INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Educational Timetabling resuelto con Recocido Simulado y modelado bajo una arquitectura Web. Generalización y aplicación práctica: casos PATAT y UADY.

Tesis de Maestría

Autor:
Cinhtia Maribel González Segura

Sometido al Programa de Graduados en Informática y Computación en cumplimiento parcial con los requerimientos para obtener el grado de:

Maestra en Ciencias de la Computación

Asesores:
Dr. Juan Frausto Solís
Dra. Mónica Larre Bolaños Cacho

Cuernavaca, Morelos. **Noviembre, 2005**

Resumen

En el presente documento se aborda uno de los principales problemas de la optimización combinatoria: la asignación de horarios y salones en instituciones educativas, conocido como *Educational Timetabling* (ETT).

Se presenta la solución al problema empleando el algoritmo Recocido Simulado (RS) así como un nuevo algoritmo que surge de sintonizar los parámetros del RS y que se ha denominado Recocido Simulado Sintonizado (RSS). Ambos algoritmos son implementados y se realiza una comparación en cuanto a la calidad de las soluciones encontradas y al tiempo que toma obtenerlas.

Los resultados obtenidos se comparan con los publicados por el PATAT (*Practice and Theory of Automated Timetabling*), una organización internacional formada por un grupo de investigadores reconocidos. Posteriormente, se implementa el algoritmo RSS para resolver el problema real de asignación de horarios y cargas académicas para una institución universitaria: la Universidad Autónoma de Yucatán (UADY). Se realiza una comparación con los resultados obtenidos entre el algoritmo RSS implementado y el algoritmo genético con diversidad forzada (AGDF), con el que se resolvió el problema de la Universidad Juárez Autónoma de Tabasco (UJAT).

Se presentan los resultados obtenidos tanto para el caso teórico (PATAT) como el práctico (UADY), así como un análisis comparativo de los algoritmos RS y RSS implementados.

Dedicatoria

A mis padres y hermanos.

Ese quinteto de ángeles
que sin importar tiempos ni distancias,
han sabido estar presentes en todo momento,
apoyándome y confiando en mi.

Agradecimientos

A Dios.
Por permitirme estar aquí, por darme la fortaleza y las condiciones necesarias para que pudiera alcanzar esta meta. Por cada angelito que enviaste para que me ayudaran a lograrla. Infinitamente, gracias.

A mis padres.
Ni todas las palabras alcanzarían para expresar lo importante que ha sido su apoyo en este y cada proyecto que he emprendido en mi vida. ¡Qué afortunada soy por ser su hija!

A mis hermanos.
Gracias por ser la representación más fiel de la palabra hermandad, gracias por compartir conmigo alegrías y tristezas, éxitos y tropiezos, y sobre todo, por su gran paciencia.

A mi novio.
Por el tiempo que hemos compartido, por estar ahí en los momentos gratos y difíciles, animándome y apoyándome en todos los sentidos. Esta tesis no es sólo mía, es de los dos.

A mis amigos.
Gracias a ustedes éste proyecto fue menos pesado de lo que pudo haber sido, porque la distancia física nunca ha sido un obstáculo y aunque la tecnología no fuera tan sofisticada, el corazón encontraría sus propios medios.

A mis compañeros de la maestría.
Por todos los momentos que compartimos, por enseñarme y ayudarme más de lo que cada uno se imagina. Especialmente a Federico Alonso Pecina y José Luis Gómez Ramos, su ayuda ha sido muy importante para terminar este trabajo.

A la Universidad Autónoma de Yucatán.
Especialmente al dr. Luis Rodríguez Carvajal y a todos mis compañeros de la Unidad Tizimín. Gracias por su apoyo y su confianza.

Al Dr. Juan Frausto Solís y la Dra. Mónica Larre Bolaños Cacho.
Por su apoyo, orientación y gran paciencia para poder concluir este trabajo.

Al Dr. Jaime Mora Vargas y el Dr. Fernando Ramos Quintana
Por sus valiosas aportaciones brindadas para el enriquecimiento de esta tesis.

Al Tecnológico de Monterrey, campus Cuernavaca.
Especialmente a todas aquellas personas que contribuyeron a la realización de este proyecto

A todos ustedes, ¡Gracias!

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1. Introducción

El problema general de Timetabling[1] (*Timetabling Problem*, TTP) consiste en calendarizar diversos eventos respetando un conjunto de restricciones de varios tipos y puede aplicarse a múltiples áreas, en tareas como: calendarización de proyectos, roles deportivos, horarios en hospitales, horarios escolares y de transporte.

El problema de asignación de horarios en una institución educativa, también conocido como *Educational Timetabling*[2] (ETT) [Schaerf, 2001] [Burke, 2002] [Marte, 2002] es el tema central de esta tesis y se presenta en cada inicio de un período escolar en prácticamente todas las instituciones de educación media, media-superior y superior. Cuando esta tarea se realiza manualmente resulta tediosa y consume gran cantidad de tiempo; además, a menudo se opta por ignorar algunas restricciones cuyo cumplimiento no es indispensable para que el horario sea factible. Es por eso que se han creado diversos métodos de solución, con el fin de automatizar el proceso y conseguir que se respeten, hasta donde sea posible, las restricciones que se presentan en cada institución.

En ésta tesis se implementa uno de los métodos existentes para solucionar el problema: la metaheurística Recocido Simulado (RS) [Kirkpatrick, 1983] [Kostuch, 2003] [Burke, 2003a]. Se implementan dos algoritmos de tipo Recocido Simulado, el RS simple y el RS con sintonización analítica de parámetros, al que aquí se ha denominado Recocido Simulado Sintonizado (RSS); ambos se ponen en práctica demostrando su eficacia para las

[1] Se utiliza éste término debido a que es común utilizar "Timetabling" para referirse al problema de Asignación de Horarios y una traducción podría desviar el significado.

[2] En este trabajo se emplea el término "Educational Timetabling" para referirse al problema de asignación de horarios y salones en instituciones educativas, conservando la terminología en inglés para no desviar el significado.

instancias generales del PATAT y para algunos casos de prueba con datos reales de una institución de educación superior, la Universidad Autónoma de Yucatán (UADY). Además, se realiza el diseño de una arquitectura y una base de datos general, que proporciona las bases para resolver el problema que se presenta en tres instituciones educativas de nivel superior: el Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM) campus Cuernavaca, la Universidad Juárez Autónoma de Tabasco (UJAT) y la UADY. El caso el ITESM está siendo resuelto por Federico Alonso Pecina [Alonso, 2005] y el caso de la UJAT fue resuelto por José Luis Gómez Ramos [Gómez, 2005].

En el presente capítulo se introduce al tema de Timetabling, presentando los antecedentes, la motivación, el objetivo, la justificación y la hipótesis de la tesis, así como las contribuciones del trabajo y la organización general del documento.

1.1 Antecedentes

El TTP consiste en asignar un conjunto de eventos tales como cursos, conferencias y talleres, a ciertos espacios de tiempo definidos en un período, satisfaciendo un conjunto de restricciones de diversos tipos [Burke, 2002]. Generalmente, el período considerado es de una semana de cinco o seis días hábiles debido a que, por lo general, en las instituciones educativas se elaboran los horarios considerando una semana, de tal forma que la calendarización realizada se pueda repetir durante el tiempo que dure el ciclo escolar. Los espacios de tiempo considerados para cada evento pueden ser de 30 minutos, una hora o 90 minutos, entre otros.

Debido a que cada institución cuenta con sus propias políticas y logística para asignar los horarios escolares, existe una gran variedad de versiones del problema, en cada una de las cuales se consideran ciertas restricciones que son propias del problema particular que se desea resolver. En el TTP se consideran dos tipos de restricciones [Burke, 1997] [Schaerf, 1999] [Melicio, 1999]: las restricciones duras (aquellas que resulta indispensable satisfacer para que el horario resulte factible[3]), y las restricciones suaves, aquellas que son deseables de cumplir, más no indispensables.

Como puede intuirse, encontrar una solución que satisfaga todas las restricciones del problema puede resultar una tarea realmente difícil de resolver, esto debido a la cantidad de combinaciones posibles (factibles e infactibles) para realizar la asignación del horario. Por lo tanto, el problema de asignación de horarios se ha abordado como un problema de optimización [Scherf, 1999] [Burke, 2002], buscando satisfacer todas las restricciones duras (indispensables) del problema y maximizar el número de restricciones suaves (deseables) satisfechas. Una forma de resolverlo [Melicio, 1999] [Rossi-Doria, 2002] es bajo el enfoque de un Problema de Programación Lineal (PPL), asociándole un costo a cada restricción, siendo las duras más costosas que las suaves, de tal forma que se persigue minimizar el costo total de la solución encontrada.

[3] Un horario factible es aquél que no presenta inconsistencias en cuanto a tiempos y espacios físicos.

1.2 Motivación

En las instituciones de educación superior existe la necesidad de asignar y coordinar, eficaz y eficientemente los recursos económicos, materiales y humanos. Uno de los problemas que se presenta en cada inicio de un período escolar es la necesidad de organizar y distribuir los horarios de clases teniendo en cuenta la presencia de algunas restricciones que complican su asignación. Dichos factores están relacionados con la disponibilidad de los recursos materiales y humanos, así como los diversos planes de estudio existentes para los estudiantes.

El proceso típico de asignación de horarios es realizado empleando métodos manuales y empíricos que requieren invertir un tiempo considerable, por lo que es deseable contar con un método automatizado que realice este proceso. El TTP es un tema de investigación ampliamente abordado en la literatura reciente [Gómez, 2005] [Burke, 2003] [Marte, 2002] [Socha, 2002].

A pesar de la gran variedad de enfoques existentes para la resolución del TTP, existen grupos dedicados a mejorar los métodos y algoritmos actuales, uno de estos grupos es el PATAT (Practice and Theory of Automated Timetabling) [PATAT, 2005], el cual se describe con detalle en el Capítulo 4, sección 4.1.

En el mercado existen sistemas comerciales (ver Capítulo 2, sección 2.5) que resuelven el problema de asignación de horarios, sin embargo, tales sistemas no se encuentran al alcance de las universidades de tamaño mediano (pues su precio está alrededor de los $500.00 USD anuales por licencia), hacia las cuales se enfoca el presente trabajo. Además, en muchos casos estos sistemas no cumplen con las necesidades particulares de cada institución.

1.3 Objetivo

El objetivo principal consiste en resolver el problema ETT encontrando una asignación que satisfaga la mayor cantidad de restricciones posibles mediante un algoritmo de tipo Recocido Simulado, e implementar el prototipo de un sistema de interfaz con cualidades mínimas para una utilización agradable por parte del usuario.

