UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA
CENTRO DE INVESTIGACIÓN EN INGENIERÍA
Y CIENCIAS APLICADAS

# Heurística Constructiva para el Modelo de Restricciones de Cursos Universitarios de la Facultad de Ciencias Químicas e Ingeniería

TESIS PROFESIONAL
PARA OBTENER EL GRADO DE:

DOCTOR EN INGENIERÍA Y CIENCIAS APLICADAS
OPCIÓN TERMINAL EN TECNOLOGÍA ELÉCTRICA

P R E S E N T A:

M.C.C. MIREYA FLORES PICHARDO

ASESOR: DR. MARCO ANTONIO CRUZ CHÁVEZ

CUERNAVACA, MORELOS OCTUBRE 2019

Cuernavaca, Morelos, a 07 de octubre de 2019.

**DR. ROSENBERG JAVIER ROMERO DOMÍNGUEZ**
**COORDINADOR DEL POSGRADO EN**
**INGENIERÍA Y CIENCIAS APLICADAS**
**P R E S E N T E**

Atendiendo a la solicitud para emitir DICTAMEN sobre la revisión de la TESIS titulada **"Heurística Constructiva para el Modelo de Restricciones de Cursos Universitarios de la Facultad de Ciencias Químicas e Ingeniería"** que presenta la alumna **Mireya Flores Pichardo,** para obtener el título de **Doctorado en Ingeniería y Ciencias Aplicadas** con opción terminal en **Tecnología Eléctrica.**

Nos permitimos informarle que nuestro voto es:

| NOMBRE | DICTAMEN | FIRMA |
|---|---|---|
| DR. DAVID JUÁREZ ROMERO | | |
| DRA. MARGARITA TECPOYOTL TORRES | Aprobado | |
| DR. ÁLVARO ZAMUDIO LARA | Aprobado | |
| DR. MARTÍN GERARDO MARTÍNEZ RANGEL (FCAeI-UAEM) | Aprobado | |
| DR. MARTÍN HERIBERTO CRUZ ROSALES (FCAeI-UAEM) | Aprobado | |
| DRA. JESÚS DEL CARMEN PERALTA ABARCA (FCQeI-UAEM) | Aprobado | |
| DR. MARCO ANTONIO CRUZ CHÁVEZ | Aprobado | |

**PLAZO PARA LA REVISIÓN 20 DÍAS HÁBILES (A PARTIR DE LA FECHA DE RECEPCIÓN DEL DOCUMENTO)**

**NOTA. POR CUESTION DE REGLAMENTACIÓN LE SOLICITAMOS NO EXCEDER EL PLAZO SEÑALADO, DE LO CONTRARIO LE AGRADECEMOS SU ATENCIÓN Y NUESTRA INVITACIÓN SERÁ CANCELADA.**

# *Resumen*

En este trabajo de investigación se propone un modelo matemático de satisfacción de restricciones que define el problema real de Programación de Cursos Universitarios en la Facultad de Ciencias Químicas en Ingeniería (FCQeI) de la Universidad Autónoma del Estado de Morelos, México.

Se desarrolló un algoritmo de enfoque constructivo para obtener soluciones factibles del modelo propuesto que permita el tratamiento de instancias de los semestres de la FCQeI. Analizando las características de la instancia del semestre Agosto - Diciembre 2015 de la FCQeI se observó que ésta excede en tamaño a la instancia más grande de la clasificación de Metaheuristics Network.

Para el algoritmo de enfoque constructivo se llevó a cabo un análisis de eficiencia con la finalidad de conocer su comportamiento. Los resultados experimentales demostraron que en el 40% de las ejecuciones se obtuvieron soluciones factibles para la programación de cursos universitarios de la FCQeI. Se compararon los resultados del Algoritmo Constructivo y los horarios generados por la administración de la FCQeI. Se observó que, usando el modelo de satisfacción de restricciones es posible mejorar la asignación de horarios para cursos universitarios de la FCQeI de modo que se satisfagan las restricciones del problema y el uso de los salones sea más eficiente, por lo que el algoritmo propuesto cumple con las expectativas planteadas en este trabajo de investigación.

# *Abstract*

In this research work a real mathematical constraint satisfaction model which defines the timetabling problem in the Faculty of Chemical Sciences and Engineering (FCSE) at the Autonomous University of Morelos State, Mexico.
A constructive approach algorithm was developed to obtain feasible solutions of the proposed model that allows the treatment of instances of the semesters of the FCQeI. Analyzing the characteristics of the instance of the August - December 2015 semester of the FCQeI, it was observed that it exceeds in size the largest instance of the Metaheuristics Network classification.

For the constructive approach algorithm an efficiency analysis was carried out in order to know its behavior. The experimental results showed that in 40% of the executions, feasible solutions were obtained for the programming of university courses of the FCQeI. The results of the Construction Algorithm and the schedules generated by the administration of the FCQeI were compared. It was observed that, using the constraint satisfaction model, it is possible to improve the allocation of schedules for university courses of the FCQeI so that the restrictions of the problem are satisfied and the use of the classrooms is more efficient, so that the proposed algorithm complies with the expectations raised in this research work.

# *Agradecimientos*

A CONACYT por brindarme el apoyo económico para poder realización este trabajo de investigación

Al Centro de Investigación en Ingeniería y Ciencias Aplicadas por la oportunidad brindad a una servidora para el estudio del posgrado.

A mi director de tesis, el Dr. Marco Antonio Cruz Chávez, por su dirección y apoyo para llevar a cabo este trabajo de investigación,

A los integrantes de mi comité tutoral y revisores de tesis: Dr. Álvaro Zamudio Lara, Dr. David Juárez Romero, Dra. Margarita Tecpoyotl Torres, Dr. Martín G. Martínez Rangel, Dr. Martín Heriberto Cruz Rosales y a la Dra. Jesús del Carmen Peralta Abarca, por sus consejos y comentarios para la culminar este trabajo de investigación, así como por todo lo que aportaron a mi persona.

# *Nomenclatura*

| | |
|---|---|
| *benchmark* | Instancia de prueba de un problema que es utilizada como paradigma en diversas investigaciones para probar el desempeño de los algoritmos desarrollados |
| *UCTP* | *University Course Timetabling Problem* - Problema de Programación de Cursos Universitarios |
| *ETTP* | Examination *Timetabling Problem* - Problema de Programación de Exámenes |
| *STP* | *School Timetabling Problem* - Problema de Programación de Horarios Escolares |
| *instancia* | Corresponde a los datos de entrada a procesar para un problema determinado. |
| $E$ | Conjunto de eventos. Se utiliza el término evento para generalizar una actividad académica donde participan estudiantes (clase, examen, conferencia, asesoría, etc.) |
| $n_E$ | Número total de eventos |
| $E_l$ | Conjunto de eventos que requieren equipo de laboratorio |
| $n_{El}$ | Número total de eventos que requieren equipo de laboratorio |
| $T$ | Conjunto de ranuras de tiempo |
| $n_T$ | Número total de ranuras de tiempo |
| $D$ | Conjunto de días |
| $n_D$ | Número total de días |
| $P$ | Conjunto de periodos |
| $n_P$ | Número total de periodos |
| $R$ | Conjunto de salones |
| $n_R$ | Número total de salones |
| $R_l$ | Conjunto de salones que cuentan con equipo de laboratorio |

| | |
|---|---|
| $n_{Rl}$ | Número total de salones que cuentan con equipo de laboratorio |
| $S$ | Matrícula de estudiantes |
| $n_S$ | Número total de estudiantes |
| $M$ | Conjunto de matrícula de maestros, compuesto por dos subconjuntos $M = \{M_T, M_N\}$ |
| $M_T$ | Subconjunto de matrícula de maestros titulares |
| $M_N$ | Sbconjunto de matrícula de maestros no titulares |
| $C$ | Conjunto de Clases |
| $n_C$ | Número total de clases |
| $C_i$ | Subconjunto de eventos de cada clase $c_i = \{e_i, e_{j,\ldots}\}$, |
| $F$ | Conjunto de facilidades de un salón |
| $n_F$ | Número total de facilidades |

# *Contenido*

# *Índice de Figuras*

# *Índice de Tablas*

# CAPÍTULO 1. Introducción

La programación o calendarización de recursos es un tema de gran interés dada la gran repercusión en problemas de diversas áreas como la distribución de tareas del personal, problemas de manufactura donde se requiere determinar la secuencia adecuada de operaciones que permita reducir los tiempos de procesamiento de tareas o productos, calendarización de sistemas de transporte, entre otros.

Dentro del área de distribución y logística, se encuentran problemas tan variados como los de distribución de redes hidráulicas, redes eléctricas y problemas de transporte, de este último se derivan múltiples variantes que tratan de modelar problemas reales mediante un enfoque general y siendo el problema del Ruteo Vehicular uno de lo más estudiados debido a su gran similitud con problemas reales presentados en la industria, donde el encontrar una solución que cumpla con todas las restricciones de este tipo de problemas es una tarea complicada, ya que esto dependerá sustancialmente de la complejidad del problema y del tamaño de la instancia [Martínez, Cruz, 2011] [Martínez, et.al., 2012]

En el área académica encontramos la calendarización de horarios, generalmente, donde el problema considera de forma general un conjunto de restricciones: asignación de recursos, asignación de tiempo, restricciones de tiempo, capacidad de los salones, entre otras.

Algunas de las variantes más estudiadas de la calendarización de horarios son [Schaerf, 1999]:

- El Problema de Programación de Exámenes (*ETTP* por sus siglas en inglés – Examination *Timetabling Problem*) El cronograma de un conjunto de exámenes de un conjunto de cursos universitarios, evitando la superposición de exámenes de cursos que tengan estudiantes en común, y espaciando los exámenes de los estudiantes, tanto como sea posible. Este problema ha sido tratado en [Pillay, 2018], [Taha, et.al.,2019]

- El Problema de Programación de Horarios Escolares (*STP* por sus siglas en inglés – *School Timetabling Problem*) Se trata de programar el horario semanal de todas las clases de una secundaria o preparatoria, evitando que los maestros impartan dos grupos al mismo tiempo y viceversa (un grupo con dos maestros al mismo tiempo). Las referencias [Christos, 2003], [Vassilios, 2017] y [Landir, et.al., 2018] son dos ejemplos del tratamiento del STP.

- El Problema de Programación de Cursos Universitarios (*UCTP* por sus siglas en inglés – *University Course Timetabling Problem*) Donde se requiere generar el horario semanal de todas las sesiones de un conjunto de cursos universitarios, minimizando el traslape de sesiones de cursos que tienen estudiantes en común. [Hamed, 2015]
  En esta tesis se aborda un modelo real del UCTP y el estudio experimental se ha publicado en la referencia [Cruz-Chávez, 2016]

En el caso particular de la calendarización de horarios universitarios lo que se busca al solucionar un problema de calendarización de horarios universitarios es obtener la distribución adecuada de materias, profesores y

alumnos en los salones disponibles en la institución, tratando de proporcionar, en la medida de lo posible, los requerimientos necesarios para que cada materia pueda ser impartida satisfactoriamente sin que se presenten traslapes entre los elementos involucrados [Cruz, Martínez, 2013].

La dificultad para encontrar una solución que cumpla con todas las restricciones del problema de programación de cursos universitarios depende por completo de la complejidad del problema y del tamaño de la instancia. Una instancia corresponde a los datos de entrada a procesar para un problema determinado. Por lo que, para instancias pequeñas resulta más sencillo encontrar una solución factible, debido a que el número de variables involucradas es menor, tal sería el caso de abordar el problema de programar horarios para una escuela de educación básica de primaria donde normalmente se cuenta con un número fijo de profesores, salones que son adecuados para el número de alumnos por grado y donde los recursos complementarios están también establecidos y disponibles (un pizarrón en cada salón, proyector, mesas, etc.), la complejidad que se va incrementando conforme crece el tamaño y características de la instancia, como ocurre con el caso de tratar el problema de la programación de cursos universitarios en donde debe ocurrir una asignación eficiente de recursos materiales y recursos humanos para cumplir con los requerimientos del problema.

En este trabajo de investigación se presenta una propuesta de solución al Problema de Programación de Cursos Universitarios de la Facultad de Ciencias Químicas e Ingeniería (FCQeI) de la Universidad Autónoma del Estado de Morelos. Se presenta un Algoritmo Constructivo (AC) adaptado para ser aplicado a las restricciones propias de la instancia de la FCQeI para el semestre Agosto - Diciembre del año 2015. Cabe mencionar que las instancias de la FCQeI varían cada semestre, por lo que la entrada para el Algoritmo

Constructivo debe adaptarse para que se pueda ejecutar y obtener soluciones factibles para cada instancia.

Parte importante del diseño de algoritmos eficientes para resolver un problema, consiste en analizar la complejidad del problema a tratar, ya que existen diversos métodos de optimización cuyas características condicionan su aplicación de acuerdo a la complejidad de los problemas. Para llevar a cabo este análisis se considera la teoría de la complejidad algorítmica y la complejidad de los problemas, temas que se encuentran descritos en los puntos 1.1 y 1.2 de este capítulo. En el punto 1.3 se encuentra la descripción del Problema de Programación de Cursos Universitarios, así como el estado del arte, en el punto 1.4 se describen algunos Métodos Heurísticos y finalmente el objetivo y alcance de la tesis.

## 1.1. Complejidad Algorítmica

Dentro de las Ciencias Computacionales se encuentra un área que se conoce como Complejidad Algorítmica o Complejidad Computacional de los algoritmos. La Complejidad Computacional es una métrica teórica de la que se dispone para poder comparar diferentes soluciones algorítmicas y conocer cómo se comportan éstas al tratar de resolver un problema de computación. [Papadimitriou, Steiglitz, 1998]. Lo anterior nos permite elegir el algoritmo más eficiente para la solución del problema.

La Complejidad Algorítmica estudia la eficiencia de los algoritmos basado en el tiempo y espacio necesarios para resolver un problema computacional, y para ello se analiza la Complejidad Temporal y Complejidad Espacial [Sipser, 2013].

Una vez que se cuenta con un algoritmo que funciona para resolver un problema, es necesario definir los criterios que permitan medir su rendimiento

o comportamiento. Estos deben considerar el uso eficiente de los recursos y la simplicidad del algoritmo.

El uso eficiente de los recursos puede medirse en función de dos indicadores: complejidad espacial (medida de la cantidad de memoria necesaria para ejecutarlo hasta su término) y complejidad temporal (medida del tiempo empleado por el programa para ejecutarse y dar resultado a partir de los datos de entrada, y que considera una aproximación al número de pasos de ejecución que el algoritmo emplea para resolver un problema).

Si para resolver un problema P un algoritmo A requiere de poca memoria del equipo de cómputo o ejecuta un pequeño número de instrucciones comparado con el resto de los algoritmos conocidos que resuelven P, entonces se puede afirmar que A es más eficiente que los restantes cuando se resuelve el problema P [Cruz, et.al., 2016].

El análisis de la complejidad temporal permite observar la eficiencia de un algoritmo, dado que el algoritmo más eficiente es el que toma menos tiempo en ejecutarse para resolver un problema determinado. Ya que los algoritmos deben ejecutarse con problemas grandes la recompensa de diseñar buenos algoritmos se verá en el tiempo de ejecución. [Fethi, 1999].

El tiempo exacto de ejecución podría constituir un posible criterio para medir la eficiencia de un algoritmo, sin embargo, este enfoque se ve afectado por diversas variables entre ellos: la computadora utilizada para ejecutarlo, el compilador empleado, etc. Además, se debe considerar el tamaño de la entrada dado que a menudo está relacionada con la estructura de datos utilizada en el algoritmo. Incluso con el mismo tamaño de entrada, un algoritmo podría realizar una cantidad diferente de trabajo, particularmente si el trabajo realizado depende de los valores de los datos. La complejidad en el peor de los casos es definida como el máximo trabajo ejecutado por un algoritmo. La complejidad en el mejor de los casos es el mínimo trabajo de ejecución de instrucciones de un algoritmo.

Por otro lado, para el cálculo de la *complejidad espacial*, se toma en cuenta la cantidad de memoria requerida para almacenar los datos utilizados por el algoritmo, debido a que la cantidad de datos que pueden ser almacenados está limitada por la cantidad de memoria con que cuenta el equipo. Otros de los factores que comúnmente afectan la eficiencia de un algoritmo son principalmente la programación, la velocidad del procesador y el tipo de compilador.

Un algoritmo eficiente se centra fundamentalmente en la simplicidad y en el manejo adecuado de los recursos, siendo la sencillez una característica fundamental para el diseño de algoritmos, debido a que facilita el cálculo de la función temporal, además de optimizar el número de instrucciones evaluadas por el procesador, lo cual se ve reflejado en el tiempo de ejecución. [A.V. Aho, 1974].

| Función | Tasa de Crecimiento | Ejemplo | Tiempo para n = 1000 (asumiendo 1 paso = 1 microsegundo) |
|---|---|---|---|
| *O(1)* | Constante | 10 | 10 microsegundos |
| *O log(n)* | Logarítmica | 2 log *n* +3 | 23 microsegundos |
| *O (n)* | Lineal | n/2 + 50 | 0.55 milisegundos |
| *O* ($n^2$) | Cuadrática | $2n^2 - 3n$ | 2 segundos |
| *O* ($2^n$) | Exponencial | $2^n$ | $10^{86}$ siglos |
| *O (n!)* | Factorial | n! | $10^{2552}$ siglos |

**Tabla 1. 1** Ejemplos de funciones y su tasa de crecimiento [Fethi, 1999].

La Tabla 1.1 muestra las funciones asintóticas más comunes en el aumento de la tasa de crecimiento, indica cuánto tiempo de ejecución se necesitaría en una computadora hipotética.

