instancias y en los parámetros definidos previamente. El algoritmo que menos veces encontró el resultado óptimo fue el algoritmo de coloración por Menor Grado. El algoritmo que más tiempo de ejecución tomo fue el Algoritmo Genético. El algoritmo que más veces encontró el resultado óptimo teniendo un tiempo de ejecución razonable fue el coloración por Búsqueda Tabú. En la Figura 3.1 se pueden ver los resultados de cada coloración en sus instancias graficados, las instancias están ordenadas de menor a mayor en cuanto a número de vértices.

En la Figura 3.2 se pueden ver los resultados en cuanto a su razón de aproximación. Si el algoritmo encontró el resultado óptimo este tendría una razón de aproximación de 1.0. Las instancias se ordenan de la misma manera que en los resultados.

En la Figura 3.3 se pueden ver los tiempos de ejecución de cada algoritmo.

Esta gráfica esta en escala logarítmica debido a que ciertas instancias tuvieron un tiempo de ejecución mucho mayor.Las instancias se ordenan de la misma manera que en las gráficas anteriores.

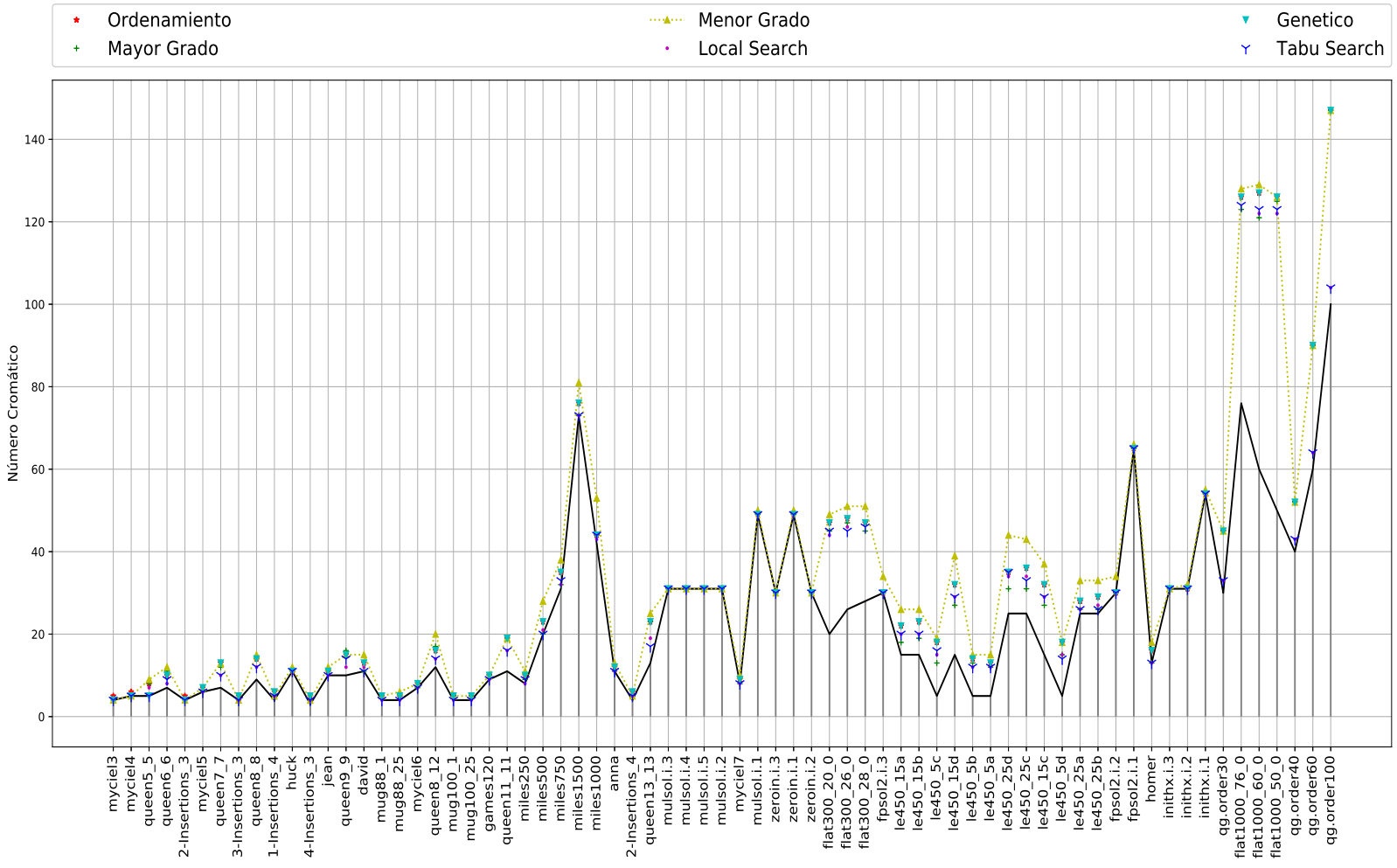


Figura 3.1: Número Cromático aproximado generado por los algoritmos.

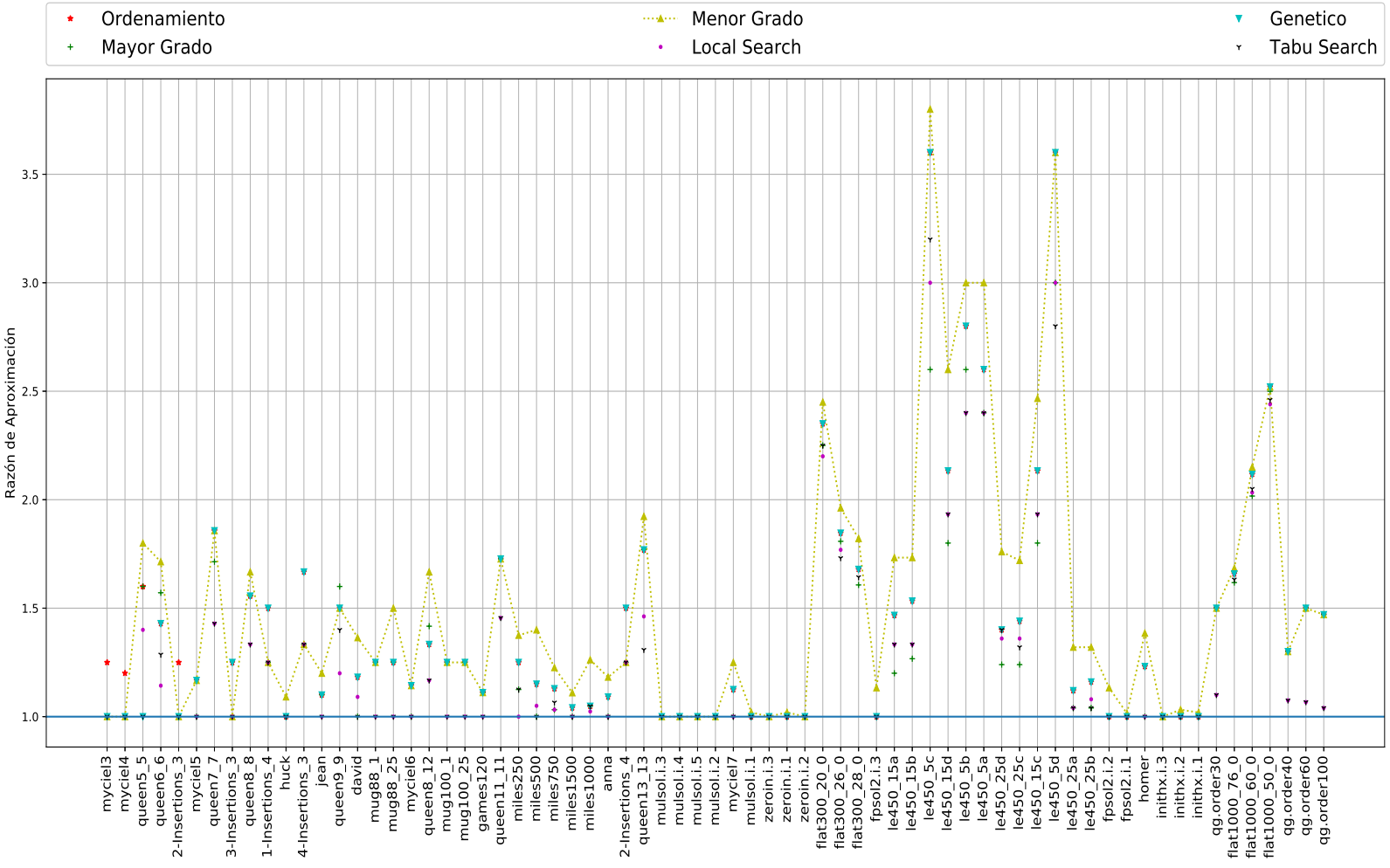


Figura 3.2: Razón de Aproximación de los algoritmos.

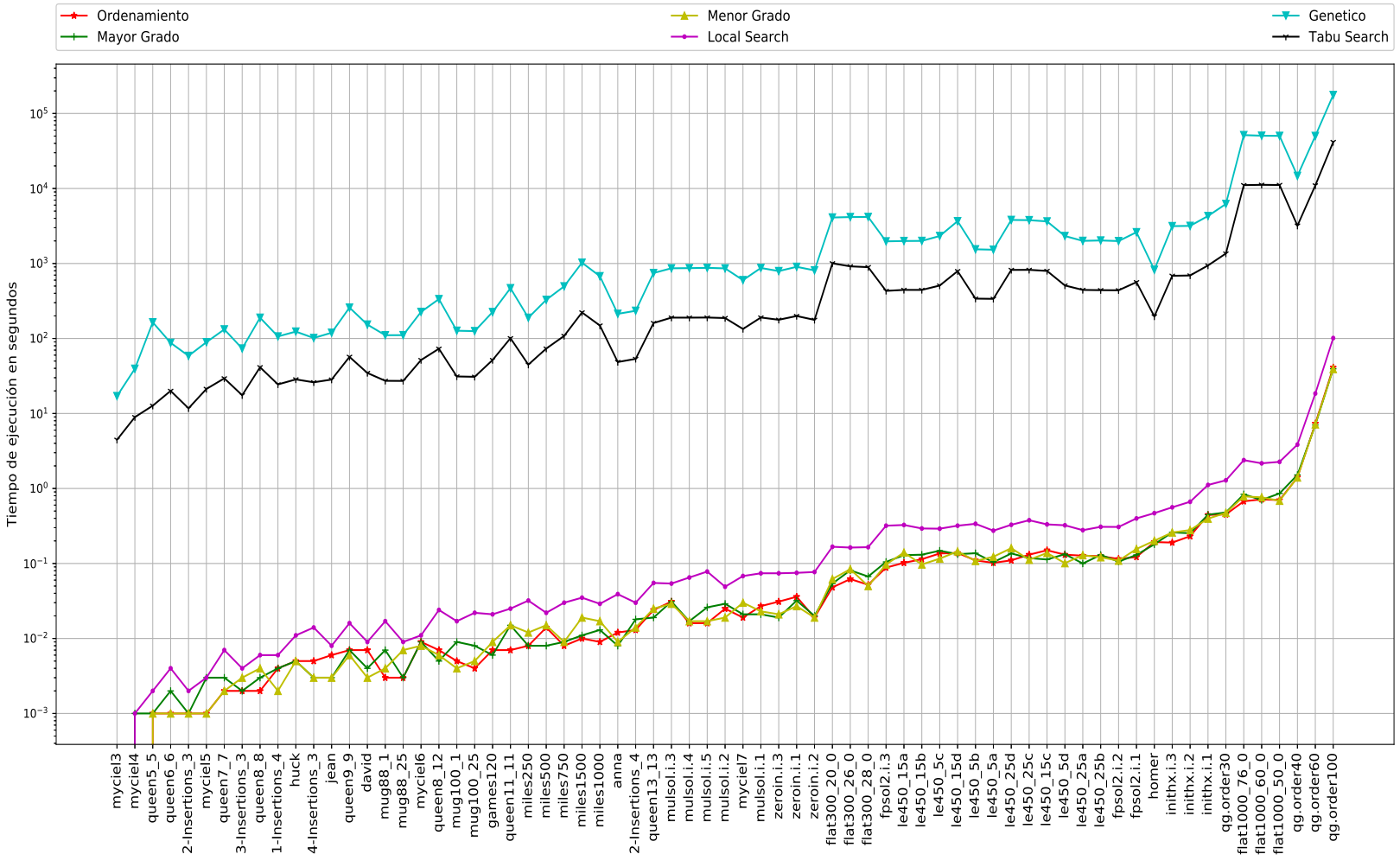


Figura 3.3: Tiempo de Ejecución de los algoritmos.

3.2. Prototipo del Sistema de Asignación de Horarios

Con base en los algoritmos implementados se creo un sistema de asignación de horarios. Este sistema implementa los algoritmos de coloración para poder determinar las materias que se pueden impartir en el mismo horario. El sistema de asignación de horarios es una herramienta web que permite la generación de horarios para la facultad de ingeniería. El sistema ejecuta la coloración y el balanceo así como la asignación de salones en base a una carga dada por el usuario en un formulario que se puede ver en la Figura 3.4. El sistema asume

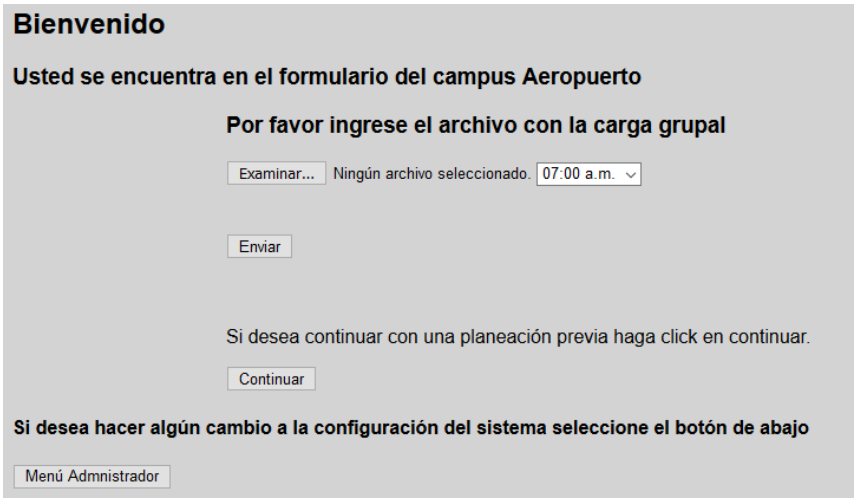


Figura 3.4: Formulario de carga.

un espacio temporal de cada grupo de clases de 5 horas, divididas en 3 bloques, 2 bloques tienen un tamaño de hora y media y el otro bloque de 2 horas. Esto es debido a que la mayoría de las clases ofrecidas por la facultad de ingeniería se dividen en 5, 3 o 2 horas a la semana. Con esta división se asegura de que las clases puedan ser acomodadas en estos espacios temporales. Este sistema también facilita la asignación permitiendo que el usuario mueva los clústers de

tiempo al horario que se necesite. Esto se puede ver en la Figura 3.5.

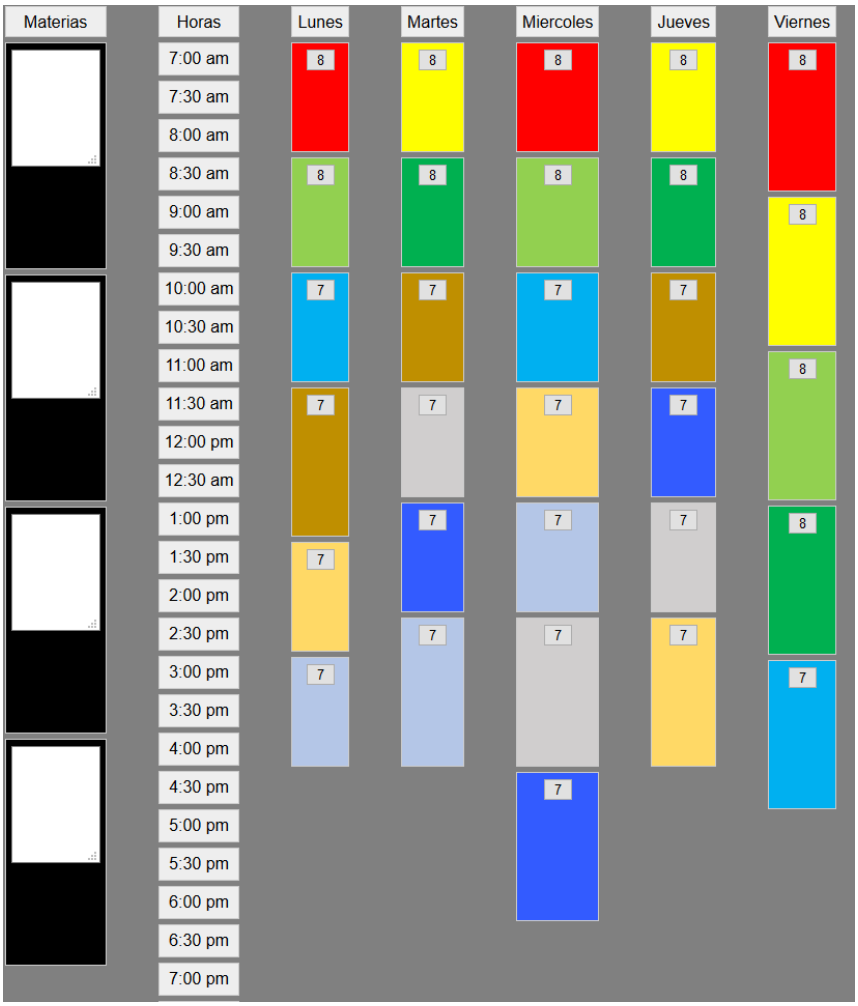


Figura 3.5: Pantalla de trabajo para asignación manual de clústers.

El sistema cuenta con una alarma que indica si se esta poniendo más de un clúster del mismo color en el mismo día. Esta alarma se puede ver en la Figura 3.6.

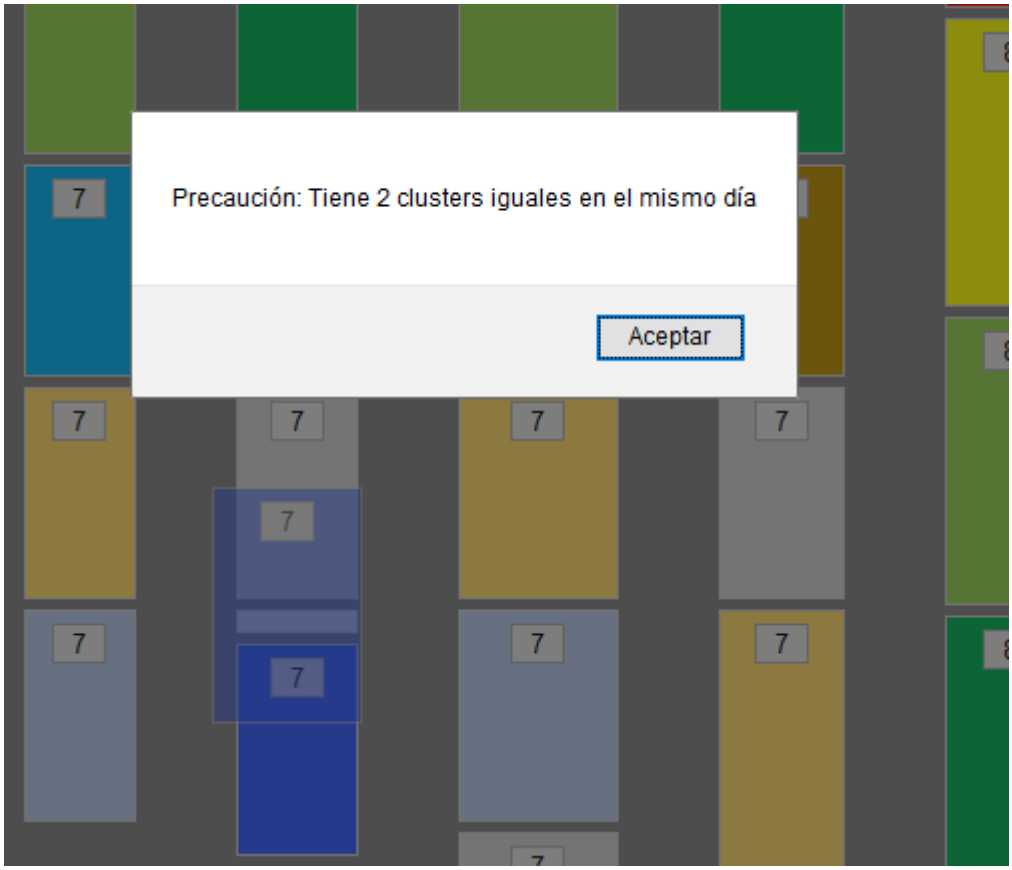


Figura 3.6: Alarma de precaución.