Los objetivos específicos son los siguientes:

- Analizar metaheurísticas empleadas en la literatura para solucionar instancias del problema ETT,
- Seleccionar la metaheurística que se considere mejor,
- Implementar el algoritmo correspondiente a la metaheurística seleccionada,
- Probar el desempeño del algoritmo con instancias tomadas del PATAT, con el fin de asegurar la calidad de la solución obtenida,
- Definir el modelo para el problema real que se desea resolver, con base en el conjunto de restricciones que lo define, y
- Construir un sistema prototipo que permita introducir los datos del problema y ejecutar el algoritmo implementado.

1.4 Justificación e Hipótesis

Existe una gran variedad de trabajos previos que han abordado el problema de Timetabling desde diversos enfoques [Abramson, 1992] [Burke, 1994] [Burke, 2002] [Carter, 1998] [Schaerf, 1999] [Kingston, 2001]. Sin embargo, cada uno de ellos presenta una solución al problema particular que plantea, de forma tal que no existe un sistema estándar general para resolver este problema. Lo anterior se debe principalmente a que las políticas y restricciones de horarios varían de una institución a otra. Muchos de los sistemas construidos son semi-automáticos, es decir, requieren de la intervención del usuario para validar la bondad de la solución proporcionada por el sistema o para participar en la construcción del horario.

El problema de Timetabling ha sido clasificado como NP-duro en casi todas sus variantes, lo cual se demuestra en [Cooper, 1995]. Es decir que, no existe un método determinista que lo resuelva en tiempo polinomial. Por lo tanto, ha sido abordado utilizando algoritmos aproximados llamados métodos heurísticos.

La hipótesis del presente trabajo es la siguiente: "Es posible mejorar la solución para el problema de ETT en instituciones educativas de tamaño mediano, utilizando el algoritmo de Recocido Simulado con sintonización de parámetros".

Por instituciones educativas de tamaño mediano se entiende de 50 a 100 grupos, 1000 a 3000 estudiantes, 100 a 200 profesores). Una de las contribuciones a la investigación consiste en probar que con la sintonización de parámetros mejora el desempeño del algoritmo recocido simulado en la resolución del problema ETT.

1.5 Contribuciones

Con la implementación del método desarrollado en este trabajo, se pretende disminuir la carga administrativa de las personas encargadas de la programación de horarios, además de que permite obtener un horario que satisfaga un número de restricciones mayor de las que se logran satisfacer empleando el método tradicional de asignación manual.

Para la resolución del ETT, en este trabajo se utiliza el algoritmo Recocido Simulado clásico [Kirkpatrick, 1983] [Cerny, 1984] con características especiales de sintonización de parámetros [Sanvicente, 2004], cuya eficiencia se compara posteriormente con el algoritmo Recocido Simulado clásico. El algoritmo desarrollado en esta tesis ha sido denominado Recocido Simulado Sintonizado (RSS) y proporciona soluciones con calidad similar a las obtenidas con el algoritmo Recocido Simulado clásico, con la ventaja de que el tiempo que se tarda en obtener la solución es reducido aproximadamente en un 50%.

Finalmente, el algoritmo se implementa en un sistema de software que posee las características necesarias para resolver el problema ETT con datos reales de una institución.

1.6 Organización de la tesis

En el capítulo dos se describen los aspectos generales del problema Timetabling, su definición y clasificación, así como el papel que ocupa en el área de optimización combinatoria. Se proporciona un panorama general del problema abordado en la tesis, así como su ubicación en el ámbito científico actual. También se presentan algunos enfoques y técnicas de solución existentes en la literatura para resolverlo.

El capítulo tres contiene la descripción y características del algoritmo desarrollado, se especifican las consideraciones del modelo utilizado, al igual que las características de las bases de datos empleadas en la programación de horarios haciendo uso del método descrito. Este capítulo incluye la metodología de desarrollo del método y la complejidad del algoritmo final desarrollado.

En el capítulo cuatro se presenta una descripción general del problema planteado por el PATAT, se describe el modelo matemático implementado para resolver el problema, y se incluyen las pruebas realizadas con los datos del *benchmark*[4] del PATAT.

El capítulo cinco describe la aplicación del modelo desarrollado para el caso particular que se presenta en la Universidad Autónoma de Yucatán (UADY), presentando las características generales del problema, el modelo matemático mediante el cual se representa el mismo, la implementación realizada y los resultados obtenidos.

En el capítulo seis se realiza el análisis y la comparación de los resultados obtenidos para las diversas instancias con las que se prueba el algoritmo.

En el capítulo siete se presenta una arquitectura general para resolver el problema de la asignación de horarios en instituciones educativas de nivel superior, se realiza un modelado bajo una arquitectura Web y se describe la estructura de la base de datos empleada para resolver el problema de tres universidades: la Universidad Autónoma de Yucatán (UADY), la Universidad Juárez Autónoma de Tabasco (UJAT) y el Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), Campus Cuernavaca. El caso de UADY se resuelve en este trabajo de tesis, el caso de la UJAT fue resuelto en [Gómez, 2005] y el caso del ITESM se está resolviendo actualmente [Alonso, 2005].

Finalmente, en el capítulo ocho se presentan las conclusiones obtenidas, las aportaciones hechas y el trabajo futuro considerado.

[4] Se conoce como *benchmark* al estándar de comparación para evaluar el rendimiento de un algoritmo.

Capítulo 2. Problema Educational Timetabling y Métodos de solución

En este capítulo se aborda el problema Educational Timetabling (ETT), empezando por los aspectos básicos del problema, la definición del mismo, una clasificación para los subproblemas existentes, algunos enfoques y algoritmos con los que se ha resuelto el problema particular que se presenta en ciertas instituciones, así como una breve descripción del software comercial existente para resolver el problema.

2.1. Aspectos generales

La optimización combinatoria es una de las áreas con mayor número de aplicaciones prácticas en la vida real, aunque a primera vista ésto no se perciba. En esta área, resolver un problema consiste en encontrar la mejor solución entre un número finito o infinito numerable de soluciones alternativas [Papadimitriou, 1982]. Los problemas que conciernen a éste área son aquellos que pueden ser formulados de manera no ambigua utilizando notación y terminología matemática. Además, se supone que el conjunto de soluciones es finito, que la calidad de una solución es cuantificable y que puede ser comparada con cualquier otra solución [Aarts, 1989].

En un problema de optimización existen diferentes soluciones y un criterio para discriminar entre ellas. En otras palabras, el problema consiste en encontrar el valor de una de las

variables de decisión para el que una determinada función objetivo alcanza su valor máximo o mínimo. El valor de las variables suele estar sujeto a ciertas restricciones [Martí, 2003].

Existe una gran cantidad de problemas de optimización, tanto en la industria como en la ciencia. Entre los problemas de optimización combinatoria más comunes se encuentran: el problema del agente viajero (TSP, *Travelling Salesman Problem*), el problema general de asignación, *Scheduling* y Timetabling. Hasta ahora, no se conoce ningún algoritmo determinístico capaz de generar la solución óptima de esos problemas en tiempo polinomial, por lo que se considera que están entre los problemas computacionales más difíciles de resolver. En el lenguaje computacional, estos problemas se califican con el nombre de NP-duros, particularmente en el contexto de la teoría de la NP-Completez. [Marti, 2003].

La teoría de la NP-Completez fue presentada inicialmente por Cook en 1971. Cook probó que el problema conocido como Satisfactibilidad o SAT, tiene la propiedad de que cualquier problema de la clase NP, puede ser reducido a él a través de una transformación de tipo polinomial. Esto significa que si el problema SAT pudiera ser resuelto en tiempo polinomial, todos los problemas no polinomiales también podrían resolverse en tiempo polinomial, lo que significaría que la clase NP dejaría de existir. De esta forma, si cualquier problema en NP es intratable, entonces SAT es un problema intratable. Cook también sugirió que el problema de satisfactibilidad y otros problemas NP tenían la característica de ser los problemas más difíciles. Estos problemas son de dos tipos: problemas de decisión y problemas de optimización. Los primeros conforman el conjunto que se denominan NP Completos, mientras que sus correspondientes problemas de optimización, reciben el nombre de NP Duros. [Frausto, 2000] (Asesor del presente trabajo).

En términos coloquiales se puede decir que un problema de optimización duro o difícil es aquel para el que no podemos garantizar el encontrar la mejor solución posible en un tiempo razonable.

La existencia de una gran cantidad y variedad de problemas difíciles, que aparecen en la práctica y que necesitan ser resueltos de forma eficiente, impulsó el desarrollo de diversos procedimientos 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 heurísticos o aproximados [Colorni, 1998].

Otros de los problemas concernientes al área de optimización combinatoria son los de secuenciación y *scheduling*, problemas de toma de decisiones que tienen como meta la optimización de uno o más objetivos y juegan un papel crucial en la industria. En la competencia actual de las empresas, la secuenciación y el *scheduling* se han convertido en una necesidad para sobrevivir en el mercado. Se tienen que asignar actividades de tal forma que se usen los recursos disponibles en una forma eficiente [Pinedo, 2002].

Un caso particular de los problemas de *scheduling* es el problema *Educational Timetabling* (ETT), donde los recursos a asignar son profesores, aulas y recursos audiovisuales; las

actividades por asignar son los eventos, tales como cursos, conferencias, talleres, etc. Estableciendo una secuencia jerárquica, el problema de Timetabling es un subproblema del problema de *Scheduling*; a su vez, éste último está dentro del problema general de asignación, el cual es un subconjunto del problema de planeación [Alvarez, 2000]. Esto se ilustra en la Figura 2.1.

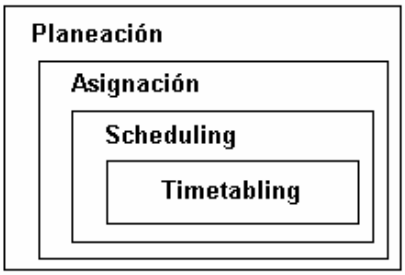

Figura 2-1. Ubicación del problema Timetabling

2.2. Definición del problema Timetabling

Existen diversas variedades del problema Timetabling entre las que se encuentran: el Timetabling de Empleados (ETP, *Employee Timetabling Problem*), la Asignación de Roles de Enfermeras (NRP, *Nurse Rostering Problem*), el problema de los Horarios de trenes (*Railway Timetabling*) y el problema *Educational Timetabling* (*Educational Timetabling Problem,* ETT). El problema ETT consiste en asignar un conjunto de eventos (asignaturas, cursos, conferencias, talleres), los cuales son impartidos por ciertos profesores, en espacios de tiempo específicos (intervalos de 30, 45, 60 ó 90 minutos) definidos en un período de tiempo -generalmente una semana-, satisfaciendo un conjunto de restricciones de diversos tipos.