De acuerdo a la clasificación mostrada en la tabla 1.1, el algoritmo más eficiente corresponde a aquel que presenta una complejidad constante *T(n) = 1*, lo que significa que, si el tamaño de la entrada aumenta, el tiempo de ejecución del algoritmo se mantiene constante. En el caso de una complejidad logarítmica *T(n) = log(n)*, si el tamaño de la entrada se incrementa 100 veces, el tiempo de ejecución del algoritmo únicamente se duplicará, por lo que se

considera un algoritmo eficiente. Por el contrario, si un algoritmo presenta una complejidad exponencial $T(n) = 2^n$, este se considera sumamente ineficiente debido a que su tiempo de ejecución aumenta exponencialmente conforme se incrementa el tamaño de la entrada.

También puede verse que las tasas de crecimiento exponencial y factorial para un gran tamaño de entrada están más allá de la capacidad de cualquier computadora e incluso más allá de la edad del universo.

## 1.2. Complejidad de los Problemas

La complejidad de un problema es equivalente a la complejidad del mejor algoritmo para resolver tal problema. Un problema es tratable (o fácil) si existe un algoritmo de tiempo polinómico para resolverlo. Un problema es intratable (o difícil) si no existe un algoritmo de tiempo polinómico para resolver el problema. [Talbi, E., 2009]. La mayor parte de los problemas en teoría de la complejidad tienen que ver con los problemas de decisión, que corresponden a poder dar una respuesta positiva o negativa a un problema dado.

Un aspecto importante de la teoría computacional es clasificar los problemas en Clases por su complejidad. Una clase de complejidad representa el conjunto de todos los problemas que se pueden resolver utilizando una cantidad dada de recursos computacionales.

Hay tres clases importantes de problemas: P, NP y NP completo, las clases P y NP completo pertenecen a la clase NP, como se muestra en la Figura 1.1.

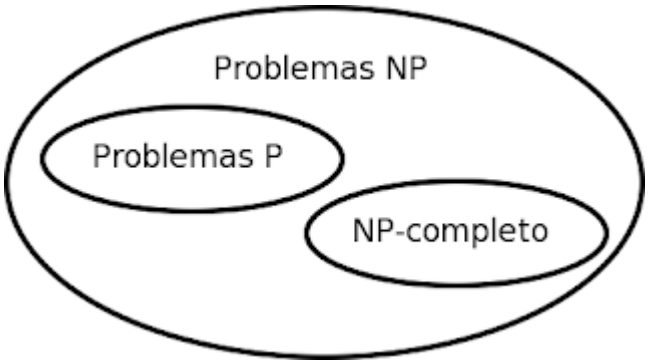


**Fig. 1. 1** Clases de Problemas de acuerdo a su complejidad

La complejidad de la Clase P representa el conjunto de todos los problemas de decisión que pueden ser resueltos por una máquina determinística en tiempo polinomial. Un algoritmo determinístico es polinomial para un problema de decisión A si su peor complejidad está limitada por una función polinómica p(n) donde n representa el tamaño de la instancia de entrada *I*. Por lo tanto, la clase P representa la familia de problemas para los que existe un algoritmo acotado en tiempo polinomial para resolverlos. Los problemas que pertenecen a la Clase P son entonces relativamente fáciles de resolver [Talbi, E., 2009].

Algunos de los problemas clásicos que pertenecen a la Clase P son: árbol de expansión mínimo, problemas de ruta más corta, red de flujo máximo, coincidencia bipartita máxima y modelos continuos de programación lineal. En el libro de Garey y Johnson se puede encontrar una lista más exhaustiva de Problemas fáciles y difíciles de la Clase P [Garey., 1990].

La clase de complejidad NP representa el conjunto de problemas de decisión que puede ser resuelto por un algoritmo no determinístico en un tiempo polinomial. [Cook, 1971]. Un algoritmo no determinístico tiene uno o más puntos de decisión que pueden ser elegidos para explorar las posibles soluciones a un problema.

Los problemas NP completos son problemas de optimización cuyos problemas de decisión asociados son NP completos. Este tipo de problemas pueden ser tratados por algoritmos no-determinísticos que son métodos estocásticos basados en la experiencia y son mejor conocidos como heurísticos [Colorni et al., 1998], [Garey, Johnson, 1990], y también pueden ser tratados por métodos metaheurísticos, mismos que sustentan su funcionamiento con base a fenómenos naturales. Dado que el resultado de los métodos heurísticos y metaheurísticos son una aproximación al resultado óptimo, éstos no tienen prueba de optimalidad y debido a la complejidad de los problemas a los que son aplicados, hasta el momento no se conoce la solución óptima para todas sus instancias.

La mayoría de los problemas de optimización del mundo real son problemas NP completos, para los cuales probablemente no existe un algoritmo eficiente para resolverlos. Estos problemas requieren un tiempo exponencial para ser resueltos de forma óptima. Las metaheurísticas constituyen una buena alternativa para resolver este tipo de problemas.

El presente estudio se centra en la clase de problemas NP que contiene los problemas de decisión que son resueltos por una Máquina de Turing no determinista en tiempo polinómico. El problema específico que se trata es el de asignación de recursos, problema clasificado dentro de la teoría de la complejidad como NP-completo [S. Even, et.al., 1976], [Garey, Johnson, 1990], y cuya importancia estriba en su aplicación práctica puesto que se observa en la asignación de equipo en la industria de la manufactura, en el transporte público o en la calendarización de horarios de escuelas.

## 1.3. Problema de Programación de Cursos Universitarios

### 1.3.1. Descripción del Problema

Los problemas de calendarización de horarios consisten en generar horarios para tareas definidas, buscando cumplir de la mejor manera con condiciones o requerimientos específicos. Estos problemas son muy comunes y se encuentran en distintos tipos de actividades tales como: Actividades Educacionales, Universidades [Andre, Dinata, 2018], [Siam, et.al, 2019], Colegios, Institutos, Facultades, Departamentos [Ben-Ayed, Hamzaoui, 2012], Actividades Deportivas [Ribeiro, 2012], Actividades de Transporte [Shen, et.al., 2015], y otras actividades que involucran a personas y el uso de equipos [Saldaña, 2007], [S. Kırıs, 2014].

En el área educativa, se requiere la creación y planeación de horarios de la forma más eficiente posible en cuanto a la designación y organización del personal, horarios, salones y equipos para conseguir un mejor aprovechamiento de los recursos y tendiente a lograr un mejor desempeño de los alumnos y el personal.

La calendarización de horarios académicos es un problema particular que se encuentra dentro del problema general de asignación de recursos. Este problema de horarios, se conoce en la comunidad científica como *University Timetabling Problem* (UTTP) [Banczyk, 2006].

En el UTTP el personal, que tiene a su cargo la organización de los horarios de alumnos y profesores, se encuentra cada periodo académico con la necesidad de proporcionar una solución lo más adecuada posible de acuerdo a una serie de restricciones. Para un programa educativo que pretende observar medidas de calidad, esto implica un gran esfuerzo.

El problema de programación de horarios en el área educativa consiste en programar las asignaturas de un periodo académico (trimestre, cuatrimestre, semestre), considerando:

- los profesores necesarios en cada asignatura
- los grupos de alumnos que toman un conjunto de asignaturas,
- los días o periodos disponibles,
- los salones requeridos (tratando de que satisfagan un conjunto de restricciones relacionadas con el programa de estudio),
- el personal con que se cuenta y
- las instalaciones.

El problema de Programación de Cursos Universitarios se considera lo siguiente: de todos los recursos que hay que programar, los que se consideran críticos o escasos son los salones de clases puesto que no son recursos que sean fáciles de adquirir, además de ello, la capacidad de los salones juegan un rol importante, mientras que las otras variantes del problema STP y ETTP este factor es frecuentemente ignorado, ya que en la mayoría de los casos de STP y ETTP se asume que cada grupo tiene su propio salón. Por lo anterior es que los salones de clase son los que se van a candelarizar tomando en cuenta sus características y las restricciones de la institución educativa.

El problema de Programación de Cursos Universitarios está sujeto a muchas restricciones, que usualmente se dividen en dos categorías: [Burke, 1997].

Restricciones duras: Son restricciones que forzosamente deben cumplirse para que la solución obtenida sea factible. Ejemplo: Un evento se programa solo en un periodo de tiempo.

Restricciones suaves: Son restricciones que es deseable cumplir, pero su cumplimiento no es esencial para que la solución se considere factible. Se consideran restricciones de preferencia. Ejemplo: los profesores pueden preferir impartir sus materias en un salón en particular

Dada la complejidad del problema de calendarización de horarios académicos su solución se centra en la búsqueda de métodos heurísticos que encuentren buenas soluciones. Los Métodos heurísticos son procedimientos eficientes para encontrar buenas soluciones. En estos métodos, la rapidez del proceso es tan importante como la calidad de la solución obtenida. Un método heurístico es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución [Díaz, 1996].

### 1.3.2. Estado del Arte

#### 1.3.2.1. Modelos Teóricos

Para el tratamiento de Modelos Teóricos, se hace uso de las instancias problema que se conocen como benchmarks, debido a que son utilizadas como paradigmas por la comunidad científica ya que les permite probar el comportamiento de los algoritmos que se implementan para resolver el problema del UCTP. Estas instancias están disponibles en la página del grupo Metaheuristic Network [Rossi-Doria et al., 2003].

El grupo de investigación Metaheuristic Network creó un generador de instancias para producir los benchmarks problema con diferentes características y valores diferentes de parámetros dependiendo de los recursos a programar. El grupo garantiza que todas las instancias que

producen tienen una solución óptima, es decir, aquella una solución que cumple las restricciones duras o suaves. El generador accede a una semilla aleatoria y además toma ocho parámetros proporcionados desde la línea de comandos, estos parámetro especifican los recursos de la instancia. Si se da al generador la misma estructura de parámetros y la misma semilla se producirá la misma instancia problema; pero si se utiliza la misma estructura de parámetros con una semilla diferente entonces producirá una instancia diferente. [Cruz-Rosales, 2010].

Basados en sus investigaciones y pruebas sobre instancias teóricas del UTTP, el grupo Metaheuristic Network, hizo una clasificación de las instancias o benchmarks que reunieron en tres clases de acuerdo al grado de dificultad y tamaño según el número de recursos a programar y las denominaron: pequeñas, medianas y grandes [Rossi-Doria, et al., 2003]. Las características de las tres clases de instancias se muestran a continuación en la Tabla 2.

| Clases | pequeña | mediana | grande |
|---|---|---|---|
| Número de eventos | 100 | 400 | 400 |
| Número de salones | 5 | 10 | 10 |
| Número de características | 5 | 5 | 10 |
| Número de estudiantes | 80 | 200 | 400 |
| Número máximo de estudiantes por evento | 20 | 50 | 100 |

**Tabla 1. 2** Clasificación de instancias

En la literatura, existe información de modelos teóricos de la Programación de Cursos Universitarios (UCTP por sus siglas en inglés *University Course Timetabling Problem*), así mismo de algoritmos para tratar esos modelos. Por ejemplo, en [O. Rossi-Doria, 2003], se presenta el estudio completo de un modelo teórico que no considera a los profesores como una variable. Este modelo es tratado mediante cinco diversas heurísticas computacionales tales como: algoritmos genéticos, colonia de hormigas, Búsqueda local iterada, Recocido Simulado y Búsqueda Tabú.

También se ha trabajado con otras instancias definidas con un valor distinto en los parámetros que las listadas en la tabla de Rossi-Doria y que consideran 400 eventos, hasta 20 salones, máximo 20 características y 1000 estudiantes [Solís, et.al., 2008].

En el Centro de Investigación en Ingeniería y Ciencias Aplicadas (CIICAp) de la Universidad Autónoma del Estado de Morelos (UAEM), se ha desarrollado el proyecto incluido en la referencia [Cruz-Rosales, 2010], en donde se programó un planificador que provee un modelo teórico para la generación de horarios utilizando Recocido Simulado aplicando a benchmarks (medium 01–05, hard 01, and hard 02) de Metaheuristic Network, un proyecto de la Comisión Europea, cuyo objetivo es comparar empíricamente y analizar el desempeño de varias meta heurísticas en diferentes problemas de optimización combinatoria, incluida la programación de horarios [C. Di Stefano, 2003]. Se obtuvieron soluciones factibles para las instancias medianas y grandes observando mayor eficiencia y eficacia de las que otros algoritmos reportaron en la referencia [C. Di Stefano, 2003].

La referencia [T. Thepphakorn, 2015] se propone una herramienta basada en un algoritmo evolutivo para resolver problemas de programación de cursos. Para evaluar la eficiencia de este algoritmo, el equipo de investigación trabajó con catorce benchmarks de los problemas de programación de cursos universitarios.

Los autores de [Song, 2018] proponen un algoritmo de búsqueda local iterada para encontrar soluciones factible para el problema de programación de cursos universitario. El algoritmo propuesto involucra tres fases clave: inicialización, intensificación y diversificación. Una vez que se construye un horario inicial parcialmente factible, se realiza una búsqueda local basada en recocido simulado y un procedimiento de diversificación que produce una perturbación moderada o incluso una mejora en la solución actual de manera

iterativa hasta que se cumpla una condición de parada. El algoritmo propuesto se evalúa en un conjunto 60 instancias del problema. Destacan que el algoritmo puede encontrar soluciones viables para 58 instancias en un tiempo.

Más recientemente, en [Peyman, et.al., 2019], trataron el programa de cursos universitarios considerando el riesgo de cancelación de cursos por falta de estudiantes inscritos. Ellos desarrollaron un modelo de programación estocástica en dos etapas con un enfoque heurístico. El desempeño del algoritmo desarrollado se analizan utilizando instancias de prueba generadas aleatoriamente.

#### 1.3.2.2. Modelos Reales

La programación de cursos universitarios es aún más complicada cuando se trata de resolver una instancia real. Las instancias de éstos problemas a menudo implican restricciones que escapan de una representación exacta tales como restricciones relacionadas con las preferencias personales del usuario [Even, 1976].

El UCTP se ha abordado utilizando modelos aplicados a casos reales de universidades del mundo, haciendo uso de diversas heurísticas que permiten descartar del espacio de búsqueda aquellas regiones donde no se espera que exista un óptimo.

Por ejemplo, en [C. Di Stefano, 2003], se aplica un algoritmo evolucionario en el Sistema Escolar Italiano con la hibridación de dos heurísticas basadas en búsqueda local. En [E. Yu, 2002], aplican un algoritmo genético basado en sectores, reportando buenos resultados al ser aplicado a un caso real.

En [Causmaecker, et.al. 2009] presentan un enfoque metaheurístico para resolver un problema de programación de cursos universitarios en el mundo real en la Escuela de Ingeniería KaHo Sint-Lieven En este problema se consideran los traslapes y los horarios semanales irregulares. Se incluye una metaheurística de búsqueda que intenta resolver las restricciones una por una, en lugar de tratar de encontrar una solución para todas las restricciones a la vez.

La referencia [Kahar, 2010] presenta un estudio de programación de exámenes en la Universidad Pahang de Malasia, su estudio comprende la comparación de los resultados de una heurística constructiva con un software. Este estudio considera cuatro conjuntos de restricciones duras y tres restricciones suaves. Los autores indican que las soluciones obtenidas por la heurística superan aquellas obtenidas por el software que fue implementado en la Universidad para tratar todas las restricciones del caso. En [Basir, 2013], fue implementada la meta heurística de Recocido Simulado, y el trabajo preliminar presentado trata con el problema de programación de cursos en la Universidad Tamhidi de Malasia. La instancia que se trabajó incluía 800 estudiantes, agrupados en cinco programas de estudio, y los estudiantes asistían a los mismos cursos. Los autores trabajaron con tres conjuntos de restricciones duras y tres de restricciones suaves.

En [Gunawan, 2012], obtienen una solución inicial por medio de programación matemática basada en la relajación Lagrangiana. Éste método ha sido probado en instancias de la Universidad de Indonesia. Fueron considerados once requerimientos del problema, que fueron divididos en dos grandes categorías: cuatro requerimientos representan el conjunto de requisitos para la asignación de profesores, mientras el resto de requerimientos estaban relacionados con el problema de programación de cursos. Los experimentos computacionales fueron ejecutados con éxito con dos conjuntos de problemas reales de la Universidad de Indonesia.

En la referencia [Chacha, 2013] desarrollaron tres modelos del Problema de Programación de Cursos Universitarios para la Universidad Mkwawa College of Education, para las pruebas realizadas se utilizaron problemas reales de la Universidad mencionada. Fue posible obtener la solución óptima para instancias del problema real por medio de reformulaciones de modelos que implicaron una mezcla de variables binarias indexadas.

En el artículo publicado por [Kaviani, et.al, 2014] se propuso un algoritmo heurístico para el problema de programación de cursos universitarios. Dado que el algoritmo propuesto tiene un comportamiento aleatorio o estocástico, proporcionaron más de un horario como solución al problema y después eligieron la mejor solución. El algoritmo propuesto fue probado con datos reales de la Universidad Najaf Abad Islamic Azad, de los departamentos de Ingeniería Industrial de Educación Superior y Maestría en el semestre del ciclo 2012-2013. Los resultados revelaron que el enfoque aplicado fue capaz de generar un horario que puede satisfacer los fines de la Universidad.