El sistema también informa de cuántos espacios de tiempo son necesarios así de cuántas clases hay en este. Finalmente el sistema genera un reporte basado en la asignación dada como se ve en la Figura 3.7

Lunes 7:00-8:30							
Clave	Curso	Profesor	Clave Profesor	Nomenclatura	Programa Solicitante	Salón	Edificio
214	PROBABILIDAD Y ESTADISTICA	Instructor 1	Clave 1	B1,N1	Biomedica	A12	A
822	FISICA	Instructor 2	Clave 2	B2	Biomedica	A1	A
844	SUSTENTABILIDAD	Instructor 3	Clave 3	B5	Biomedica	B12	B
894	BIOMECANICA	Instructor 4	Clave 4	B7	Biomedica	C1	C
898	PROCESAMIENTO DE BIOSENALES (PBS)	Instructor 5	Clave 5	B6	Biomedica	C3	C
483	OPTATIVA V	Instructor 6	Clave 6	B8	Biomedica	C1	C
484	FISIOLOGIA I	Instructor 7	Clave 7	B3	Biomedica	C2	C
486	BIOMAGNETISMO	Instructor 8	Clave 8	B4	Biomedica	E3	E

Figura 3.7: Reporte de Asignación.

Si se desea conocer más del sistema de asignación favor de referirse al manual de usuario que se puede ver en el apéndice H.

CAPÍTULO 4

Conclusiones

> *Nothing in this world can take the place of persistence. Talent will not; nothing is more common than unsuccessful men with talent. Genius will not; unrewarded genius is almost a proverb. Education will not; the world is full of educated derelicts. Persistence and determination alone are omnipotent. The slogan Press On! has solved and always will solve the problems of the human race.*
>
> *Calvin Coolidge*

En este trabajo se implementaron y probaron diversos algoritmos heurísticos y metaheurísticos para optimizar el problema de asignación de horarios escolares, el cual fue primeramente convertido a un problema de coloración de grafos, el cual se sabe que es un problema NP-Completo y se conjetura que no puede existir un algoritmo eficiente que pueda resolverlo. Este trabajo también muestra una aplicación práctica del trabajo realizado con anterioridad (Montuffar-Otero, 2017).

Según los resultados obtenidos se demostró que los algoritmos son eficientes para dar una solución práctica al grafo generado en un tiempo lo suficientemente corto para que esta solución tenga un uso práctico. Ahora que se sabe qué vértice pertenece a qué color se puede hacer referencia a los colores como un espacio de tiempo que se puede compartir con vértices a los que se

asignó el mismo color. Este estudio demostró ser efectivo en el diseño de un grafo que puede representar visualmente el programa de la facultad de ingeniería de la Universidad Autónoma de Querétaro.

En este trabajo también se produjo un prototipo de sistema de asignación que es eficiente en cuanto a tiempo de ejecución. Este sistema utiliza todos los algoritmos diseñados para afrontar dicho problema.

El sistema creado a partir de la investigación realizada muestra que es eficiente en la creación horarios. Este sistema cuenta con la flexibilidad de movimiento deseada a la hora de generar horarios. También garantiza que el horario generado no tendrá ningún tipo de empalmes ni de maestros o alumnos. Por lo que se refiere al modelo básico que se ha presentado en esta tesis, se puede concluir que los resultados representan la cota mínima de requerimientos de infraestructura, ya que se asume que el estudiante que cursa el programa académico es regular.

El trabajo futuro incluirá la habilidad de restringir mas la modelación del problema de tal manera que esta sea capaz de resolver más problemas de la vida real, por ejemplo que un profesor de una clase en mas de un campus. También se agregarán todas las otras facultades de la Universidad Autónoma de Querétaro y considere cualquier otro desafío que estas facultades puedan presentar, como sus áreas correspondientes y diferentes líneas terminales en las que los estudiantes pueden especializarse. A pesar de la flexibilidad del sistema actualmente no es capaz de tomar en cuenta todas las restricciones de los programas sin que se definan previamente con modificaciones al código fuente.

El prototipo de sistema desarrollado en este trabajo fue entregado en su totalidad a la Universidad Autónoma de Querétaro, con la esperanza de que este

prototipo facilite el proceso de asignación de horarios, reduciendo la cantidad de tiempo requerido para poder planear horarios de manera eficiente. Este prototipo se registrará por los medios correspondientes ante el Instituto Nacional del Derecho de Autor.

Finalmente queda señalar que el código fuente de los algoritmos, así como del prototipo del sistema de asignación de horarios están disponibles en la dirección web opcodingai.com. Sin embargo debido a cuestiones de seguridad y privacidad para obtener acceso a este recurso es necesario una clave de acceso. Dicha clave de acceso puede ser obtenida mandando un correo electrónico solicitándola, a el correo hans_mntfr@hotmail.com

Referencias

Ahuja, R. K. (2017). *Network flows: Theory, algorithms, and applications* (1st ed.). Pearson Education.

Birkhoff, G. D. (1912). A determinant formula for the number of ways of coloring a map. *Annals of Mathematics*, *14*(1/4), pp. 42-46.

Burkard, R., Dell'Amico, M., y Martello, S. (2009). *Assignment problems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Cela, E. (1963, julio). The quadratic assignment problem. *Manage. Sci.*, *9*(4), 586–599. Descargado de http://dx.doi.org/10.1287/mnsc.9.4.586 doi: 10.1287/mnsc.9.4.586

Cook, S. A. (1971). The complexity of theorem-proving procedures. En *Proceedings of the third annual acm symposium on theory of computing* (pp. 151–158). New York, NY, USA: ACM. Descargado de http://doi.acm.org/10.1145/800157.805047 doi: 10.1145/800157.805047

Garey, M., Johnson, D., y Mahoney, M. S. (1979). *Computers and intractability: A guide to the theory of np-completeness*. W. H. Freeman.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & OR*, *13*(5), 533–549.

Gonzalez, T. F. (Ed.). (2007). *Handbook of approximation algorithms and metaheuristics*. Chapman and Hall/CRC. Descargado de https://doi.org/10.1201/9781420010749 doi: 10.1201/9781420010749

Grimaldi, R. P. (1998). *Discrete and combinatorial mathematics: An applied introduction* (4th ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Hertz, d. W. D., A. (1987, diciembre). Using tabu search techniques for graph coloring. *Computing*, *39*(4), 345–351. Descargado de http://dx.doi.org/10.1007/BF02239976 doi: 10.1007/BF02239976

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Karp, R. M. (1972). Reducibility among combinatorial problems. En R. E. Miller y J. W. Thatcher (Eds.), *Complexity of computer computations* (p. 85-103). Plenum Press.

Leighton, T. (2010). Mathematics for computer science. En *6.042j / 18.062j.* Cambridge MA: MIT. Descargado de https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2010/ (MIT OpenCourseWare)

Mitchem, J. (1976, 05). On Various Algorithms for Estimating the Chromatic Number of a Graph. *The Computer Journal*, *19*(2), 182-183. Descargado de https://doi.org/10.1093/comjnl/19.2.182 doi: 10.1093/comjnl/19.2.182

Montuffar-Otero, G.-G. A. D. G.-G. F. M. e. C. . S.-G. C. M. e. D. M., H. A. (2017). Optimization of scheduling problems by coloring planning time slots for courses offered by the school of engineering: A case study. *CONIIN Looking for Solutions in Applied Engineering*(1), 96-100.

Munkres, J. (1957). *Algorithms for the assignment and transportation problems.*

Musa M. Hindi, R. V. Y. (2010). Genetic algorithm applied to the graph coloring problem. *J.B. Speed School of Engineering*.

Pearl, J. (1984). *Heuristics: Intelligent search strategies for computer problem solving*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Tannenbaum, P. (1992). *Excursions in modern mathematics with mini-excursions*. Pearson.

Thomas, R. (1998). An update on the four-color theorem. *Notices of the AMS*, 848–857.

Thomson Leighton, F. (1979, 11). A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, *84*. doi: 10.6028/jres.084.024

Wilson, R. J. (1986). *Introduction to graph theory*. New York, NY, USA: John Wiley & Sons, Inc.

A. DIMACS(*Center for Discrete Mathematics and Theoretical Computer Science*)

El desafío de DIMACS es un conjunto de instancias de grafos bidireccionales. Estos grafos vienen con sus características:

- Nombre
- Número de nodos
- Número de aristas
- En algunas de las instancias, número óptimo de coloración $\chi(G)$
- Fuente

A continuación se enlistan las fuentes de los grafos:

- DSJ: (De David Johnson (dsj@research.att.com)) Gráficos aleatorios utilizados en su trabajo con Aragon, McGeoch y Schevon, ”Optimización por el simulacro de anulación: una evaluación experimental”. Part II, Graph Coloring and Number Partitioning, Operations Research, 31, 378-406 (1991). DSJC son gráficos aleatorios estándar (n, p). DSJR son gráficos geométricos, con DSJR..c complementos de gráficos geométricos.
- CUL: (De Joe Culberson (joe@cs.ualberta.ca)) Problema de coloración generado de manera pseudo aleatoria.
- REG: (De Gary Lewandowski (gary@cs.wisc.edu)) Problema basado en la asignación de registros de memoria para variables en códigos reales.

- LEI: (De Craig Morgenstern (morgenst@riogrande.cs.tcu.edu)) Gráficos de Leighton con número cromático garantizado.
- SCH: (De Gary Lewandowski (lewandow@cs.wisc.edu)) Gráficos de programación de clases, con y sin aulas de estudio.
- LAT: (De Gary Lewandowski (lewandow@cs.wisc.edu)) Problema de la plaza latina.
- SGB: (De Michael Trick (trick@cmu.edu)) Gráficos de Stanford GraphBase de Donald Knuth. Estos se pueden dividir en: Gráficos de libros: Dado un trabajo de literatura, se crea un gráfico donde cada nodo representa un personaje. Dos nodos están conectados por una arista si los personajes correspondientes se encuentran entre sí en el libro. Knuth crea las gráficas de cinco obras clásicas: Anna Karenina (Anna) de Tolstoi, David Copperfield (David) de Dicken, Ilíada de Homer (jonrón), Huckleberry Finn de Twain (huck) y Les Miserables (Hugo) de Hugo.

  Gráficos de juego. Un grafo que representa los juegos jugados en una temporada de fútbol americano universitario puede representarse mediante una grafo donde los nodos representan a cada equipo universitario. Dos equipos están conectados por una arista si jugaron entre ellos durante la temporada. Knuth da el grafo para la temporada de fútbol americano universitario de 1990.

  Grafos de millas. Estos grafos son similares a los grafos geométricos en que los nodos se colocan en el espacio con dos nodos conectados si están lo suficientemente cerca. Estos grafos, sin embargo, no son aleatorios. Los nodos representan un conjunto de ciudades de los Estados Unidos y la distancia entre ellas viene dada por el kilometraje de la carretera desde 1947. Estos gráficos también se deben a Knuth.

Grafos de Reina. Dado un tablero de ajedrez n por n, un grafo de reina es un gráfico en n^2 nodos, cada uno correspondiente a un cuadrado del tablero. Dos nodos están conectados por una arista si los cuadrados correspondientes están en la misma fila, columna o diagonal.

- MYC: (De Michael Trick (trick@cmu.edu)) Grafos basados en la transformación de Mycielski.

- CAR: (De M. Caramia caramia@iac.rm.cnr.it y
  P. Dell'Olmo paolo.dellolmo@uniroma1.it) son una generalización de grafos myciel incrementando el número de nodos pero no la densidad.

- KOS: (De Arie Koster koster@zib.de) De casos de la vida real de diseño de redes ópticas. Cada vértices corresponde a un camino de luz en la red; las aristas corresponden a caminos que se intersectan

Nombre	Nodos	Aristas	$\chi(G)$	Fuente	Nombre	Nodos	Aristas	$\chi(G)$	Fuente
1-FullIns_3	30	100	?	CAR	latin_square_10	900	307350	?	LAT
1-FullIns_4	93	593	?	CAR	le450_15a	450	8168	15	LEI
1-FullIns_5	282	3247	?	CAR	le450_15b	450	8169	15	LEI
1-Insertions_4	67	232	4	CAR	le450_15c	450	16680	15	LEI
1-Insertions_5	202	1227	?	CAR	le450_15d	450	16750	15	LEI
1-Insertions_6	607	6337	?	CAR	le450_25a	450	8260	25	LEI
2-FullIns_3	52	201	?	CAR	le450_25b	450	8263	25	LEI
2-FullIns_4	212	1621	?	CAR	le450_25c	450	17343	25	LEI
2-FullIns_5	852	12201	?	CAR	le450_25d	450	17425	25	LEI
2-Insertions_3	37	72	4	CAR	le450_5a	450	5714	5	LEI
2-Insertions_4	149	541	4	CAR	le450_5b	450	5734	5	LEI
2-Insertions_5	597	3936	?	CAR	le450_5c	450	9803	5	LEI
3-FullIns_3	80	346	?	CAR	le450_5d	450	9757	5	LEI
3-FullIns_4	405	3524	?	CAR	miles1000	128	3216	42	SGB
3-FullIns_5	2030	33751	?	CAR	miles1500	128	5198	73	SGB

3-Insertions_3	56	110	4	CAR	miles250	128	387	8	SGB
3-Insertions_4	281	1046	?	CAR	miles500	128	1170	20	SGB
3-Insertions_5	1406	9695	?	CAR	miles750	128	2113	31	SGB
4-FullIns_3	114	541	?	CAR	mug100_1	100	166	4	MIZ
4-FullIns_4	690	6650	?	CAR	mug100_25	100	166	4	MIZ
4-FullIns_5	4146	77305	?	CAR	mug88_1	88	146	4	MIZ
4-Insertions_3	79	156	3	CAR	mug88_25	88	146	4	MIZ
4-Insertions_4	475	1795	?	CAR	mulsol.i.1	197	3925	49	REG
5-FullIns_3	154	792	?	CAR	mulsol.i.2	188	3885	31	REG
5-FullIns_4	1085	11395	?	CAR	mulsol.i.3	184	3916	31	REG
DSJC1000.1	1000	99258	?	DSJ	mulsol.i.4	185	3946	31	REG
DSJC1000.5	1000	499652	?	DSJ	mulsol.i.5	186	3973	31	REG
DSJC1000.9	1000	898898	?	DSJ	myciel3	11	20	4	MYC
DSJC125.1	125	1472	?	DSJ	myciel4	23	71	5	MYC
DSJC125.5	125	7782	?	DSJ	myciel5	47	236	6	MYC
DSJC125.9	125	13922	?	DSJ	myciel6	95	755	7	MYC
DSJC250.1	250	6436	?	DSJ	myciel7	191	2360	8	MYC
DSJC250.5	250	31366	?	DSJ	qg.order100	10000	990000	100	GOM
DSJC250.9	250	55794	?	DSJ	qg.order30	900	26100	30	GOM
DSJC500.1	500	24916	?	DSJ	qg.order40	1600	62400	40	GOM
DSJC500.5	500	125249	?	DSJ	qg.order60	3600	212400	60	GOM
DSJC500.9	500	112437	?	DSJ	queen10_10	100	1470	?	SGB
DSJR500.1	500	7110	?	DSJ	queen11_11	121	1980	11	SGB
DSJR500.1c	500	242550	?	DSJ	queen12_12	144	2596	?	SGB
DSJR500.5	500	117724	?	DSJ	queen13_13	169	3328	13	SGB
abb313GPIA	1557	53356	?	HOS	queen14_14	196	4186	?	SGB
anna	138	493	11	SGB	queen15_15	225	5180	?	SGB
ash331GPIA	662	4185	?	HOS	queen16_16	256	6320	?	SGB
ash608GPIA	1216	7844	?	HOS	queen5_5	25	160	5	SGB
ash958GPIA	1916	12506	?	HOS	queen6_6	36	290	7	SGB
david	87	406	11	SGB	queen7_7	49	476	7	SGB
flat1000_50_0	1000	245000	50	CUL	queen8_12	96	1368	12	SGB

flat1000_60_0	1000	245830	60	CUL	queen8_8	64	728	9	SGB
flat1000_76_0	1000	246708	76	CUL	queen9_9	81	1056	10	SGB
flat300_20_0	300	21375	20	CUL	school1	385	19095	?	SCH
flat300_26_0	300	21633	26	CUL	school1_nsh	352	14612	?	SCH
flat300_28_0	300	21695	28	CUL	wap01a	2368	110871	?	KOS
fpsol2.i.1	496	11654	65	REG	wap02a	2464	111742	?	KOS
fpsol2.i.2	451	8691	30	REG	wap03a	4730	286722	?	KOS
fpsol2.i.3	425	8688	30	REG	wap04a	5231	294902	?	KOS
games120	120	638	9	SGB	wap05a	905	43081	?	KOS
homer	561	1629	13	SGB	wap06a	947	43571	?	KOS
huck	74	301	11	SGB	wap07a	1809	103368	?	KOS
inithx.i.1	864	18707	54	REG	wap08a	1870	104176	?	KOS
inithx.i.2	645	13979	31	REG	will199GPIA	701	6772	?	HOS
inithx.i.3	621	13969	31	REG	zeroin.i.1	211	4100	49	REG
jean	80	254	10	SGB	zeroin.i.2	211	3541	30	REG

Tabla A.1: Información de las instancias de DIMACS

B. Resultados de coloración por ordenamiento

$\chi(G)$ representa el número cromático, es decir el mínimo (óptimo) número de colores necesarios para colorear apropiadamente el grafo.

$\hat{f}$ representa el número cromático aproximado generado por el algoritmo.

$\frac{\hat{f}}{\chi(G)}$ representa la razón de aproximación.

Notas:

1. El algoritmo de coloración por ordenamiento encontró el número óptimo de colores en 15 instancias de un total de 69.
2. La razón de aproximación máxima $\frac{\hat{f}}{\chi(G)}$ es de 3.6.
3. El promedio de la razón de aproximación es 1.450464.