Este problema ha sido estudiado desde 1960 [Appleby, 1960]. En la literatura se ha propuesto un gran número de variantes del problema, las cuales difieren entre sí dependiendo del tipo de institución involucrada y el tipo de restricciones que se consideran.

En un principio, la mayoría de las técnicas empleadas para resolver el problema ETT estaban basadas en la simulación de los procedimientos humanos (manuales), sin embargo, la solución manual del problema normalmente requiere varios días de trabajo. Además, la solución obtenida podría no ser satisfactoria en algunos aspectos, por ejemplo, un estudiante podría no tomar todos los cursos que requiera debido a que están programados al mismo tiempo. Estas técnicas, también conocidas como heurísticas directas, estaban basadas en un incremento sucesivo de eventos calendarizados; es decir, el horario se extendía evento a evento, hasta que todos los eventos estuvieran en el horario. [Alonso, 2002].

El problema de Timetabling pertenece a la clase de problemas NP-completos en casi todas sus variantes, por lo que sólo es factible una solución exacta para casos pequeños; mientras que cuando el tamaño del problema aumenta, el tiempo de ejecución de un algoritmo exacto aumenta de forma exponencial, lo cual hace prácticamente imposible resolverlo en

un tiempo finito. Por lo tanto, en aplicaciones prácticas deben emplearse métodos aproximados, es decir, algoritmos capaces de obtener soluciones próximas a la solución óptima en un tiempo de cómputo razonable, aunque no se garantice llegar a la solución óptima.

Tanto los métodos exactos como los heurísticos empleados para la resolución del problema, se presentan brevemente, en la sección 2.4.

Debido a la importancia del problema Timetabling, han surgido algunas organizaciones tales como el grupo de trabajo WATT (Working Group on Automated Timetabling) del cual forma parte el PATAT (International Series of Conferences on the Practice and Theory of Automated Timetabling, Serie Internacional de Conferencias sobre la Teoría y Práctica de Timetabling Automatizado) [PATAT, 2005], un foro de intercambio de ideas para investigadores y especialistas de Timetabling.

De los resultados aportados por los miembros del WATT, y en general, en los artículos publicados en la bibliografía actual, existen diversas clasificaciones del problema ETT, desde las más generales hasta las más detalladas, como se describe en las siguientes secciones.

2.3. Clasificación del problema *Educational Timetabling*

Diversos autores [Schaerf, 1999] [Burke, 1997] [Carter, 1998] [Melicio, 1999] han clasificado el problema ETT de varias maneras, de las cuales sobresalen tres categorías principales presentadas a continuación, con las definiciones dadas en [Schaerf, 1999]. Esta clasificación se ilustra en la Figura 2-2[5].

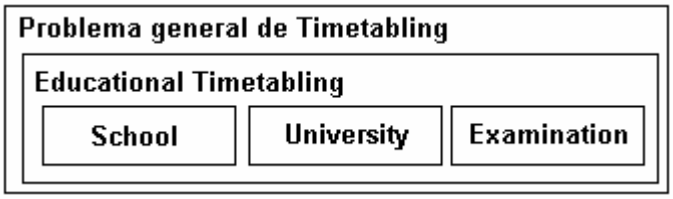

Figura 2-2 Clasificación del Problema Timetabling

- *School Timetabling*: Consiste en la calendarización semanal de todos los grupos de un nivel de enseñanza media (secundaria o preparatoria), evitando que un profesor imparta dos clases al mismo tiempo y viceversa. En esta modalidad, existen grupos de estudiantes ya predefinidos que cursan materias específicas asignadas según su grado escolar.
- *University Timetabling* o *Course Timetabling*: Consiste en calendarizar semanalmente todas las sesiones de un conjunto de cursos universitarios, minimizando el traslape de sesiones entre cursos con estudiantes o profesores en común. La principal diferencia con la modalidad anterior está en que los estudiantes

[5] Se incluye la clasificación empleando la terminología en inglés con el fin de no desviar el significado con una traducción al español.

pueden elegir las materias que desean cursar, por lo que en vez de calendarizar horarios grupales, se trabaja con horarios prácticamente individuales.

- *Examination Timetabling*: Esta variedad consiste en la calendarización de los exámenes de un conjunto de cursos universitarios, evitando el traslape de exámenes de cursos con estudiantes en común, además de procurar una distribución esparcida de exámenes para los estudiantes, tanto como sea posible.

En particular, el presente trabajo está orientado a un problema que tiene características tanto del *University Timetabling* como del *School Timetabling*, con el fin de poder implementar los resultados obtenidos en la asignación de horarios en la Universidad Autónoma de Yucatán.

En el problema ETT existen diversos tipos de restricciones, las cuales difieren entre instituciones pero que en general pueden ser agrupadas en dos tipos principales [Schaerf, 1999]:

- *Restricciones duras*. Son restricciones físicas que deben cumplirse necesariamente para que una solución sea factible. En esta clasificación se encuentra, por ejemplo, que un profesor, salón o estudiante no tenga programada más de una sesión a la vez.
- *Restricciones suaves*. Son aquellas restricciones deseables en el problema, pero que no es indispensable que se cumplan para que el horario sea factible pues no representan un conflicto físico. Estas varían mucho de una institución a otra, y entre ellas se encuentran, por ejemplo, las preferencias de los maestros por impartir clases a ciertas horas y el número máximo de sesiones por día.

Las restricciones del problema dependen de las características de cada institución, sin embargo entre las restricciones duras más comunes se consideran las siguientes:

- Los participantes (profesores o alumnos) no pueden asistir a dos eventos asignados en el mismo horario.
- En un salón no debe programarse más de un evento a la vez.
- La cantidad de alumnos programados en un salón de clases no debe ser mayor que la capacidad del mismo.

Las restricciones suaves difieren mucho entre las instituciones pues están directamente relacionadas con las políticas propias de cada una de ellas. Algunos ejemplos de éste tipo de restricciones son:

- Un estudiante debe tener la menor cantidad de espacios vacíos en su horario.
- Los espacios vacíos deben quedar preferentemente al final de cada horario.
- Existe un máximo número de horas al día frente a grupo para cada profesor.

Cuando el problema consiste en encontrar un horario que satisfaga todas las restricciones, el problema es formulado como un problema de búsqueda. En otros casos, si se requiere un horario que satisfaga todas las restricciones duras y maximice (o minimice) una función objetivo planteada con base en las restricciones suaves, se tiene un problema de optimización.

Las primeras técnicas empleadas en 1960 estuvieron basadas en simular la forma manual en que se resolvía el problema [Appleby, 1960]. Estas técnicas, conocidas como heurísticas directas, se basaban en un aumento sucesivo; es decir, se incluían las sesiones una por una hasta que todas estuvieran contempladas. La idea básica de estas técnicas consistía en calendarizar primero las sesiones con un mayor número de restricciones, y todas diferían entre sí únicamente en el criterio para seleccionar las "más restringidas".

A partir de 1970, aproximadamente, los investigadores empezaron a aplicar técnicas generales para resolver este problema. Por consiguiente, existen algoritmos basados en programación entera [Smith, 1975], flujo de redes [Werra, 1971] [Cheng, 2003], y otros. Además, el problema ETT también ha sido abordado mediante una transformación al problema de coloreo de grafos [Brelaz, 1979] [Burke, 1994], donde los eventos se representan mediante vértices y las aristas representan los conflictos existentes. Posteriormente surgen algunos enfoques basados en técnicas de búsqueda también usadas en Inteligencia Artificial, como por ejemplo el recocido simulado [Elmohamed, 1997] [Abramson, 2001], búsqueda tabú [Hertz, 1992a], algoritmos genéticos [Abramson, 1992] y satisfacción de restricciones [Faber, 1998]. Tales enfoques se conocen como Heurísticas y se describen con más detalle en la siguiente sección.

2.4. Heurísticas y Técnicas de solución

El término Heurística deriva de la palabra griega *heuriskein* que significa encontrar o descubrir y se usa en el ámbito de la optimización para describir una clase de algoritmos de resolución de problemas. También la expresión utilizada por Arquímedes, heureka (que significa "lo he encontrado") al descubrir el denominado principio de Arquímedes, está relacionada con el término heurístico. En [Díaz, 1996] se presentan diversas definiciones de algoritmo heurístico, a partir de las cuales se ha formulado la siguiente:

"Un método heurístico es un procedimiento basado en el sentido común, que se supone ofrecerá una solución cercana a la óptima (aunque no necesariamente la óptima), para problemas difíciles, de un modo más fácil y rápido que la mayoría de los métodos exactos."

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 tal método existe, es de un orden de magnitud muy superior al de la heurística (en muchos casos pudiendo llegar a ser de una complejidad tan grande que el algoritmo correspondiente a la heurística resulta inaplicable).

La justificación de los métodos heurísticos se fundamenta en el hecho de la existencia de problemas de optimización pertenecientes a la categoría denominada NP. Es decir, problemas para los que no existe un algoritmo de resolución que sea polinomial con el tamaño del problema. Si se demuestra que un problema de optimización pertenece a la categoría de problemas NP, es común abordarlo por medio de heurísticas de optimización.

Desde un punto de vista práctico, tiene sentido la utilización de algoritmos heurísticos en aquellos problemas donde la búsqueda exhaustiva sea ineficiente o aquellos donde la cardinalidad del espacio de búsqueda aumenta exponencialmente con el tamaño del problema.

Por lo que respecta a las ventajas derivadas de la utilización de heurísticas para resolver problemas de optimización, quizás la más importante sea la flexibilidad en el manejo del problema en contraposición con la limitación que ocasionan las técnicas de la investigación operativa clásica. Además, las ideas en las que se basan estas técnicas heurísticas son más comprensibles al compararlas con las técnicas matemáticas de resolución.

El gran inconveniente derivado de la utilización de los métodos heurísticos radica en que no es posible conocer a priori la calidad de la solución obtenida con los mismos, desconociéndose, por tanto, la cercanía de dicha solución con respecto a la solución óptima global.