En China, [Zhang, 2014] utilizaron un enfoque heurístico codicioso para el UCTP. Se consideraron condiciones complejas con múltiples restricciones, basadas en el profesorado, recursos de la Universidad (que incluye profesores y estudiantes), y salones de clases con ranuras de tiempo disponibles. En particular, respecto a las Escuelas de Posgrado en China, hay más de 400 cursos en cada semestre académico y un horario de cursos debe satisfacer nueves requerimientos básicos. En la implementación fueron consideradas las características del conjunto de datos de dos semestres de la Escuela de Posgrado.

En la referencia [Shimazaki, 2015], es propuesto un modelo de programación matemática con un enfoque denominado meta modelo. El modelo puede incluir restricciones implícitas y funciones objetivas implícitas

para la asignación de un conjunto de clases temporales de la Universidad de Toyama. Cada clase tiene restricciones duras y suaves durante un periodo de tiempo específico. Inicialmente, la programación era hecha de forma manual y ese trabajo duraba cerca de tres semanas, con una preparación de 5 o 6 días. Para validar el modelo matemático propuesto aplicaron un algoritmo Branch & Bound. También en 2015, [Phillips, et.al., 2015] presentan un método de programación de lineal para resolver la asignación de salones del Problema de programación de cursos universitarios. Aseguran que funciona incluso para grandes instancias, resultados que han validado a través de experiencias en la Universidad de Auckland, y en instancias de la ITC2007.

En la referencia [Pereira, 2016] presentan un modelo de programación lineal entera que resuelve el problema de programación en el Departamento de Rio de Janeiro, Brasil. El modelo fue diseñado para generar soluciones que satisfagan las preferencias de los administradores de las Facultades. Específicamente, esto significaba asignar el número máximo de profesores con el título académico más alto y minimizando costos al fusionar cursos con programas equivalentes. Otros autores en [Méndez -Díaz, et.al. 2016] trabajaron también un modelo de programación lineal y una aproximación heurística para abordar el problema de programación de horarios en una Universidad de Argentina. Ellos trataron la generalización del problema de programación de cursos universitarios post inscripción PECTP (según las siglas en inglés - Post Enrolment based Course Timetabling) [McCollum, 2010], donde el objetivo es programar un conjunto de eventos en salones y ranuras de tiempo (usualmente en una semana) basándose en las selecciones hechas por estudiantes que atenderán a los cursos. Este algoritmo fue implementado y probado con instancias reales de la Universidad de Argentina mostrando que el enfoque fue factible en la práctica y produce soluciones de buena calidad.

Los autores de [Phillips, et.al., 2017] presentan un enfoque general basado en la programación entera para el problema de perturbación mínima en el horario de los cursos universitarios. Este problema surge cuando un horario existente contiene violaciones de restricciones duras o inviabilidades, que deben ser resueltos El objetivo es resolver estas inviabilidades mientras minimiza la interrupción o perturbación en el resto del horario. Esta situación ocurre comúnmente en horarios prácticos, por ejemplo cuando hay son cambios inesperados en la matrícula de cursos o en las habitaciones disponibles. Su método intenta resolver cada inviabilidad en el vecindario más pequeño posible, y utiliza la exactitud de la programación entera. Operando dentro de un vecindario de tamaño mínimo mantiene los cálculos rápidos y no permite grandes movimientos de eventos del curso, que causan una interrupción generalizada en la estructura del horario. Demostraron la aplicación de su propuesta usando un ejemplo basado en datos reales de la Universidad de Auckland.

Los autores de [Borchani, 2017] investigaron el enfoque de vecindarios variables para abordar un problema específico del problema de programación de cursos relacionado con el caso de la Facultad de Economía y Ciencias de gestión de Sfax en Túnez. El objetivo fue minimizar el total número de ranuras de tiempo desocupadas y el número de clases aisladas para todos los grupos de estudiantes. Desarrollaron once estructuras de vecindario específicas: seis de ellas están diseñadas para corregir las ranuras de tiempo desocupadas, mientras que los otros cinco están diseñados para ajustar las clases aisladas.

Los cálculos se realizan en instancias reales y los resultados muestran que su enfoque supera la solución existente con la eliminación del 52,47% de ranuras de tiempo desocupadas y clases aisladas, en promedio.

En la referencia [Lindahl, et.al., 2018] proponen un algoritmo basado en el método de restricciones para resolver instancias del ITC2007 (International Timetabling Competition) que son ejemplos reales del problema de programación de cursos universitarios de la Universidad Udine de Italia. El algoritmo les permitió resolver los problemas y analizar el impacto de tener diferentes recursos disponibles en la mayoría de los problemas de horarios que resolvieron. Concluyeron que la disponibilidad de los salones, la disponibilidad de periodos y la calidad de los horarios generados son variables que influyen unas en otras.

En el Centro de Investigación en Ingeniería y Ciencias Aplicadas (CIICAp) de la Universidad Autónoma del Estado de Morelos (UAEM), se han desarrollado dos proyectos que toman en consideración el modelo real del Problema de Programación de Cursos Universitarios de la Facultad de Ciencias Químicas e Ingeniería (FCQeI) de la UAEM.

El primero proyecto implicó el desarrollo de un sistema administrativo básico de generación de horarios para la FCQeI que incluyó una interfaz web, hizo uso de un modelo de tres capas, servicio web y bases de datos. A través de este proyecto se desarrolló un sistema de consulta de horarios académicos para profesores y estudiantes. Se propuso un modelo básico de restricciones para evitar materias con traslapes para estudiantes de la FCQeI, pero sin tomar en cuenta restricciones en la asignación de profesores para las asignaturas que imparten [B. Martínez-Bahena, 2008].

El segundo proyecto es el que se documenta en el presente trabajo (ver Capítulos 2, 3 y 4) y cuyos resultados experimentales se han publicado en [Cruz-Chávez, 2016].

## 1.4. Métodos Heurísticos

La existencia de una gran variedad de problemas difíciles de resolver que aparecen en la práctica y que necesitan ser resueltos de forma eficiente, impulsó el desarrollo de procedimientos eficientes para encontrar buenas soluciones aunque no fueran óptimas. Estos métodos, en los que la rapidez del proceso es tan importante como la calidad de la solución obtenida, se denominan métodos heurísticos o aproximados. [Martí, Reinelt 2011].

Un método heurístico es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución [Díaz, 1996].

En contraposición a los métodos exactos que proporcionan una solución óptima del problema, los métodos heurísticos se limitan a proporcionar una buena solución no necesariamente óptima. Lógicamente, el tiempo invertido por un método exacto para encontrar la solución óptima de un problema difícil, si es que existe tal método, es de un orden de magnitud muy superior al del heurístico (pudiendo llegar a ser tan grande en muchos casos, que sea inaplicable).

Existen diversas razones para utilizar métodos heurísticos: [Martí, Reinelt 2011]

- El problema es de una naturaleza tal que no se conoce ningún método exacto para su resolución.
- O bien, aunque existe un método exacto para resolver el problema, su uso es computacionalmente muy costoso.

- El método heurístico es más flexible que un método exacto, permitiendo, por ejemplo, la incorporación de condiciones de difícil modelación.
- El método heurístico se utiliza como parte de un procedimiento global que garantiza el óptimo de un problema donde existe la posibilidad de que el método heurístico proporcione una buena solución inicial de partida.

Existen muchos métodos heurísticos de naturaleza muy diferente, por lo que es complicado dar una clasificación completa. Además, muchos de ellos han sido diseñados para un problema específico sin posibilidad de generalización o aplicación a otros problemas similares. A continuación se dan unas categorías amplias, no excluyentes, en donde ubicar a los heurísticos más conocidos:

**Métodos de Descomposición**

El problema original se descompone en subproblemas más sencillos de resolver, teniendo en cuenta, aunque sea de manera general, que ambos pertenecen al mismo problema.

**Métodos Inductivos**

La idea de estos métodos es generalizar de versiones pequeñas o más sencillas al caso completo. Propiedades o técnicas identificadas en estos casos más fáciles de analizar pueden ser aplicadas al problema completo.

**Métodos de Reducción**

Consiste en identificar propiedades que se cumplen mayoritariamente por las buenas soluciones e introducirlas como restricciones del problema. El objeto es restringir el espacio de soluciones simplificando el problema. El riesgo obvio es dejar fuera las soluciones´optimas del problema original.

**Métodos Constructivos**

Consisten en construir literalmente paso a paso una solución del problema. Usualmente son métodos deterministas y suelen estar basados en la mejor elección en cada iteración. Estos métodos han sido muy utilizados en problemas clásicos como el del viajante.

**Métodos de Búsqueda Loca**l

A diferencia de los métodos anteriores, los procedimientos de búsqueda o mejora local comienzan con una solución del problema y la mejoran progresivamente. El procedimiento realiza en cada paso un movimiento de una solución a otra con mejor valor. El método finaliza cuando, para una solución, no existe ninguna solución accesible que la mejore.

Una serie de métodos con el nombre de metaheurísticas han surgido con el propósito de obtener mejores resultados que los alcanzados por los métodos heurísticos tradicionales.

Las metaheurísticas son estrategias para diseñar o mejorar los procedimientos heurísticos con miras a obtener un alto rendimiento. El término metaheurística fue introducido por Fred Glover en 1986 y a partir de entonces han aparecido muchas propuestas de pautas o guías para diseñar mejores procedimientos de solución de problemas combinatorios.

Las metaheurísticas se sitúan conceptualmente" por encima" de las heurísticas en el sentido de que guían el diseño de éstas, pueden estar compuestas por una combinación de algunas heurísticas, por ejemplo, una metaheurística puede usar una heurística constructiva para generar una solución inicial y luego usar otra heurística de búsqueda para encontrar una mejor solución.

Dentro de las metaheurísticas se encuentran Búsqueda Tabú (Tabu Search), Recocido Simulado (Simulated Annealing), Algoritmos Genéticos, Búsqueda Dispersa (Scatter Search), entre otros [Arindam, 2010].

A continuación se describen de forma general algunos de los algoritmos heurísticos: [Silva, 2017].

Métodos de búsqueda local (Local Search):
Procedimiento basado en la suposición de que es posible encontrar una secuencia de soluciones entre la solución inicial y la final tal que cada una de ellas es ligeramente diferente a la inmediatamente anterior.
Tiene la ventaja de que en poco tiempo puede encontrar soluciones suficientemente buenas para un conjunto amplio de problemas. Este procedimiento puede ofrecer una medida de bondad de la solución encontrada, pero no garantiza que la solución obtenida sea el óptimo global del problema considerado

Búsqueda tabú (Tabu Search):
Este método es un tipo de búsqueda por espacios de búsqueda (entornos), que permite moverse a una solución de entorno aunque no sea tan buena como la actual, de este modo se puede escapar de óptimos locales y continuar la búsqueda de soluciones aún mejores. La forma de evitar viejos óptimos locales es clasificando un determinado número de los más recientes movimientos como “movimientos tabú”, los cuales no son posibles repetir durante un determinado horizonte temporal.
Así, el escape de los óptimos locales se produce de forma sistemática y no aleatoria.

Algoritmos genéticos (Genetic Algorithms):
Los Algoritmos Genéticos son métodos adaptativos, generalmente usados en problemas de búsqueda y optimización de parámetros, basados en la reproducción sexual y en el principio de supervivencia del más apto

Algoritmos Evolutivos (Evolutionary Algorithms):
Son un esquema de representación que aplica una técnica de búsqueda de soluciones enfocada a problemas de optimización, inspirada en la teoría de la evolución de Charles Darwin. Se basa en el algoritmo de selección propio de la naturaleza, con la esperanza de que consiga éxitos similares, en relación a la capacidad de adaptación a un amplio número de ambientes diferentes.

Colonias de hormigas (Ant Colony):
Los algoritmos de colonias de hormigas están basados en el comportamiento de las hormigas. Los biólogos y los entomólogos han descubierto la facilidad que tienen las hormigas para encontrar siempre el camino más corto entre el hormiguero y la fuente de alimento, este comportamiento ha sido estudiado y se ha encontrado que las hormigas mantienen una comunicación indirecta con una sustancia volátil llamada feromona. Con la feromona las hormigas son capaces de crear una ruta y a través del tiempo optimizar sus recorridos obteniendo así un camino corto sin una visión global del terreno. Así que estos algoritmos se caracterizan por simular el comportamiento de las hormigas cuando forman las rutas entre el nido y la fuente de alimento, en base a un rastro de feromonas que depositan en la trayectoria efectuada

Algoritmos Voraces (Greedy Algorithms):
Los algoritmos voraces son algoritmos en los que se toman decisiones locales para llegar a una solución óptima parcial. No deshacen una selección ya realizada; una vez incorporado un elemento a la solución, este permanece

hasta el final y cada vez que un candidato es rechazado, lo es permanentemente.

Recocido simulado (Simulated Annealing):
Es una técnica que hace uso de conceptos originalmente descritos por la mecánica estadística. Tiene su base en el proceso físico de recocido, el cual primero reblandece un sólido mediante su calentamiento a una temperatura elevada, y luego va enfriando lentamente hasta que las partículas se van posicionando por sí mismas en el “estado fundamental” del sólido. El recocido simulado es capaz de encontrar la solución óptima, sin embargo, esta se alcanzará tras un número infinito de pasos, en el peor de los casos.

## 1.5. Objetivos

### *Objetivo General*

Propuesta y desarrollo de una estrategia de solución al problema de calendarización de recursos en horarios de escuela de la Facultad de Ciencias Químicas e Ingeniería de la UAEM.

### *Objetivos Específicos*

1) Definir un modelo matemático para la calendarización de recursos en Facultad de Ciencias Químicas e Ingeniería de la UAEM.
2) Desarrollar un algoritmo basado en una heurística constructiva cuya complejidad temporal máxima esté representada por una función polinomial.
3) Aplicar el algoritmo desarrollado en el caso real de Programación de Cursos Universitarios de la Facultad de Ciencias Químicas e Ingeniería de la UAEM.
4) Comparar los resultados de la implementación del Algoritmo Constructivo con el horario generado por la administración de la Facultad de Ciencias Químicas e Ingeniería de la UAEM.

## 1.6. Descripción y Alcance del Trabajo

El problema tratado es un problema difícil de resolver en el área de Ciencias Computacionales.

Las soluciones a un problema de programación de horarios que son construidas de manera manual, además de tomar días o semanas, no garantizan una solución ideal pues puede darse el caso de que los alumnos se encuentren en la situación de no poder tomar dos materias que deberían porque los horarios se traslapan, puede ocurrir que se pasen por alto algunas restricciones, entre otros casos más.

Esta situación da lugar a un problema cuya resolución se complica cuanto mayor sea el número de restricciones se deban tomar en cuenta.

Así mismo, el problema de asignación de horarios es un problema de gran aplicación práctica y cualquier mejora en su solución se podría adaptar a otro tipo de problemas dentro del área de asignación de recursos.

Por este motivo, se propone para su solución el uso un Algoritmo Constructivo que permitirá automatizar la programación de horarios de acuerdo a los recursos con que se cuentan (salones con diversas características de tamaño, equipamiento, etc., profesores, materias) y observando restricciones duras.

La infraestructura con la que se contó para el desarrollo de la solución al problema de asignación de horarios se puede ver en la sección 4.1.

## 1.7. Organización de la Tesis

En el capítulo 1 se incluye el marco teórico del problema de University Course Timetabling, en el capítulo 2 se define el Problema de Programación de cursos universitarios de la Facultad de Ciencias Químicas e Ingeniería que se aborda en la presente tesis. El capítulo 3 se describe la Metodología de solución del problema de la Facultad de Ciencias Químicas e Ingeniería. En el capítulo 4 se presentan las pruebas experimentales y el análisis de resultados. Las conclusiones y trabajo futuro se encuentran en el capítulo 5.

# CAPÍTULO 2. Fundamentos del Problema de la Facultad de Ciencias Químicas e Ingeniería (FCQeI)

La Facultad de Ciencias Químicas e Ingeniería de la Universidad Autónoma del Estado de Morelos se constituyó como tal el año de 1977 como respuesta a las necesidades del campo industrial del Estado. Actualmente, la FCQeI ofrece las carreras de Químico Industrial, Ingeniería Química, Ingeniería Industrial, Ingeniería Mecánica e Ingeniería Eléctrica.

Las instalaciones con que cuenta la FCQeI para impartir las licenciaturas que ofrece son 41 espacios que incluyen:

- 21 salones de clases ubicados en el edificio principal de la FCQeI
- 1 Laboratorio de Centro de Cómputo Académico
- 2 Laboratorios de Análisis Industriales
    - Laboratorio de Química 3
    - Laboratorio de Química 5
- 2 Laboratorios de Química
    - Laboratorio de Química 1
    - Laboratorio de Química 2
- Taller Multidisciplinario Básico (TAMULBA)
    - Con 1 Laboratorio de Física
    - Y cuenta 10 salones de clase.
- Laboratorio de Operaciones Unitarias (LOU)
    - Con 1 Laboratorio de Operaciones Unitarias
    - Y cuenta con 3 salones de clase.

Las diferentes licenciaturas se estudian en 10 semestres y en un máximo de 12, en turno matutino y/o vespertino. Los primeros dos semestres se estudian en grupos rígidos donde las materias son definidas por la administración de la Facultad, después los grupos son flexibles y se avanza conforme la capacidad del alumno, a partir del tercer semestre, cada estudiante puede elegir las materias que desea llevar en el semestre académico y toma sus materias con el profesor que él quiera. El proceso de admisión a la FCQeI se efectúa cada año y se inician actividades en septiembre y en febrero.