Nombre de la instancia	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$	**Nombre de la instancia**	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$
1-FullIns_3	9	?	0.001		le450_15b	23	15	0.114	1.533
1-FullIns_4	12	?	0.008		le450_15c	32	15	0.15	2.133
1-FullIns_5	15	?	0.053		le450_15d	32	15	0.138	2.133
1-Insertions_4	6	4	0.004	1.5	le450_25a	28	25	0.127	1.12
1-Insertions_5	7	?	0.027		le450_25b	29	25	0.125	1.16
1-Insertions_6	8	?	0.171		le450_25c	36	25	0.131	1.44
2-FullIns_3	11	?	0.003		le450_25d	35	25	0.11	1.4
2-FullIns_4	15	?	0.036		le450_5a	13	5	0.102	2.6
2-FullIns_5	19	?	0.411		le450_5b	14	5	0.11	2.8
2-Insertions_3	5	4	0.001	1.25	le450_5c	18	5	0.136	3.6
2-Insertions_4	6	4	0.013	1.5	le450_5d	18	5	0.131	3.6

Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$	Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$
2-Insertions_5	7	?	0.159		miles1000	44	42	0.009	1.048
3-FullIns_3	13	?	0.006		miles1500	76	73	0.01	1.041
3-FullIns_4	18	?	0.125		miles250	10	8	0.008	1.25
3-FullIns_5	23	?	2.255		miles500	23	20	0.014	1.15
3-Insertions_3	5	4	0.002	1.25	miles750	35	31	0.008	1.129
3-Insertions_4	6	?	0.056		mug100_1	5	4	0.005	1.25
3-Insertions_5	7	?	1.122		mug100_25	5	4	0.004	1.25
4-FullIns_3	15	?	0.011		mug88_1	5	4	0.003	1.25
4-FullIns_4	21	?	0.298		mug88_25	5	4	0.003	1.25
4-FullIns_5	27	?	9.053		mulsol.i.1	49	49	0.027	1.0
4-Insertions_3	5	3	0.005	1.667	mulsol.i.2	31	31	0.025	1.0
4-Insertions_4	6	?	0.125		mulsol.i.3	31	31	0.031	1.0
5-FullIns_3	17	?	0.012		mulsol.i.4	31	31	0.016	1.0
5-FullIns_4	24	?	0.642		mulsol.i.5	31	31	0.016	1.0
DSJC1000.1	32	?	0.633		myciel3	5	4	0.0	1.25
DSJC1000.5	128	?	0.657		myciel4	6	5	0.0	1.2
DSJC1000.9	322	?	0.957		myciel5	7	6	0.001	1.167
DSJC125.1	8	?	0.011		myciel6	8	7	0.009	1.143
DSJC125.5	26	?	0.016		myciel7	9	8	0.019	1.125
DSJC125.9	57	?	0.016		qg.order100	147	100	40.958	1.47
DSJC250.1	13	?	0.031		qg.order30	45	30	0.451	1.5
DSJC250.5	44	?	0.036		qg.order40	52	40	1.404	1.3
DSJC250.9	100	?	0.039		qg.order60	90	60	7.334	1.5
DSJC500.1	20	?	0.169		queen10_10	19	?	0.01	
DSJC500.5	74	?	0.213		queen11_11	19	11	0.007	1.727
DSJC500.9	173	?	0.212		queen12_12	21	?	0.013	
DSJR500.1	15	?	0.125		queen13_13	23	13	0.024	1.769
DSJR500.1c	110	?	0.221		queen14_14	23	?	0.033	
DSJR500.5	145	?	0.194		queen15_15	25	?	0.044	

Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$	Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$
abb313GPIA	15	?	1.411		queen16_16	26	?	0.041	
anna	12	11	0.012	1.091	queen5_5	8	5	0.001	1.6
ash331GPIA	11	?	0.283		queen6_6	10	7	0.001	1.429
ash608GPIA	9	?	0.792		queen7_7	13	7	0.002	1.857
ash958GPIA	11	?	2.001		queen8_12	16	12	0.007	1.333
david	13	11	0.007	1.182	queen8_8	14	9	0.002	1.556
flat1000_50_0	126	50	0.7	2.52	queen9_9	15	10	0.007	1.5
flat1000_60_0	127	60	0.716	2.117	school1	44	?	0.091	
flat1000_76_0	126	76	0.675	1.658	school1_nsh	41	?	0.103	
flat300_20_0	47	20	0.048	2.35	wap01a	62	?	3.114	
flat300_26_0	48	26	0.062	1.846	wap02a	61	?	3.274	
flat300_28_0	47	28	0.052	1.679	wap03a	73	?	11.289	
fpsol2.i.1	65	65	0.122	1.0	wap04a	69	?	12.944	
fpsol2.i.2	30	30	0.115	1.0	wap05a	64	?	0.433	
fpsol2.i.3	30	30	0.088	1.0	wap06a	62	?	0.506	
games120	10	9	0.007	1.111	wap07a	73	?	1.885	
homer	16	13	0.193	1.231	wap08a	73	?	1.921	
huck	11	11	0.005	1.0	will199GPIA	13	?	0.262	
inithx.i.1	54	54	0.444	1.0	zeroin.i.1	49	49	0.036	1.0
inithx.i.2	31	31	0.231	1.0	zeroin.i.2	30	30	0.019	1.0
inithx.i.3	31	31	0.19	1.0	zeroin.i.3	30	30	0.031	1.0

Tabla B.1: Resultados de Coloración por Ordenamiento

C. Resultados de coloración por mayor grado

$\chi(G)$ representa el número cromático, es decir el mínimo (óptimo) número de colores necesarios para colorear apropiadamente el grafo.

$\hat{f}$ representa el número cromático aproximado generado por el algoritmo.

$\frac{\hat{f}}{\chi(G)}$ representa la razón de aproximación.

Notas:

1. El algoritmo de coloración por mayor grado encontró el número óptimo de colores en 27 instancias de un total de 69.
2. La razón de aproximación máxima $\frac{\hat{f}}{\chi(G)}$ es de 3.
3. El promedio de la razón de aproximación es 1.347826.

Nombre de la instancia	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$	**Nombre de la instancia**	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$
1-FullIns_3	5	?	0.001		le450_15b	19	15	0.131	1.267
1-FullIns_4	6	?	0.009		le450_15c	27	15	0.113	1.8
1-FullIns_5	7	?	0.059		le450_15d	27	15	0.133	1.8
1-Insertions_4	5	4	0.004	1.25	le450_25a	26	25	0.1	1.04
1-Insertions_5	6	?	0.021		le450_25b	26	25	0.13	1.04
1-Insertions_6	7	?	0.235		le450_25c	31	25	0.118	1.24
2-FullIns_3	6	?	0.001		le450_25d	31	25	0.136	1.24
2-FullIns_4	7	?	0.024		le450_5a	12	5	0.104	2.4
2-FullIns_5	8	?	0.355		le450_5b	13	5	0.137	2.6
2-Insertions_3	4	4	0.001	1.0	le450_5c	13	5	0.148	2.6
2-Insertions_4	5	4	0.018	1.25	le450_5d	15	5	0.133	3.0

Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$	Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$
2-Insertions_5	6	?	0.233		miles1000	44	42	0.013	1.048
3-FullIns_3	7	?	0.007		miles1500	73	73	0.011	1.0
3-FullIns_4	8	?	0.092		miles250	9	8	0.008	1.125
3-FullIns_5	9	?	2.274		miles500	20	20	0.008	1.0
3-Insertions_3	4	4	0.002	1.0	miles750	32	31	0.009	1.032
3-Insertions_4	5	?	0.061		mug100_1	5	4	0.009	1.25
3-Insertions_5	6	?	1.074		mug100_25	5	4	0.008	1.25
4-FullIns_3	8	?	0.011		mug88_1	5	4	0.007	1.25
4-FullIns_4	9	?	0.227		mug88_25	5	4	0.003	1.25
4-FullIns_5	10	?	9.805		mulsol.i.1	49	49	0.021	1.0
4-Insertions_3	4	3	0.003	1.333	mulsol.i.2	31	31	0.029	1.0
4-Insertions_4	5	?	0.139		mulsol.i.3	31	31	0.031	1.0
5-FullIns_3	9	?	0.01		mulsol.i.4	31	31	0.017	1.0
5-FullIns_4	10	?	0.683		mulsol.i.5	31	31	0.026	1.0
DSJC1000.1	30	?	0.632		myciel3	4	4	0.0	1.0
DSJC1000.5	122	?	0.792		myciel4	5	5	0.001	1.0
DSJC1000.9	315	?	0.968		myciel5	6	6	0.003	1.0
DSJC125.1	8	?	0.007		myciel6	7	7	0.009	1.0
DSJC125.5	25	?	0.017		myciel7	8	8	0.021	1.0
DSJC125.9	54	?	0.011		qg.order100	147	100	39.413	1.47
DSJC250.1	12	?	0.028		qg.order30	45	30	0.479	1.5
DSJC250.5	40	?	0.063		qg.order40	52	40	1.518	1.3
DSJC250.9	93	?	0.048		qg.order60	90	60	7.031	1.5
DSJC500.1	18	?	0.149		queen10_10	18	?	0.009	
DSJC500.5	72	?	0.234		queen11_11	19	11	0.015	1.727
DSJC500.9	170	?	0.203		queen12_12	21	?	0.014	
DSJR500.1	14	?	0.108		queen13_13	23	13	0.019	1.769
DSJR500.1c	100	?	0.307		queen14_14	23	?	0.036	
DSJR500.5	135	?	0.172		queen15_15	25	?	0.044	

Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$	Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$
abb313GPIA	16	?	1.34		queen16_16	28	?	0.036	
anna	11	11	0.008	1.0	queen5_5	8	5	0.001	1.6
ash331GPIA	10	?	0.243		queen6_6	11	7	0.002	1.571
ash608GPIA	9	?	0.87		queen7_7	12	7	0.003	1.714
ash958GPIA	10	?	2.104		queen8_12	17	12	0.005	1.417
david	11	11	0.004	1.0	queen8_8	14	9	0.003	1.556
flat1000_50_0	125	50	0.859	2.5	queen9_9	16	10	0.007	1.6
flat1000_60_0	121	60	0.703	2.017	school1	33	?	0.1	
flat1000_76_0	123	76	0.836	1.618	school1_nsh	33	?	0.094	
flat300_20_0	45	20	0.054	2.25	wap01a	52	?	3.372	
flat300_26_0	47	26	0.081	1.808	wap02a	49	?	3.328	
flat300_28_0	45	28	0.067	1.607	wap03a	55	?	10.542	
fpsol2.i.1	65	65	0.13	1.0	wap04a	51	?	12.464	
fpsol2.i.2	30	30	0.106	1.0	wap05a	51	?	0.456	
fpsol2.i.3	30	30	0.105	1.0	wap06a	48	?	0.598	
games120	10	9	0.006	1.111	wap07a	51	?	1.981	
homer	13	13	0.18	1.0	wap08a	52	?	1.885	
huck	11	11	0.005	1.0	will199GPIA	12	?	0.244	
inithx.i.1	54	54	0.447	1.0	zeroin.i.1	49	49	0.032	1.0
inithx.i.2	31	31	0.253	1.0	zeroin.i.2	30	30	0.02	1.0
inithx.i.3	31	31	0.26	1.0	zeroin.i.3	30	30	0.019	1.0

Tabla C.1: Resultados de Coloración por Mayor Grado

D. Resultados de coloración por menor grado

$\chi(G)$ representa el número cromático, es decir el mínimo (óptimo) número de colores necesarios para colorear apropiadamente el grafo.

$\hat{f}$ representa el número cromático aproximado generado por el algoritmo.

$\frac{\hat{f}}{\chi(G)}$ representa la razón de aproximación.

Notas:

1. El algoritmo de coloración por menor grado encontró el número óptimo de colores en 11 instancias de un total de 69.
2. La razón de aproximación máxima $\frac{\hat{f}}{\chi(G)}$ es de 3.8.
3. El promedio de la razón de aproximación es 1.521536.

Nombre de la instancia	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$	**Nombre de la instancia**	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$
1-FullIns_3	7	?	0.001		le450_15b	26	15	0.097	1.733
1-FullIns_4	7	?	0.004		le450_15c	37	15	0.138	2.467
1-FullIns_5	9	?	0.054		le450_15d	39	15	0.145	2.6
1-Insertions_4	5	4	0.002	1.25	le450_25a	33	25	0.129	1.32
1-Insertions_5	7	?	0.021		le450_25b	33	25	0.121	1.32
1-Insertions_6	8	?	0.212		le450_25c	43	25	0.112	1.72
2-FullIns_3	8	?	0.001		le450_25d	44	25	0.16	1.76
2-FullIns_4	10	?	0.028		le450_5a	15	5	0.123	3.0
2-FullIns_5	13	?	0.379		le450_5b	15	5	0.108	3.0
2-Insertions_3	4	4	0.001	1.0	le450_5c	19	5	0.116	3.8
2-Insertions_4	5	4	0.014	1.25	le450_5d	18	5	0.101	3.6

Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$	Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$
2-Insertions_5	6	?	0.212		miles1000	53	42	0.017	1.262
3-FullIns_3	10	?	0.006		miles1500	81	73	0.019	1.11
3-FullIns_4	11	?	0.111		miles250	11	8	0.012	1.375
3-FullIns_5	15	?	2.344		miles500	28	20	0.015	1.4
3-Insertions_3	4	4	0.003	1.0	miles750	38	31	0.009	1.226
3-Insertions_4	5	?	0.062		mug100_1	5	4	0.004	1.25
3-Insertions_5	6	?	1.085		mug100_25	5	4	0.005	1.25
4-FullIns_3	12	?	0.011		mug88_1	5	4	0.004	1.25
4-FullIns_4	14	?	0.308		mug88_25	6	4	0.007	1.5
4-FullIns_5	16	?	9.429		mulsol.i.1	50	49	0.023	1.02
4-Insertions_3	4	3	0.003	1.333	mulsol.i.2	31	31	0.019	1.0
4-Insertions_4	5	?	0.105		mulsol.i.3	31	31	0.029	1.0
5-FullIns_3	14	?	0.01		mulsol.i.4	31	31	0.017	1.0
5-FullIns_4	17	?	0.596		mulsol.i.5	31	31	0.017	1.0
DSJC1000.1	34	?	0.613		myciel3	4	4	0.0	1.0
DSJC1000.5	130	?	0.792		myciel4	5	5	0.0	1.0
DSJC1000.9	331	?	0.966		myciel5	7	6	0.001	1.167
DSJC125.1	9	?	0.013		myciel6	8	7	0.008	1.143
DSJC125.5	27	?	0.017		myciel7	10	8	0.03	1.25
DSJC125.9	58	?	0.017		qg.order100	147	100	39.137	1.47
DSJC250.1	14	?	0.036		qg.order30	45	30	0.473	1.5
DSJC250.5	44	?	0.042		qg.order40	52	40	1.402	1.3
DSJC250.9	102	?	0.041		qg.order60	90	60	7.161	1.5
DSJC500.1	21	?	0.156		queen10_10	17	?	0.011	
DSJC500.5	77	?	0.213		queen11_11	19	11	0.015	1.727
DSJC500.9	184	?	0.231		queen12_12	20	?	0.015	
DSJR500.1	18	?	0.139		queen13_13	25	13	0.025	1.923
DSJR500.1c	108	?	0.253		queen14_14	24	?	0.021	
DSJR500.5	200	?	0.192		queen15_15	25	?	0.042	

Nombre de la instancia	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$	**Nombre de la instancia**	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$
abb313GPIA	15	?	1.311		queen16_16	27	?	0.04	
anna	13	11	0.009	1.182	queen5_5	9	5	0.001	1.8
ash331GPIA	9	?	0.26		queen6_6	12	7	0.001	1.714
ash608GPIA	9	?	0.849		queen7_7	13	7	0.002	1.857
ash958GPIA	9	?	2.031		queen8_12	20	12	0.006	1.667
david	15	11	0.003	1.364	queen8_8	15	9	0.004	1.667
flat1000_50_0	126	50	0.686	2.52	queen9_9	15	10	0.006	1.5
flat1000_60_0	129	60	0.762	2.15	school1	47	?	0.099	
flat1000_76_0	128	76	0.784	1.684	school1_nsh	44	?	0.093	
flat300_20_0	49	20	0.062	2.45	wap01a	65	?	3.157	
flat300_26_0	51	26	0.084	1.962	wap02a	66	?	3.278	
flat300_28_0	51	28	0.05	1.821	wap03a	74	?	10.527	
fpsol2.i.1	66	65	0.156	1.015	wap04a	75	?	12.419	
fpsol2.i.2	34	30	0.109	1.133	wap05a	64	?	0.481	
fpsol2.i.3	34	30	0.097	1.133	wap06a	63	?	0.544	
games120	10	9	0.009	1.111	wap07a	72	?	1.738	
homer	18	13	0.201	1.385	wap08a	71	?	1.873	
huck	12	11	0.005	1.091	will199GPIA	12	?	0.269	
inithx.i.1	55	54	0.396	1.019	zeroin.i.1	50	49	0.027	1.02
inithx.i.2	32	31	0.279	1.032	zeroin.i.2	30	30	0.019	1.0
inithx.i.3	31	31	0.259	1.0	zeroin.i.3	30	30	0.021	1.0

Tabla D.1: Resultados de Coloración por Menor Grado

E. Resultados de coloración por Búsqueda Local

$\chi(G)$ representa el número cromático, es decir el mínimo (óptimo) número de colores necesarios para colorear apropiadamente el grafo.

$\hat{f}$ representa el número cromático aproximado generado por el algoritmo.

$\frac{\hat{f}}{\chi(G)}$ representa la razón de aproximación.

Notas:

1. El algoritmo de coloración por Búsqueda Local encontró el número óptimo de colores en 32 instancias de un total de 69.
2. La razón de aproximación máxima $\frac{\hat{f}}{\chi(G)}$ es de 3.
3. El promedio de la razón de aproximación es 1.287710.

Nombre de la instancia	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$	**Nombre de la instancia**	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$
1-FullIns_3	4	?	0.004		le450_15b	20	15	0.293	1.333
1-FullIns_4	6	?	0.025		le450_15c	29	15	0.332	1.933
1-FullIns_5	6	?	0.119		le450_15d	29	15	0.319	1.933
1-Insertions_4	5	4	0.006	1.25	le450_25a	26	25	0.278	1.04
1-Insertions_5	7	?	0.066		le450_25b	27	25	0.308	1.08
1-Insertions_6	7	?	0.488		le450_25c	34	25	0.377	1.36
2-FullIns_3	5	?	0.007		le450_25d	34	25	0.328	1.36
2-FullIns_4	7	?	0.081		le450_5a	12	5	0.274	2.4
2-FullIns_5	8	?	1.036		le450_5b	12	5	0.338	2.4
2-Insertions_3	4	4	0.002	1.0	le450_5c	15	5	0.291	3.0
2-Insertions_4	5	4	0.03	1.25	le450_5d	15	5	0.323	3.0

Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$	Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$
2-Insertions_5	6	?	0.52		miles1000	43	42	0.029	1.024
3-FullIns_3	6	?	0.01		miles1500	73	73	0.035	1.0
3-FullIns_4	7	?	0.222		miles250	8	8	0.032	1.0
3-FullIns_5	10	?	5.808		miles500	21	20	0.022	1.05
3-Insertions_3	4	4	0.004	1.0	miles750	32	31	0.03	1.032
3-Insertions_4	5	?	0.104		mug100_1	4	4	0.017	1.0
3-Insertions_5	7	?	2.79		mug100_25	4	4	0.022	1.0
4-FullIns_3	7	?	0.015		mug88_1	4	4	0.017	1.0
4-FullIns_4	8	?	0.661		mug88_25	4	4	0.009	1.0
4-FullIns_5	9	?	24.194		mulsol.i.1	49	49	0.074	1.0
4-Insertions_3	4	3	0.014	1.333	mulsol.i.2	31	31	0.049	1.0
4-Insertions_4	5	?	0.294		mulsol.i.3	31	31	0.054	1.0
5-FullIns_3	8	?	0.05		mulsol.i.4	31	31	0.065	1.0
5-FullIns_4	9	?	1.652		mulsol.i.5	31	31	0.078	1.0
DSJC1000.1	31	?	1.527		myciel3	4	4	0.0	1.0
DSJC1000.5	126	?	2.466		myciel4	5	5	0.001	1.0
DSJC1000.9	308	?	4.699		myciel5	6	6	0.003	1.0
DSJC125.1	7	?	0.025		myciel6	7	7	0.011	1.0
DSJC125.5	24	?	0.031		myciel7	8	8	0.068	1.0
DSJC125.9	54	?	0.031		qg.order100	104	100	101.26	1.04
DSJC250.1	12	?	0.112		qg.order30	33	30	1.285	1.1
DSJC250.5	41	?	0.147		qg.order40	43	40	3.849	1.075
DSJC250.9	94	?	0.154		qg.order60	64	60	18.462	1.067
DSJC500.1	18	?	0.387		queen10_10	14	?	0.014	
DSJC500.5	72	?	0.627		queen11_11	16	11	0.025	1.455
DSJC500.9	166	?	0.948		queen12_12	17	?	0.043	
DSJR500.1	13	?	0.373		queen13_13	19	13	0.055	1.462
DSJR500.1c	100	?	0.869		queen14_14	19	?	0.054	
DSJR500.5	143	?	0.601		queen15_15	21	?	0.083	

Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$	Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$
abb313GPIA	14	?	3.337		queen16_16	22	?	0.097	
anna	11	11	0.039	1.0	queen5_5	7	5	0.002	1.4
ash331GPIA	6	?	0.634		queen6_6	8	7	0.004	1.143
ash608GPIA	6	?	2.077		queen7_7	10	7	0.007	1.429
ash958GPIA	6	?	5.083		queen8_12	14	12	0.024	1.167
david	12	11	0.009	1.091	queen8_8	12	9	0.006	1.333
flat1000_50_0	122	50	2.266	2.44	queen9_9	12	10	0.016	1.2
flat1000_60_0	122	60	2.166	2.033	school1	39	?	0.228	
flat1000_76_0	126	76	2.385	1.658	school1_nsh	38	?	0.222	
flat300_20_0	44	20	0.167	2.2	wap01a	55	?	8.143	
flat300_26_0	46	26	0.163	1.769	wap02a	53	?	8.795	
flat300_28_0	47	28	0.165	1.679	wap03a	59	?	28.288	
fpsol2.i.1	65	65	0.399	1.0	wap04a	58	?	32.961	
fpsol2.i.2	30	30	0.307	1.0	wap05a	53	?	1.234	
fpsol2.i.3	30	30	0.319	1.0	wap06a	51	?	1.453	
games120	9	9	0.021	1.0	wap07a	57	?	4.633	
homer	13	13	0.469	1.0	wap08a	56	?	4.837	
huck	11	11	0.011	1.0	will199GPIA	9	?	0.692	
inithx.i.1	54	54	1.114	1.0	zeroin.i.1	49	49	0.075	1.0
inithx.i.2	31	31	0.665	1.0	zeroin.i.2	30	30	0.077	1.0
inithx.i.3	31	31	0.561	1.0	zeroin.i.3	30	30	0.074	1.0

Tabla E.1: Resultados de Coloración por Búsqueda Local

F. Resultados de coloración por Búsqueda Tabú

$\chi(G)$ representa el número cromático, es decir el mínimo (óptimo) número de colores necesarios para colorear apropiadamente el grafo.