Tratando de establecer criterios genéricos para decir en qué condiciones se aconseja utilizar procedimientos heurísticos, se afirma que su uso es adecuado cuando se verifica una o más de las siguientes condiciones [Marti, 2003] [Blum, 2003]:

- No existe un método exacto de resolución, o en el caso de que dicho método exacto exista, requiere de mucho esfuerzo computacional y/o de memoria.
- No es necesario encontrar la solución óptima, en el sentido del óptimo global sino que es suficiente con obtener una solución de calidad aceptable.
- Los datos son poco fiables y por tanto no tiene sentido el tratar de encontrar el óptimo global para dichos datos, ya que el mismo no sería más que una aproximación al óptimo global que correspondería a los datos correctos.
- Existen limitaciones de tiempo en proporcionar la respuesta y/o de memoria en la computadora que va a efectuar los cálculos.
- Se va a utilizar el resultado proporcionado por la heurística de optimización como solución inicial para un algoritmo exacto de tipo iterativo, el cual reduciría considerablemente el número de iteraciones si parte de una solución inicial suficientemente buena.

Durante las últimas décadas [Kirkpatrick, 1983] [Goldberg, 1989] [Glover, 1997] [Larrañaga, 2002] se han venido desarrollando distintos algoritmos heurísticos de optimización, algunos de los cuales son genéricos e independientes del problema que tratan de optimizar. A este tipo de algoritmos heurísticos se les conoce bajo el nombre de metaheurísticos [Blum, 2003]. La gran ventaja de los mismos radica en que una pequeña modificación o adaptación con respecto de la formulación general es suficiente para que puedan ser aplicados a un problema concreto.

La Tabla 2.1 contiene una clasificación tomada de [Blum, 2003] para algunas metaheurísticas considerando los elementos que intervienen en cada uno de ellas. Los algoritmos mencionados en la Tabla 2-1 son los siguientes: Recocido Simulado (RS)

[Kirkpatrick 1983] [Cerny, 1984], Búsqueda Tabú (TS) [Glover, 1997], Cómputo Evolutivo (EC) [Fogel, 1962], Colonia de Hormigas (ACO) [Dorigo, 1992], Búsqueda de Vecindad Variable (VNS) [Hansen, 1997], GRASP (*Greedy Randomized Adaptive Search Procedure*) [Feo, 1995] y Búsqueda Local Guiada (GLS) [Voudouris, 1995].

Tabla 2-1. Características de algunas metaheurísticas

Metaheurística	Componentes de Intensificación y Diversidad
RS	Criterio de aceptación y esquema de enfriamiento.
TS	Selección de vecindad (lista tabú) y criterio de aceptación.
EC	Recombinación, mutación y selección.
ACO	Actualización de feromonas y construcción probabilística
ILS	Búsqueda local "caja negra", movimientos de impacto, criterio de aceptación.
VNS	Búsqueda local "caja negra", esquema de vecindad, fase de perturbación, criterio de aceptación.
GRASP	Búsqueda local "caja negra", lista de candidatos restringidos.
GLS	Función de penalización.

En general, los métodos para resolver un problema de optimización combinatoria pueden clasificarse en dos tipos: los métodos exactos y los aproximados (o heurísticos); aunque también han surgido hibridaciones entre diversos métodos de cualquier tipo [HEUR, 2005].

2.4.1. Métodos exactos

Los métodos exactos exploran de manera determinística el espacio de soluciones del problema buscando una solución óptima, y aseguran encontrarla en caso de que ésta exista. Además, todos los métodos exactos utilizan condiciones de optimalidad como condición de paro. Cuando el tamaño del problema es relativamente pequeño, el tiempo de ejecución de este tipo de métodos no resulta un gran inconveniente, por lo que resulta adecuado emplearlos, máxime cuando es de gran importancia encontrar la mejor solución del problema. Ejemplos de estos métodos son: la ramificación y acotación, la programación dinámica o las técnicas de "divide y vencerás" [Papadimitrou, 1982]. El gran problema de los métodos exactos es que pueden llegar a ser exhaustivos, de manera que para problemas NP-Duros no garantizan llegar a la solución óptima en tiempo Polinomial.

2.4.1.1. Ramificación y acotamiento

El método de ramificación y acotamiento (B&B, *Branch and Bound* en inglés) está basado en la idea de enumerar de manera "inteligente" todos los puntos de un problema de optimización combinatoria, y probar que una solución es óptima con base en un particionamiento sucesivo del espacio de soluciones. Como su nombre lo indica, se trata de

ramificar (particionar) y acotar (encontrar límites que son usados para construir una prueba de optimalidad sin realizar una búsqueda exhaustiva sobre el espacio de soluciones). [Papadimitrou, 1982].

Características del método Ramificación y acotamiento:

- Generaliza dos algoritmos conocidos como *Best First Search* (Mejor búsqueda primero) y *Depht First Search* (Búsqueda primero en profundidad);
- Para cada estado se guarda el costo de llegar del estado inicial a dicho estado;
- Guarda el costo óptimo global hasta el momento;
- Deja de explorar una rama cuando su costo es mayor que el mínimo actual;
- Si el costo de los nodos es uniforme equivale a realizar una búsqueda por niveles.

El algoritmo de Ramificación y acotamiento se puede aplicar a problemas de programación lineal enteros puros y mixtos. Una desventaja básica de este algoritmo es que prácticamente es necesario resolver un problema de programación lineal completo en cada nodo, lo cual puede consumir mucho tiempo, sobre todo cuando la única información necesaria en el nodo puede ser el valor óptimo de la función objetivo. Sin embargo, independientemente de esta desventaja, este método es uno de los más efectivos para resolver ciertos problemas de gran tamaño que se presentan en la práctica. [Taha, 1995].

Para emplear este método en la resolución del problema ETT, ha resultado conveniente abordarlo bajo el enfoque de coloreo de grafos, es decir, trasladar el problema original a su representación gráfica para posteriormente emplear este método de solución. Aunque, por el tamaño de los problemas reales, la exploración total de un árbol consumiría demasiado tiempo. En [Marte, 2000] se menciona una investigación parcial al respecto y en [Qualizza, 2005] se presenta un enfoque de programación lineal denominado generación de columnas, el cual es resuelto mediante ramificación y acotamiento.

2.4.1.2. Programación dinámica

La programación dinámica (PD) [Guerequeta, 2000] [Fallahi, 2004] es un procedimiento diseñado principalmente para mejorar la eficiencia del cálculo de problemas de programación matemática seleccionados, descomponiéndolos en subproblemas de menor tamaño y, por consiguiente, más fáciles de resolver.

La programación dinámica comúnmente resuelve el problema en etapas, donde en cada etapa interviene exactamente una variable de optimización; los cálculos en las diferentes etapas se enlazan a través de cálculos recursivos de manera que se genere una solución óptima factible para todo el problema. La idea básica de este método consiste en eliminar el efecto de la interdependencia entre etapas, asociando una definición de estado con cada etapa. Una etapa en PD se define como la parte del problema que posee un conjunto de alternativas mutuamente excluyentes, de las cuales se seleccionará la mejor alternativa. Un estado se define normalmente como aquel que refleja la condición (o estado) de las restricciones que enlazan las etapas.

El problema Timetabling ha sido abordado mediante este método en [Marte, 2002], [Abba, 2001] específicamente con un enfoque conocido como satisfacción de restricciones (CSP, *Constraint Satisfaction Problem*).

2.4.1.3. Programación Lineal

La programación lineal (PL) [Taha, 1995] es una de las herramientas más efectivas de la Investigación de Operaciones. Su éxito se debe a la flexibilidad para describir un gran número de situaciones reales en diversas áreas, así como la disponibilidad de programas de computadora para resolver este tipo de problemas.

La programación lineal, que trata exclusivamente con funciones objetivos y restricciones lineales, es una parte de la programación matemática, y una de las áreas más importantes de la matemática aplicada. Se utiliza en campos como la Ingeniería, la Economía, la Gestión, y muchas otras áreas de la ciencia, la técnica y la industria.

Cualquier problema de programación lineal requiere identificar cuatro componentes básicos [Castillo, 2002]:

1. El conjunto de datos.
2. El conjunto de variables involucradas en el problema, junto con sus dominios respectivos de definición.
3. El conjunto de restricciones lineales del problema que definen el conjunto de soluciones admisibles.
4. La función lineal que debe ser optimizada (minimizada o maximizada).

En [Castillo, 2002] se presenta una representación del problema Timetabling desde el enfoque de programación lineal entera-mixta[6].

Se considera que están disponibles n_c aulas y n_h horas, respectivamente, para enseñar n_s asignaturas. Estas asignaturas están agrupadas por cursos y profesores. Se denomina Ω, al conjunto de todas las asignaturas, Ω_i, al conjunto de las n_i asignaturas enseñadas por el profesor i, y Δb, al conjunto de las n_b asignaturas agrupadas en el curso académico b. Los índices s, c, h, i, y b indican respectivamente asignatura, clase, hora, profesor y bloque. Este problema consta de los siguientes elementos:

Datos
n_c: número de aulas
n_h: número de horas
n_s: número de asignaturas
n_i: número de asignatura que debe impartir el profesor i
n_b: número de cursos
Ω: conjunto de todas las asignaturas
Ω_i: conjunto de asignaturas que ha de impartir el profesor i

[6] La programación lineal entera-mixta (PPLEM) es un PPL en el que algunas de las variables toman valores enteros. Si todas las variables enteras son binarias (0/1) el problema se denomina 0/1 PPLEM. Si todas las variables son enteras, el problema se denomina problema de programación lineal entera (PPLE).

Δb: conjunto de asignaturas del curso *b*

Variables
v(s, c, h): variable binaria que toma el valor 1 si la asignatura s se imparte en el aula c a la hora h, y 0 en otro caso.

Restricciones
(a) Cada profesor imparte todas sus asignaturas:

$$\sum_{s\in\Omega_i}\sum_{c=1}^{n_c}\sum_{h=1}^{n_h} v(s,c,h) = n_i, \quad \forall i \tag{2.1}$$

(b) Cada profesor imparte no más de una asignatura por hora:

$$\sum_{s\in\Omega_i}\sum_{c=1}^{n_c} v(s,c,h) \le 1, \quad \forall h, \quad \forall i \tag{2.2}$$

(c) Cada asignatura se imparte una sola vez:

$$\sum_{c=1}^{n_c}\sum_{h=1}^{n_h} v(s,c,h) = 1, \quad \forall s \tag{2.3}$$

(d) En cada clase y hora se imparte una asignatura, como máximo:

$$\sum_{s\in\Omega} v(s,c,h) \le 1, \quad \forall c, \quad \forall h \tag{2.4}$$

(e) En cada hora, no se enseña más de una asignatura de cada curso:

$$\sum_{s\in\Delta b}\sum_{c=1}^{n_c} v(s,c,h) \le 1, \quad \forall h, \quad \forall b \tag{2.5}$$

Función a optimizar. En este caso el objetivo es lograr un horario compacto, lo cual se logra minimizando la función

$$\sum_{s\in\Omega}\sum_{c=1}^{n_c}\sum_{h=1}^{n_h}(c+h) \quad v(s,c,h) \tag{2.6}$$

sujeto a las restricciones (2.1) – (2.5).