La matrícula de alumnos que existe en la FCQeI es 1,500 en promedio y es variable cada semestre. La FCQeI utiliza en cada semestre, un promedio de 150 maestros titulares y otros más que no son titulares de materias. La FCQeI realiza la programación de horarios de forma manual tratando de tomar en cuenta el aforo de los salones y sus características, disponibilidad de horario de profesores titulares, días y periodos existentes, así como las materias que pueden tomar los alumnos y las restricciones propias en la toma de materias.

## 2.1 Descripción del Problema de Programación de Cursos Universitarios en la FCQeI

Las consideraciones que se tienen en la administración de la FCQeI para generar los horarios escolares son las siguientes:

- Una materia está formada por un subconjunto de 4 o 6 eventos. El número de eventos que forma una materia depende de la materia misma. Los eventos que se programen en un salón de clases, deben de tener las facilidades necesarias aportadas por la infraestructura del salón de clases (televisión, proyectos, conexiones eléctricas necesarias y otros).

- Buscar que todas las materias junto con sus eventos puedan ser programadas en los periodos y salones disponibles en la semana.
- Evitar que los salones de clase sean subutilizados, esto es, que para un grupo pequeño de estudiantes no se asigne un salón con aforo muy grande, sino que, de acuerdo al tamaño del grupo de estudiantes, sea el aforo del salón de clases que se asigne. También, el salón de clases que se asigne al grupo de estudiantes que tome una materia debe de tener el aforo necesario.
- Los maestros titulares pueden decidir el día y periodo en que impartirán sus materias. Debido a que la asignación de maestros titulares es conocido y es elegido por el conjunto de maestros titulares en una calendarización de horarios de materias para un semestre escolar. Lo único que se debe de asignar a los maestros titulares es el salón de clases donde se impartirá el evento.
- Los estudiantes no deben de tomar clases en horario discontinuo, esto es, que no deban asistir a clases por la mañana y tarde en el mismo día y también que los eventos de la misma materia deben de presentarse en periodos consecutivos si aparecen en el mismo día. Las materias que se imparten se componen de 4 o 6 eventos. Esos eventos deben ser programados considerando que la mitad de los eventos que corresponden a una materia se deben impartir en el mismo día en periodos consecutivos. Por ejemplo, si una materia se compone de 4 eventos y el evento 1 es programado el día 1 en el periodo 1, entonces el evento 2 estará programado de forma consecutiva en el periodo 2 del mismo día, mientras que, si el evento 3 es programado en el periodo 4, entonces el evento 4 estará programado de forma consecutiva en el periodo 5 del mismo día.

- Buscar una asignación de salones en la que se haya logrado la programación de horarios que evite el traslape de materias para los alumnos.
- Buscar una asignación de salones en la que se haya logrado la programación de horarios que evite el traslape de materias que imparten los profesores. Las materias de maestros no titulares se programan en los días y periodos que quedan libres después de la programación de las materias de los maestros titulares.

Dado los antecedentes y características en base a las consideraciones definidas del problema de asignación de horarios en la FCQeI, donde nunca no se ha podido generar una asignación de horarios adecuada, que cumpla el total de las consideraciones definidas, en este trabajo, se propone un modelo de satisfacción de restricciones, con una propuesta de solución a través de un Algoritmo Constructivo para la calendarización de recursos de la FCQeI, que permita la generación de horarios de clase que cumpla con las consideraciones más duras que actualmente se toman en cuenta y que en el modelo propuesto en este trabajo para la solución del problema de asignación de horarios en la FCQeI, llamamos restricciones.

El resultado del presente trabajo permite mejorar el proceso de generación de horarios de clases tratando de eliminar los problemas que se presentan actualmente en el cumplimiento de las restricciones, así como de disminuir el tiempo que lleva este proceso de calendarización de recursos en forma manual que es hasta de dos semanas. Lo anterior permitirá obtener horarios con los que se puedan aprovechar los recursos con que cuenta la FCQeI de una forma más eficiente y que a su vez también permitan a los alumnos obtener un mejor aprovechamiento escolar.

## 2.2 Representación simbólica de la instancia del problema para la FCQeI

Una representación del *University Course Timetabling Problem*, es a través de una matriz en 3D. Partiendo de una matriz en 2D, que representa a un salón de clase disponible y que consta de un conjunto de ranuras de tiempo, en este caso 90 ranuras. Una ranura de tiempo *t* es definida como un espacio en 2D en función de las coordenadas de *(día, periodo)*, ver figura 2.

La programación de los eventos para un salón de clases, se debe realizar en las 90 ranuras de tiempo disponibles, por lo que se tiene una matriz en 2D para cada uno de los 41 salones de clase (que incluyen laboratorios) con que cuenta la FCQeI. La figura 1, muestra la representación simbólica para cada salón de clases.

| (d, p) | Lunes | Martes | Miércoles | Jueves | Viernes | Sábado |
|---|---|---|---|---|---|---|
| 1 | *t 1* | *t 2* | *t 3* | *t 4* | *t 5* | *t 6* |
| 2 | *t 7* | *t 8* | *t 9* | *t 10* | *t 11* | *t 12* |
| . | . | . | . | . | . | |
| . | . | . | . | . | . | |
| . | . | . | . | . | . | |
| 15 | *t 85* | *t 86* | *t 87* | *t 88* | *t 89* | *t 90* |

**Fig. 2. 1** Representación simbólica de una tabla de horarios

En la tabla de horarios de la figura 2.1, se pueden observar los periodos de tiempo en que se debe realizar la programación de los eventos en un salón de clases.

Las dos dimensiones que se observan se conforman del conjunto de los días D y de los periodos P del tiempo en que se labora en la FCQeI.

Dado que la instancia cuenta con 6 días y 15 periodos se tienen entonces 90 ranuras de tiempo para un salón de clases, numeradas de t1 a t90 en la figura 2.1.

| (d, p) | Lunes | Martes | Miércoles | Jueves | Viernes | Sábado |
|---|---|---|---|---|---|---|
| 1 | 49 | | | 332 | 50 | |
| 2 | 49 | | 15 | 332 | 50 | |
| 3 | | | 15 | | 29 | |
| 4 | | | 15 | | | |
| ... | | | | | 212 | |
| 12 | | 120 | | | 212 | |
| 13 | | 120 | | | | |
| 14 | | 120 | | | | |
| 15 | | 120 | | | | |

**Fig. 2. 2** Abstracción de una tabla de horarios. Caso FCQeI

La figura 3, presenta un ejemplo de una asignación de eventos para un salón de clase que contempla las variables del caso de la FCQeI con 6 días (Lunes, Martes, Miércoles, Jueves, Viernes, Sábado), 15 periodos de tiempo (p = 1 a p = 15) y la asignación de 16 eventos que corresponde a 7 materias con un identificado ID (15, 29, 49, 50, 120, 212 y 332).

Conforme se realiza la programación de los eventos, a las ranuras de tiempo, se les coloca el número de identificador ID de la materia que corresponde a los eventos que fueron asignados en tales periodos, por ejemplo, en la figura 2, la materia con ID 120 está formada por 4 eventos. Los espacios de ranura de tiempo mostrados en blanco en la figura 2, representan los periodos de tiempo disponibles para programar otros eventos.

Al hacer la programación de eventos en los diferentes salones, la variable *R* (de salones) crea una tercera dimensión para formar el conjunto de Tablas de Horarios que corresponden al conjunto de salones disponibles, tal como se muestra en la figura 4.

Por lo tanto, con el conjunto de salones se obtiene una estructura tridimensional con coordenadas para cada ranura de tiempo *t = (d, p, r)*, por lo que al asignar el total de eventos del conjunto *E* existentes en la FCQeI, en las ranuras disponibles de *t* en la matriz 3D, cumpliendo el total de las restricciones duras (del 1 al 8) presentadas en esta sección en el modelo de satisfacción de restricciones, se genera una solución del UCTP para la FCQeI.

En la estructura tridimensional que se puede ver en la Figura 4, se presenta una abstracción que contempla las variables que definen a las ranuras de tiempo t = (p, d, r), con 5 días (LU, MA, MI, JU, VI), 9 periodos de tiempo (p = 1 a p = 9), y 10 salones (r = 1 a r = 10), con lo que se tendría un espacio de 450 ranuras de tiempo disponibles para asignación de eventos en salones de clase.

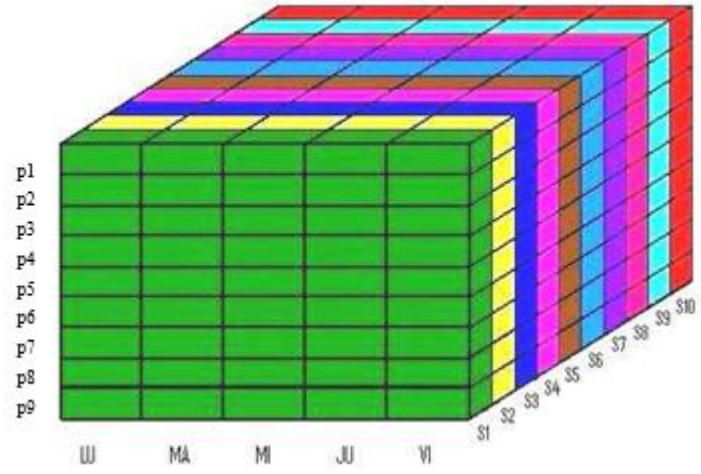


**Fig. 2. 3** Estructura tridimensional de abstracción de tablas de horarios

Al caracterizar de esa forma las variables del caso de estudio de la FCQeI se tienen 6 días (L, M, Mi, J, V, S), 15 periodos de tiempo (Periodo 1 a Periodo 15) , y 41 salones (r = 1 a r = 41), con lo que se tiene un espacio de 3060 (6x15x34) ranuras de tiempo disponibles en aulas de clase y 630 (6x15x7) de esas ranuras de tiempo son disponibles únicamente para los laboratorios en donde se imparten exclusivamente materias que requieren equipo de laboratorio.

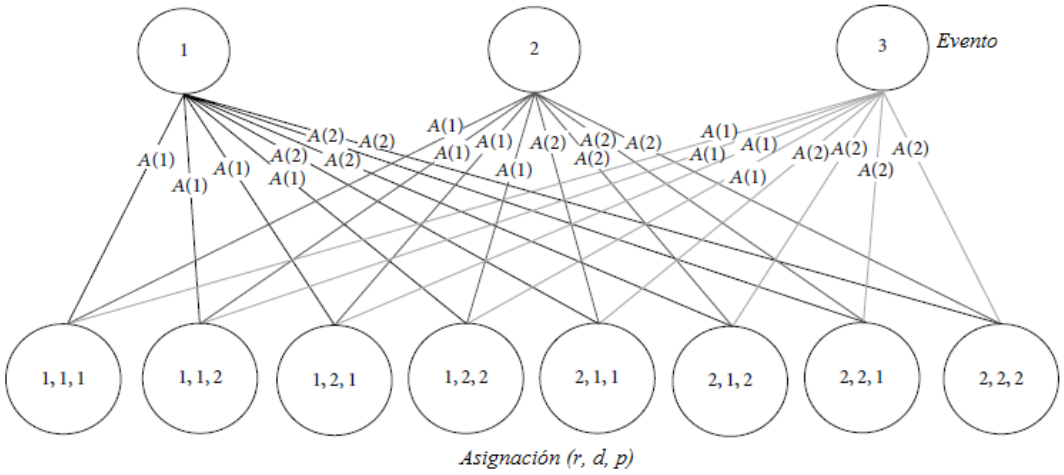


**Fig. 2. 4** Modelo de Grafo bipartita para el problema de Programación de Cursos Universitarios

La Figura 2.4 representa una instancia de 3 eventos y 2 salones para el problema de Programación de Cursos Universitarios, usando un modelo de grafo bipartita. El grafo consiste por un lado, del conjunto de eventos y por otro lado, las asignaciones a ranuras de tiempo $t$ = ($r$, $d$, $p$) que pueden ser ocupadas por eventos. En esta instancia, el problema tiene solo 2 días $d$ y 2 periodos $p$. Cada salón $r$ tiene una capacidad $A(k)$ y puede ser asignado a un solo evento en cada ranura de tiempo $t$. El evento $e_i$ que es asignado a la ranura de tiempo $t$ indica que el evento se llevará a cabo en el salón $r$ el día $d$

durante el periodo $p$. El objetivo es determinar la ranura de tiempo $t$ para cada evento $e_i$ en el grafo bipartita, tomando en cuenta la capacidad ($k$) del salón $r$.

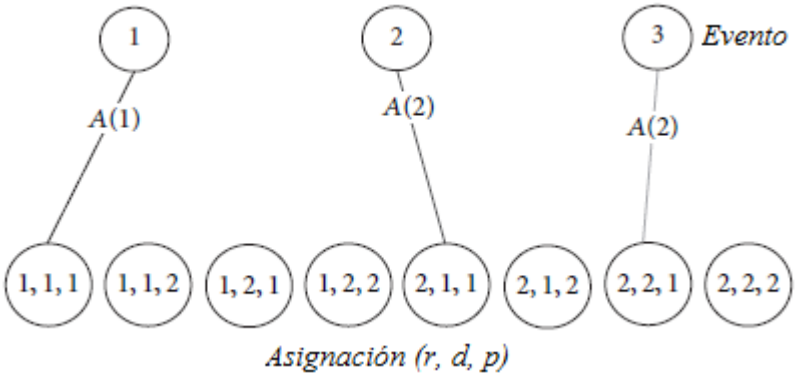


**Fig. 2. 5** Una solución al problema de Programación de Cursos Universitarios presentado en la Figura 2.4

La Figura 2.5 presenta una solución al problema presentado en la Figura 2.4. Por ejemplo: el evento $e_1$ es asignado al día $d_1$ durante el periodo $p_1$, en el salón $r_1$. Naturalmente, para asignar eventos a los salones en el caso de la instancia de la FCQeI, deben tomarse en cuenta las restricciones presentadas en el modelo de satisfacción de restricciones presentado en la sección 2.4

## 2.3 Instancia del problema

Los valores de los parámetros de la instancia del problema de la FCQeI, se definen de acuerdo a una asignación de horarios realizada por la administración en la FCQeI y aplicada como solución en un semestre académico, luego de un análisis se pudo observar que la solución aplicada en ese semestre no era factible dado la cantidad de traslapes encontrados en los horarios de los estudiantes que atendieron ese semestre.

La instancia considerada corresponde a la toma de materias del Semestre Agosto - Diciembre del año 2015 y los valores ahí contemplados se obtuvieron del estudio de campo realizado en la FCQeI con información proporcionada por la persona Responsable administrativa de la Programación de Horarios de la FCQeI. La definición de la instancia del Semestre Agosto - Diciembre del año 2015 de la FCQeI se presenta en la tabla 3.

| Parámetro | Valor |
|---|---|
| Eventos | 1514 |
| Salones | 41 (salones y laboratorios) |
| Estudiantes | 1426 |
| Días | 6 |
| Periodos | 15 |
| Ranuras de tiempo | 90 |
| Facilidades | 7 |
| Maestros Titulares | 156 |

**Tabla 2. 1** Parámetros de la instancia del semestre académico, Agosto-Diciembre de 2015 en la FCQeI

La instancia de la FCQeI tiene 1514 eventos, 41 salones, 7 características y 1426 estudiantes. Aunado a ello se cuenta con 156 maestros titulares y los eventos que ellos imparten se deben programar en primera instancia. Con lo anterior, se puede observar que en la instancia de la FCQeI se trabaja con un tamaño que sobrepasa la clase más grande de la clasificación del grupo Metaheuristic Network (ver sección 1.3.2.1), así mismo el número de restricciones duras del caso de estudio es mayor a la cantidad de restricciones duras contempladas en los benchmarks de [Rossi-Doria et al., 2003].

## 2.4 Modelo de satisfacción de restricciones

En el modelo de restricciones se consideran varios conjuntos de variables, tales como:

- El conjunto de eventos $E = \{1,2,3,...,n_E\}$,
- El conjunto de eventos que requieren equipo de laboratorio $E_l = \{1,2,3,...,n_{El}\}$,
- El conjunto de ranuras de tiempo $T = \{1,2,3,...,n_T\}$,
- El conjunto de días $D = \{1,2,3,...n_D\}$,
- El conjunto de periodos $P = \{1,2,3,...,n_P\}$,
- El conjunto de salones $R = \{1,2,3,...,n_R\}$,
- El conjunto de salones que cuentan con equipo de laboratorio $R_l = \{1,2,3,...,n_{Rl}\}$,
- La matrícula de estudiantes $S = \{1,2,3,...,n_S\}$,
- El conjunto de matrícula de maestros $M = \{M_T, M_N\}$, compuesto por dos subconjuntos, el de matrícula de maestros titulares $M_T$ y el de matrícula de maestros no titulares $M_N$,
- El conjunto de Clases $C = \{1,2,3,...n_C\}$,
- El subconjunto de eventos de cada clase $c_i = \{e_i, e_{j,...}\}$,
- Y finalmente, el conjunto de facilidades $F = \{1,2,3,...,n_F\}$.