$\hat{f}$ representa el número cromático aproximado generado por el algoritmo.

$\frac{\hat{f}}{\chi(G)}$ representa la razón de aproximación.

Notas:

1. El algoritmo de coloración por Búsqueda Tabú encontró el número óptimo de colores en 34 instancias de un total de 69.
2. La razón de aproximación máxima $\frac{\hat{f}}{\chi(G)}$ es de 3.2.
3. El promedio de la razón de aproximación es 1.284478.

Nombre de la instancia	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$	**Nombre de la instancia**	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$
1-FullIns_3	4	?	11.66		le450_15b	20	15	444.195	1.333
1-FullIns_4	5	?	46.474		le450_15c	29	15	795.811	1.933
1-FullIns_5	6	?	183.074		le450_15d	29	15	790.479	1.933
1-Insertions_4	5	4	24.413	1.25	le450_25a	26	25	444.169	1.04
1-Insertions_5	6	?	91.348		le450_25b	26	25	440.712	1.04
1-Insertions_6	7	?	400.249		le450_25c	33	25	822.75	1.32
2-FullIns_3	5	?	19.984		le450_25d	35	25	820.837	1.4
2-FullIns_4	7	?	106.847		le450_5a	12	5	337.617	2.4
2-FullIns_5	7	?	772.331		le450_5b	12	5	339.905	2.4
2-Insertions_3	4	4	11.718	1.0	le450_5c	16	5	508.016	3.2
2-Insertions_4	5	4	53.465	1.25	le450_5d	14	5	509.086	2.8

Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$	Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$
2-Insertions_5	6	?	295.451		miles1000	44	42	148.815	1.048
3-FullIns_3	6	?	30.256		miles1500	73	73	222.344	1.0
3-FullIns_4	8	?	227.305		miles250	9	8	44.776	1.125
3-FullIns_5	8	?	1983.039		miles500	20	20	73.343	1.0
3-Insertions_3	4	4	17.51	1.0	miles750	33	31	108.122	1.065
3-Insertions_4	5	?	103.088		mug100_1	4	4	31.164	1.0
3-Insertions_5	7	?	740.412		mug100_25	4	4	30.84	1.0
4-FullIns_3	7	?	45.472		mug88_1	4	4	27.255	1.0
4-FullIns_4	8	?	438.215		mug88_25	4	4	27.166	1.0
4-FullIns_5	9	?	4556.417		mulsol.i.1	49	49	190.424	1.0
4-Insertions_3	4	3	26.054	1.333	mulsol.i.2	31	31	187.233	1.0
4-Insertions_4	5	?	191.445		mulsol.i.3	31	31	189.723	1.0
5-FullIns_3	8	?	64.982		mulsol.i.4	31	31	189.639	1.0
5-FullIns_4	9	?	745.017		mulsol.i.5	31	31	190.247	1.0
DSJC1000.1	30	?	2347.488		myciel3	4	4	4.457	1.0
DSJC1000.5	125	?	11040.007		myciel4	5	5	8.933	1.0
DSJC1000.9	311	?	19830.036		myciel5	6	6	21.224	1.0
DSJC125.1	7	?	58.972		myciel6	7	7	51.566	1.0
DSJC125.5	24	?	183.781		myciel7	8	8	133.866	1.0
DSJC125.9	52	?	305.499		qg.order100	104	100	41664.691	1.04
DSJC250.1	12	?	172.045		qg.order30	33	30	1358.577	1.1
DSJC250.5	42	?	659.462		qg.order40	43	40	3192.756	1.075
DSJC250.9	93	?	1054.887		qg.order60	64	60	10995.666	1.067
DSJC500.1	19	?	622.35		queen10_10	15	?	76.859	
DSJC500.5	70	?	2746.301		queen11_11	16	11	100.846	1.455
DSJC500.9	167	?	4674.998		queen12_12	17	?	128.34	
DSJR500.1	14	?	257.399		queen13_13	17	13	160.68	1.308
DSJR500.1c	96	?	5104.116		queen14_14	20	?	197.398	
DSJR500.5	148	?	2633.488		queen15_15	22	?	238.399	

Nombre de la instancia	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$	**Nombre de la instancia**	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$
abb313GPIA	14	?	2965.857		queen16_16	22	?	287.757	
anna	11	11	48.551	1.0	queen5_5	5	5	12.71	1.0
ash331GPIA	6	?	319.831		queen6_6	9	7	20.026	1.286
ash608GPIA	6	?	616.107		queen7_7	10	7	29.399	1.429
ash958GPIA	6	?	1009.358		queen8_12	14	12	73.041	1.167
david	11	11	34.692	1.0	queen8_8	12	9	41.346	1.333
flat1000_50_0	123	50	11129.372	2.46	queen9_9	14	10	56.855	1.4
flat1000_60_0	123	60	11192.591	2.05	school1	40	?	846.061	
flat1000_76_0	124	76	11085.139	1.632	school1_nsh	36	?	654.448	
flat300_20_0	45	20	1005.406	2.25	wap01a	55	?	5963.675	
flat300_26_0	45	26	915.81	1.731	wap02a	52	?	5994.684	
flat300_28_0	46	28	893.376	1.643	wap03a	60	?	14558.956	
fpsol2.i.1	65	65	563.535	1.0	wap04a	58	?	15433.044	
fpsol2.i.2	30	30	439.416	1.0	wap05a	53	?	2295.844	
fpsol2.i.3	30	30	431.344	1.0	wap06a	52	?	2373.197	
games120	9	9	51.61	1.0	wap07a	56	?	5273.266	
homer	13	13	196.817	1.0	wap08a	56	?	5261.96	
huck	11	11	28.465	1.0	will199GPIA	11	?	468.575	
inithx.i.1	54	54	941.245	1.0	zeroin.i.1	49	49	199.313	1.0
inithx.i.2	31	31	692.045	1.0	zeroin.i.2	30	30	176.98	1.0
inithx.i.3	31	31	685.17	1.0	zeroin.i.3	30	30	177.604	1.0

Tabla F.1: Resultados de Coloración por Búsqueda Tabú

G. Resultados de coloración por Algoritmo Genético

$\chi(G)$ representa el número cromático, es decir el mínimo (óptimo) numero de colores necesarios para colorear apropiadamente el grafo.

$\hat{f}$ representa el número cromático aproximado generado por el algoritmo.

$\frac{\hat{f}}{\chi(G)}$ representa la razón de aproximación.

Notas:

1. El algoritmo genético de coloración encontró el número óptimo de colores en 19 instancias de un total de 69.
2. La razón de aproximación máxima $\frac{\hat{f}}{\chi(G)}$ es de 3.6.
3. El promedio de la razón de aproximación es 1.431623.

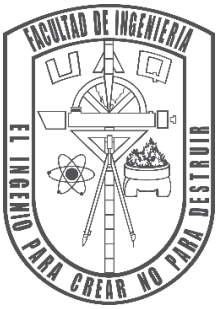
Nombre de la instancia	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$	**Nombre de la instancia**	$\hat{f}$	$\chi(G)$	**Tiempo (seg)**	$\frac{\hat{f}}{\chi(G)}$
1-FullIns_3	4	?	75.003		le450_15b	23	15	2000.69	1.533
1-FullIns_4	12	?	195.249		le450_15c	32	15	3640.041	2.133
1-FullIns_5	15	?	841.999		le450_15d	32	15	3667.508	2.133
1-Insertions_4	6	4	106.469	1.5	le450_25a	28	25	2002.793	1.12
1-Insertions_5	7	?	410.311		le450_25b	29	25	2027.348	1.16
1-Insertions_6	8	?	1825.175		le450_25c	36	25	3781.81	1.44
2-FullIns_3	7	?	214.632		le450_25d	35	25	3807.506	1.4
2-FullIns_4	15	?	481.317		le450_5a	13	5	1526.394	2.6
2-FullIns_5	19	?	3290.07		le450_5b	14	5	1546.11	2.8

Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$	Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$
2-Insertions_3	4	4	58.638	1.0	le450_5c	18	5	2324.405	3.6
2-Insertions_4	6	4	233.856	1.5	le450_5d	18	5	2323.31	3.6
2-Insertions_5	7	?	1324.71		miles1000	44	42	676.534	1.048
3-FullIns_3	13	?	136.838		miles1500	76	73	1023.991	1.041
3-FullIns_4	18	?	1032.678		miles250	10	8	188.994	1.25
3-FullIns_5	23	?	9137.908		miles500	23	20	327.753	1.15
3-Insertions_3	5	4	73.52	1.25	miles750	35	31	495.462	1.129
3-Insertions_4	6	?	439.869		mug100_1	5	4	126.855	1.25
3-Insertions_5	7	?	3374.922		mug100_25	5	4	125.332	1.25
4-FullIns_3	15	?	201.401		mug88_1	5	4	110.58	1.25
4-FullIns_4	21	?	1968.67		mug88_25	5	4	110.713	1.25
4-FullIns_5	27	?	20691.124		mulsol.i.1	49	49	871.378	1.0
4-Insertions_3	5	3	101.503	1.667	mulsol.i.2	31	31	860.523	1.0
4-Insertions_4	6	?	779.218		mulsol.i.3	31	31	863.279	1.0
5-FullIns_3	17	?	281.337		mulsol.i.4	31	31	867.114	1.0
5-FullIns_4	24	?	3376.094		mulsol.i.5	31	31	874.704	1.0
DSJC1000.1	32	?	11327.141		myciel3	4	4	17.047	1.0
DSJC1000.5	128	?	51304.364		myciel4	5	5	39.476	1.0
DSJC1000.9	322	?	90897.99		myciel5	7	6	89.231	1.167
DSJC125.1	8	?	245.427		myciel6	8	7	226.31	1.143
DSJC125.5	26	?	792.224		myciel7	9	8	600.001	1.125
DSJC125.9	57	?	1322.347		qg.order100	147	100	176460.797	1.47
DSJC250.1	13	?	795.581		qg.order30	45	30	6213.531	1.5
DSJC250.5	44	?	2981.662		qg.order40	52	40	14713.913	1.3
DSJC250.9	100	?	5087.06		qg.order60	90	60	50172.59	1.5
DSJC500.1	20	?	2902.98		queen10_10	19	?	350.382	
DSJC500.5	74	?	12380.172		queen11_11	19	11	468.373	1.727
DSJC500.9	173	?	21638.044		queen12_12	21	?	584.228	
DSJR500.1	15	?	1178.327		queen13_13	23	13	746.565	1.769

Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$	Nombre de la instancia	$\hat{f}$	$\chi(G)$	Tiempo (seg)	$\frac{\hat{f}}{\chi(G)}$
DSJR500.1c	110	?	23362.38		queen14_14	23	?	910.971	
DSJR500.5	145	?	11852.914		queen15_15	25	?	1132.038	
abb313GPIA	15	?	12750.27		queen16_16	26	?	1333.41	
anna	12	11	213.154	1.091	queen5_5	5	5	164.34	1.0
ash331GPIA	11	?	1468.007		queen6_6	10	7	87.514	1.429
ash608GPIA	9	?	2780.82		queen7_7	13	7	132.172	1.857
ash958GPIA	11	?	4466.855		queen8_12	16	12	335.036	1.333
david	13	11	153.289	1.182	queen8_8	14	9	189.188	1.556
flat1000_50_0	126	50	50288.52	2.52	queen9_9	15	10	258.187	1.5
flat1000_60_0	127	60	50563.83	2.117	school1	44	?	4017.109	
flat1000_76_0	126	76	51491.694	1.658	school1_nsh	41	?	3035.814	
flat300_20_0	47	20	4104.894	2.35	wap01a	62	?	26581.04	
flat300_26_0	48	26	4160.531	1.846	wap02a	61	?	27337.024	
flat300_28_0	47	28	4170.498	1.679	wap03a	73	?	61369.098	
fpsol2.i.1	65	65	2611.324	1.0	wap04a	69	?	59549.95	
fpsol2.i.2	30	30	1990.193	1.0	wap05a	64	?	9931.503	
fpsol2.i.3	30	30	1973.226	1.0	wap06a	62	?	10088.149	
games120	10	9	225.089	1.111	wap07a	73	?	22456.502	
homer	16	13	823.992	1.231	wap08a	73	?	20169.198	
huck	11	11	123.572	1.0	will199GPIA	13	?	2071.382	
inithx.i.1	54	54	4295.946	1.0	zeroin.i.1	49	49	902.598	1.0
inithx.i.2	31	31	3172.178	1.0	zeroin.i.2	30	30	809.545	1.0
inithx.i.3	31	31	3148.584	1.0	zeroin.i.3	30	30	791.859	1.0

Tabla G.1: Resultados de Coloración por Algoritmo Genético

H. Manual de Usuario

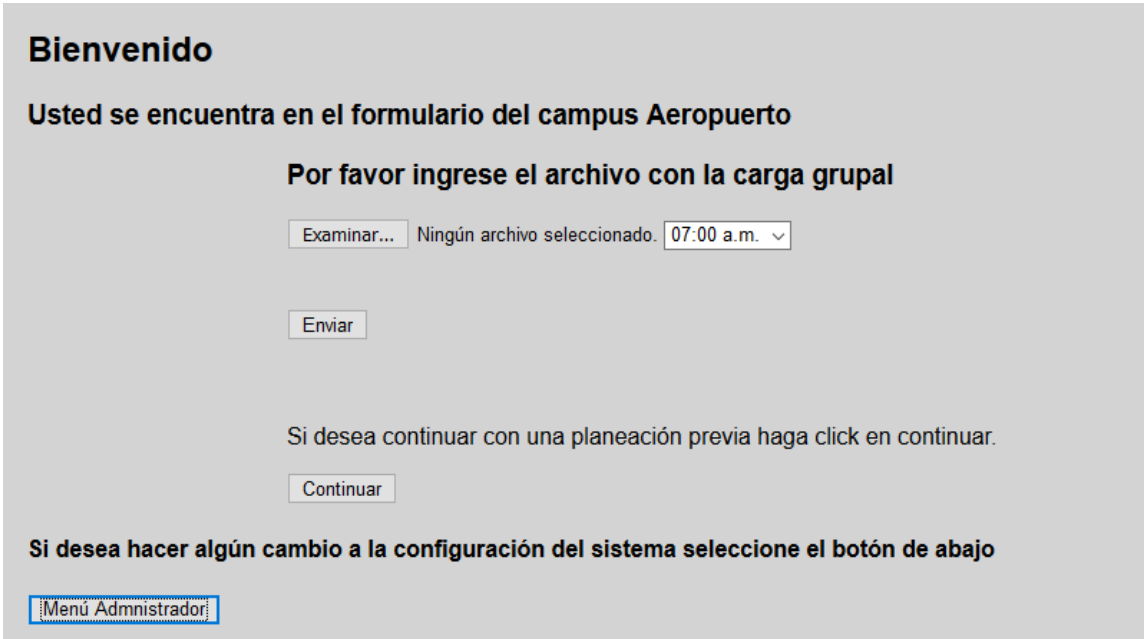


Manual de Usuario del Sistema de Asignación de Horarios

El sistema de asignación de horarios es una herramienta web que permite la generación de horarios para la facultad de ingeniería.

Para hacer uso de este sistema el primer paso es escoger en la pantalla inicial el campus del cual se requiera hacer el horario (campus Aeropuerto o campus Centro Universitario):

Después de escoger un campus el sistema rediccionara al usuario a un formulario



En este formulario se deberá cargar un archivo con la información de la carga horaria de las asignaturas que se enseñan por maestro.

Dicho archivo deberá ser una hoja de Excel, esta hoja deberá de contener 2 columnas con la siguiente información:

- En la primera columna se tiene la clave del maestro.
- En la misma fila en la siguiente columna se tiene la clave de la materia.
- En la misma fila en la siguiente columna se tiene el nombre del programa que solicita la materia
- Si el maestro enseña la misma materia a más de un grupo la misma fila deberá de estar repetida el mismo número de veces que esa materia se enseñe.
- En la misma fila en la siguiente columna se indicará el número de estudiantes que tomarán la materia

Por ejemplo:

9161	228	Biomédica	50
9161	228	Biomédica	60
9161	1278	Biomédica	60
9161	1278	Biomédica	50
9161	1327	Biomédica	30

Consideraciones:

- En este caso el maestro con la clave 9161 enseña las materias 228, 1278 y 1327, sin embargo, enseña las materias 228 y 1278 a más de un grupo por lo que aparecen más de una vez.
- En este caso todas las materias son requeridas por la carrera de Ingeniería Biomédica.
- La otra entrada en el formulario es un número que refiere al número de salones con los que se puede contar para las materias.

También se deberá de poner el número de salones con los que se cuentan para la asignación horarios.

El sistema viene precargado con la información de que materias son requeridas por cada semestre para cada carrera por lo que esta es la única información que el usuario deberá de ingresar.

Esta información puede ser editada en el menú de administrador.

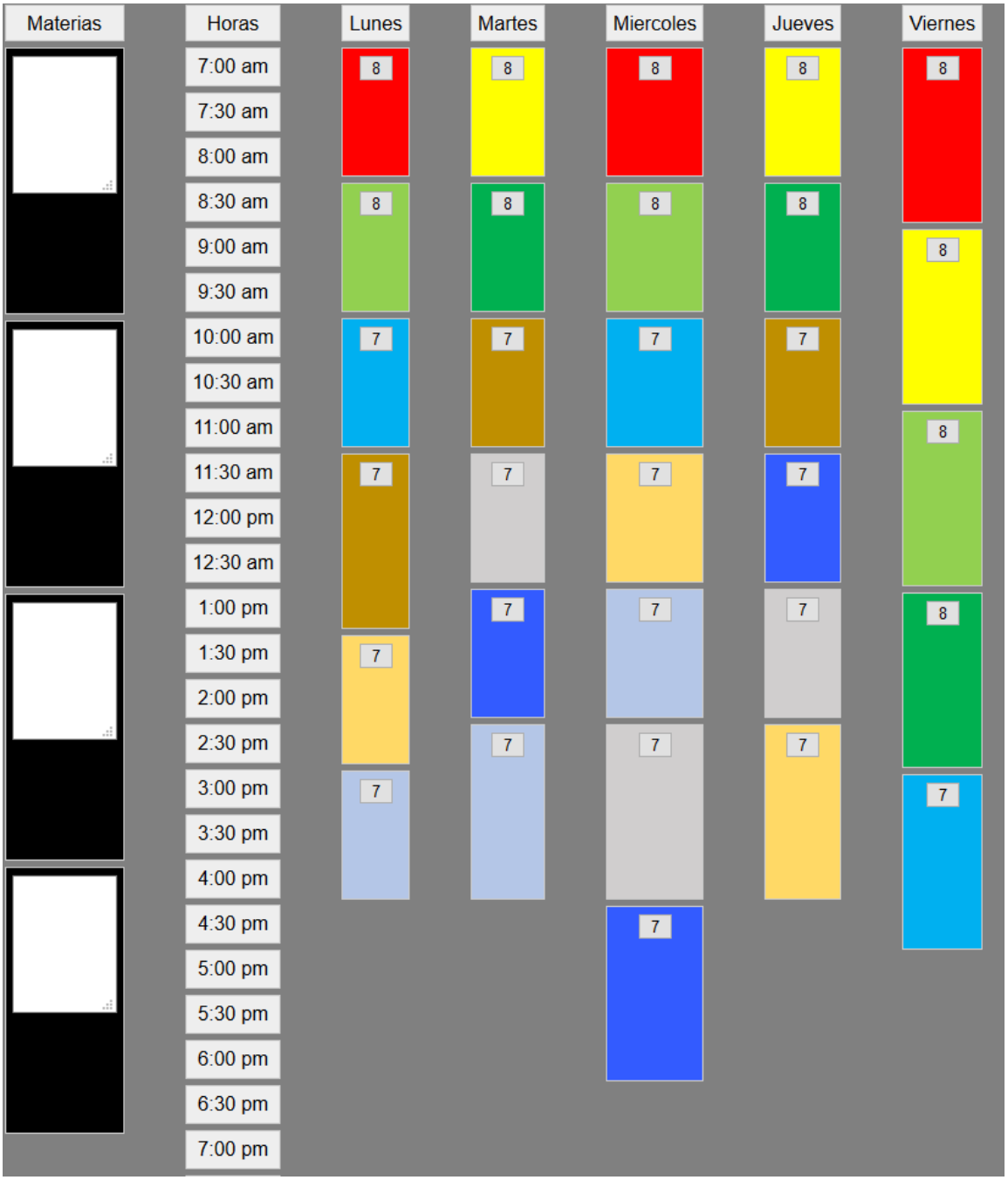
Menú de administrador para Campus Aeropuerto

El formulario deberá de ser enviando dando click en el botón de Enviar.