Se ha elegido esta función objetivo dado que penaliza el que las variables *v*(*s, c, h*) tomen el valor 1 para los últimos valores de *c* y *h*. Por tanto, su objetivo es compactar el horario. Se hace que los números que identifican la aulas y las horas sean lo más bajos posibles.

Cabe mencionar que en [Castillo, 2002] únicamente se presenta el planteamiento del problema, más no el método de solución empleado para resolverlo. Además, el ejemplo presentado es muy pequeño (3 aulas, 5 horas, 8 asignaturas, 2 profesores, y 2 cursos), a diferencia de los problemas reales que se presentan en las instituciones educativas.

2.4.2. Métodos aproximados o heurísticos

Muchos problemas de optimización no pueden ser abordados por métodos exactos, ya sea, por su alto grado combinatorio o por la dificultad de generar un modelo basado en programación matemática que represente exactamente una situación real. Para situaciones

de ésta naturaleza se han venido generando desde la década de los sesenta métodos conocidos como heurísticos o aproximados.

Los métodos aproximados son aquellos que no garantizan encontrar la mejor solución existente, sin embargo, son algoritmos capaces de obtener soluciones próximas al mínimo (o máximo, según sea el caso) absoluto en un tiempo de computación razonable. En aplicaciones prácticas de problemas cotidianos, estos métodos suelen ser incluso más eficientes que los métodos exactos, sobre todo si se comparan con respecto al tiempo de ejecución.

A continuación, se describen algunos de los métodos aproximados, muy populares en la actualidad debido al éxito obtenido en su implementación, para resolver problemas de diversa naturaleza.

2.4.2.1. Búsqueda Tabú

Los orígenes de la búsqueda tabú (TS, por sus siglas en inglés) [Glover, 1997] pueden situarse en diversos trabajos publicados hace alrededor de 20 años. Sin embargo, oficialmente, el nombre y la metodología fueron introducidos por Fred Glover. En la literatura han aparecido numerosas aplicaciones, al igual que artículos y libros para difundir el conocimiento teórico del procedimiento [Glover,1989] [Wright, 1996] [Hertz, 1991].

La idea de la búsqueda tabú es partir de una solución aleatoria y trasladarse sucesivamente a otras que se encuentren en el espacio de soluciones vecino. Cada vez que se realiza un movimiento, es decir, que se encuentra una nueva solución, se prohíbe regresar a la solución anterior, es decir, no se permite realizar la acción que conduciría a obtener la solución anterior a la actual.

Esto se realiza con la inclusión de la solución o del movimiento en una lista tabú, donde permanecerá por algunas iteraciones. Para una solución dada, no todos los candidatos son siempre aceptados. Si la nueva solución mejora la actual, pero tiene el status tabú, sólo será aceptada si supera la mejor solución global encontrada hasta ese momento, de lo contrario, será rechazada (nivel de aspiración).

La heurística TS ha sido implementada con éxito para resolver el problema ETT [Werra, 1985], [Hertz, 1992], [Hertz, 1992a], [Colorni 1998].

2.4.2.2. Algoritmos genéticos

Los Algoritmos Genéticos (AG, por sus siglas en inglés) [Goldberg, 1989] son una herramienta de propósito general basada en la teoría de la evolución de Darwin para asegurar la perpetuación de las especies mediante la selección de los mejores individuos y han sido exitosamente aplicados en una amplia variedad de problemas [Ross, 1994].

Los AG operan sobre una población codificada de soluciones. Cada miembro de la población consta de un número de genes, cada uno de los cuales es una unidad de información. Se obtienen nuevas soluciones mediante la combinación de genes de diferentes miembros de poblaciones (cruzamiento) para producir su descendencia o mediante la alteración de miembros existentes en la población (mutación).

Generalmente se emplea una representación binaria para codificar las soluciones del problema, lo cual se expresa como cadenas de bits, aunque también se han utilizado cadenas de enteros para realizar la codificación. La solución se mejora mediante diferentes métodos de cruce, mutación y selección. Un algoritmo genético aplica tanto el operador de recombinación de dos soluciones como el de mutación aleatoria del contenido de los individuos. [Holland, 1992].

Como su nombre lo indica, los algoritmos genéticos intentan resolver problemas de optimización combinatoria en una forma similar a la selección natural biológica y a los esquemas de supervivencia. Los algoritmos genéticos se han aprovechado en la solución de muchos problemas de optimización, en particular, de programación entera.

Una de las ventajas que presenta un AG frente a otros métodos heurísticos es la búsqueda en diversos puntos dentro del espacio de soluciones al mismo tiempo, y una manera de aumentar la eficiencia de un algoritmo genético es combinarlo con algún otro método que permita encontrar el óptimo (global), siempre y cuando éste exista.

Para la solución del problema ETT se han utilizado heurísticas que han demostrado encontrar soluciones óptimas en tiempos razonables, se han utilizado algoritmos genéticos en [Paecher, 1994] [Mahdi, 2003] [Abramson, 1992] [Fernandes, 1999], entre otros.

2.4.2.3. Recocido Simulado

Una de las heurísticas más exitosas empleadas para la resolución de este problema es el Recocido Simulado (RS por sus siglas en inglés: *Simulated Annealing*). Este concepto fue introducido por Kirkpatrick en 1983 [Kirkpatrick, 1983] e independientemente por Cerny en 1985 [Cerny, 1985]. El nombre proviene de una analogía con el proceso físico realizado en la metalurgia, que consiste en recalentar un sólido hasta dejarlo en su estado líquido, para posteriormente enfriarlo lentamente con el objetivo de que sus átomos alcancen una configuración óptima, logrando así una mejor calidad en la consistencia del material.

En general, el algoritmo RS consta de 2 ciclos, uno externo regulado por la temperatura del algoritmo y limitado por el criterio de paro, y otro interno limitado por el equilibrio estocástico, es decir, por el máximo número de iteraciones permitidas para buscar la mejor solución a cada temperatura.

RS ha demostrado ser una herramienta muy exitosa para resolver una amplia gama de problemas de Optimización combinatoria. El RS es una variante de la búsqueda local que permite aceptar movimientos “malos” con una probabilidad dada por la distribución de Boltzman, con la finalidad de escapar de óptimos locales. [Dowsland, 2001]

Estrictamente hablando, RS no es un algoritmo sino una heurística aleatoria que en su implementación requiere de la adecuada selección de parámetros para desempeñarse adecuadamente y encontrar así soluciones muy cercanas al óptimo sin quedar atrapado prematuramente en un óptimo local[7].

Existen resultados teóricos basados en la teoría de las cadenas de Markov que demuestran que con un programa de enfriamiento infinitamente lento, el algoritmo de Recocido Simulado convergería al óptimo con probabilidad de 1 [Hajek, 1988] Sin embargo, con programas de enfriamiento finitos no se puede garantizar esta convergencia al óptimo, lo cual conduce a la necesidad de seleccionar parámetros adecuados que conduzcan a obtener buenas soluciones en una cantidad de tiempo considerable. [Aarts, 1989].

Las decisiones que hay que tomar para una buena selección de parámetros del algoritmo pueden ser clasificadas en dos grupos [Downsland, 2001]: las genéricas y las específicas del problema. Las decisiones genéricas están básicamente relacionadas con los parámetros que dirigen el enfriamiento del sistema, tales como la temperatura inicial y final, la velocidad a la que disminuirá dicha temperatura (velocidad directamente relacionada con el factor de decremento) y el criterio de paro del algoritmo. Las decisiones específicas del problema se refieren sobre todo a la definición del espacio de soluciones, la estructura de los entornos (criterio de vecindad), la función de costo o función objetivo, y la elección de la solución inicial.

Se han propuesto diversos esquemas de enfriamiento y sugerencias para una adecuada elección de parámetros. En [Sanvicente, 2004] se propone un método que permite encontrar de manera analítica los mejores parámetros del algoritmo, dicho método se ha llamado "sintonización de parámetros" y permite disminuir considerablemente el número de ensayos realizados en la búsqueda de los mejores parámetros del algoritmo. El método asegura que las temperaturas inicial y final están en función de los incrementos máximo y mínimo en el costo, obtenidos de la estructura de la vecindad empleada.

Dueck y Sheuer [Dueck, 1990] proponen una ligera modificación al algoritmo estándar de RS, dando origen a la heurística denominada "Umbral de Aceptación" (TA, Treshold Accepting), la cual propone el uso de un umbral de tolerancia en vez de la distribución de Boltzmann para aceptar los movimientos "malos", que ayudan a escapar de los óptimos locales. Dueck y Scheuer aplican el algoritmo a un problema del agente viajero y aseguran que TA es mejor que RS. Originalmente, los autores sugieren que el umbral debe decrementarse cuando el algoritmo no mejore la solución por mucho tiempo. Sin embargo, no queda claro cuándo y cuánto decrementarlo.

La principal diferencia entre RS y TA, es el mecanismo para aceptar la solución elegida aleatoriamente del conjunto de vecinos de la solución actual. RS usa un modelo estocástico y TA usa un modelo estadístico. En TA, si la diferencia entre el valor objetivo de la nueva solución y el valor objetivo de la solución actual es menor que un umbral T, TA acepta la

[7] Un óptimo local es el mejor punto en una región limitada y difiere del óptimo global en que éste último es el mejor punto en todo el espacio de soluciones y no únicamente en una región específica.

nueva solución, en caso contrario la rechaza y conserva la solución actual. El umbral es un parámetro de control positivo que se decrementa (converge a cero) conforme se incrementa el número de iteraciones. Es decir que, cuando una solución es aceptada, el umbral se reduce. De esta manera, en cada iteración se permiten movimientos que no deterioran la solución actual en un nivel mayor que el permitido por el umbral actual T. Al final de la ejecución, sólo se aceptan las mejores soluciones. Por lo tanto, los parámetros del algoritmo TA son: el umbral inicial, el factor de reducción, el número de vecinos a ser examinados en una iteración, el número de iteraciones sin cambio en la función objetivo, y finalmente, el criterio de paro.