El modelo de satisfacción de restricciones se muestra a continuación:

$$\sum_{t=1}^{n_T} \sum_{r=1}^{n_R} X(t, R(r,e)) = 1 \quad \forall e, \tag{1}$$

$$\sum_{t=1}^{n_T} \sum_{r=1}^{n_R} X(e, \Phi(t,r), R(r,e)) = 2 \quad \forall e, \tag{2}$$

$$\sum_{e=1}^{n_E} X(e, \Phi(t,r)) \leq 1 \quad \forall (t,r), \tag{3}$$

$$\sum_{s=1}^{n_S} Z(s, X(e, \Phi(t,r))) \leq \text{Capacity}(r) \quad \forall (e,t,r), \tag{4}$$

$$\sum_{t=1}^{n_T} \sum_{r=1}^{n_R} X(e_{tt}, \Phi(t,r), R(r,e_{tt})) = 2 \quad \forall e_{tt}, \tag{5}$$

$$\sum_{e=1}^{n_E} \sum_{r=1}^{n_R} Z(s, X(e, \Phi(t,r))) \leq 1 \quad \forall (s,t), \tag{6}$$

$$\sum_{e=1}^{n_E} \sum_{r=1}^{n_R} Z(m, X(e, \Phi(t,r))) \leq 1 \quad \forall (m,t), \tag{7}$$

$$\sum_{t=1}^{n_T} \sum_{r_l=1}^{n_{R_l}} X(e_l, \Phi(t,r_l), R_l(r_l,e_l)) = 2 \quad \forall e_l, \tag{8}$$

$$(t,r,e,s,m) \in N.$$

El modelo de satisfacción de restricciones está sujeto a ocho conjuntos de restricciones duras, las cuales tienen que ser satisfechas para que la solución de la instancia obtenida sea factible.

Una solución factible se encuentra cuando los eventos son asignados a ranuras de tiempo. Cada salón disponible tiene un conjunto de ranuras de tiempo, en este caso $n_T$ ranuras de tiempo. Una ranura de tiempo $t$ está definido en un espacio 2D como una función de coordenadas ($d$, $p$). Por lo tanto, $T = \{t \in N^2 \mid t = (d, p), t = 1, 2, 3,..., n_T\}$. El total de las asignación de los eventos toma lugar en un espacio 3D, como una función de Φ ($t$, $r$), donde $r \in R$. Entonces, para cada Φ ($t$, $r$)

$$\Phi(t,r) = \begin{cases} 1, & si\ es\ asignado\ un\ evento \\ 0, & de\ otra\ \ forma \end{cases} \qquad (9)$$

El salón $r \in R$ debe satisfacer las necesidades del evento $e \in E$ que se da en la ranura de tiempo $t \in T$.

El conjunto de restricciones en (1) implica que el salón $r \in R$ debe satisfacer las facilidades del evento $e \in E$ que ocurre en una ranura de tiempo $t \in T$.

Para cada salón $r \in R$ y cada evento $e \in E$, se define lo siguiente:

$$R(r,e) = \begin{cases} 1, & si\ el\ salón\ r\ satisface\ las\ necesidades\ del\ evento\ e \\ 0, & de\ otra\ \ forma \end{cases} \qquad (10)$$

Todos y cada uno de los eventos $e \in E$, tienen que ser programados, esto es, cada evento debe de estar programado en alguna ranura de tiempo $t \in T$ de algún salón.

El conjunto de restricciones en (2) implica que todos y cada uno de los eventos debe ser programado. Cada evento debe ser programado en una ranura de tiempo de un salón que tenga las facilidades adecuadas para el evento.

$$X\left(e,\Phi\left(t,r\right),R(r,e)\right)=\begin{cases}1, & si\ \Phi\left(t,r\right)+R\left(r,e\right)=2\\ 0, & de\ otra\ \ forma\end{cases} \qquad (11)$$

Un salón $r\in R$, puede admitir solamente un evento $e\in E$ por cada ranura de tiempo $t\in T$. Lo anterior implica que para una ranura de tiempo $t\in T$ de un salón $r\in R$, máximo puede estar programado un evento.

El conjunto de restricciones en (3) requiere que un salón admita solo un evento por ranura de tiempo. Esto implica que por cada ranura de tiempo $t \in T$ de un salón $r \in R$, puede ser programado un evento como máximo.

El salón $r \in R$ debe tener la capacidad suficiente para alojar el grupo de estudiantes $s \in S$ que participante en el evento $e \in E$. La capacidad de cada salón $r \in R$ es un dato de entrada.

Para saber cuándo un estudiante $s$ atiende un evento, se define la función presentada en (12) Esta función indica que estudiante $s$ atiende el evento que ocurre en la ranura de tiempo $t$ en el salón $r$. Las restricciones en (4) deben cumplirse; el salón debe tener la capacidad suficiente para que acuda el grupo de estudiantes que atienden el evento $e \in E$. La capacidad de cada salón $r$ es un dato de entrada.

$$Z(s,X\left(e,\Phi\left(t,r\right)\right))=\begin{cases}1, & si\ X\left(e,\Phi\left(t,r\right)\right)=1\ \text{entonces}\ s\ \text{sirve para}\ e\\ 0, & de\ otra\ \ forma\end{cases} \qquad (12)$$

Todos y cada uno de los eventos $e \in E$ impartidos por los maestros titulares $M_T$, se deberán programar en primera instancia, como una asignación parcial, en los periodos solicitados.

En el conjunto de restricciones en (5), los eventos de un profesor titular (tt) son programados en primera instancia. El salón asignado debe contar con las facilidades requeridas por el profesor. Esto da como resultado una programación parcial de eventos.

Un estudiante $s \in S$ no debe tener eventos traslapados. El conjunto de restricciones (6) evita los traslapes en los eventos a los que acude el estudiante $s \in S$. Como se indica en la función (12), si el estudiante atiende un evento $e$ en el salón $r$, entonces éste estudiante no debe atender otro evento $e$ durante el mismo periodo de tiempo $t$.

Por lo tanto, $Z(s, (e,\Phi(t, r)))=0$, $\forall e \in E$ con $e' \neq e$ La restricción (6) indica que el estudiante $s$ no debe tener eventos traslapados:

$$Z(m, X(e, \Phi(t,r))) = \begin{cases} 1, & si\ X(e, \Phi(t,r)) = 1 \text{ entonces } m \text{ atiende } e \\ 0, & de\ otra\ forma \end{cases} \qquad (13)$$

Un maestro $m \in M$ no debe tener eventos traslapados. El conjunto de restricciones en (7) prohíbe el traslape de eventos para todo maestro $m \in M$. La función (13) asegura que, si el maestro atiende el evento $e$ en el salón $r$, entonces el maestro no puede atender otro evento $e$ en el mismo periodo de tiempo $t$. Esto significa que $(m,(e,\Phi(t, r))) = 0$, $\forall e \in E$ con $e' \neq e$. La restricción (7) indica que el maestro $m$ no puede presentar traslape de eventos.

En el conjunto de restricciones (8), los eventos $e_l$ que corresponden a clases de laboratorio deben programarse en salones con equipo de laboratorio; el salón asignado debe cumplir con los requerimientos del maestro (10).

Los valores que pueden tomar las variables $t$, $r$, $e$, $s$, y $m$ pertenecen al conjunto de los números naturales.

El conjunto de maestros se divide en dos subconjuntos $M = M_T \cup M_N$. El conjunto de ranuras de tiempo también se divide en dos subconjuntos $T = T_T \cup T_N$. Cada una de las ranuras de tiempo *t* que forman parte de $T_T$ se asigna a los maestros titulares $M_T$ de materias a solicitud de ellos. Cada maestro titular solicita impartir en una ranura de tiempo *t* específica $M_T = \{m | \forall m \, \exists t \in T_T, t = Asignado(m)\}$, la asignación de *t* a *m* es un dato de entrada. Para el conjunto de ranuras de tiempo $T_N$ que será asignado a maestros no titulares $M_N$ de materias, no se conoce su asignación específica de cada *t* para cada maestro *m* no titular.

Dado lo anterior, el modelo de la FCQeI comienza con una solución parcial factible, debido a que la asignación de $T_T$ es conocido y es elegido por el conjunto de maestros titulares $M_T$ en una calendarización de horarios de materias para un semestre escolar. Lo que resta por asignar a los maestros titulares $M_T$, es la elección del salón *r* donde se impartirá el evento.

Se comienza con una solución parcial factible, con valores de $T_T$ conocidos que son elegidos por el conjunto de maestros titulares $M_T$ en una calendarización de horarios de materias para un semestre escolar. Esta calendarización parcial debe ser factible y autorizada por la dirección de la FCQeI. Los salones donde se imparten estas materias no se conocen.

Consideraciones de la instancia a resolver del problema de la FCQeI:

- Las necesidades de cada materia se establecieron de forma aleatoria, ya que no hay información de primera mano.

- Las necesidades que puede ofrecer cada salón se establecieron de acuerdo a las conexiones de energía eléctrica que se tiene en cada uno de estos para la conexión de equipo.
- Se considera que la facultad cuenta con el equipo suficiente para atender las necesidades de cada materia siempre y cuando el salón tenga los contactos adecuados de energía, a excepción del uso de televisores, los cuales se definieron en número de acuerdo a la existencia real.
- El equipo para atender necesidades no está fijo y es movible al salón solicitado siempre y cuando éste cuente con conexiones de energía eléctrica.
- Una materia está formada por un subconjunto de eventos. El número de eventos que forma una materia depende de la materia misma, puede ir de 4 eventos hasta 6 eventos.
- El modelo que se presenta, es un modelo relajado que no toma en cuenta varias restricciones de la FCQeI. Es un primer intento para representar el problema de asignación de horarios en esta Facultad.

  Se podrían agregar las siguientes restricciones:
  - La restricción de que cuando se imparta más de un evento de la misma materia en el mismo día, estos eventos se impartan en el mismo salón, por comodidad de los maestros y estudiantes.
  - La restricción de que, si se imparte más de un evento de la misma materia en el mismo día, estos eventos se impartan de manera seguida, por comodidad de los maestros y estudiantes.
- La restricción de seriación de materias se considera de manera implícita y no se toman en este modelo. Se hace la suposición que esta restricción se hace cumplir por los jefes de carrera en la toma de materias realizada por los estudiantes.

El objetivo es buscar una solución a una instancia del modelo de satisfacción de restricciones de la Facultad de Ciencias Químicas e Ingeniería (FCQeI). Una instancia del problema se tiene en cada semestre del año escolar. La instancia se genera en base a la oferta de materias, toma de materias de cada alumno y el número de matrícula de alumnos y de profesores.

# CAPÍTULO 3. Metodología de solución del problema de la FCQeI

## 3.1. Metodología para tratar la información de horarios de cursos universitarios de la FCQeI

Para la ejecución de la instancia del problema de la FCQeI, por medio del algoritmo constructivo que obtiene soluciones factibles, los datos de entrada o instancia del problema, se deben encontrar almacenados en archivos de texto con el formato adecuado para que puedan ser leídos.

El formato utilizado para representar la instancia del semestre de la FCQeI es el mismo que se aplicó a los benchmarks presentados en [Rossi-Doria et al., 2003].

### 3.1.1 Formato de la instancia del problema

Dado que el número de datos de la instancia de la FCQeI es muy grande, la instancia se tiene que almacenar en más de un archivo.

Los datos de la instancia del problema están organizados en 6 archivos con el formato que se muestra a continuación.

**Archivo de texto 1. Definición del problema**

- **Primera línea:** El número de eventos, número de salones, número de características, número de estudiantes. (todos los números son enteros).
- **Una línea para cada salón:** Cada línea indicará el aforo de un salón.

La tabla 3.1 presenta un ejemplo para la instancia de la FCQeI.
En este ejemplo, se observa que la primera línea presenta al número de eventos, número de salones, número de características y número de estudiantes, en ese orden. A partir de la línea 2 hasta la 42, se presenta el aforo para cada uno de los 41 salones de la FCQeI. Lo único que aparece en el archivo de texto 1, es la codificación.

| Línea | Aforo del salón | Número de eventos | Número de salones | Número de características | Número de estudiantes |
|---|---|---|---|---|---|
| | Codificación | | | | |
| 1 | --- | 1514 | 41 | 7 | 1426 |
| 2 | 10 | --- | --- | --- | --- |
| 3 | 20 | --- | --- | --- | --- |
| 4 | 50 | --- | --- | --- | --- |
| … | … | --- | --- | --- | --- |
| 42 | 60 | --- | --- | --- | --- |

**Tabla 3. 1** Codificación para la definición del problema de la FCQeI

**Archivo de texto 2. Asignación de eventos por estudiante**

- **Una línea para cada estudiante/evento:** La codificación se hace con un cero o uno. Cero indica que el estudiante no atiende el evento, uno indica que el estudiante atiende el evento. La tabla 3.2 presenta un ejemplo para 3 estudiantes y 4 eventos programados. Lo único que aparece en el archivo de texto 2, es la codificación.
  En este ejemplo, se observa que el primer estudiante atiende el evento número dos, el segundo estudiante atiende los eventos uno y dos, finalmente el tercer estudiante atiende los eventos número tres y cuatro.

| Línea | Estudiante | Codificación | Número de evento |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 2 |
| 3 | 1 | 0 | 3 |
| 4 | 1 | 0 | 4 |
| 5 | 2 | 1 | 1 |
| 6 | 2 | 1 | 2 |
| 7 | 2 | 0 | 3 |
| 8 | 2 | 0 | 4 |
| 9 | 3 | 0 | 1 |
| 10 | 3 | 0 | 2 |
| 11 | 3 | 1 | 3 |
| 12 | 3 | 1 | 4 |

**Tabla 3. 2** Codificación para la asignación de eventos por cada estudiante

### Archivo de texto 3. Asignación de eventos por maestro

- **Una línea para cada maestro/evento:** Con un cero o uno. Cero indica que el maestro no imparte el evento; uno indica que el maestro imparte el evento. La tabla 3.3 presenta un ejemplo para 3 maestros y 4 eventos programados. Lo único que aparece en el archivo de texto 3, es la codificación.

| Línea | Maestro | Codificación | Número de evento |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 2 |
| 3 | 1 | 0 | 3 |
| 4 | 1 | 0 | 4 |
| 5 | 2 | 1 | 1 |
| 6 | 2 | 1 | 2 |
| 7 | 2 | 0 | 3 |
| 8 | 2 | 0 | 4 |
| 9 | 3 | 0 | 1 |
| 10 | 3 | 0 | 2 |
| 11 | 3 | 1 | 3 |
| 12 | 3 | 1 | 4 |

**Tabla 3. 3** Codificación para la asignación de eventos por cada maestro

En este ejemplo, se observa que el primer maestro atiende el evento número dos, el segundo maestro atiende los eventos uno y dos, finalmente el tercer maestro atiende los eventos número tres y cuatro.

**Archivo de texto 4. Horario maestros titulares**

- **Una línea para cada horario/evento:** Con cuatro columnas que contienen números enteros.

  La tabla 3.4, presenta la codificación para la asignación de eventos a cada maestro titular.

  La primera columna, indica el identificador del maestro titular que solicita impartir su materia en un horario.

  La segunda y tercera columna indican el día (Lunes = 1,…, Viernes = 5) y periodo (periodo = 1,…, periodo = 15) en que un maestro titular solicita impartir el evento, los números diferentes de cero indican que el evento tiene un horario solicitado por un maestro titular, los ceros indican que el evento no tiene un horario solicitado para ser impartido por el maestro titular.

  La cuarta columna indica el identificador de la materia a la que pertenece el evento.

  La tabla 3.4, presenta un ejemplo para ocho eventos programados para el maestro titular con ID = 1. Lo único que aparece en el archivo de texto 4, es la codificación.

  El maestro, solicita impartir dos materias, una con ID = 1 y la otra con ID = 2, cada materia tiene cuatro eventos. El maestro solicita impartir dos eventos de la materia ID = 1, el día martes en el primero y segundo periodo.

  También solicita impartir los otros dos eventos de la materia ID = 1, el día jueves en el primero y segundo periodo. Para la materia ID = 2, el maestro no requiere de un horario en especial y se ajusta al horario que le puedan asignar.

| ID Maestro | Día | Periodo | ID Materia |
|---|---|---|---|
| | Codificación | | |
| 1 | 2 | 1 | 1 |
| | 2 | 2 | 1 |
| | 4 | 1 | 1 |
| | 4 | 2 | 1 |
| | 0 | 0 | 2 |
| | 0 | 0 | 2 |
| | 0 | 0 | 2 |
| | 0 | 0 | 2 |

**Tabla 3. 4** Codificación para la asignación de eventos a cada maestro titular

### Archivo de texto 5. Características por salón

- **Una línea para cada salón/característica:** La tabla 3.5, presenta la codificación para la asignación de características por cada salón de clase.

  La primera columna, es el número de línea en el archivo.

  La segunda columna indica el ID del salón de clase.

  La tercera columna indica las características que pudiera satisfacer un salón de clase. Un cero si el salón no satisface la característica, o un uno si el salón si satisface la característica.

  Para el ejemplo que se ilustra en la tabla 3.5, encontramos en la columna dos la información de tres salones (ID = 1, 2, 3), en la columna tres se muestra que pueden existir cuatro características que el salón pudiera cumplir; en el caso del salón ID = 1, solo cumple la característica número 2; en el caso del salón ID = 2, cumple las características 1 y 2; en el caso del salón ID = 3, solo cumple las características 3 y 4. Lo único que aparece en el archivo de texto 5, es la codificación.

| Línea | ID Salón | Codificación |
|---|---|---|
| | | Característica |
| 1 | 1 | 0 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |
| 4 | 1 | 0 |
| 5 | 2 | 1 |
| 6 | 2 | 1 |
| 7 | 2 | 0 |
| 8 | 2 | 0 |
| 9 | 3 | 0 |
| 10 | 3 | 0 |
| 11 | 3 | 1 |
| 12 | 3 | 1 |

**Tabla 3. 5** Codificación para la asignación de características por cada salón de clase

### Archivo de texto 6. Necesidades por evento

- **Una línea para cada evento/necesidad:** La tabla 3.6, presenta la codificación para la asignación de necesidades por cada clase en función de los eventos que componen la clase.