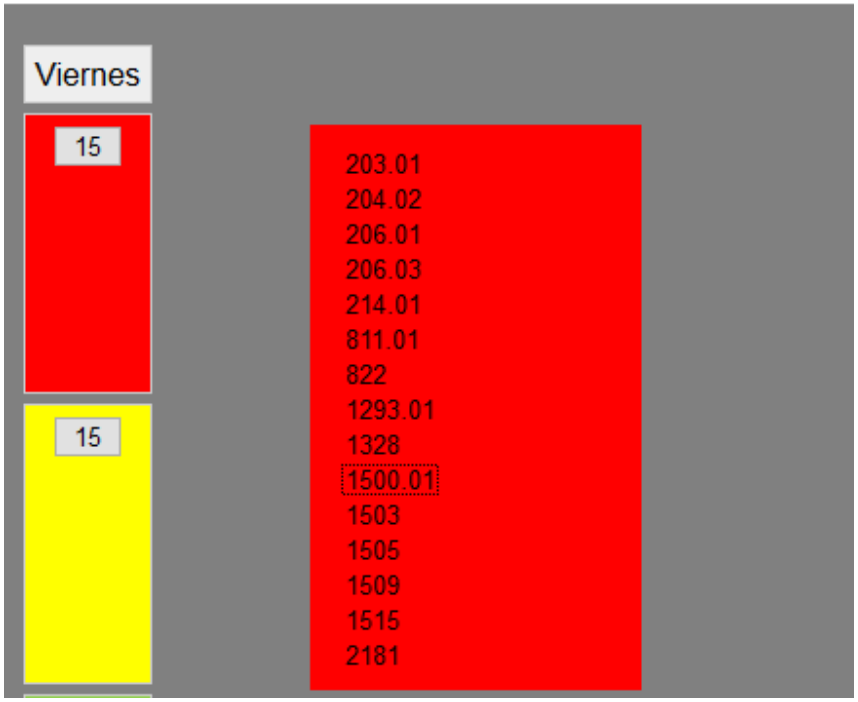
Ya enviado el formulario con el archivo el sistema calculara cuantos espacios temporales se necesitarán para poder ofrecer las materias sin ningún tipo de cruzamiento.

Por cruzamiento se refiere a que un alumno regular necesite llevar más de una materia que se ofrezca en el mismo espacio temporal.

Después de que el sistema termine de calcular el cómo ofrecer la carga horaria el usuario será redirigido a una página donde podrá acomodar el horario a su conveniencia.

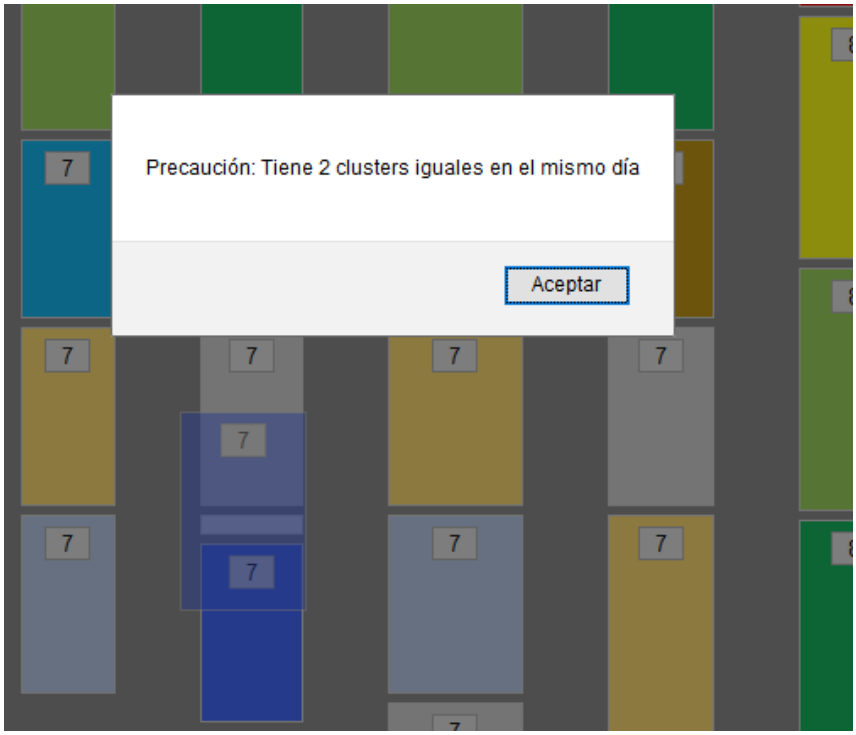


Este formulario indica los clústeres de materias que se necesitan separados por colores, cada bloque del mismo color indica que las mismas materias se enseña en él, al hacerle click en los bloques se despliega que materias se enseña en él. Los clústeres color negro en la parte izquierda están fuera del horario puesto que están diseñados para representar idiomas. Las materias de idiomas cambian cada semestre y estas pueden ser escritas en el espacio en blanco.



También es posible el mover el clúster a otra ubicación haciéndole click y arrastrándolo a la nueva posición.

Sin embargo, si se trata de poner 2 clústeres del mismo color en el mismo día el sistema soltará una alarma.



Esta asignación puede ser guardada para continuar trabajando en ella.

Finalmente, al hacer click en generar reporte se lleva al usuario a una página que contiene la tabla de asignación de horario, maestro y salón.

Lunes 7:00-8:30							
Clave	Curso	Profesor	Clave Profesor	Nomenclatura	Programa Solicitante	Salón	Edificio
214	PROBABILIDAD Y ESTADISTICA	Instructor 1	Clave 1	B1,N1	Biomedica	A12	A
822	FISICA	Instructor 2	Clave 2	B2	Biomedica	A1	A
844	SUSTENTABILIDAD	Instructor 3	Clave 3	B5	Biomedica	B12	B
894	BIOMECANICA	Instructor 4	Clave 4	B7	Biomedica	C1	C
898	PROCESAMIENTO DE BIOSENALES (PBS)	Instructor 5	Clave 5	B6	Biomedica	C3	C
483	OPTATIVA V	Instructor 6	Clave 6	B8	Biomedica	C1	C
484	FISIOLOGIA I	Instructor 7	Clave 7	B3	Biomedica	C2	C
486	BIOMAGNETISMO	Instructor 8	Clave 8	B4	Biomedica	E3	E

En este reporte se indican:

- La clave de la materia
- El nombre de la materia
- El profesor que la imparte
- La clave del profesor que la imparte
- Una clave de nomenclatura que indica la carrera y el semestre al que este pertenece. Por ejemplo, B1, N1 quiere decir que esta materia es impartida en el primer semestre de biomédica y nanotecnología. Si en la nomenclatura solo se encuentra un número este indica que pertenece a todas las carreras en el semestre indicado por el número.
- El salón donde se imparte
- El edificio donde se encuentra el salón

Este reporte puede ser descargado como un archivo Excel.

I. Artículo 11vo Coloquio de Posgrado

“Coloreo de grafos: Un caso de estudio para la planeación de horarios escolares.”

“Graph coloring: a case study for planning academic schedules”

Resumen

En este artículo se modela el problema de planeación de horarios escolares mediante el problema de coloreo de grafos. Un grafo apropiadamente coloreado consiste en una asignación de color a cada vértice del grafo tal que ningún par de vértices tienen el mismo color si existe una arista que los una. La modelación consiste en representar cada materia mediante un vértice, y si dos materias no se deben ofrecer en el mismo horario, porque el plan de estudios lo exige y previene que ambas materias se deben cursar simultáneamente, o porque existe al menos un estudiante que pueda llevar ambas, entonces se construye una arista uniendo tales dos vértices. Particularmente se modela la operación de los programas académicos de ingeniería en: Nanotecnología, Biomédica, y Física, que ofrece la Facultad de Ingeniería de la Universidad Autónoma de Querétaro en Campus Aeropuerto. Para la solución del problema de coloreo de grafos se utilizan dos algoritmos voraces para determinar el mínimo número de colores (número cromático) para colorear apropiadamente el grafo. El primero consiste en colorear los vértices de acuerdo al orden en que se encuentran los vértices etiquetados, hasta que no sea posible colorear un vértice más con el mismo color; el segundo, lleva a cabo el proceso de coloración conforme al orden dictado por el grado de los vértices. El número de colores necesarios corresponde al número de ranuras de tiempo requeridas para ofertar las materias que un alumno regular pudiera tomar, donde cada subconjunto de vértices del mismo color representa las materias que pueden ofrecerse al mismo tiempo sin riesgos de conflicto de horario. El problema de coloreo de grafos es NP-completo, por lo que no existe un algoritmo eficiente que produzca soluciones óptimas. Sin embargo, los algoritmos descritos en este articulo producen resultados aproximados con soluciones cercanas al óptimo.

Palabras Clave: *Optimización, Teoría de grafos, Número Cromático, Planeación*

Abstract

In this article the problem of academic schedule planning is modeled by the graph coloring problem. A properly colored graph consists of an assignment of a color to each vertex of the graph such that there is no pair of vertices of the graph that have the same color if there is an edge that joins them. The modeling consists of representing each course as a vertex, and if two courses must be taken at the same time, or there is a student who must take both, then an edge is created joining such two vertices. Particularly, the operation of the engineering academic programs in Nanotechnology, Biomedical, and Physics, offered by the School of Engineering of the Autonomous University of Queretaro in Airport Campus, is modeled. To solve the graph coloring problem, two greedy algorithms are used to determine the minimum number of colors (chromatic number) that are needed to properly color the graph. The first one consists of coloring the vertices in the order given by how the vertices are labeled until it is not possible to color with that color anymore; and the second one makes the process according to the order given by the degree of the vertices. The number of colors that are needed corresponds to the number of time slots required to offer the courses that a regular student could take, where each subset of vertices of the same color represents the courses that can be offered at the same time without any risk of conflict in the academic schedules. The graph coloring problem is NP-complete and there does not exist an efficient algorithm that produces optimal solutions. However, the algorithms described in this article produce approximated results which are near optimal.

Keywords: *Optimization, Graph Theory, Chromatic Number, Scheduling*

1. Introducción

En teoría de grafos, la coloración de grafos es un caso de etiquetado en el que se le asigna una color a cada nodo de manera que ningún vértice tenga el mismo color que un vértice adyacente [2]. Este problema fue desarrollado a partir del problema de los 4 colores [3], y fue planteado por primera vez en 1852 el matemático

Francis Guthrie cuando trataba de colorear el mapa de Inglaterra. El problema plantea la noción de que cualquier mapa puede tener todos los territorios coloreados con solo 4 colores de manera que cada territorio tenga un color diferente a sus territorios adyacentes, y fue demostrado en 1912 por George David Birkhoff [4] cuando desarrollo el polinomio cromático. A pesar de que el problema está resuelto para mapas, esto solo es debido a que no es posible plantear más de 4 territorios compartiendo fronteras entre sí. Sin embargo, en teoría de grafos el problema se vuelve más interesante, ya que se pueden requerir más de 4 colores. Considere el grafo que tiene nodos con aristas apuntando a todos los otros nodos, al cual se le conoce como un grafo completo; este tipo de grafos no tiene un límite en cuanto a tamaño por lo que es posible necesitar más de 4 colores para colorear los vértices.

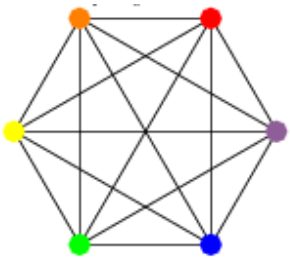


Figura 1. Grafo completo de tamaño 6

En la figura 1 se muestra como ejemplo un grafo completo de tamaño 6 el cual necesita 6 colores para poder ser coloreado. El problema de coloreo de vértices fue clasificado como un problema NP-completo, planteando un gran desafío computacional, ya que la solución óptima solo es encontrada utilizando grandes cantidades de tiempo de cómputo. En el caso particular del coloreo de grafos, la solución se encuentra generando el polinomio cromático. Dada la naturaleza de los problemas NP-completos rara vez se trata de encontrar la solución óptima, por lo que se trata de encontrar una solución viable utilizando algoritmos de aproximación. Estos algoritmos tratan de dar resultados cercanos a la solución óptima de tal manera que la solución sea viable; por ejemplo, supongamos que un grafo tiene un número cromático óptimo de 9 ($n_k = 9$), lo cual quiere decir que se necesitan como mínimo 9 colores diferentes para colorearlo. Un algoritmo de aproximación podría dar de resultado 10 ($n_k = 10$), el cual es un resultado aceptable ya que se puede producir con tiempos de cómputo muy razonables.

2. Modelo de planeación de horarios basado en coloración de grafos.

El proponer la coloración de grafos para el desarrollo de horarios fue una idea propuesta por primera vez en 1979 [5]. Para poder aplicar el coloreo de vértices a un sistema de planeación es necesario definir los eventos como nodos. Si un nodo tiene una arista que lo conecta con otro nodo esto quiere decir que ambos nodos son eventos que deben ocurrir al mismo tiempo. De esta manera se puede planear la carga de materias que tiene la facultad de ingeniería campus aeropuerto, que se compone de las carreras de Ingeniería Biomédica, Ingeniería Física e Ingeniería en Nanotecnología. Para poder modelar la carga horario como un grafo fue necesario obtener la información de las materias que se ofrecen, particularmente la clave de la materia así como el semestre al que pertenecen. En la figura 2 se presentan las materias del 1er semestre de la carrera de Ingeniería Biomédica, modelado mediante un grafo y conforme al problema establecido de coloreo de grafos..

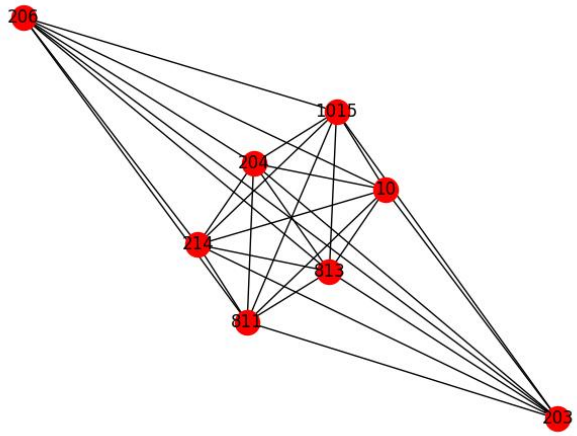


Figura 2. Modelado del primer semestre de Ingeniería Biomédica

Se puede apreciar como todas las 8 materias están conectadas entre sí indicando la necesidad de que todas se ofrezcan en un horario diferente. Si aplicamos ese proceso a cada uno de los semestres de todas las carreras se pude construir el grafo general de todo el campus aeropuerto. En la figura 3 se muestra la modelación del primer semestre de Ingeniería Física. Este caso a diferencia del primer semestre de Ingeniería Biomédica cuenta con 9 materias.

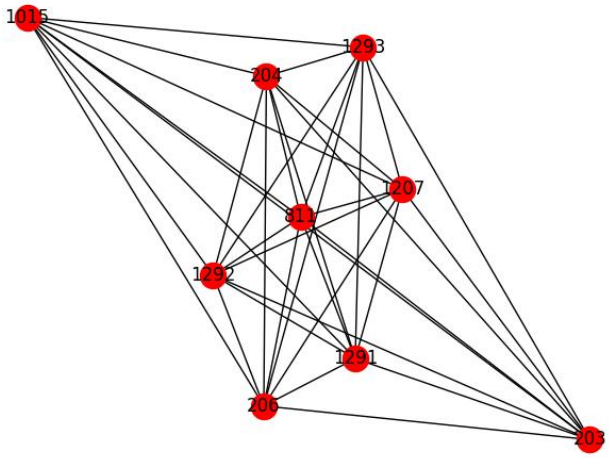


Figura 3. Modelado del primer semestre de Ingeniería Física

Sin embargo, debido a que diferentes carreras ofrecen las mismas materias, el grafo puede ser complementado. Tomando como ejemplo las figuras 2 y 3 se puede ver que el primer semestre de Ingeniería Física y el primer semestre de Ingeniería Biomédica comparten las materias con clave 203, 1015, 206, 204 y 811. Dado que se trata de materias comunes no hay problema de que ambas carreras la tomen al mismo tiempo. En la figura 4 se muestra el grafo resultante al combinar ambos grafos de las figuras 2 y 3.

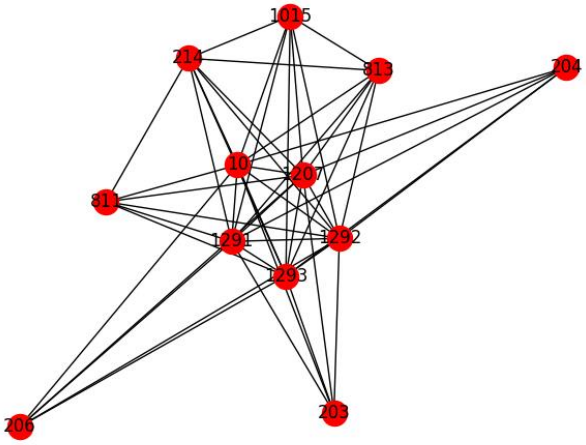


Figura 4. Modelado del primer semestre de Ingeniería Física junto con el primer semestre de Ingeniería Biomédica

El proceso de modelación se hizo para todos los semestres de las 3 carreras. Sin embargo, antes de aplicarse los algoritmos se aplicó un filtro para reducir el número de materias, cancelando aquellas como las que se indican como Servicio Social y Prácticas Profesionales ya que no son planeadas para ser tomadas en un aula y a pesar de tener un semestre asignado. La misma situación se da con las materias de Deportes y Cultura Deportiva, ya que aunque se dan en la escuela no requieren aula. Finalmente no se consideraron las materias de Inglés y de Segundo Idioma puesto que el horario y aula para estas es determinado en un espacio específico por el personal administrativo de la facultad.

Una vez haciendo efectivos esos filtros se pudo modelar un grafo que describe las materias que son ofrecidas para todas las carreras. La figura 5 muestra el grafo que representa todas las materias de los tres programas académicos.