Otra variante del algoritmo RS, presentada en [Bilbro, 1992] a la cual se ha denominado Mean-Field Annealing (MFA), es una aproximación determinista significativamente más rápida que el RS. En vez de simular directamente la transición[8] estocástica del recocido simulado, se utiliza el comportamiento medio (o promedio) de las transiciones para caracterizar un sistema estocástico. De esta manera, en el ciclo interno del algoritmo se logra un equilibrio más rápidamente, por lo que se realiza un relajamiento a cada temperatura mucho más rápido que en el recocido simulado estándar.

En [Ingber, 1993a] se presenta otra mejora al algoritmo estándar de Recocido simulado, a la cual se ha denominado Recocido Simulado Adaptativo (ASA, *Adaptive Simulated Annealing*), y cuyo desempeño está basado en una demostración que afirma que el espacio de soluciones puede ser muestreado mucho más eficientemente que de la forma aleatoria comúnmente usada. Es decir que, el algoritmo se adapta al espacio de soluciones reducido en vez de explorar de manera aleatoria todo el espacio de búsqueda. Originalmente, este algoritmo se llamó Recocido Simulado Muy Rápido (VFSA, Very Fast Simulated Annealing) [Ingber, 1993b], y fue en 1993 cuando el autor lo rebautizó, después de hacerle otras mejoras al algoritmo, cuyo código se encuentra disponible en la página Web del autor [Ingber, 2005].

En [Abramson, 1991] se presenta otro trabajo donde los átomos que corresponden al RS son reemplazados por elementos, cada elemento corresponde a una tupla[9] formada por una clase de estudiantes, un profesor, una asignatura y un salón de clases. Los autores suponen que los estudiantes ya están agrupados en clases, y que las clases no tienen estudiantes en común. Para medir la calidad de la solución, se define una función objetivo o función de costo, de tal forma que el sistema de energía de la heurística es reemplazada por el costo de la asignación de horarios. Se genera una solución inicial situando los elementos en un período elegido aleatoriamente, y se calculan el costo y la temperatura inicial. Además, se crean dos límites para el número de intercambios y el número de intercambios exitosos, elegidos proporcionalmente al número de elementos. Las condiciones para terminar una corrida son dos: que el costo sea cero, o bien que el costo no cambie en cierto número de iteraciones.

[8] Al cambio de una solución a otra se le conoce como transición.
[9] Una tupla es un elemento formado por la combinación de otros elementos.

Otra aplicación del RS se encuentra en el artículo [Wright, 2003], donde se reportan los resultados de varios experimentos llevados a cabo para probar el valor de la aleatoriedad en el criterio de aceptación para el Recocido Simulado. En el caso del problema de Timetabling, la función de costo es una combinación lineal de los conflictos de enseñanza preferentes e insuficiencias departamentales, penalizados cuando las lecciones que necesitan recursos en común se calendarizan en el mismo período de tiempo y cuando las lecciones que requieren ser simultáneas no se calendarizan en el mismo período de tiempo. Las permutaciones que se consideran corresponden al intercambio de lecciones entre períodos de tiempo. La búsqueda procede sistemáticamente a través de los vecinos sin aleatoriedad. Se usa un esquema de enfriamiento geométrico, el inicio y fin de la temperatura se fija después de cierta cantidad de experimentos par asegurar que estén cerca del óptimo. El número total de perturbaciones se fijó en 300,000. En este caso el algoritmo de RS resultó ser mejor que el VT (Variable Threshold), contra el cual se comparó.

2.5. Software comercial existente

Actualmente, se encuentra disponible en el mercado software comercial que permite a ciertas instituciones realizar la asignación de horarios. Sin embargo, sólo son útiles para aquellas instituciones cuya estructura y políticas pueden adaptarse al modelo que los proveedores consideraron al realizar dicho software, además de que se requiere contar con el capital suficiente para comprar las licencias necesarias (cuyo costo está alrededor de los $500.00 dólares anuales por licencia). Lo anterior pocas veces ocurre, pues es difícil considerar todas las restricciones posibles del problema real en cuestión debido a la falta de homogeneidad entre las políticas de cada institución.

Por otra parte, las instituciones públicas de tamaño mediano difícilmente cuentan con los recursos económicos necesarios para solventar el pago de las licencias de este tipo de software, considerando el número de licencias que tendrían que adquirir. Por ejemplo, la Universidad Autónoma de Yucatán cuenta con 15 Facultades y 2 Preparatorias, por lo que, si se consideran 10 licencias por Facultad, se tendría que pagar aproximadamente $8,500 dólares al año.

A continuación se describen algunos de los programas disponibles en la actualidad, que intentan encontrar una solución al problema.

En [Lierenski, 2003] se presenta un Sistema Automatizado de Timetabling (SAT), donde para resolver el problema se utiliza programación de restricciones junto con una técnica de búsqueda local. Para la implementación de la interfaz se utilizó MySQL, PHP, Javascript y HTML, VC++ y el *solver*[10] Mozart/OZ. Este artículo parte del supuesto de que los lenguajes que se proponen en [Kingston, 2001], [Reis, 2001] y [De Causmaecker, 2002], contemplan todas las restricciones posibles de una universidad.

[10] Se conoce como *solver* a aquellos programas disponibles en la Web para resolver instancias de problemas.

Existe también un software denominado GHOL (Generador de Horarios On Line) [Parada, 2005] desarrollado por un grupo de alumnos de la Universidad Santiago de Chile, quienes implementaron el algoritmo de solución y realizaron un sitio web en el que se puede descargar libremente la aplicación, introducir los datos de manera local y posteriormente subir el archivo generado por dicha aplicación para que de manera remota se realice la asignación del horario. Sin embargo, esta aplicación está más orientada a la interfaz, y no tanto al algoritmo de resolución, por lo que una de las desventajas es la pérdida de generalidad en los casos que pueden ser resueltos. El sistema no menciona cuál es el método de resolución empleado, aunque corresponde al área de optimización.

En la Facultad de Ciencias de la Universidad Autónoma de Baja California también se ha implementado un sistema generador de horarios [SIGEH, 2005]. Sin embargo, el sistema fue diseñado específicamente para cubrir las necesidades de dicha institución, y no se proporciona información sobre la manera de realizarlo. Más que un proyecto de optimización, corresponde al área de ingeniería de software.

La Universidad de Vigo de España también cuenta con una aplicación denominada GeHoWeb (Generación de horarios) [GeHoWeb, 2005]. El sistema fue diseñado específicamente con las políticas de dicha universidad y no se proporciona información sobre la manera de encontrar la solución.

El Ministerio de Educación y Ciencia de España [MECE, 2005] cuenta con un sistema para la generación de Horarios para IES (Institutos de Educación Secundaria). Este sistema es de distribución gratuita aunque presenta limitaciones en cuanto al número de períodos diarios que permite y al número de relaciones entre sesiones de clases. Tampoco se proporciona información sobre el método empleado para encontrar la solución presentada.

Estos son algunos de los trabajos revisados, y además, existen otros programas comerciales [Lantiv, 2004] [Mimosa, 2004] [Gti, 2004] [Timetabler, 2004] cuyo método para encontrar el resultado del problema Timetabling permanece totalmente oculto, y para utilizarlos libremente es necesario adquirir la licencia correspondiente, por lo que quedan fuera de los alcances del presente trabajo.

2.6. Resumen del capítulo

En este capítulo se presentó un panorama general para contextualizar el problema ETT, desde la definición hasta el software comercial existente y las razones por las cuales en muchas ocasiones éste resulta inadecuado. También se ha presentado la clasificación más general realizada acerca del problema ETT, así como también las heurísticas y técnicas de solución empleadas para su resolución. Lo anterior con el fin de proporcionar las herramientas básicas para comprender los siguientes capítulos, en los cuales se describirá lo realizado para resolver los casos teórico y práctico que se presentan.

Capítulo 3. Recocido Simulado y el problema de Sintonización de Parámetros

En este capítulo se aborda el método empleado para resolver el problema ETT: un algoritmo basado en la metaheurística Recocido Simulado, realizando una sintonización de parámetros con el fin de disminuir el tiempo de ejecución del algoritmo implementado. Primero, se describen los aspectos básicos de dicha heurística, seguidamente se especifican las consideraciones tomadas en cuenta para la implementación del algoritmo y se detallan las características del algoritmo Recocido Simulado Sintonizado codificado. Finalmente se realiza un análisis de eficiencia del algoritmo implementado.

3.1. Recocido Simulado Clásico

Como se mencionó en el capítulo anterior, Recocido Simulado (RS) es una de las metaheurísticas empleadas con mayor éxito en la resolución del problema Timetabling. El algoritmo RS permite algunos movimientos "malos" para escapar de óptimos locales con el propósito de alcanzar el óptimo global. Esta metaheurística ha demostrado ser una herramienta muy exitosa para resolver una amplia gama de problemas de Optimización combinatoria.

Los fundamentos del mecanismo de escape del algoritmo RS se encuentran en el trabajo de Metropolis [Metropolis, 1953], quien modeló el proceso de recocido físico, simulando los cambios energéticos en un sistema de partículas conforme decrece la temperatura hasta que converge a un estado de equilibrio. Cada configuración del sistema S_i, definida por la

solución del problema en ese estado, tiene asociado un factor de probabilidad de Boltzmann, $\exp(E(S_i)/K_BT)$, donde $E(S_i)$ representa la energía de la configuración S_i, K_B es la constante de Boltzmann (con valor $1.38x10^{-23}$) y T es la temperatura.

El algoritmo de Recocido Simulado consta de dos ciclos: uno externo (o ciclo de temperatura) que es controlado por un criterio de paro regulado por el factor de equilibro térmico, y uno interno (o ciclo de Metrópolis) que es regulado por el factor de equilibrio estocástico. Los parámetros que el algoritmo emplea son: la temperatura inicial T_I , la temperatura final T_F , una función para realizar el enfriamiento de la temperatura, y un número máximo de intentos por encontrar la mejor solución a cada temperatura. Además, en el algoritmo se genera una solución inicial S_I que se obtiene de manera aleatoria (random) del conjunto de soluciones posibles.

En el diagrama de flujo de la Figura 3-1, la variable *Maxiter* toma un valor inicial *x* que representa el máximo número de iteraciones realizadas en cada ciclo de metrópolis; la variable *iter* representa el número de iteraciones ya realizadas en ese ciclo. *T* corresponde a la temperatura con la que trabaja el algoritmo en cada ciclo de temperatura, T_I y T_F son las temperaturas inicial y final, respectivamente. S_I y S_{new} corresponden a las soluciones actual y generada a partir de la actual, respectivamente. Finalmente, α es el factor de decremento de la temperatura, $0.7 \leq \alpha < 1$, empleada en la función geométrica de enfriamiento.