  La primera columna, es el número de línea en el archivo.

  La segunda columna indica el ID de la materia de clase.

  La tercera columna indica las necesidades requeridas por los eventos de la ID clase que deberá aportar el salón de clases dada las características de este salón. Un cero si el evento no requiere de la necesidad, o un uno si el evento si requiere la necesidad.

  Por ejemplo, la tabla 3.6, en la columna dos presenta tres salones (ID = 1, 2, 3), en la columna tres se muestra que pueden existir cuatro necesidades que la clase pudiera requerir; en el caso de los eventos que conforman la clase ID = 1, solo requieren la necesidad número 2; en el caso de los eventos que conforman la clase ID = 2, solo requieren las necesidades 1 y 2; en el caso de los eventos que conforman la clase ID = 3, solo requieren las necesidades 3 y 4. Lo único que aparece en el archivo de texto 6, es la codificación.

| Línea | ID Clase | Codificación |
|---|---|---|
| | | Necesidad |
| 1 | 1 | 0 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |
| 4 | 1 | 0 |
| 5 | 2 | 1 |
| 6 | 2 | 1 |
| 7 | 2 | 0 |
| 8 | 2 | 0 |
| 9 | 3 | 0 |
| 10 | 3 | 0 |
| 11 | 3 | 1 |
| 12 | 3 | 1 |

**Tabla 3. 6** Codificación para la asignación de necesidades por cada evento de clase

## 3.2. Desarrollo de una Heurística Constructiva

Las Heurísticas Constructivas son procedimientos que se encargan de obtener una solución a partir de un criterio inicial, esto es, construyen una solución factible. [Glover et al., 2001].

En esta sección del trabajo de investigación se presenta un Algoritmo Constructivo (AC) adaptado para ser aplicado a las restricciones propias de la instancia del Problema de Programación de Cursos Universitarios de la Facultad de Ciencias Químicas e Ingeniería de la Universidad Autónoma del Estado de Morelos que permite generar soluciones factibles de Programación de Cursos Universitarios satisfaciendo las restricciones duras que se observan en el modelo de satisfacción de restricciones planteado en la sección 2.4.

En la Figura 5 se muestra el pseudocódigo del Algoritmo Constructivo desarrollado en el presente trabajo de investigación. El detalle de los pasos que ejecuta el Algoritmo Constructivo se encuentra enseguida en esta misma sección.

```
(1)  Leer la instancia de entrada
(2)  Clasificar eventos (tipo, características)
(3)  for i = 0 a NumSoluciones do
(4)       Inicializar en 0 y vacíos los arreglos
(5)       while eventosTitularesNoProgramados do
(6)                 Asignar eventosTitulares en ranuras vacías que cumplan restricciones duras(RD)
(7)                 Asignar eventos no programados(eventosNP) en arreglos temporales
(8)                 if (eventosNP)
(9)                   Asignar eventosNP en ranuras vacías elegidas al azar que cumplan RD *
(10)      endwhile
(11)      while j < etapasMax or todosEventosProgramados do
(12)                Ordenar eventosNoTitulares(aforo)
(13)                Asignar eventosNoTitulares en ranuras vacías elegidas al azar que cumplan RD
(14)                Asignar eventosNoTitulares no programados(eventosNPNT) en arreglos
                    temporales
(15)                while j < intentosMax do
(16)                    for k = 0 a eventosNPNT do
(17)                        Asignar eventosNPNT en ranuras vacías sin validar RD
(18)                    endfor
(19)                    for m=0 a eventos programados en ranuras vacías sin validar RD do
(20)                        while s < intentosBMax do
(21)                        Seleccionar eventos programados en ranuras vacías sin validar RD
(22)                        Seleccionar al azar de solucionTemporal ranuras ocupadas o desocupadas
(23)                        Intercambiar los eventos si cumplen RD o Insertar eventos en ranuras
                            vacías que cumplen RD
(24)                        endwhile
(25)                    endfor
(26)                    Reprogramar eventos
(27)                endwhile
(28)      endwhile
(29)      if solucionTemporal factible
(30)                Imprime Solución Factible y tiempo en segundos
(31)                guardaSolucion
(32)      else      Imprime Solución Infactible y tiempo en segundos
(33)      endfor
```

**Fig. 3. 1** Pseudocódigo Algoritmo Constructivo

A continuación, se describen los pasos del Algoritmo Constructivo

*Línea (1)*. El algoritmo inicia con la instrucción de lectura de la instancia de entrada. La información de la instancia se obtiene de 6 archivos de texto. La codificación de los 6 archivos de texto ha sido explicada en la sección 3.2.1.

*Línea (2)*. Antes de la programación del horario de los eventos, éstos son clasificados por tipo (tomando en cuenta los maestros titulares y las materias que imparten) y facilidades (considerando los requerimientos de los eventos a programar).

*Línea (3)*. El ciclo desde $i = 0$ *hasta NumSoluciones* verifica la condición para cada ejecución para procesar el número de soluciones (NumSoluciones) que son generadas.

*Línea (4)*. Con la instrucción *inicializar arreglos en cero y vaciar estructura,* son inicializadas las estructuras que son utilizadas para manipular la información durante el proceso de creación de la solución. Se hace uso de arreglos tridimensionales para guardar las soluciones. Estos arreglos son necesarios dada la organización de la información que se guarda. (Ver modelo de grafos en la Figura 5) Tal información puede ser usada para obtener la programación de horarios de los salones.

*Líneas (5) a (10)*. La programación de eventos inicia con la búsqueda del cumplimiento de la restricción dura número (5) del modelo (ver sección 2.4) La programación de los eventos que impartirán los maestros titulares (línea (6)) es ejecutada. Se hace el intento de programar los eventos en el día y periodo que los maestros titulares solicitaron para impartir las materias que les corresponden.

Se hace una asignación aleatoria de un salón y los eventos son programados en el día y periodo requerido solo si las restricciones duras son satisfechas. De otra forma (línea (9)), se hace un número máximo de intentos para programar los eventos en ranuras de tiempo elegidas de forma aleatoria para satisfacer las restricciones duras. Una vez que todos los eventos de los maestros titulares han sido programados, el resultado es una calendarización parcial factible.

*Líneas (11)* a *(27)*. Se continúa con la programación del resto de los eventos que han sido clasificados como pertenecientes a maestros no titulares. Se hace un número máximo de intentos para programar tales eventos o bien, hasta que se encuentra una solución factible, que es el criterio de parada del algoritmo. Se ejecutan las actividades de las líneas (12) a (25) para encontrar una solución.

*Línea (12)*. Los eventos son ordenados tomando en cuenta la capacidad (número de estudiantes que atienen el evento) Estos eventos son ordenados en orden decreciente de acuerdo a la capacidad que requieren. Se realiza este ordenamiento, de modo que los eventos que requieren salones con mayor aforo sean programados en primera instancia. Los eventos que requieren menor capacidad para ser impartidos se programan al final.

*Línea (13).* Los eventos a ser programados se asignan de forma aleatoria a ranuras de tiempo desocupadas que satisfacen las restricciones duras de dicho evento. Esto se hace a través de una selección aleatoria de salones y ranuras de tiempo (día, periodo y salón), como puede verse en la Figura 6, se hace el intento de programar tantos eventos de maestros no titulares como es posible, siempre y cuando se satisfagan las restricciones duras.

*Línea (14)*. Los eventos que no pudieron ser asignados en un número máximo de intentos so almacenados en el arreglo temporal denominado NPNT.

*Líneas (16) y (17)*. Los eventos almacenados en el arreglo temporal NPNT son programados en ranuras de tiempo sin verificar si con ello se cumple o no con las restricciones duras. La solución se vuelve infactible.

*Líneas (19)* a *(25)*. Para intentar recobrar la factibilidad de la solución, se lleva a cabo un paso de reubicación de eventos. Este paso de reubicación de eventos consiste en ejecutar repetidos intentos de reprogramación de eventos que no cumplen con las restricciones duras.

*Líneas (22) y (23)* Se seleccionan de forma aleatoria ranuras de tiempo y se analiza la factibilidad de hacer un intercambio. Si las ranuras de tiempo están siendo ocupadas con eventos que ya cumplen con las restricciones duras, el análisis que se hace consiste en determinar si se siguen satisfaciendo las restricciones duras luego del intercambio de eventos. Si las ranuras de tiempo seleccionadas para la reubicación están vacías, el análisis permitirá determinar si los eventos pueden ser movidos a las ranuras de tiempo satisfaciendo sus restricciones duras.

La reubicación de eventos tiene lugar solo para los eventos de los maestros no titulares.

*Línea (26).* Cada que termina el ciclo para programar los eventos almacenados en el arreglo temporal NPNT, se incluye un paso de reprogramación de eventos, que consiste en intercambios de eventos programados en la solución siempre que continúen satisfaciendo las restricciones duras, asegurándose de que los eventos de maestros titulares no sean reprogramados.

*Líneas (29) a (32)*. Una vez que finaliza el ciclo de programación de los eventos de maestros no titulares, el algoritmo imprime si la solución encontrada es factible o infactible, además de ello se imprime el tiempo que empleó el algoritmo en obtener tal solución.

Para validar la satisfacción de restricciones duras RD del modelo (ver Sección 2.4) en el Algoritmo Constructivo, es necesario observar cada estudiante (RD = 7) y maestro (RD = 8) para asegurarse de que las restricciones fueron satisfechas. Esto significa que no debe haber traslapes. Para asegurar que no existen traslapes para estudiantes y maestros o cualquier otro incumplimiento de restricciones duras, se ejecuta una búsqueda iterativa de estas ocurrencias en cada ranura de tiempo en todos los salones. Este es el proceso que consume la mayor parte del tiempo de ejecución del algoritmo. Para efectuar ésta búsqueda, se aplica una combinación de dos algoritmos: búsqueda binaria y búsqueda BTREE, haciendo más eficiente la búsqueda requerida para la Programación de los Cursos Universitarios. [Cruz-Chávez, 2014].

# CAPÍTULO 4 Pruebas experimentales y análisis de resultados

## 4.1 Entorno de ejecución

Las Pruebas Experimentales del Algoritmo Constructivo fueron ejecutadas en una computadora personal con Procesador Intel Core i7, cuya velocidad de CPU era 2.2GHz, 8GB RAM, Sistema Operativo Windows 7 Home Premium, Microsoft Visual Studio 2008 Ver. 9.0.21022.8. Para cada ensayo experimental, se realizaron 30 ejecuciones del Algoritmo Constructivo para generar soluciones factibles.

El análisis de eficiencia del Algoritmo para generar soluciones factibles se muestra en la Figura 4.1. Se muestran los tiempos que tomó el Algoritmo para obtener cada una de las 12 soluciones factibles en la ejecución de las 30 pruebas.

El tiempo más corto que se requirió para encontrar una solución factible fue 3 horas con 33 minutos y el mayor tiempo fueron 6 horas con 22 minutos. Se obtuvieron soluciones factibles en el 40% de las ejecuciones. En los casos en que no se obtuvo una solución factible, el tiempo que llevó realizar el total de número de iteraciones antes de llegar al criterio de parada para el algoritmo varió entre 4 y 8 horas.

El camino que sigue el Algoritmo Constructivo siempre es diferente, dado que es un algoritmo no-determinístico. Por lo tanto, el tiempo que lleva

realizar las operaciones para encontrar las soluciones parciales serán siempre diferentes, principalmente debido a la búsqueda iterativa que verifica que no haya traslapes para estudiantes y profesores.

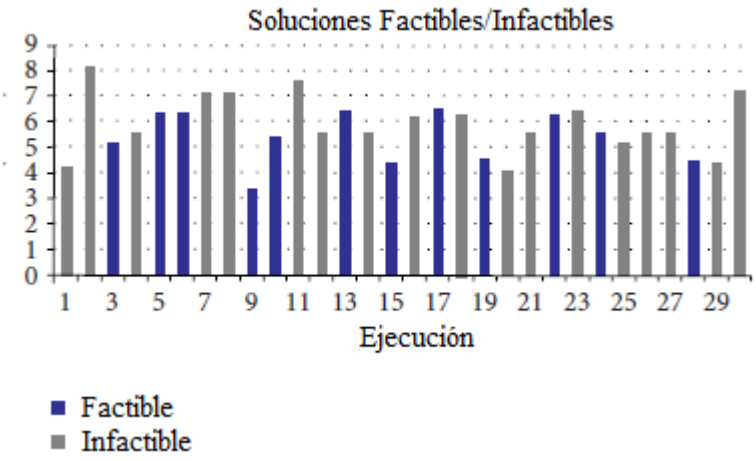


**Fig. 4. 1** Tiempo de generación para 30 ejecuciones del Algoritmo Constructivo

## 4.2 Análisis de resultados

### 4.2.1. Frecuencia de asignación de eventos en los salones

La Figura 4.1 presenta el promedio de ocupación por periodo de los 41 salones que se consideraron en la instancia de la FCQeI. La gráfica muestra que ningún salón es ocupado en los 90 periodos disponibles que tiene cada uno. El promedio más alto de ocupación fue la del salón 2 con 65 periodos ocupados de los 90 disponibles, lo que representa el 77.8%. El salón 2 tiene capacidad para 70 estudiantes. Se observa que los salones con mayor aforo tienen el promedio más alto de ocupación.

Las barras azules que se observan en la Figura 4.2 permiten identificar los laboratorios que se tuvieron disponibles para la instancia considerada. Estos salones que corresponden a laboratorios tuvieron siempre el mismo

número de periodos ocupados, mismos que estuvieron en el rango de 14 a 42. Solamente las asignaturas que requieren equipo de laboratorio pueden ser programadas en este tipo de salones; por las características propias de los laboratorios, no pueden ser empleadas para programar ahí clases teóricas dado que carecen de la infraestructura necesaria para tal actividad como butacas para alumnos, proyector, etc.

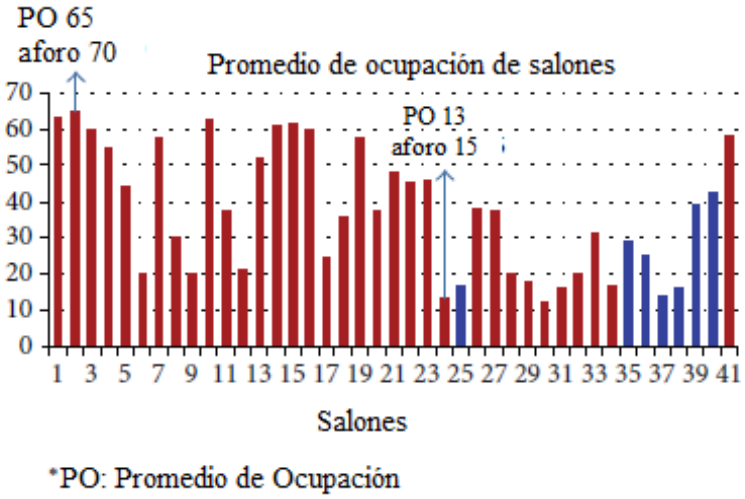


**Fig. 4. 2** Promedio de ocupación de periodos por salón

La existencia de siete laboratorios y varios salones con poca capacidad disminuye el número de periodos disponibles para calendarizar eventos. Los siete laboratorios tienen 630 periodos disponibles (7 * 90) pero solo 203 periodos fueron ocupados porque solo pueden ser ocupados cuando la asignatura requiere de algún laboratorio.

Hay 18 salones con poca capacidad para albergar a los estudiantes, esta capacidad va de 15 a 30 estudiantes (ver Figura 7). Para los salones con poca capacidad, se ocuparon en promedio entre 13 y 38 periodos, dejando desocupados entre 52 y 77 periodos. Por ejemplo, en la Figura 4 se puede observar que el salón 24, con una capacidad de 15, tiene solo 13 periodos de

los 90 disponibles. Los 18 salones con poca capacidad son poco factibles de utilizar para la programación de eventos debido a que la mayoría de los eventos requieren de salones con mayor capacidad.

La Figura 4.3 presenta un diagrama de Pareto para la implementación del Algoritmo Constructivo para el análisis de las 12 soluciones factibles que se obtuvieron en 30 ejecuciones del algoritmo. El diagrama de Pareto muestra la frecuencia del promedio de programación de eventos en los 9 salones con la mayor capacidad, los cuales tienen una capacidad de 70 estudiantes. Se observa que, en promedio, para cada salón con capacidad de 70, el Algoritmo Constructivo programa aproximadamente 58 eventos por semana en dos ejecuciones, 59 eventos por semana en dos ejecuciones, 60 eventos por semana en dos ejecuciones, 63 eventos por semana en dos ejecuciones y 61 eventos por semana en dos ejecuciones. Cada evento programado un salón con la mayor capacidad no puede exceder del máximo de la capacidad que es 70 estudiantes. Dado que el número máximo de eventos que pueden ser programados en cada salón de clases es 90 por semana (ver Tabla 1), para los nueve salones con capacidad de 70 hay en promedio 33% de periodos desocupados por semana.