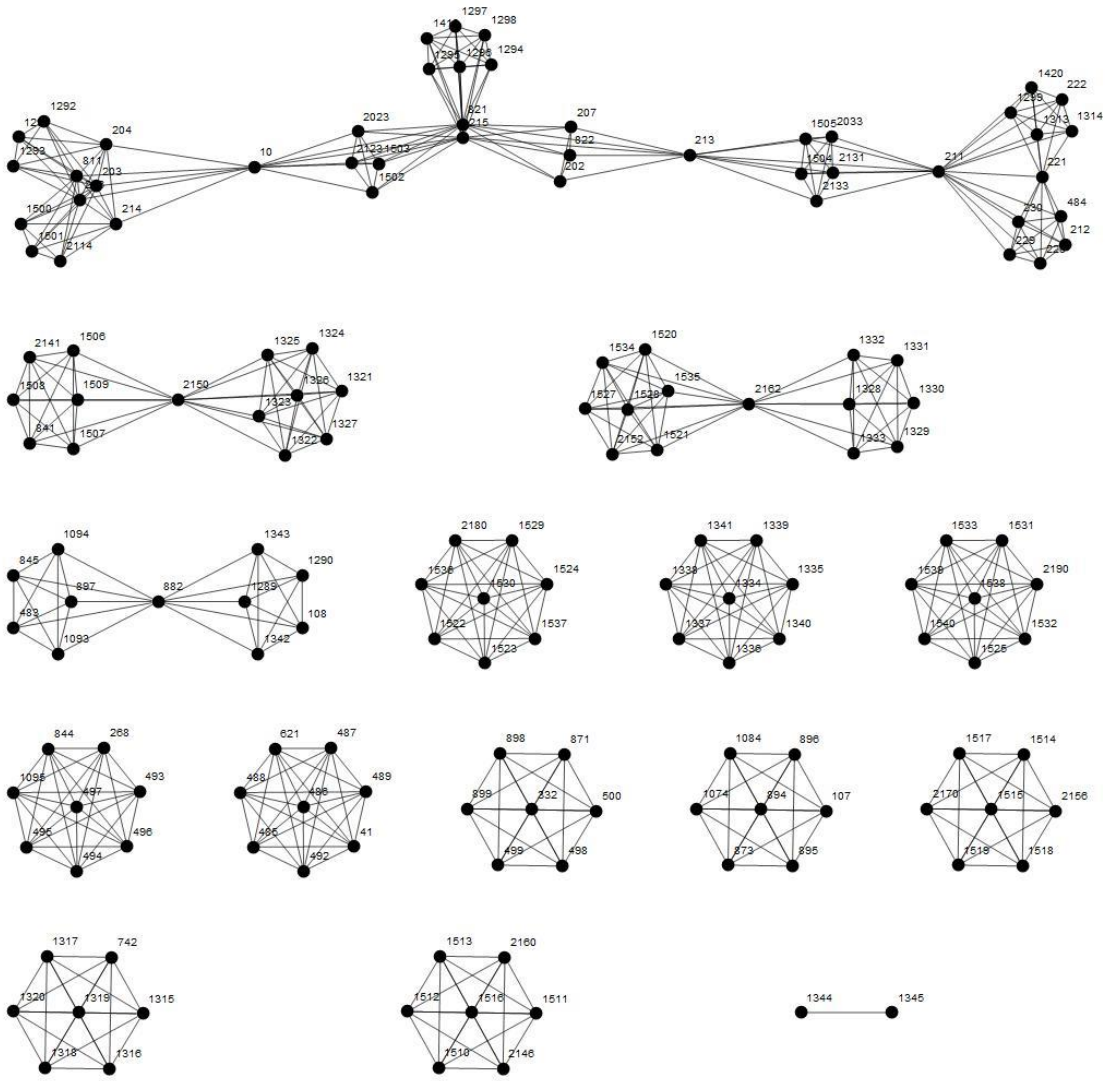


Figura 5. Modelado de todas las materias del campus aeropuerto

Ya que se tuvo modelado el grafo de campus aeropuerto se procedió a aplicarle los algoritmos de coloración. Los algoritmos utilizados en este proyecto son derivaciones del algoritmo conocido como coloreo por ordenamiento el cual es un algoritmo tipo voraz. El algoritmo es como sigue:

Paso 1: Asignar el primer color(c_1) al primer nodo(v_1)

Paso 2: Al vértice $2(v_2)$ le es asignado el c_1 si no es adyacente al $v_1$de otra forma se le es asignado el nuevo color (c_2)

Paso 3, 4, ..., n: Al vértice v_i le es asignado el primer color posible en la lista de prioridad de colores (es decir el primer color que no ha sido asignado a uno de los vecinos ya coloreados de v_i

Tomemos por ejemplo el grafo en la figura 6. Si a este grafo le aplicamos este algoritmo encontramos que su solución óptima corresponde al número cromático de 2 ($n_k = 2$), como se demuestra en la misma figura 6.

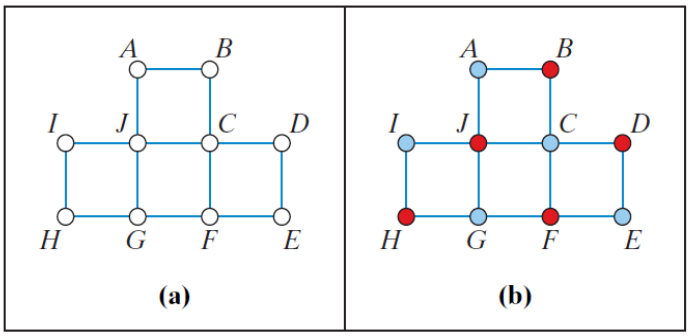


Figura 6. Ejemplo de el algoritmo de coloreo por ordenamiento

El problema de este algoritmo es que el número cromático obtenido varia drásticamente dependiendo del orden que se les dé a los nodos del grafo. Como se puede apreciar en la figura 7, el número cromático se duplica si se altera el orden de los nodos.

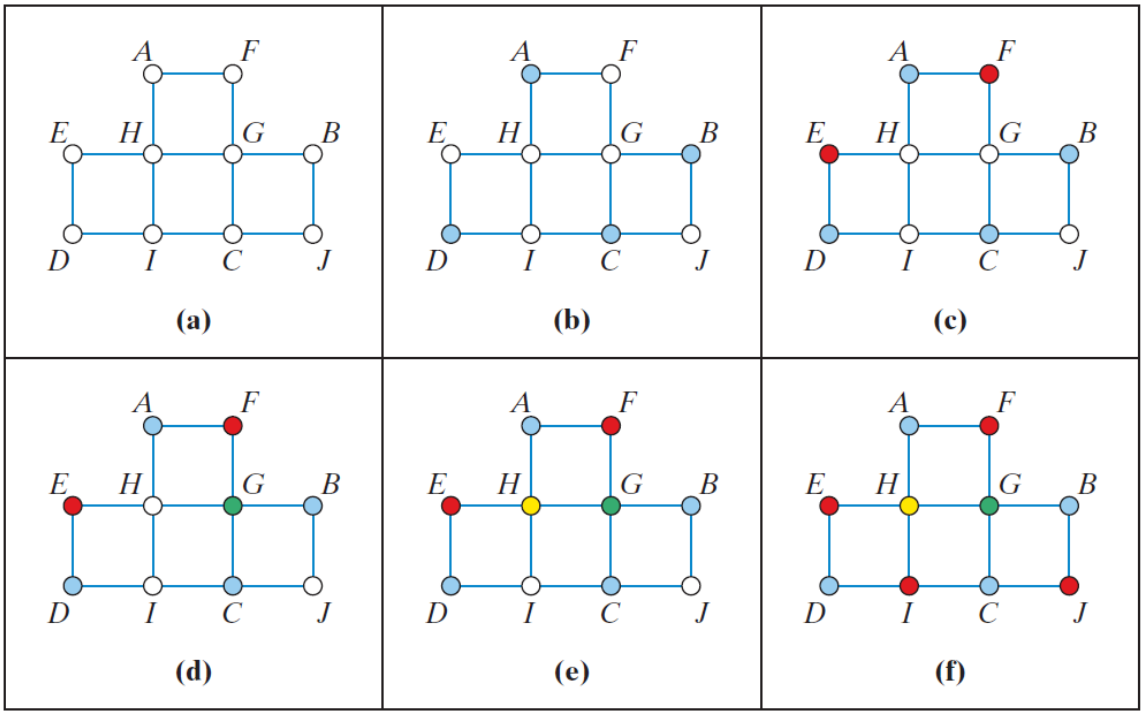


Figura 7. Problema de el algoritmo ordenamiento

Debido a este problema se desarrollaron 2 algoritmos. El primero es llamado algoritmo por ordenamiento, el cual recorre los nodos siguiendo el orden numérico de las claves de las materias; y el segundo será llamado algoritmo por mayor grado, el cual le da un orden a los nodos de manera descendente, dándole el primer lugar al nodo que cuente con mayor número de vecinos. Es decir, el que tenga más aristas conectándolo con otros nodos, y se sigue el criterio para asignarles una posición en el orden a los demás nodos.

Finalmente, considerando como límite la cantidad máxima de salones disponibles (como se describe en la sección de resultados), se obtuvo un color que tiene asignados 24 materias. Pero el campus aeropuerto solo tiene 19 salones dispobibles, por lo que el algoritmo fue modificado para asignar otro color al nodo en caso de que ya tuviera 19 materias asignadas. También se tomó en cuenta el dejar los horarios para inglés.

4. Resultados y discusión

Después de haber modelado las materias ofrecidas como un grafo, se sometió el grafo a los algoritmos de coloreo con diferentes números de límite de salones.

Tabla 1. Resultados de los algoritmos

Número de Salones	**Número cromático obtenido con los diferentes algoritmos**		
	Ordenamiento	Mayor Grado	Saturación Máxima
19	13	11	13
20	11	11	13
21	11	11	13
22	11	11	13
23	11	11	13
24	11	11	13
25	11	11	13
Sin Limite	9	8	13

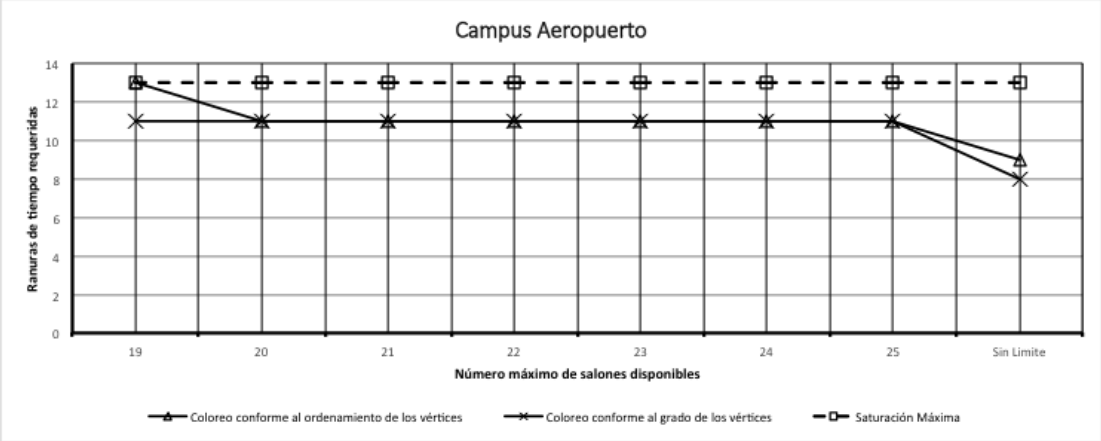


Figura 8. Cantidad de ranuras de tiempo requeridas conforme al número máximo de salones disponibles.

En la tabla 1 y en la figura 8 se ilustran los resultados de los algoritmos. Se puede notar como que no se necesitan más de 13 espacios de horarios diferentes para poder ofertar todas las materias. Debido al análisis hecho a las materias que se ofertan en el campus aeropuerto fue posible también generar un diagrama de Venn en el que se demuestran las materias en común que tienen las carreras. En la figura 9 se ilustra dicho diagrama junto con un análisis de inclusión-exclusión. El diagrama refleja la poca cantidad de materias que en común cuentan las diferentes carreras, así como el hecho de que Nanotecnología cuenta con más materias que las otras carreras.

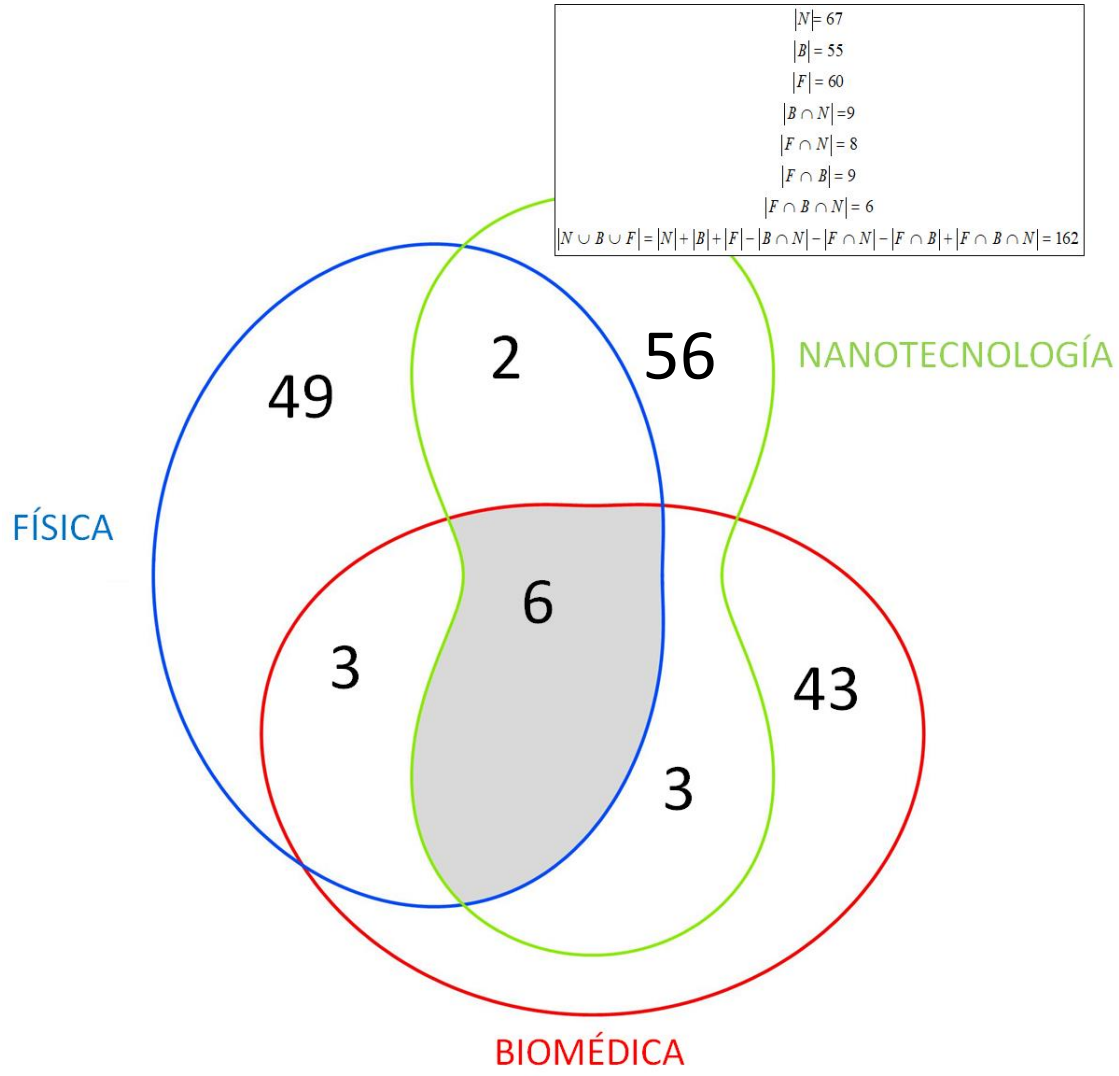


Figura 9. Análisis de inclusión-exclusión

Como resultado final se propone un horario que cumpla las condiciones prestablecidas. Primero se ilustra en las figuras 10 y 11 un horario diseñado para el caso que considera número ilimitado de salones. Este caso requiere sólo 8 ranuras de tiempo. En la figura 10 también se muestran las materias que constituyen las 8 ranuras de tiempo sin restricción en el número máximo de salones. A pesar de ello, se puede afirmar que se requieren como máximo 24 salones para poder ofertar todas las materias dentro de un horario de 7:00 am a 6:30 pm. Este es un resultado que no tiene en cuenta las limitaciones físicas del campus aeropuerto

En las figuras 12 y 13 se presentan los resultados considerando el límite físico de 19 salones.

Sistema de Horarios

Universidad Autónoma de Querétaro/Facultad de Ingeniería

Campus Aeropuerto

Politica de Simulación: Alumno Regular y # de Salones Ilimitado

Cluster	Cursos/Ranuras de Tiempo	Tamaño del Cluster
0	41, 211, 215, 332, 742, 882, 1074, 1095, 1334, 1344, 1526, 1536, 1538, 2146, 2150, 2156, 2162	17
1	108, 203, 221, 268, 621, 821, 871, 1084, 1093, 1315, 1321, 1329, 1335, 1345, 1527, 1537, 1539, 2131, 2141, 2160, 2170,	21
2	10, 107, 212, 213, 222, 488, 841, 844, 898, 1094, 1289, 1291, 1294, 1316, 1322, 1328, 1336, 1513, 1514, 1528, 1529, 1540, 2114	23
3	202, 206, 228, 492, 493, 845, 873, 899, 1290, 1295, 1299, 1317, 1323, 1331, 1337, 1506, 1515, 1516, 1535, 2123, 2133, 2180, 2190	23
4	207, 229, 489, 494, 498, 811, 894, 897, 1296, 1313, 1318, 1324, 1330, 1338, 1342, 1502, 1504, 1509, 1510, 1517, 1523, 1525, 2152	23
5	204, 230, 483, 485, 496, 499, 822, 895, 1297, 1314, 1319, 1325, 1332, 1339, 1343, 1500, 1503, 1505, 1507, 1511, 1518, 1520, 1524, 1531	24
6	214, 484, 486, 497, 500, 896, 1292, 1298, 1320, 1326, 1333, 1340, 1420, 1508, 1512, 1519, 1521, 1522, 1532, 2023, 2033	21
7	487, 495, 1293, 1327, 1341, 1419, 1501, 1530, 1533, 1534	10
	Total	162
	Promedio	20
	Desviacion estandar	5

Figura 10. Materias organizadas por color sin límite de salones

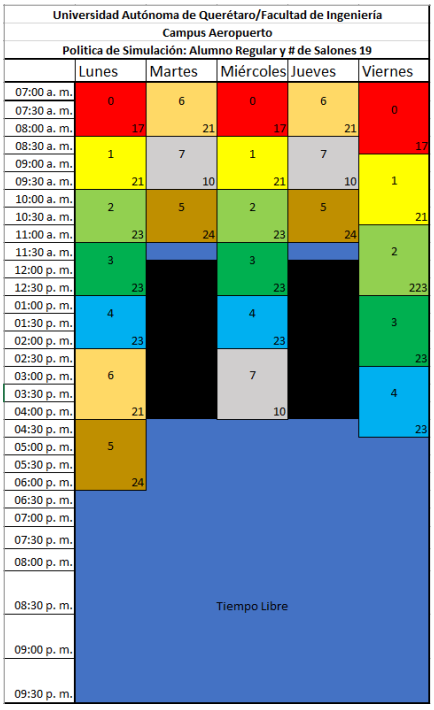


Figura 11. Horario propuesto sin límite de salones

Sistema de Horarios

Universidad Autónoma de Querétaro/Facultad de Ingeniería

Campus Aeropuerto

Politica de Simulación: Alumno Regular y # de Salones 19

Cluster	Cursos/Ranuras de Tiempo	Tamaño del Cluster
0	41, 211, 215, 332, 742, 882, 1074, 1095, 1334, 1344, 1526, 1536, 1538, 2146, 2150, 2156, 2162	17
1	108, 221, 268, 621, 821, 871, 1084, 1093, 1315, 1321, 1329, 1335, 1345, 1527, 1537, 1539, 2141, 2160, 2170	19
2	10, 107, 212, 213, 222, 488, 841, 844, 898, 1294, 1316, 1322, 1328, 1336, 1513, 1514, 1528, 1529, 1540	19
3	203, 228, 492, 493, 873, 899, 1295, 1299, 1317, 1323, 1331, 1337, 1515, 1516, 1535, 2123, 2131, 2180, 2190	19
4	206, 229, 489, 494, 894, 1296, 1313, 1318, 1324, 1330, 1338, 1502, 1506, 1510, 1517, 1523, 1525, 2133, 2152	19
5	230, 485, 496, 811, 895, 1297, 1314, 1319, 1325, 1332, 1339, 1503, 1504, 1509, 1511, 1518, 1520, 1524, 1531	19
6	204, 484, 486, 497, 896, 1298, 1320, 1326, 1333, 1340, 1420, 1505, 1507, 1512, 1519, 1521, 1522, 1532, 202	19
7	202, 214, 487, 495, 498, 1094, 1289, 1291, 1327, 1341, 1419, 1508, 1530, 1533, 1534, 2034	16
8	207, 499, 845, 1290, 1292, 2114	6
9	500, 822, 897, 1293, 1342, 1500	6
10	483, 1343, 1501	3
	Total	162
	Promedio	14
	Desviacion estandar	6

Figura 12. Materias organizadas por color con un límite de 19 salones

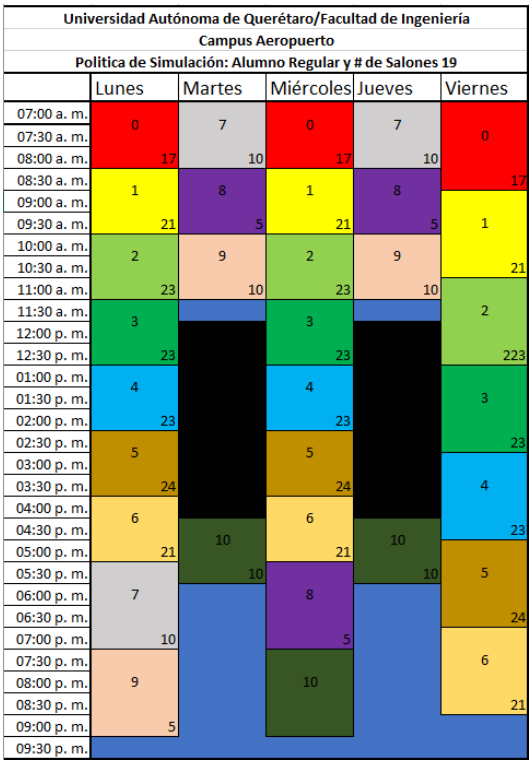


Figura 13. Horario propuesto con límite de 19 salones

5. Conclusiones

El sistema de horarios propuesto cuenta con la capacidad de ofrecer todas las materias de los tres programas de ingeniería ofrecidos en Campus Aeropuerto por la Facultad de Ingeniería de la Universidad Autónoma de Querétaro, con la garantía de que no exista conflictos de horarios, conforme al límite de salones establecido.