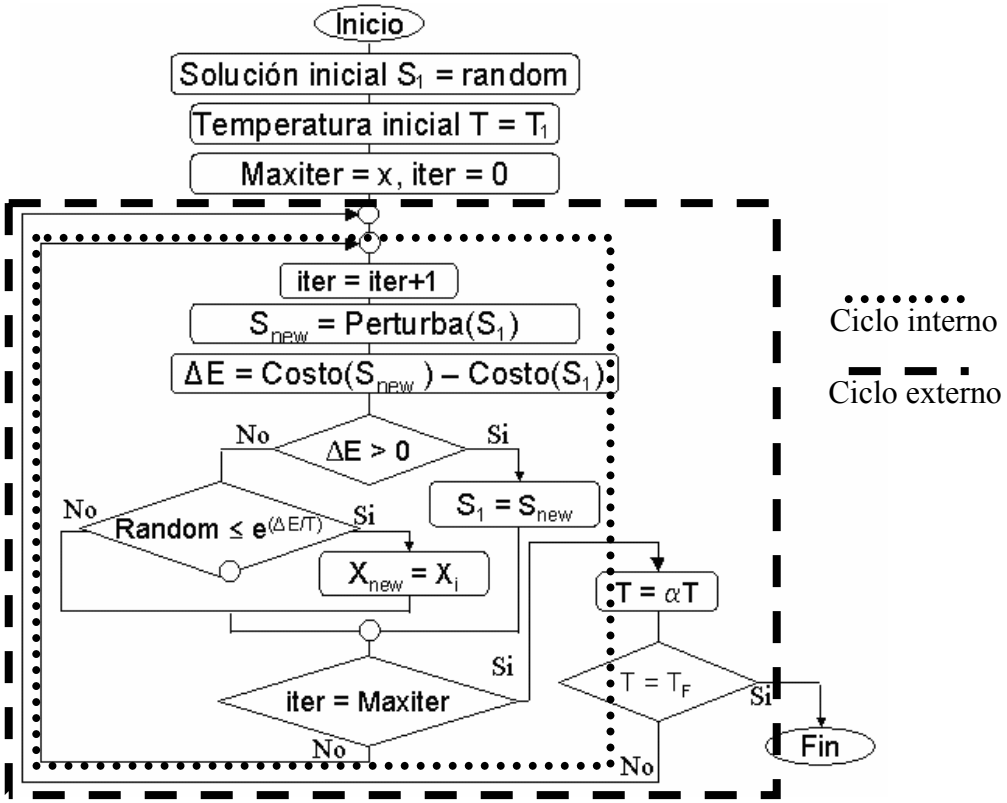

Figura 3-1. Diagrama de flujo del RS general para Maximización

3.2. Planteamiento del problema de sintonización de parámetros

El algoritmo de recocido simulado inicialmente debe aceptar casi todas las soluciones, explorando de manera aleatoria todo el espacio de soluciones. Progresivamente la temperatura debe ir disminuyendo, lo cual ocasiona que la probabilidad de aceptar malas soluciones sea menor. La bondad de toda solución encontrada se evalúa en función del costo asociado a la función objetivo del problema particular que se desea resolver. Al final del algoritmo, sólo se deben aceptar los movimientos que mejoran la función objetivo.

En este trabajo se ha denominado Sintonización de Parámetros al hecho de encontrar los parámetros adecuados (temperatura inicial, temperatura final y factor de enfriamiento) para el buen desempeño del algoritmo. En la literatura se han propuesto diversos esquemas de enfriamiento[11], así como sugerencias para una adecuada selección de parámetros [Lundy, 1986] [Hajek, 1988] [Ingber, 1993a]. En la mayoría de los métodos se realiza una búsqueda iterativa mediante una secuencia de ensayos, con los que se busca encontrar una razón de aceptación de transiciones óptima, tanto al inicio como al final de la ejecución del algoritmo.

Se puede considerar que, una vez establecido el espacio de soluciones y el esquema de vecindad[12], el funcionamiento del algoritmo RS consiste en una secuencia de pasos donde se parte de la solución actual y se llega a alguna de las soluciones vecinas con cierta probabilidad. Este proceso puede verse como una Cadena de Markov, es decir, una secuencia de experimentos en la que el resultado de cada experimento depende únicamente del resultado del experimento anterior [Aarts, 1989]. En el caso del algoritmo RS, los experimentos equivalen a las transiciones y el resultado de una transición depende únicamente del estado previo.

Formalmente, una Cadena de Markov se describe por medio de un conjunto de probabilidades condicionales $P_{ij}(k, k+1)$ para cada par de resultados (i, j), donde $P_{ij}(k, k+1)$ es la probabilidad de que el experimento $(k+1)$-ésimo sea j, suponiendo que el k-ésimo resultado sea i. [Andreu, 2000]. En el caso del algoritmo RS, la probabilidad condicional $P_{ij}(k, k+1)$ representa la probabilidad de que la k-ésima transición ocurra del resultado i al resultado j.

En [Hajek, 1988] se demuestra, con base en la teoría de las cadenas de Markov, que con un programa de enfriamiento infinitamente lento, el algoritmo de Recocido Simulado converge al óptimo global con probabilidad de 1. Sin embargo, y pese al buen desempeño observado en diferentes aplicaciones, el algoritmo requiere de una adecuada selección de parámetros para desempeñarse adecuadamente y encontrar así soluciones muy cercanas al óptimo sin quedar atrapado prematuramente en un óptimo local. [Downsland, 2003].

[11] Se le llama **Esquema de enfriamiento** a los parámetros T_1, T_F y α pues son éstos quienes determinan la velocidad de enfriamiento del algoritmo.

[12] El **criterio de vecindad** es aquél que determina a cuáles soluciones es posible llegar, partiendo de la solución que se tiene en un momento dado.

El método para sintonizar los parámetros propuesto en [Sanvicente, 2004], permite disminuir considerablemente el número de ensayos realizados en la búsqueda de los mejores parámetros del algoritmo. Dicho método asegura que las temperaturas inicial y final están en función de los incrementos máximo y mínimo en el costo, obtenidos del criterio de vecindad empleado. En la siguiente sección se presenta el método empleado para sintonizar el esquema de enfriamiento del algoritmo RS implementado.

3.3. Esquema de enfriamiento

En el algoritmo RS, el esquema de enfriamiento está determinado por tres parámetros que determinan en gran medida la eficiencia que tendrá el algoritmo RS. Tales parámetros son: la temperatura inicial (T_I), la temperatura final (T_F) y el parámetro que determina la velocidad de la función de enfriamiento (α).

La selección de la función de enfriamiento está basada en la analogía con el recocido de sólidos, donde la convergencia a la solución óptima global depende de la velocidad dada por el esquema de enfriamiento. La función geométrica de reducción (3.1) es una manera fácil e intuitiva de establecer dicha velocidad.

$$T_i = \alpha\, T_{i-1} \tag{3.1}$$

Para establecer el valor de T_I y T_F así como el número de iteraciones del ciclo de Metrópolis, generalmente se realiza una sintonización de parámetros de manera experimental, ejecutando varias veces el algoritmo hasta encontrar a aquellos con los que se obtienen los mejores resultados. Una nueva forma de sintonizarlo [Sanvicente, 2004] consiste en aplicar un método analítico basado en la teoría de procesos de Markov [Aarts, 1989]. En este último caso, para obtener el valor de T_I se busca encontrar una razón de aceptación de transiciones muy cercana a uno, es decir, es deseable que al principio se acepte prácticamente cualquier solución. Por el contrario, para encontrar el valor de T_F, se busca encontrar un valor (punto de congelamiento) mediante un criterio de paro en el que la función de costo no experimente cambio alguno, o bien, el cambio sea mínimo.

Los métodos que realizan la búsqueda de parámetros de manera totalmente experimental, requieren de invertir mucho tiempo en la realización de ejecuciones del algoritmo, además de que los parámetros encontrados únicamente serán útiles para problemas de cierta naturaleza (la del problema que se esté resolviendo), y si se desea resolver algún problema de otro dominio será necesario realizar nuevamente una serie de iteraciones en búsqueda de los mejores parámetros.

En [Sanvicente, 2004] se presenta un método analítico que permite determinar los valores para T_I y T_F , sin tener que realizar demasiada experimentación. Dicho método considera los siguientes términos:

- El criterio de vecindad
- Los deterioros máximos y mínimos
- La probabilidad de aceptación deseada al principio y al final del algoritmo.

El criterio de vecindad empleado es importante en cuanto a que establece la forma en que se realiza la búsqueda a través del espacio de soluciones del problema. Una mala definición del esquema de vecindad para un problema particular, puede ocasionar una falta de exploración del espacio de soluciones, lo cual muy probablemente conduciría a no encontrar el óptimo global deseado.

El deterioro máximo está dado por la diferencia máxima en el costo (establecido mediante la función objetivo) que se puede producir entre dos soluciones vecinas. De manera análoga, el deterioro mínimo está dado por la mínima diferencia que se puede producir entre dos soluciones vecinas. Como puede observarse, ambos deterioros están en función del esquema de vecindad definido para cada problema, sin embargo, su obtención se realiza de manera general: considerando los casos "peor" y "mejor", respectivamente, que pueden presentarse entre dos soluciones vecinas.

La probabilidad de aceptación es aquella que se proporciona para decidir si se admite una solución específica, y puede establecerse tomando en cuenta el criterio de Metrópolis, es decir, la probabilidad de aceptar una solución al principio debe ser cercana a 1 y al final, cercana a 0.

El método para sintonizar las temperaturas, propuesto en [Sanvicente, 2004] surge del siguiente análisis:

Sea $P_A(s_j)$ la probabilidad de aceptar la nueva solución s_j generada a partir de la solución actual s_i. De donde, la probabilidad de rechazar dicha solución está dada por:

$$P_R(s_j) = 1 - P_A(s_j) \tag{3.2}$$

Ahora bien, la aceptación o rechazo de s_j únicamente depende del deterioro que dicha solución provoque en la función de costo, lo cual puede ser expresado como sigue:

$$P_A(s_j) = g(Z(s_i) - Z(s_j)) = g(\Delta Z_{ij}) \tag{3.3}$$

Donde $Z(s_i)$ y $Z(s_j)$ son los costos asociados a s_i y a s_j, respectivamente; $g(\Delta Z_{ij})$ es una función que define la probabilidad de aceptar la diferencia de costo $\Delta Z_{ij} = Z(s_i) - Z(s_j)$.