La Figura 4.4 presenta el diagrama de Pareto para la implementación del Algoritmo Constructivo para las 12 soluciones factibles obtenidas en 30 ejecuciones. El diagrama de Pareto muestra el promedio de ocupación de eventos programados en los 4 salones con la menor capacidad, estos salones tienen una capacidad máxima de 15 estudiantes. Se observa que, en promedio, para cada salón con la menor capacidad, el Algoritmo Constructivo programó menos de 15 eventos por semana en una ejecución, menos de 16 eventos por semana en tres ejecuciones, menos de 17 eventos por semana en tres ejecuciones, menos de 18 eventos por semana en tres ejecuciones y menos de 20 eventos por semana en dos ejecuciones. Cada evento programado en los salones de menor capacidad no debe exceder el máximo de la capacidad que es de 15 estudiantes. Dado que el número máximo de

eventos que pueden ser programados en un salón son 90 por semana, se puede concluir que, en promedio, los cuatro salones con la menor capacidad, que cuenta con lugares para 15 personas, tiene cerca del 78.6% de periodos desocupados por semana.

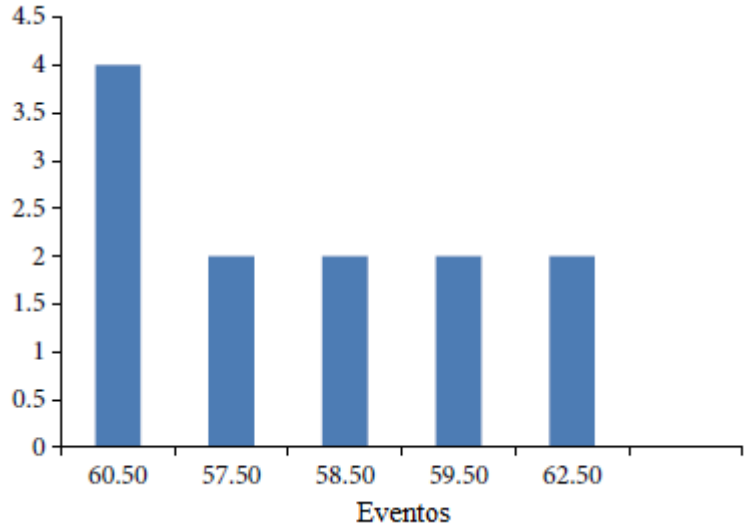


**Fig. 4. 3** Promedio de eventos semanales asignados a salones con aforo para 70 estudiantes (en 12 soluciones factibles)

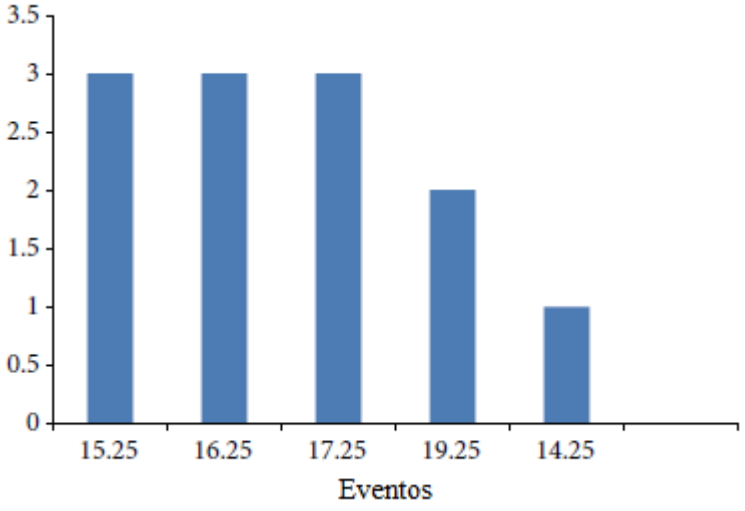


**Fig. 4. 4** Promedio de eventos semanales asignados a salones con aforo para 15 estudiantes (en 12 soluciones factibles)

### 4.2.2. Comparación de resultados

Los resultados obtenidos por el Algoritmo Constructivo fueron comparados con la programación realizada por la Administración de la FCQeI para el semestre de la instancia considerada: Agosto – Diciembre 2015.

La solución generada por la Administración de la FCQeI tiene las siguientes características: de los 656 eventos que debían ser impartidos por profesores titulares, 3 eventos no pudieron ser asignados a periodos que había solicitado el profesor titular de la materia. Los eventos fueron programados en otros periodos, por lo tanto, el conjunto de restricciones 5 no fue satisfecho.

De los 858 eventos que correspondían a profesores no titulares, hubo 28 eventos que fueron asignados a periodos que ya estaban ocupados por otro evento, de modo que el conjunto de restricciones en 3 no fue satisfecho.

Derivado del análisis que se hizo de la solución obtenida por la FCQeI, se pudo observar que la solución obtenida por la administración de la FCQeI no era factible. Así mismo, se pudo observar que, de los 1514 eventos programados en esta instancia, hubo 459 eventos que presentaron traslapes (ver Figura 4.5) de modo que el conjunto de restricciones en 7 no se satisfizo. El resto de restricciones si fueron satisfechas.

También se observó que si el traslape era parte de clases de 4 eventos, había entonces 114 clases con traslapes, por otro lado, si el evento que presentaba traslape era parte de una clase de 6 eventos esto provocaba la existencia de 76 clases con traslapes. El número de eventos con traslapes en la solución propuesta por la Administración de la FCQeI va de 76 a 114 eventos con traslapes para el semestre de Agosto – Diciembre 2015. La solución obtenida por la FCQeI tenía clases con traslapes, por tal motivo, para los estudiantes no era posible acudir a todas sus clases.

La solución obtenida por el Algoritmo Constructivo presenta los siguientes resultados: había 656 eventos impartidos por profesores titulares y estos eventos, en su totalidad, pudieron ser asignados a los periodos

requeridos por los profesores, satisfaciendo con ello el conjunto de restricciones en 5. Respecto a los 858 eventos que impartieron profesores no titulares, todos los eventos fueron asignados a periodos desocupados, satisfaciendo con ello el conjunto de restricciones en 3. El resto de restricciones de modelo propuesto fueron satisfechas.

La solución obtenida por el Algoritmo Constructivo para la FCQeI es factible. Se observó que no existen traslapes de eventos para estudiantes.

Las Figuras 4.5 y 4.6 muestra la ocupación de salones y el comportamiento de los traslapes, permitiendo una comparación entre las soluciones generadas por la Administración de la FCQeI y el Algoritmo Constructivo. Para la solución generada por la Administración de la FCQeI (Figura 4.5), un total de 459 eventos tuvieron traslapes para estudiantes.

Cuando un periodo (ranura) en un salón es ocupado, se entiende que el periodo está ocupado por un evento. De este modo, para el salón 1, hay 67 eventos (ranuras de tiempo ocupadas) por semana, de un total de 90 ranuras disponibles de ese salón. De esos 67 eventos, 39 presentaban traslapes. La capacidad del salón 1 es de 70 estudiantes.

En la solución propuesta por la Administración de la FCQeI hay solo 13 salones donde no se presentaron traslapes (26,27, 28,29, 31, 32, 33, 36, 37, 38, 39, 40, y 41). También se puede observar que hay cinco salones en los cuales no hubo eventos asignados (28, 32, 33, 39 y 40).

Se muestran a continuación las tablas 4.1, 4.2 y 4.3 que permiten observar tanto la ocupación por salones como el comportamiento de los traslapes para la solución generada por la FCQeI:

| Salón | Periodos ocupados | Eventos con Traslapes | % traslapes |
|---|---|---|---|
| 1 | 67 | 39 | 58% |
| 2 | 64 | 32 | 50% |
| 3 | 66 | 13 | 19% |
| 4 | 56 | 30 | 53% |
| 5 | 45 | 25 | 55% |
| 6 | 51 | 27 | 52% |
| 7 | 61 | 16 | 26% |
| 8 | 46 | 21 | 45% |
| 9 | 51 | 21 | 41% |
| 10 | 62 | 19 | 30% |
| 11 | 60 | 32 | 53% |
| 12 | 43 | 13 | 30% |
| 13 | 64 | 19 | 29% |
| 14 | 60 | 16 | 26% |
| 15 | 63 | 16 | 25% |
| 16 | 63 | 16 | 25% |
| 17 | 43 | 10 | 23% |
| 18 | 61 | 24 | 39% |
| 19 | 48 | 10 | 20% |
| 20 | 28 | 6 | 21% |

**Tabla 4. 1** Ocupación por salones y porcentaje de traslapes (Salones 1 al 20)

| Salón | Periodos ocupados | Eventos con Traslapes | % traslapes |
|---|---|---|---|
| 21 | 53 | 16 | 30% |
| 22 | 38 | 10 | 26% |
| 23 | 47 | 7 | 14% |
| 24 | 38 | 8 | 21% |
| 25 | 32 | 4 | 12% |
| 26 | 12 | 0 | 0% |
| 27 | 4 | 0 | 0% |
| 28 | 0 | | |
| 29 | 2 | 0 | 0% |
| 30 | 30 | 3 | 10% |
| 31 | 17 | 0 | 0% |
| 32 | 0 | | |
| 33 | 0 | | |
| 34 | 9 | 3 | 33% |
| 35 | 37 | 3 | 8% |
| 36 | 8 | 0 | 0% |
| 37 | 16 | 0 | 0% |
| 38 | 8 | 0 | 0% |
| 39 | 0 | | |
| 40 | 0 | | |
| 41 | 30 | 0 | 0% |

**Tabla 4. 2** Ocupación por salones y porcentaje de traslapes (Salones 21 al 41)

| Salón | Periodos ocupados | Eventos con Traslapes | % traslapes |
|---|---|---|---|
| 1 | 67 | 39 | 58% |
| 5 | 45 | 25 | 55% |
| 4 | 56 | 30 | 53% |
| 11 | 60 | 32 | 53% |
| 6 | 51 | 27 | 52% |
| 2 | 64 | 32 | 50% |
| 8 | 46 | 21 | 45% |
| 9 | 51 | 21 | 41% |
| 18 | 61 | 24 | 39% |
| 34 | 9 | 3 | 33% |
| 10 | 62 | 19 | 30% |
| 12 | 43 | 13 | 30% |
| 21 | 53 | 16 | 30% |
| 13 | 64 | 19 | 29% |
| 7 | 61 | 16 | 26% |
| 14 | 60 | 16 | 26% |
| 22 | 38 | 10 | 26% |
| 15 | 63 | 16 | 25% |
| 16 | 63 | 16 | 25% |
| 17 | 43 | 10 | 23% |

**Tabla 4. 3** Salones con mayor porcentaje de traslapes

La solución generada por el Algoritmo Constructivo no tiene eventos con traslapes (Figura 4.6) y se puede observar que todos los salones fueron ocupados. Ninguno de los salones alcanzó el límite máximo de ocupación (90 ranuras). Los salones con la menor capacidad tuvieron el menor número de eventos programados. Por ejemplo: los salones 24, 25, 30 y 31 tienen 13, 17, 13 y 16 eventos programados respectivamente. Cada uno de estos salones tiene una capacidad de 15 estudiantes.

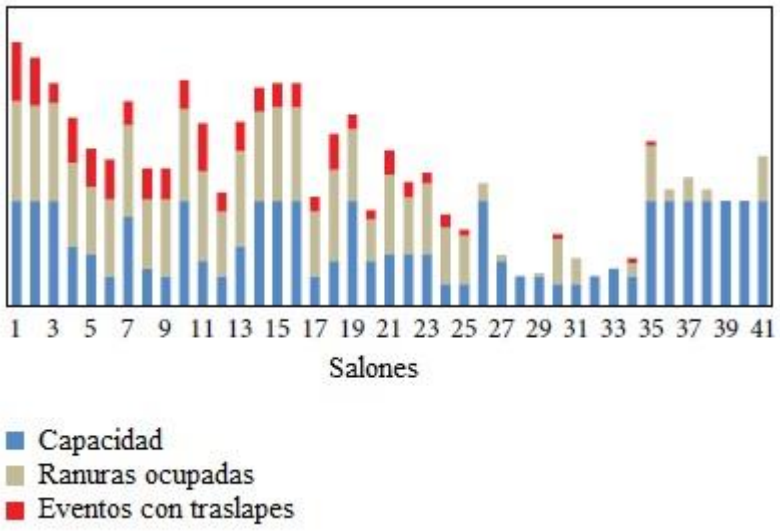


**Fig. 4. 5** Solución obtenida para el UCTP de la FCQeI. Administración de la FCQeI. Semestre Agosto-Diciembre 2015

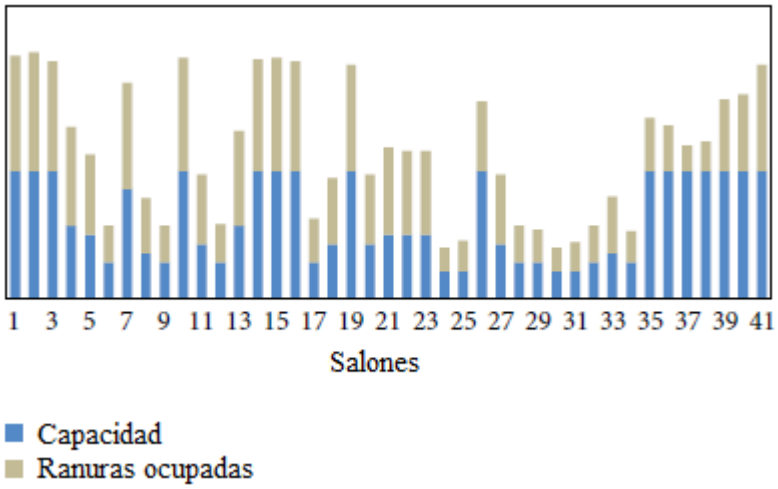


**Fig. 4. 6** Solución obtenida para el UCTP de la FCQeI. Algoritmo Constructivo. Semestre Agosto-Diciembre 2015

# CAPÍTULO 5 Conclusiones y trabajo futuro

## 5.1 Conclusiones

El modelo matemático de satisfacción de restricciones propuesto en el presente proyecto considera considera el desempeño de las restricciones duras identificadas en el caso real de programación de cursos universitarios en la FCQeI.

Fue muy complicado obtener soluciones que cumplieran con todas las restricciones propuestas en el modelo matemático. Solo en el 40% de las 30 pruebas ejecutadas se obtuvieron soluciones factibles. Las soluciones fueron encontradas en un promedio de 7 horas, lo cual muestra la complejidad de las restricciones duras del modelo propuesto. Lo anterior es comparado con las soluciones infactibles obtenidas por la administración de la FCQeI después de alrededor de dos semanas de trabajo.

El modelo matemático propuesto es el que ha sido implementado manualmente en el FCQeI durante 20 años. Dado el incremento de la matrícula de estudiantes y requerimientos adicionales de la Facultad, el número de horarios traslapados para las materias se incrementa. Esto ha dificultado aún más a la administración de la FCQeI la tarea de encontrar buenas soluciones. Adicionalmente, el número de salones construidos en años recientes no es proporcional al incremento de la demanda.

El modelo matemático propuesto es una oportunidad de programar horarios eficientes de cursos universitarios sin traslapes en el FCQeI. El modelo puede ser resuelto con alguna técnica heurística. Para el caso del presente proyecto, se utilizó un algoritmo de enfoque constructivo, lo cual es relativamente sencillo de analizar.

En el análisis de los resultados, se observa que los salones con menor capacidad son subutilizados. Una propuesta para mejorar la asignación de estas aulas es la formación de varios grupos pequeños de estudiantes en clases, en lugar de grandes grupos como actualmente sucede.

La formación de pequeños grupos generar una mayor cantidad de materias (eventos) para programar, lo que a su vez requeriría una mayor inscripción de profesores que imparten una mayor cantidad de asignaturas. Eso facilitaría la asignación de aulas pequeñas lo que conlleva a aumentar el uso de los salones.

La matrícula estudiantil podría aumentar en la FCQeI y aún sería posible obtener soluciones factibles sin invertir en la construcción de salones adicionales para los nuevos grupos si son pequeños.

Otra propuesta es utilizar otras técnicas heurísticas para resolver el modelo propuesto con mayor eficacia. Esto podría facilitar el encontrar soluciones apropiadas de acuerdo a la capacidad de los salones de clase. Sería necesario realizar un cambio en el modelo matemático propuesto para que las nuevas restricciones sean incluidas. El cambio podría permitir por ejemplo, que la heurística ocupara primero los salones con menor capacidad y luego los de mayor capacidad.

## 5.2 Trabajo Futuro

Para mejorar las soluciones factibles obtenidas se podría implementar una metaheurística que permita explorar y explotar el espacio de soluciones con el objetivo de optimizar las soluciones ya obtenidas. Así mismo, se podría considerar la evaluación de adaptación de cada solución para el proceso de selección de los individuos por medio de restricciones suaves que se pueden observar para el caso de estudio de la FCQeI.

Se podría considerar abordar el problema de calendarización de cursos universitarios de la FCQeI con un Algoritmo Genético ya que en la literatura se ha visto que da buenos resultados tratar el UCTP con esta heurística. [Flores-Pichardo, 2011] Incluso, se ha encontrado que la combinación de Algoritmos Genéticos con métodos de búsqueda local obtienen buenos resultados al aplicarse a instancias grandes como se puede encontrar en [Abdelhalim, et.al., 2016] y [Kralev, et.al., 2019], esta sería una buena opción para abordar el problema de la programación de cursos universitarios de la FCQeI por las características propias del problema descritas en el apartado 2.3. Por otro lado, se podría hacer uso de alguna búsqueda en vecindarios variables, métodos con los que se ha experimentado para su uso en optimización abordando problemas NP-completos, como es el caso del UCTP. [Cruz-Chavez, et.al., 2014].