Por lo que se refiere al modelo básico que se ha presentado en este artículo, se puede concluir que los resultados representan la cota mínima de requerimientos de infraestructura, ya que se asume que el estudiante que cursa el programa académico es regular y solo se ofrece un grupo por materia.

El modelo puede incorporar requerimientos más finos, como por ejemplo el que un subconjunto de materias tienen la característica de que se deben ofrecer varios grupos de la misma materia. Esto sucede a menudo cuando los cursos corresponden a los primeros semestres de los programas académicos, donde se tienen la mayor población estudiantil. Asimismo, se puede modelar el requerimiento de que los grupos de una misma materia pueden estar o no en el misma ranura de tiempo. Esto constituye trabajos interesantes a futuro.

Referencias

[1] Ralph P. Grimaldi. (1999). Discrete and combinatorial mathematics. San Francisco: Pearson Addison Wesley

[2] Tom Leighton, 6.042J / 18.062J Mathematics for Computer Science, Fall 2010. (Massachusetts Institute of Technology: MIT OpenCouseWare), http://ocw.mit.edu (Accessed February 12, 2017). License: Creative Commons BY-NC-SA

[3] Peter Tannenbaum. (1992). Excursions in Modern Mathematics with Mini-Excursions. USA: Pearson.

[4]George D. Birkhoff. A determinant formula for the number of ways of coloring a map. The Annals of Mathematics, 14(1/4):42–46, 1912

[5]F. Thomson Leighton, A Graph Coloring Algorithm for Large Scheduling Problems., JOURNAL OF RESEARCH of the Notional Bureau of Standards Vol. 84, No.6, November- December 1979

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

FACULTAD DE INGENIERÍA

DIPFI POSGRADO INGENIERÍA

Se otorga el presente

RECONOCIMIENTO a:

HANSEL AMADEUS MONTÚFFAR OTERO

Por haber participado en el concurso de artículo científico con su trabajo:

"COLOREO DE GRAFOS: UN CASO DE ESTUDIO PARA LA PLANEACIÓN DE HORARIOS ESCOLARES".

11° COLOQUIO
Posgrado Facultad de Ingeniería

24 de Noviembre de 2017

Dr. Aurelio Domínguez González
DIRECTOR
FACULTAD DE INGENIERÍA

Dr. Manuel Toledano Ayala
JEFE DE LA DIVISIÓN DE INVESTIGACIÓN Y POSGRADO
FACULTAD DE INGENIERÍA

Santiago de Querétaro, Querétaro, 20 de noviembre de 2017

C. Hansel Amadeus Montúffar Otero

PRESENTE

Me permito informarle que el comité evaluador del XI Coloquio de Posgrado de la Facultad de Ingeniería ha decidido **aceptar** el artículo detallado a continuación **para presentación y publicación en las memorias del congreso:**

Título:
Coloreo de grafos: Un caso de estudio para la planeación de horarios escolares

Código de registro:
IAR_1

Autores:
Hansel Amadeus Montúffar Otero, Arturo González Gutiérrez, Carmen Sosa Garza, Guillermo Díaz Delgado, Fidel González Gutiérrez

Le reiteramos nuestro agradecimiento por su aporte al éxito de este XI Coloquio y esperamos seguir contando con su participación en eventos posteriores.

Atentamente

M.C. Stephanie Virginia Camacho Gutiérrez
Miembro del comité técnico del XI Coloquio

J. Artículo IEEE Latin American Transactions (por enviar)

Course scheduling based on Graph Coloring

H. A. Montúffar, A. González, IEEE Member, C. Sosa, F. González, IEEE Member, G. Díaz, IEEE Member, and S. Tovar, IEEE Senior Member

***Abstract*— In this paper the problem of academic schedule planning is modeled by the graph coloring problem. A properly colored graph consists of an assignment of a color to each vertex of the graph such that there is no pair of vertices of the graph that have the same color if there is an edge that joins them. The modeling consists of representing each course as a vertex, and if two courses must not be offered in the same time schedule, because the study program requires it to foresee that both courses must be taken simultaneously, or because there exists at least one student who can take both, then an edge is created joining such two vertices. Particularly, the operation of the academic programs, offered by the School of Engineering of the Autonomous University of Queretaro, are modeled. To solve the graph coloring problem, we consider several algorithmic approaches to determine the minimum number of colors (chromatic number) that are needed to properly color the graph. The number of colors that are needed corresponds to the number of time slots required to offer the courses that a regular student could take, where each subset of vertices of the same color represents the courses that can be offered at the same time without any risk of conflict in the academic schedules. It has been proven that the graph coloring problem is NP-complete, establishing the conjecture that there does not exist an efficient algorithm to produce optimal solution for all problem instances. However, the algorithms described in this article produce approximated results which are near optimal and efficient.**

***Keywords*— Academic Time Schedule, Chromatic Number, Graph Theory, Optimization, Scheduling**

I. INTRODUCCIÓN

EN teoría de grafos, la coloración de grafos es un problema que consiste en asignar un color a cada nodo de manera que ningún vértice tenga el mismo color que su vértice adyacente [6]. Este problema tiene sus inicios en el problema de los 4 colores [7], y fue planteado por primera vez en 1852 por el matemático Francis Guthrie cuando intentaba colorear el mapa de Inglaterra. El problema plantea la noción de que cualquier mapa puede tener todos los territorios coloreados con solo 4 colores, de manera que cada territorio tenga un color diferente a sus territorios adyacentes. Más adelante George David Birkhoff demostró que dicha noción era verdadera en sus estudios de polinomios cromáticos [3]. Dado un grafo no dirigido *G* la función polinomial *p (G, λ)* representa el número de formas que se puede colorear apropiadamente utilizando a lo más λ colores [8]. El problema está resuelto para mapas, ya que no es posible tener más de 4 territorios compartiendo fronteras entre sí. Sin embargo, en teoría de grafos el problema se vuelve más interesante, ya que para colorear un grafo se pueden requerir más de 4 colores. Considere, por ejemplo, el grafo cuyos nodos tienen aristas apuntando a todos los demás nodos, esto es, un grafo completo $K_n = (V, E)$ con un conjunto de vértices *V* y un conjunto de aristas *E*, donde $|V| = n$ y $|E| = n\,(n-1)\,/\,2$. Para K_n se requieren entonces n colores diferentes para colorear el grafo apropiadamente [4]. En la Figura 1 se muestra como el grafo K_6, el cual necesita 6 colores para poder ser coloreado apropiadamente.

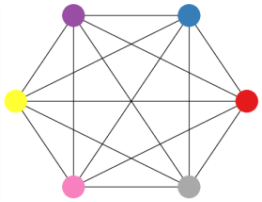


Figura 1. Grafo completo de tamaño K_6

Se ha demostrado que el problema de coloreo de vértices es un problema NP-completo [7], planteando un gran desafío computacional, ya que para encontrar la solución óptima para toda instancia del problema de coloreo de grafos, se requiere grandes cantidades de tiempo de cómputo. Dada la naturaleza de los problemas NP-completos, rara vez se trata de encontrar la solución óptima, por lo que el problema se replantea como encontrar una solución factible utilizando heurísticas, metaheurísticas o algoritmos de aproximación, los cuales eventualmente pueden producir resultados cercanos a la solución óptima. Por ejemplo, suponga que un grafo tiene un número cromático óptimo de 9 ($n_k = 9$), esto es, se necesitan como mínimo 9 colores diferentes para colorearlo de manera apropiada. Un algoritmo de aproximación o heurística podría dar como resultado 10 ($n_k = 10$), el cual podría ser un resultado aceptable pero que se puede producir en tiempos de cómputo muy razonables. Existen diversas técnicas que tratan de solucionar el problema de coloreo de grafos. Ninguna de estas soluciones garantiza soluciones optimas, sin embargo, son mucho más rápidas que tratar de encontrar soluciones óptimas

H. A. Montúffar, Universidad Autónoma de Querétaro (hans_mntfr@hotmail.com),
A. González, Universidad Autónoma de Querétaro (aglez@uaq.mx)
C. Sosa, Universidad Autónoma de Querétaro (carsosa@uaq.mx),
F. González, Universidad Autónoma de Querétaro (fglez@uaq.mx)
G. Díaz, Universidad Autónoma de Querétaro (gdiaz@uaq.mx),
S. Tovar, Universidad Autónoma de Querétaro (saul.tovar@uaq.mx)
Corresponding author: Arturo González Gutiérrez

II. MATERIALES Y MÉTODOS

Modelación

El proponer la coloración de grafos para el desarrollo de horarios fue una idea propuesta por primera vez en 1979 [5]. Para poder aplicar el coloreo de vértices a un sistema de planeación es necesario definir los eventos como nodos. Si un nodo tiene una arista que lo conecta con otro nodo esto quiere decir que ambos nodos son eventos que deben ocurrir al mismo tiempo. De esta manera se pueden planear los horarios de materias que ofrece la facultad de ingeniería en ambos campus (Centro Universitario y Campus Aeropuerto). Para poder modelar los horarios como un grafo se necesita información de las materias que se ofrecen, particularmente la clave y nombre de la materia, así como el semestre en el que un estudiante regular debe cursarla.

En la Tabla 1 se tiene la información del 1er. Semestre de la carrera de Ingeniería Biomédica.

TABLA I
PLAN DE 1er SEMESTRE DE BIOMÉDICA

Clave	Nombre	Semestre
10	BIOLOGÍA	1
203	ÁLGEBRA LINEAL	1
204	QUÍMICA	1
206	UNIVERSIDAD Y SOCIEDAD	1
214	PROBABILIDAD Y ESTADÍSTICA	1
811	CALCULO DIFERENCIAL	1

En la figura 2 se presentan las materias del primer semestre de la carrera de Ingeniería Biomédica, modelado mediante un grafo y conforme al problema establecido de coloreo de grafos.

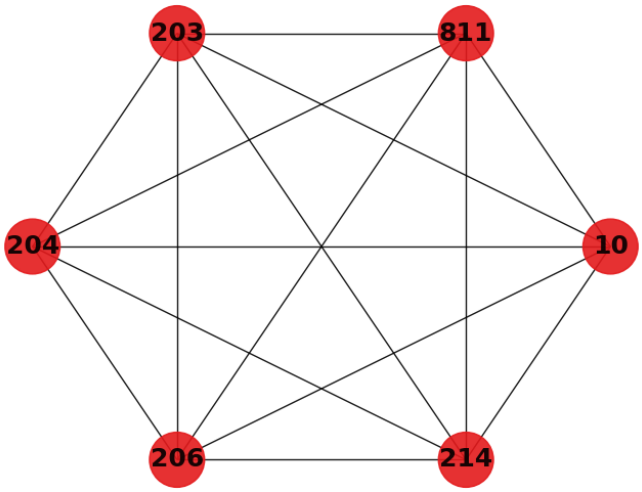


Figura 2. Modelado del primer semestre de Ingeniería Biomédica

Se puede apreciar como todas las 8 materias están conectadas entre sí, indicando la necesidad de que todas se ofrezcan en un horario diferente.

En la Tabla 2 se muestran las materias del primer semestre de Ingeniería Física.

TABLA 2
PLAN DE 1er SEMESTRE DE FÍSICA

Clave	Nombre	Semestre
203	ÁLGEBRA LINEAL	1
204	QUÍMICA	1
206	UNIVERSIDAD Y SOCIEDAD	1
811	CALCULO DIFERENCIAL	1
1291	FÍSICA I	1
1292	LABORATORIO DE QUÍMICA	1
1293	TALLER OPTATIVO 1	1

Si aplicamos el proceso de modelación a cada uno de los semestres de todas las carreras se pude construir el grafo general de todo el campus aeropuerto. En la figura 3 se muestra la modelación del primer semestre de Ingeniería Física. Este caso, a diferencia del primer semestre de Ingeniería Biomédica, cuenta con 7 materias.

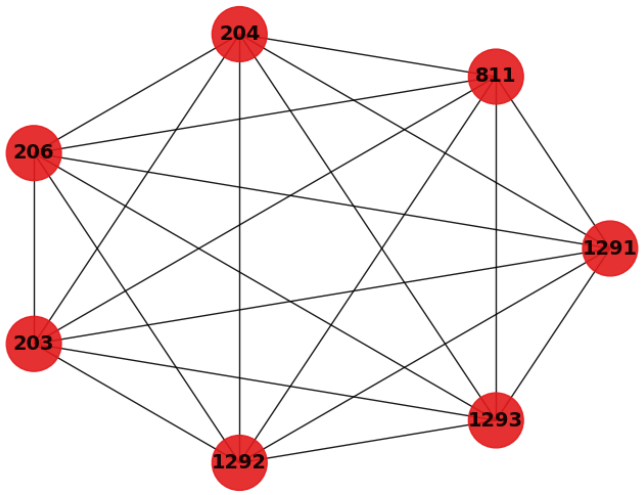


Figura 3. Modelado del primer semestre de Ingeniería Física

Sin embargo, considerando que hay materias que son comunes en varios programas académicos, los grafos de cada carrera pueden resultar interconectados. Tomando como ejemplo las figuras 2 y 3 se puede ver que el primer semestre de Ingeniería Física y el primer semestre de Ingeniería Biomédica comparten las materias con claves 203, 204, 206 y 811. Dado que se trata de materias comunes se pueden ofertar los cursos en el mismo horario para cada materia. En la figura 4 se muestra el grafo resultante al combinar ambos grafos de las figuras 2 y 3.

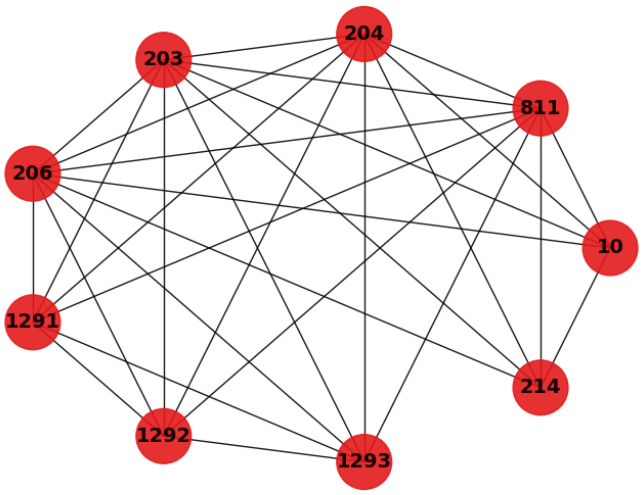


Figura 4. Modelado del primer semestre de Ingeniería Física junto con el primer semestre de Ingeniería Biomédica

El proceso de modelación se hizo para todos los semestres de las 3 carreras ofrecidas en campus aeropuerto (Ing. Física, Ing. en Biomédica e Ing. en Nanotecnología). Cabe mencionar que en el modelo propuesto no se consideran las materias de los planes de estudio como las que se indican como Servicio Social y Prácticas Profesionales, ya que no son tomadas en un aula. La misma situación se da con las materias de Deportes y Cultura Deportiva, ya que, aunque se dan en la escuela no requieren aula. Finalmente, no se consideraron las materias de inglés y de Segundo Idioma, puesto que el horario y aula para estas, es determinado en un espacio específico por el personal administrativo de la facultad. Una vez que se aplican esos filtros se modela el grafo que describe las materias que son ofrecidas por todas las carreras. La figura 5 muestra el grafo que representa todas las materias de los tres programas académicos

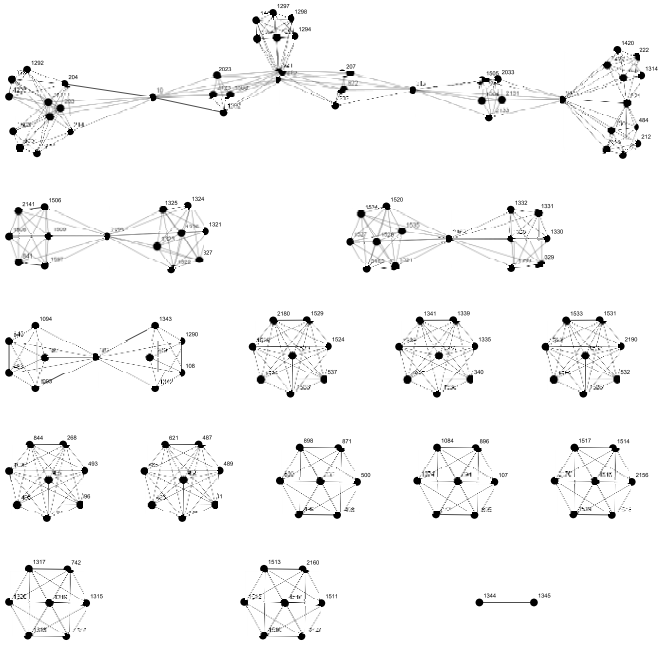


Figura 5. Modelado de todas las materias de los programas de Ingeniería en Nanotecnología, Física y Biomédica ofrecidos en el Campus Aeropuerto

Debido a que la universidad generalmente ofrece más de una clase de la misma materia al grafo se le aumentan los nodos.

Este incremento ocurre duplicando el nodo y sus aristas por cada uno de los grupos requeridos. Este resultado se puede apreciar en la Figura 6.

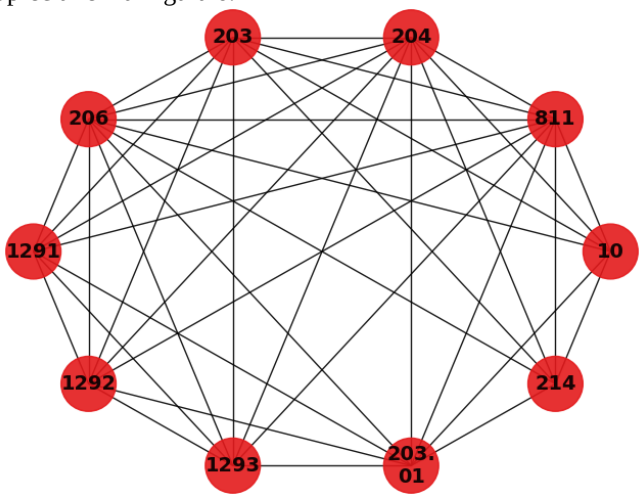


Figura 6. Clonación aplicada al nodo 203.

Finalmente se crean las aristas basadas en los maestros. Estas aristas indican que un maestro enseña ambas materias por lo que no pueden ser enseñadas en el mismo espacio temporal. Este proceso se puede observar en la Figura 7.

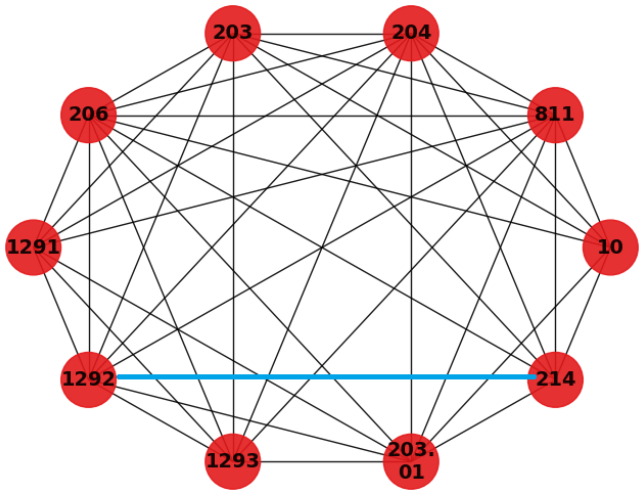


Figura 7. Arista creada entre el nodo 1292 y 214.

Algoritmos

El problema de coloreo de grafos se define formalmente de la siguiente manera:

Entrada: Dado un grafo $G = (V, E)$, con el conjunto V de vértices y el conjunto E de aristas, donde $|V| = n$, y un conjunto de colores $C = \{c_1, c_2, \ldots, c_m\}$, para $m \leq n$.

Salida: Producir una asignación $f: C \rightarrow V$ de colores en C al conjunto de V de vértices del grafo de tal modo que existe una arista $e = \{v_i, v_k\} \in E$ entonces los vértices v_i, v_k deben de ser coloreados con diferente color.

Para resolver el problema de coloreo de grafos se siguen 2 enfoques. El primero consiste en en tres heurísticas que consideran un orden previamente establecido, la primera heurística procede coloreando los vértices conforme al orden establecido por el etiquetamiento de los nodos v_1 *a* v_n del vértice 1 al vértice n. La segunda y tercera heurística siguen el orden establecido por el grado de los vértices. Así la segunda heurística ordena los nodos de mayor a menor grado, y conforme a dicho orden se procede a colorear. Similarmente la tercera heurística lleva a cabo el proceso de coloreo, pero a partir de un ordenamiento de vértices de menor a mayor grado. Estos algoritmos siguen el siguiente proceso:

Paso 1: $i = 1, j = 1$.

Paso 2: Para todos los vértices v_i:

Paso 3: Si para todas las aristas $\{v_i, v_k\} \in E$, ningún vértice v_k está coloreado con c_j entonces asignar color c_j a vértice v_i. Siendo v_k todas las aristas adyacentes con v_i

De otro modo elegir el mínimo j para colorear vértice v_i con color c_j.

Paso 4: Reportar a j como el número cromático del grafo.

En el segundo enfoque se utilizaron las siguientes metaheurísticas:

- Búsqueda Local (*Local Search*) [1]
- Búsqueda Tabú (*Tabu Search*) [2]
- Algoritmo Genético [6]

Experimentación

La eficiencia de estos algoritmos es probada utilizando el desafío de DIMACS (*The Center for Discrete Mathematics and Theoretical Computer Science*). Este desafío cuenta con 125 grafos de los cuales se conoce el número cromático óptimo de 69 de ellos. El resultado se mide utilizando 3 métricas:

- Cuantas veces el algoritmo encontró el resultado óptimo.
- La razón de aproximación del resultado (El resultado obtenido dividido entre el resultado óptimo).
- El tiempo de ejecución del algoritmo (en segundos).

Procesamiento Final

Después de que se ha obtenido un coloreo para el grafo generado por la carga horaria se aplica la función de balanceo. Esta función trata de que cada color tenga el mismo número de nodos.

En la Figura 8 se puede observar el número de materias en cada clúster antes del balanceo.

206, 211, 882	3
10, 204, 212, 213, 332, 488, 497, 896, 897	9
10.01, 203, 207, 228, 485, 495, 495.01, 498, 1084, 1093	10
41, 229, 268, 500, 811, 821, 845, 873	8
202, 214, 214.01, 230, 483, 493, 499, 621, 1074	9
107, 203.01, 204.01, 206.01, 215, 221, 489, 811.01, 871, 871.01, 1095, 1095.01	12
484, 486, 822, 844, 894, 898	6
211.01, 212.01, 221.01, 228.01, 229.01, 484.01, 487, 494, 895, 899	10
230.01, 492, 496	3
494.01, 496.01, 497.01, 844.01	4
	74

Figura 8. Coloreo antes del balanceo.

Este proceso se hace con el propósito de utilizar el mayor número de salones posibles en cada horario. En la Figura 9 se puede ver el coloreo después de que se aplica la función de balanceo.

214, 483, 484, 486, 822, 844, 894, 898	8
206, 211, 215, 871, 871.01, 882, 1095, 1095.01	8
41, 229, 268, 500, 811, 821, 845, 873	8
10, 203, 213, 230.01, 492, 496, 498, 1084	8
204, 212, 332, 488, 497, 896, 897	7
211.01, 487, 494.01, 496.01, 497.01, 844.01, 895	7
10.01, 207, 228, 485, 495, 495.01, 1093	7
107, 203.01, 204.01, 206.01, 221, 489, 811.01	7
212.01, 221.01, 228.01, 229.01, 484.01, 494, 899	7
202, 214.01, 230, 493, 499, 621, 1074	7
	74

Figura 9. Coloreo después del balanceo

III. RESULTADOS Y DISCUSIÓN

Los algoritmos fueron probados para poder saber su efectividad en un sistema web. En la Tabla 3 se pueden ver los resultados obtenidos por cada uno de los algoritmos.

TABLA 3
Resultados experimentales

Algoritmo	Veces que se encontró el óptimo	Promedio de las razones	Razón más grande
Tabu Search	34/69	1.28	3.2
Local Search	32/69	1.28	3
Mayor Grado	27/69	1.34	3
Genético	19/69	1.43	3.6
Ordenamiento	15/69	1.45	3.6
Menor Grado	11/69	1.52	3.8

Como se puede observar en la tabla Tabu Search fue el algoritmo que dio mejores resultados. Sin embargo, dado que el algoritmo debe de ser eficiente en un servidor el tiempo de ejecución es un factor a considerar.

En las Figuras 10 y 11 se presenta el tiempo de ejecución de cada uno de los algoritmos para cada uno de los casos de estudio.

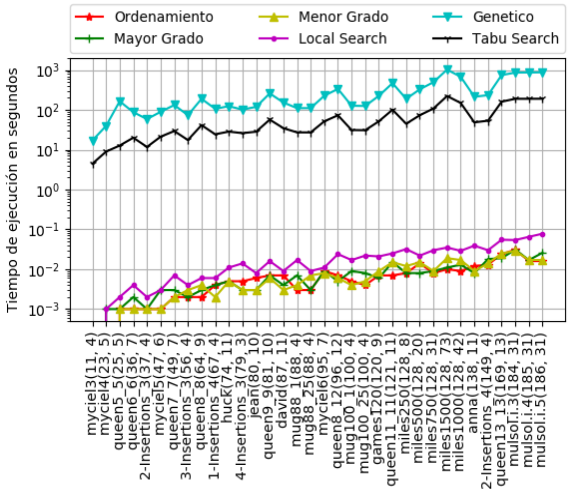


Figura 10. Resultados de tiempo de ejecución de los algoritmos (1/2).

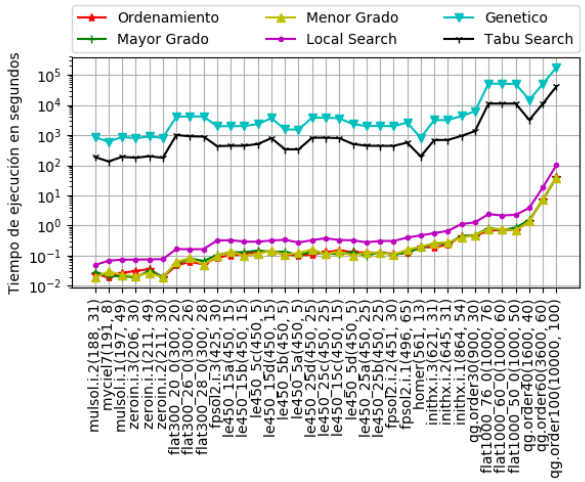


Figura 11. Resultados de tiempo de ejecución de los algoritmos (2/2).

En el eje vertical de las figuras se puede ver el tiempo en segundos que toma cada algoritmo y en el eje horizontal se ve el nombre del caso de estudio, los cuales están ordenados dependiendo del número de nodos con los que cuenta el grafo.

En la gráfica se puede apreciar que Tabu Search y el algoritmo genético tardan significativamente más tiempo que el resto de los algoritmos. Este comportamiento no los hace funcionales en un servidor web.

En las figuras 12 y 13 se puede apreciar el resultado de número cromático que obtuvo cada algoritmo.

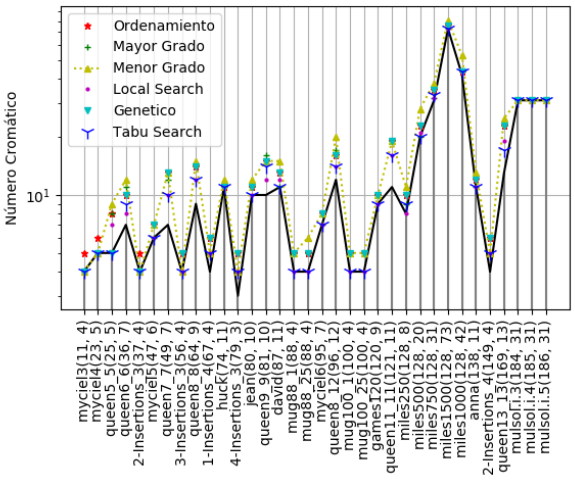


Figura 12. Resultados de número cromático obtenido por los algoritmos (1/2).

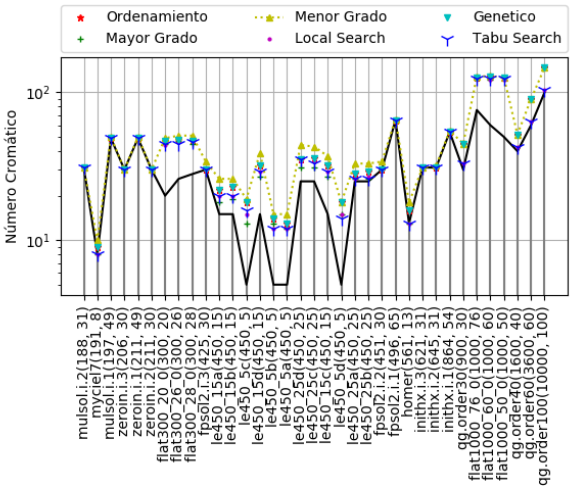


Figura 13. Resultados de número cromático obtenido por los algoritmos

En las figuras 14 y 15 se puede apreciar la razón de aproximación que tuvo cada algoritmo.

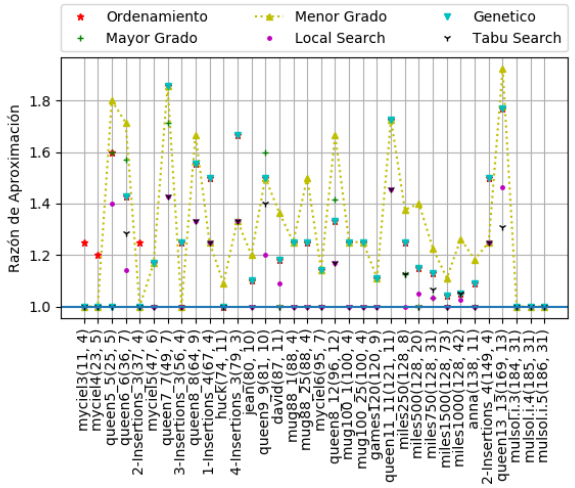


Figura 14. Resultados de razón aproximación de los algoritmos (1/2).

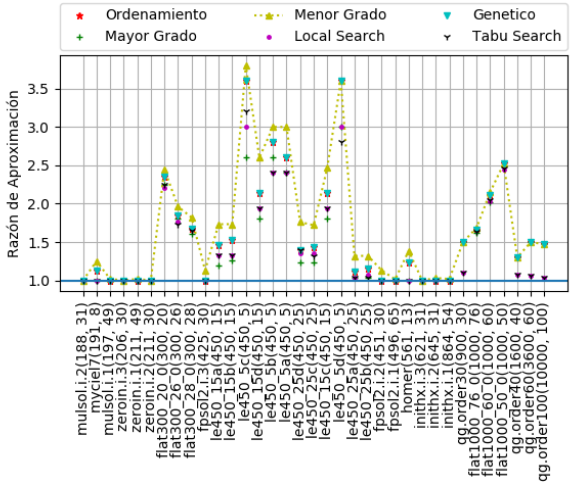


Figura 15. Resultados de razón aproximación de los algoritmos (2/2).

Sistema

Con esta información se desarrolló un sistema web que permite la planeación de horarios de la Facultad de Ingeniería.

El sistema viene precargado con la información de que materias son requeridas por cada semestre para cada carrera del programa de estudio.

Para hacer uso de este sistema el primer paso es escoger en la pantalla inicial el campus del cual se requiera hacer el horario (campus Aeropuerto o campus Centro Universitario) como se puede observar en la Figura 16.

Figura 16. Menú principal del sistema

Después de escoger un campus el sistema rediccionara al usuario a un formulario. El cual se observa en la figura 17.

Figura 17. Formulario de carga.

En este formulario se deberá cargar un archivo con la información de la carga horaria de las asignaturas que se ofertan por maestro.

La otra entrada en el formulario es un número que refiere al número de salones con los que se puede contar para las materias.

Dicho archivo deberá ser una hoja de Excel, esta hoja deberá de contener 3 columnas con la siguiente información:

- En la primera columna deberá de ir la clave del maestro.
- En la siguiente columna deberá de ir la clave de la materia.
- En la siguiente columna deberá de ir el nombre del programa que solicita la materia.
- Si el maestro enseña la misma materia a más de un grupo, esta deberá de estar repetida de acuerdo a la cantidad de veces que imparta la asignatura.

Un ejemplo de la carga necesaria para que el sistema funcione se puede observar en la Tabla 4.

TABLA 4
Ejemplo de carga

9161	228	Biomédica
9161	228	Biomédica
9161	1278	Biomédica
9161	1278	Biomédica
9161	1327	Biomédica

Consideraciones:

- En este caso el maestro con la clave 9161 enseña las materias 228, 1278 y 1327, sin embargo, enseña las materias 228 y 1278 a más de un grupo por lo que aparecen más de una vez.
- En este caso todas las materias son requeridas por la carrera de Ingeniería Biomédica.

El formulario deberá de ser enviando dando click en el botón de Enviar.

Ya enviado el formulario con el archivo el sistema calculara cuantos espacios temporales se necesitarán para poder ofrecer las materias sin ningún tipo de cruzamiento.

El sistema compara el coloreo generado por los algoritmos descritos y utiliza el coloreo que necesite menos colores mostrando una solución de horarios.

Después de que el sistema termine de calcular el cómo ofrecer la carga horaria el usuario será redirigido a una página donde podrá acomodar el horario a su conveniencia. Como se puede apreciar en la Figura 18.

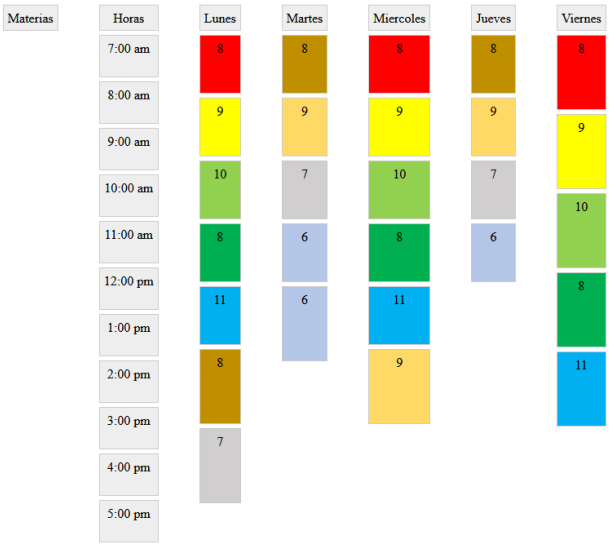


Figura 18. Ejemplo del sistema

Este formulario indica los clústeres de materias que se necesitan separados por colores, al hacerle click en los bloques se despliega las claves y nombres de las materias que corresponden al mismo conjunto de materias que se ofertan.

IV. CONCLUSIONES

El sistema creado a partir de la investigación realizada muestra que es eficiente en la creación de horarios. Este sistema demuestra la flexibilidad deseada a la hora de generar horarios. También garantiza que el horario generado no tendrá ningún tipo de empalmes ni de maestros o alumnos. Por lo que se refiere al modelo básico que se ha presentado en este artículo, se puede concluir que los resultados representan la cota mínima de requerimientos de infraestructura, ya que se asume que el estudiante que cursa el programa académico es regular.

AGRADECIMENTOS

Agradecemos al personal administrativo de la Facultad de Ingeniería por proveer la información necesaria para poder desarrollar este proyecto. También se agrade a CONACYT por proveer los fondos necesarios por medio de su programa de becas nacionales.

REFERENCIAS

[1] Avrim Blum. New Approximation Algorithms for Graph Coloring. Laboratory for Computer Science MIT, N/A, N/A. 1984.

[2] A. Hertz and D. de Werra, Lausanne. Using Tabu Search Techniques for Graph Coloring. Computing, 39, 345-351. 1987

[3] Birkhoff. George D.A. Determinant formula for the number of ways of coloring a map. The Annals of Mathematics, 14(1/4):42–46 1912

[4] Donald Alexander Mackenzie. Mechanizing Proof (103). Boston: MIT Press 2004

[5] F. Thomson Leighton, A Graph Coloring Algorithm for Large Scheduling Problems., Journal of Research of the Notional Bureau of Standards Vol. 84, No.6, November- December 1979

[6] Musa M. Hindi and Roman V. Yampolskiy. Genetic Algorithm Applied to the Graph Coloring Problem. Computer Engineering and Computer Science, I, N/A. February 2013

[7] Peter Tannenbaum. Excursions in Modern Mathematics with Mini-Excursions. Estados Unidos: Pearson. 1992.

[8] Ralph P. Grimaldi. Discrete and combinatorial mathematics. San Francisco: Pearson Addison Wesley. 1999)

[9] Wilson, R. Introduction to graph theory. New York, NY, USA. John Wiley & Sons 1986

Hansel Amadeus Montúffar Otero

Obtuvo su grado de Licenciatura en Ingeniería de Software de la Universidad Autónoma de Querétaro en el 2016. Actualmente estudia la Maestría en Ciencias en Inteligencia Artificial.

Arturo González Gutiérrez

Es profesor titular a tiempo completo en la Universidad Autónoma de Querétaro desde 1995, donde imparte cursos de posgrado y pregrado en la Facultad de Ingeniería. Obtuvo su doctorado y M.Sc. en Ciencias de la Computación en la Universidad de California en Santa Bárbara, EE. UU. También obtuvo un M.Sc. en Ciencias de la Computación en Inteligencia Artificial del Instituto de Tecnología y Educación Superior de Monterrey, y un grado de Licenciatura en Ingeniería Eléctrica por el Instituto de Tecnología de Morelia. Ha sido distinguido con una certificación PROMEP (Programa para el Mejoramiento de Maestros de Educación Superior) otorgada por el Ministerio de Educación de México. Ha colaborado en diferentes proyectos de investigación relacionados con Algoritmos de Aproximación para Redes apoyados por el Consejo Nacional de Ciencia y Tecnología (CONACyT), el Instituto de la Universidad de California para México y Estados Unidos (UC MEXUS) y el PROMEP. Ha impartido conferencias a nivel nacional e internacional, ha publicado varios artículos indexados por JCR en el área de geometría computacional, algoritmos de aproximación y complejidad computacional.

Carmen Sosa Garza

Obtuvo el grado de Licenciatura en Matemática en 1998 en la UNAM, su Maestría en Docencias de las Matemáticas en la Universidad Autónoma de Querétaro en el 2012, actualmente es la Secretaria Académica de la Facultad de Ingeniería de la Universidad Autónoma de Querétaro (desde 2012).

Fidel González Gutiérrez

Actualmente está inscrito en el programa de Doctorado en Ciencias de la Computación en la Universidad Autónoma de Querétaro. Su investigación doctoral es sobre algoritmos de ruteo en superficies de embaldosados rectilíneos. Obtuvo su Maestría en Ciencias Computacionales en Sistemas Distribuidos de la Universidad Autónoma de Querétaro, y su grado de Licenciatura en Ingeniería en Sistemas Computacionales por el Instituto Tecnológico de Querétaro. Es profesor en la Facultad de Informática de la Universidad Autónoma de Querétaro desde 1997, donde enseña en los niveles de pregrado y posgrado. Es miembro de la ACM e IEEE (IEEE Computer Society y IEEE Intelligent Systems Society).

Guillermo Díaz Delgado

Obtuvo un el grado de Licenciatura en Ingeniería de Sistemas Electrónicos del Instituto Tecnológico y de Estudios Superiores de Monterrey (México), y un M.Sc. Licenciado por la Universidad de Laval (Quebec, Canadá). Se ha desempeñado como ingeniero en varias empresas y ha sido profesor e investigador en diferentes instituciones públicas y privadas. Desde 1991, el Sr. Díaz-Delgado ha

sido profesor titular a tiempo completo en la Facultad de Ingeniería de la Universidad Autónoma de Querétaro. Sus intereses de investigación incluyen: Ad Hoc y redes de sensores, diseño de capas cruzadas para la provisión de QoS, algoritmos de optimización y aplicaciones de inteligencia artificial para redes e ingeniería. Es miembro de IEEE, IEEE Communications Society, IEEE Computer Society, IEEE Information Theory Society y IEEE Intelligent Systems Society.

Saúl Tovar Arriaga

Obtuvo su grado de Licenciatura en Ingeniería en Electrónica en el Instituto Tecnológico de Querétaro, su Maestría en Ciencias en Mecatrónica en la Universidad de Siegen, Alemania, y su Doctorado en Ciencias Biomédicas en la Universidad de Erlangen-Nuremberg, Alemania. Actualmente es profesor de tiempo completo e investigador en la Universidad Autónoma de Querétaro. Sus intereses de investigación incluyen robótica médica, diagnóstico automático por imagen y visión por computadora.