Los algoritmos genéticos son métodos adaptativos utilizados para resolver problemas de optimización. Estos algoritmos están basados en el proceso genético de los organismos biológicos y simulan el principio de selección natural, siendo posible con ellos evolucionar la forma de solucionar problemas del mundo real. Por ejemplo, pueden ser usados para diseñar estructuras de puentes o para determinar la forma en que se disminuye el desperdicio en el corte de patrones de ropa [Beasly, 1993].

Dado que en el Centro de Investigación en Ingenierías y Ciencias Aplicadas (CIICAp) se incursiona en un proyecto para estudiar modelos teóricos de tipo NP-completos en la GRID Morelense, el algoritmo genético para la Programación de Cursos Universitarios se podría implementar en ese ambiente Grid.

El ambiente Grid permite contar con equipo de cómputo con gran poder de procesamiento y es un ambiente ideal para ejecutar algoritmos que resuelvan problemas de optimización de tipo NP-completos que consideren instancias grandes.

El Cuerpo Académico de Optimización y Software adscrito al CIICAp Centro de Investigaciones en Ingeniería y Ciencias Aplicadas de la UAEM en colaboración con el Cuerpo Académico Cómputo Intensivo Aplicado a la Ingeniería, del Instituto Tecnológico de Veracruz, construyo la MiniGRID Tarántula, para trabajar en sus líneas de investigación, la cual cuenta con 13 nodos (24 procesadores), 21 GB RAM y 1740 GB HD. Actualmente esta MiniGRID está formada por dos clusters de alto rendimiento, uno localizado en Cuernavaca, Morelos (UAEM), el cluster CIICAp y otro localizado en Veracruz, Veracruz (ITVer), el cluster Nopal.Los dos Clusters de alto rendimiento se unen a través de una Red Privada Virtual red-a-red utilizando OpenVPN en lugar de usar routers (vía Hardware). Los Clusters son de diferentes subredes, el Cluster del CIICAp como subred A y el Instituto Tecnológico de Veracruz con una subred B.

## Referencias


Abdelhalim Esraa A. y El Khayat Ghada A., (2016) *A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA)*, Alexandria Engineering Journal, Volume 55, Issue 2, 2016,Pages 1395-1409, ISSN 1110-0168, https://doi.org/10.1016/j.aej.2016.02.017.

Aho A. V., Hopcroft J. E. y Ullman J. D. (1974) *The Design and Analysis of Computer Algorithms.* Addison-Wesley, ISBN 0-201-00023-7

Andre A. y Dinata H, (2018) *Interaction Design to Enhance UX of University Timetable Plotting System on Mobile Version.* Interaction, IOP Conf. Series: Materials Science and Engineering 407(1), (2018) 012174 doi:10.1088/1757-899X/407/1/012174, IOP Publishing, 2018.

Arindam Chaudhuri, Kajal De (2010) *Fuzzy genetic heuristic for university course timetable problem* Int. J. Advance. Soft Comput. Applications Volumen 2 Número 1 Páginas 100-123

Assi Maram, Halawi Bahia, Haraty Ramzi A. (2018) *Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem.* Procedia Computer Science, Volume 126, 2018, Pages 899-906, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.08.024.

Banczyk K., Boinski T., y Krawczyk H. (2006) *Parallelization of genetic algorithms for solving university timetabling problems.* in Proceedings of the International Symposium on IEEE Xplore Conference: Parallel Computing in Electrical Engineering, pp. 325–330

Ben-Ayed O. y Hamzaoui S., (2012) *Multiobjective multiproduct parcel distribution timetabling: a real-world application.* International Transactions in Operational Research, Vol. 19, No. 4, pp. 613–629, 2012.

Borchani Rahma, Elloumi Abdelkerim, Masmoudi Malek. (2017) *Variable neighborhood descent search based algorithms for course timetabling problem: Application to a Tunisian University.* Electronic Notes in Discrete Mathematics, Volume 58, 2017, Pages 119-126, ISSN 1571-0653, https://doi.org/10.1016/j.endm.2017.03.016.

Burke E., Kingston J., Jackson K., Weare R. (1997) *Automated University Timetabling: The State of the Art.* The Computer Journal 40 (9) 565-571

Causmaecker Patrick De, Demeester Peter, Berghe Greet Vanden. (2009) *A decomposed metaheuristic approach for a real-world university timetabling problem.* European Journal of Operational Research, Volume 195, Issue 1, 2009, Pages 307-318, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2008.01.043.

Chacha S. y Allen R. M., (2013) *Optimal solution strategy for university course timetabling problem.* International Journal of Advanced Research in Computer Science, Vol. 4, No. 1, p. 35, 2013.

Christos Valouxis, Efthymios Housos, (2003) *Constraint programming approach for school timetabling.*Computers & Operations Research, Volume 30, Issue 10, Pages 1555-1572, ISSN 0305-0548, https://doi.org/10.1016/S0305-0548(02)00083-7, 2003

Colorni, A., Dorigo, M. y Maniezzo, V. (1998) *Computational Optimization and Applications* (1998) 9: 275. https://doi.org/10.1023/A:1018354324992

Cook S.A., (1971) *The complexity of theorem-proving procedures.* in Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71), pp. 151–158, Machinery, 1971. Association for Computing

Cruz Chávez, M., Moreno Bernal, P. y Peralta Abarca, J. (2016). *Aplicación de la teoría de la complejidad en optimización combinatoria.* Inventio, la génesis de la cultura universitaria en Morelos, 10(20), 35-42.

Cruz-Chávez M. A. y Martínez-Oropeza A, (2013) *B-Tree algorithm complexity analysis to evaluate the feasibility of its application in the university course timetabling problem.* Journal of Artificial Intelligence and Soft Computing Research, Vol. 3, No. 4, pp. 251–263, 2013

Cruz-Chávez M.A., Flores-Pichardo Mireya, Martínez-Oropeza Alina, Moreno-Bernal Pedro y Cruz-Rosales Martín H., (2016) *Solving a Real Constraint Satisfaction Model for the University Course Timetabling Problem: A Case Study*. Mathematical Problems in Engineering, Vol. 2016, Article ID 7194864, 14 pages, 2016. https://doi.org/10.1155/2016/7194864, 2016

Cruz-Chávez Marco Antonio, Martínez-Oropeza Alina, Martínez-Rangel Martín, Moreno-Bernal Pedro, Pecina Federico Alonso, Juárez-Chavez Jazmín Yanel, Flores-Pichardo Mireya. (2014). *Experimental Analysis with Variable Neighborhood Search for Discrete Optimization Problems.* Enclyclopedia of Information Science and Technology. Third Edition. Information Resources Management Association. ISBN. 978-1-4666-5888-2, e-ISBN. 978-1-4666-5889-9. USA.

Cruz-Rosales M. H., (2010) El problema de programación de cursos en una Universidad y una propuesta de solución [Tesis], CIICAp-UAEM, 2010

Di Stefano C. y Tettamanzi A. G. B., (2001) An evolutionary algorithm for solving the school time-tabling problem in Applications of Evolutionary Computing, Vol. 2037 of Lecture Notes in Computer Science, pp. 452–462, Springer, New York, NY, USA, 2001.

Díaz, A., Glover, F., Ghaziri, H.M., Gonzalez, J.L., Laguna, M, Moscato, P. y Tseng, F.T. (1996) *Optimización Heur ística y Redes Neuronales* Editorial Paraninfo, Madrid.

Even S., Itai A., Shamir A. (1976) *On the complexity of timetable and multicommodity flow problems.* SIAM Journal on Computing, Vol. 5, No. 4, pp. 691–703

Fethi Rabhi and Guy Lapalme, (1999). *Algorithms; a Functional Programming Approach* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN:0201596040

Flores-Pichardo, Mireya, (2011) *Revisión de Algoritmos Genéticos Aplicados al Problema de la Programación de Cursos Universitarios.* Programación Matemática y Software (2011) Vol. 3. No 1. ISSN: 2007-3283

Garey M. G. y Johnson D. S. (1990) *Computers and Intractability: A Guide to theTheory of NP-Completeness.*W. H. Freeman&Co., New York, NY, USA, 1990.

Glover,F., Gutin,G., Yeo,A., Zverovich,A.(2001) *Construction Heuristics for the asymmetric TSP.* European Journal of Operational Research 129 III. 2001. 555- 568.

Gunawan A., Ming Ng K. y Leng Poh K., (2012) *A hybridized Lagrangian relaxation and simulated annealing method for the course timetabling problem* Computers & Operations Research, Vol. 39, No. 12, pp. 3074–3088, 2012.

Hamed Babaei, Jaber Karimpour, Amin Hadidi (2015) *A survey of approaches for university course timetabling problem.*Computers & Industrial Engineering, Volume 86, 2015, Pages 43-59, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2014.11.010.

Kahar M. N. M. y Kendall G., (2010) *The examination timetabling problem at Universiti Malaysia Pahang: comparison of a constructive heuristic with an existing software solution.* European Journal of Operational Research, Vol. 207, No. 2, pp. 557–565, 2010

Kaviani, Mohadese & Shirouyehzad, Hadi & Sajadi, Seyed & Salehi, Mohammadreza. (2014). *A heuristic algorithm for the university course timetabling problems by considering measure index: A case study.* Int. J. of Services and Operations Management. 18. 1 - 20. 10.1504/IJSOM.2014.060448.

Kırıs S., (2014) *AHP and multichoice goal programming integration for course planning* International Transactions in Operational Research, Vol. 21, No. 5, pp. 819–833, 2014.

Kralev, Velin, Kraleva, Radoslava y Agnihotri, Sachin. (2019). *A Modified Event Grouping Based Algorithm for The University Course Timetabling Problem.* International Journal on Advanced Science, Engineering and Information Technology. 9. 229-235. 10.18517/ijaseit.9.1.6488.

Landir Saviniec, Maristela O. Santos, Alysson M. Costa. (2018) *Parallel local search algorithms for high school timetabling problems.*European Journal of Operational Research, Volume 265, Issue 1, 2018, Pages 81-98, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2017.07.029.

Lindahl Michael, Mason Andrew J., Stidsen Thomas y Sørensen Matias. (2018) *A strategic view of University timetabling.* European Journal of Operational Research, Volume 266, Issue 1, 2018, Pages 35-45, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2017.09.022.

Martí R., Reinelt G. (2011) *Heuristic Methods. In: The Linear Ordering Problem.* Applied Mathematical Sciences, vol 175. Springer, Berlin, Heidelberg

Martínez-Bahena B. y Calderón-Segura Y.Y, (2008) Desarrollo de un sistema administrador de horarios para la FCQeI, usando un modelo de tres capas, cliente, servicio web y base de datos. [Tesis de Ingeniería], Upemor, Jiutepec, Mexico, 2008

Martínez-Oropeza Alina, Cruz-Chávez Marco Antonio, Moreno-Bernal Pedro, (2012) *Distancia Hamming: Método y Desarrollo de un Algoritmo Computacional para el Problema de Ruteo Vehicular con Ventanas de Tiempo.* 9No. Congreso Internacional de Cómputo en Optimización y Software. México, 2012

Martínez-Oropeza Alina, Cruz-Chávez Marco Antonio. (2011) Método de Agrupamiento no Supervisado para el Problema de Ruteo Vehicular con Restricciones de Capacidad en Vehículos. Memorias del 8vo. Congreso Internacional de Cómputo en Optimización y Software. México, 2011

McCollum Barry, Schaerf Andrea, Paechter Ben, McMullan Paul, Lewis Rhyd, Parkes Andrew J, et.al. (2010) *Setting the research agenda in automated timetabling: the second international timetabling competition.*INFORMSJComput2010;22 (1):120–30.

Méndez-Díaz Isabel, Zabala Paula, Miranda-Bront Juan José. (2016) *An ILP based heuristic for a generalization of the post-enrollment course timetabling problem.* Computers & Operations Research, Volume 76, 2016, Pages 195-207, ISSN 0305-0548, https://doi.org/10.1016/j.cor.2016.06.018.

N. Basir N.,Ismail W. y Norwawi N., (2013) *A simulated annealing for Tahmidi course timetabling.* Procedia Technology, Vol. 11, pp. 437–445, 2013.

Papadimitriou C. H. y Steiglitz K., (1998) *Combinatorial Optimization: Algorithms and Complexity.* Dover, New York, NY, USA, 1998

Pereira V. y Gomes Costa H, (2016) *Linear integer model for the course timetabling problem of a faculty in Rio de Janeiro.* Advances in Operations Research, Vol. 2016,Article ID 7597062, 9 pages, 2016.

Peyman Yasari, Mohammad Ranjbar, Negin Jamili, Mohammad-Hesam Shaelaie. (2019) *A two-stage stochastic programming approach for a multi-objective course timetabling problem with courses cancelation risk.*Computers & Industrial Engineering, Volume 130, 2019, Pages 650-660, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2019.02.050.

Phillips Antony E., Waterer Hamish, Ehrgott Matthias, Ryan David M. (2015) *Integer programming methods for large-scale practical classroom assignment problems.* Computers & Operations Research, Volume 53, 2015, Pages 42-53, ISSN 0305-0548

Phillips, A.E., Walker, C.G., Ehrgott, M. et al. (2017) *Integer programming for minimal perturbation problems in university course timetabling.* Annals of Operations Research (2017) 252: 283. https://doi.org/10.1007/s10479-015-2094-z

Pillay N., Qu R. Examination Timetabling Problems. In: Hyper-Heuristics: Theory and Applications. Natural Computing Series. Springer, Cham, 2018

Ribeiro C. C., (2012) *Sports scheduling: problems and applications.* International Transactions in Operational Research, Vol. 19, No. 1-2, pp. 201–226, 2012.

Rossi-Doria O., Sampels M., Birattari M., Chiarandini M., Dorigo M., y Gambardella L. M., (2003) A comparison of the performance of different metaheuristics on the timetabling problem in Practice andTheory of Automated Timetabling IV, Vol. 2740, pp.A13 329–351, Springer, Berlin, Germany, 2003

Saldaña-Crovo A., Oliva-SanMartín C., y Pradenas-Rojas L. (2007) *Integer programming models for scheduling problem on universities.*Chilen Engineering Magazine, Vol. 15, No. 3, pp. 245–259 (Spanish).

Schaerf A. (1999). *A survey of automated timetabling.* Artificial Intelligence Review, 13(2):87–127, 1999.

Shen Y., Xu J. y Zeng Z., (2015) *Public transit planning and scheduling based on AVL data in China*, International Transactions in Operational Research, 2015.

Shimazaki S., SakakibaraK K. y Matsumoto T., (2015) *Iterative optimization techniques using man-machine interaction for university timetabling problems.*SpringerPlus, Vol. 4, No. 1, article 251, 2015.

Siam A.,El Habib S.M. y Safir S. (2019) *A multi-agent system for generation of university time schedules.*Conference: 8th International Conference on Advanced Computer Science and Information Technology (ICAIT 2019), At: March 30-31, 2019, Zurich, Switzerland, DOI: 10.5121/csit.2019.90403

Silva, John Jairo, (2017) *Heurística para la Planificación de Horarios de la Universidad EAFIT* (2017) Tesis Departamento de Ciencias Matemáticas

Sipser Michael. (2013). *Introduction to the Theory of Computation*. Second Edition. Thomson Course Technology. ISBN. 0-534-95097-3.

Solís, J.F., Alonso-Pecina F., y Mora-Vargas, J. (2008). *An Efficient Simulated Annealing Algorithm for Feasible Solutions of Course Timetabling*. MICAI.

Taha Arbaoui, Jean-Paul Boufflet, Aziz Moukrim (2019) *Lower bounds and compact mathematical formulations for spacing soft constraints for university examination timetabling problems.* Computers & Operations Research, Volume 106, 2019, Pages 133-142, ISSN 0305-0548, https://doi.org/10.1016/j.cor.2019.02.013

Talbi, El-Ghazali. (2009). *Metaheuristics, From Design To Implementation.* University of Lille – CNRS – INRIA ISBN 978-0-470-27858-1. Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Thepphakorn T., Pongcharoen P., y C. Hicks C., (2015) Modifying regeneration mutation and hybridising clonal selection for evolutionary algorithms based timetabling tool, Mathematical Problems in Engineering, Vol. 2015, Article ID 841748, 16 pages, 2015

Song Ting, Liu Sanya, Tang Xiangyang, Xicheng Peng, Mao Chen. (2018) *An iterated local search algorithm for the University Course Timetabling Problem.* Applied Soft Computing, Volume 68, 2018, Pages 597-608, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2018.04.034.

Vassilios I.Skoullis, Ioannis X.Tassopoulos, Grigorios N.Beligiannis. (2017) *Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm.*Applied Soft Computing Journal, p.p. 277–289 http://dx.doi.org/10.1016/j.asoc.2016.10.038

Yu E. y and Sung K.-S., (2002) *A genetic algorithm for a university weekly courses timetabling problem.* International Transactions in Operational Research, Vol. 9, No. 6, pp. 703–717, 2002.

Zhang D., Guo S., Zhang W., y Yan S., (2014) *A novel greedy heuristic algorithm for university course timetabling problem.* in Proceedings of the 11th World Congress on Intelligent Control and Automation (WCICA ’14), pp. 5303–5308, IEEE, Shenyang, China, July 2014.