De esta manera, los deterioros máximo y mínimo producidos están dados por:

$$\Delta Z_{Vmax} = MAX \{ Z(s_i) - Z(s_j) \} \qquad \forall s_j \in Vs_i, \forall s_i \in S \tag{3.4}$$

$$\Delta Z_{Vmin} = MIN \{ Z(s_i) - Z(s_j) \} \qquad \forall s_j \in Vs_i, \forall s_i \in S \tag{3.5}$$

Donde ΔZ_{Vmax} y ΔZ_{Vmin} son los deterioros máximo y mínimo, respectivamente. Vs_i es el esquema de vecindad de una solución s_i y está definido de la siguiente manera:

$$\{ \forall s_i \in S, \exists \text{ un conjunto } Vs_i \subset S \mid Vs_i = V : S \rightarrow S \}$$

Donde $V: S \rightarrow S$ es una función de mapeo que relaciona a dos conjunto de soluciones. De lo anterior puede observarse que los vecinos de una solución s_i, así como los valores ΔZ_{Vmax}

y ΔZ_{Vmin} dependen de la estructura de vecindad Vs_i establecida, la cual puede ser de dos maneras: mediante un muestreo sin reposición de elementos o mediante un muestreo con reposición de elementos [Sanvicente, 2004]. Ambas formas difieren únicamente en la manera de seleccionar a los vecinos de una solución, el primero de ellos no permite que en un muestreo se seleccionen elementos repetidos mientras que el segundo sí permite seleccionar elementos repetidos. Como puede observarse, el tamaño de la vecindad es mayor en el segundo caso.

Ahora bien, como ΔZ_{Vmax} proporciona el máximo deterioro en costo que se puede producir mediante el esquema de vecindad empleado, y puesto que al principio de la ejecución la probabilidad de aceptación debe ser cercana a 1, entonces $P_A(\Delta Z_{Vmax}) \approx 1$. Esta probabilidad está directamente relacionada con la temperatura inicial T_1, es decir:

$$\exp\left(\frac{-\Delta Z_{V\max}}{T_0}\right) = P_A(\Delta Z_{V\max}) \tag{3.6}$$

De donde se obtiene:

$$T_1 = \frac{-\Delta Z_{V\max}}{\ln\left(P_A(\Delta Z_{V\max})\right)} \tag{3.7}$$

Donde $P_A(\Delta Z_{Vmax})$ puede tomar valores cercanos a 1, según el grado de confiabilidad que se desee y dependiendo de la naturaleza del problema. En [Sanvicente, 2004] se sugieren los valores 0.90, 0.95, 0.99, con lo cual se garantiza que al inicio del proceso se acepte cualquier solución encontrada.

Por otro lado, cuando el algoritmo se encuentra cerca del punto de congelamiento, la probabilidad de aceptar una solución que no mejore el costo dado por la función objetivo debe ser cercana a cero, es decir, $P_A(\Delta Z_{Vmin}) \approx 0$.

Puesto que ΔZ_{Vmin} establece el deterioro mínimo que puede ser producido al final del algoritmo, entonces se obtiene:

$$\exp\left(\frac{-\Delta Z_{V\min}}{T_F}\right) = P_A(\Delta Z_{V\min}) \tag{3.8}$$

Por lo que la temperatura T_F debe ser:

$$T_F = \frac{-\Delta Z_{V\min}}{\ln\left(P^A(\Delta Z_{V\min})\right)} \tag{3.9}$$

Donde $P_A(\Delta Z_{Vmin})$ puede tomar valores cercano a cero, dependiendo de la naturaleza del problema. En [Sanvicente, 2004] se sugieren valores tales como 0.10, 0.05, 0.01. Más adelante se ejemplifica el procedimiento descrito en esta sección para calcular los valores de T_1 y T_F.

La función de enfriamiento más utilizada, misma que se emplea en este trabajo por la sencillez y eficiencia que otorga, es la función de reducción geométrica propuesta en [Kirkpatrick, 1983]:

$$T_{i+1} = \alpha T_i \ , \quad \text{con } \alpha \approx 1,\ 0.7 \leq \alpha < 1 \tag{3.10}$$

Cuando $\alpha \rightarrow 1$ la velocidad de enfriamiento es pequeña, mientras que cuando $\alpha \rightarrow 0$, la velocidad es mayor.

3.4. Cadenas de Markov en el Ciclo Metrópolis

En [Aarts, 1989] se demuestra que el ciclo interno del algoritmo RS corresponde a una cadena de Markov, por lo que el número de iteraciones realizadas en ese ciclo corresponde con el valor de la longitud de la cadena de Markov. Esta longitud es otro parámetro cuya adecuada selección afecta en forma directa al tiempo que se tarda el algoritmo en encontrar la mejor solución, por lo que en el presente trabajo también se considera como parte de los parámetros que requieren sintonizarse. Es decir, el número de intentos que se deben realizar buscando la mejor solución para el valor de la temperatura t_i en cada iteración i. Donde i corresponde al índice de iteraciones del ciclo externo del algoritmo (Figura 3-1).

Los métodos empleados para determinar el mejor número de iteraciones que deben ser ejecutadas realizan un proceso de sintonización experimental, sin embargo, gastan demasiado tiempo al principio del algoritmo o bien provocan una finalización prematura del proceso, cuando dicha longitud no se selecciona adecuadamente.

En [Kirkpatrick, 1983] el ciclo interno del algoritmo RS es ejecutado un mismo número de veces L_{max}, para todas las iteraciones. El valor de L_{max} es establecido a $L_{max} = n$, donde n es el número de variables del problema a resolver. En [Aarts, 1989] la longitud máxima de la cadena de Markov se establece como un múltiplo m del tamaño de la vecindad, es decir, $L_{max} = m|V_{Si}|$. Donde $|V_{Si}|$ es el tamaño de la vecindad S_i.

En [Sanvicente, 2004] se presenta también un método analítico para determinar la longitud L_i de cada iteración i, a la respectiva temperatura T_i. En este método, la longitud L_i se determina con base en una relación establecida entre la función de enfriamiento y la longitud de cada cadena de Markov. Por lo tanto, el procedimiento puede ser aplicado para determinar el mejor número de iteraciones de cualquier algoritmo, en función del esquema de enfriamiento empleado. A continuación se describe un análisis realizado por los autores de dicho artículo:

La longitud de la i-ésima cadena de Markov (L_i), debe ser $L_i > 0$, para cada ciclo i del algoritmo. Por otro lado, la temperatura T_i debe ser calculada mediante una función de enfriamiento, es decir:

$$T_{i+1} = f(T_i) \tag{3.11}$$

Donde el valor de T_i debe satisfacer la siguiente propiedad:

$$\lim_{i\to\infty} T_i = 0 \tag{3.12}$$

Es decir,

$$T_i \geq T_{i+1}\ , \quad \forall i \geq 0 \tag{3.13}$$

De la ecuación (3.13) se puede observar que, a medida que se incrementan las iteraciones, el valor de T_i tiende a 0, lo cual significa que la temperatura final del algoritmo debe ser muy cercana a 0. De manera similar, al inicio del proceso el valor de T_i debe ser tal que permita la aceptación de cualquier solución propuesta. A medida que la temperatura desciende, también disminuye la probabilidad de aceptar soluciones que no mejoran el valor de la función objetivo, por lo que cuando la temperatura está en sus niveles mas bajos, prácticamente sólo se aceptan mejores soluciones.

Dado lo anterior, es posible establecer una relación entre la longitud de la cadena de Markov y la velocidad de enfriamiento: a temperaturas altas se requieren pocas iteraciones y a temperaturas bajas se requieren más. Es decir, cuando $T_i \to \infty$, $L_i \to 0$ y cuando $T_i \to 0$, $L_i \to \infty$. Sin embargo, la longitud L_i de la cadena de Markov para cada valor de T_i debe ser tal que permita que el sistema alcance el equilibrio, es decir, se explore al máximo las soluciones vecinas para la temperatura actual, lo cual asegura que se tiene la mejor solución para esta temperatura.

Existen dos maneras de seleccionar a los elementos de una vecindad: mediante un muestreo con reemplazo de elementos o sin reemplazo. En un muestreo con reemplazo de elementos, el número máximo de soluciones diferentes que pueden ser rechazadas (cuando $T_i \to 0$) corresponde al tamaño de la vecindad de dicha solución: el tamaño de la vecindad del elemento s_i es $|Vs_i|$. Análogamente, en un muestreo sin reemplazo, la longitud máxima de la cadena de Markov puede establecerse como:

$$L_{max} = g(|V_{Si}|) \tag{3.14}$$

Donde L_{max} es la máxima longitud de la cadena de Markov (cuando $T_i = T_F$) y $g(|V_{Si}|)$ es una función que depende del tamaño de la vecindad de s_i y determina el número máximo de intentos a realizar buscando una mejor solución, en el valor mínimo de T_i. Como puede observarse, L_{max} depende únicamente del número de elementos de V_{Si} que serán explorados en T_F.

Sea $G(s_i)$ la función de generación de vecinos empleada para algún problema particular, la cual determina el número N de soluciones s_j que pueden ser generadas a partir de s_i. Por lo general, esta función puede ser definida como:

$$G(s_i) = \begin{cases} \dfrac{1}{|Vs_i|}, & \forall S_j \in Vs_i \\ 0, & \forall S_j \notin Vs_i \end{cases} \tag{3.15}$$

Cuando se emplea un esquema de vecindad con reemplazo de muestras, la probabilidad de elegir cierta s_i tomando N muestras está dada por:

$$P(s_j) = 1 - \exp^{-\frac{N}{|Vs_i|}} \tag{3.16}$$

De donde el número de muestras N puede ser obtenido como:

$$N = -|Vs_i| \ln\left(1 - P(S_j)\right) \tag{3.17}$$

Por lo tanto, para cualquier algoritmo que emplee un muestreo con reposición la longitud máxima de la cadena de Markov puede ser establecida de la siguiente manera:

$$L_{\max} = N = -|Vs_i| \ln\left(1 - P(S_j)\right) \tag{3.18}$$

$$= C \ |Vs_i|$$

Es decir, el nivel de exploración que puede ser llevada a cabo está dado por la variable C.

Para diferentes valores de C, se obtienen diferentes valores de exploración de la vecindad [Sanvicente, 2004]. Por ejemplo, para $C = 1$ se obtiene un 63% de exploración. Para los valores de $C = 2$, 3, y 4.6, se obtienen niveles de exploración de vecindad del 86%, 95% y 99%, respectivamente.

Como se puede observar, si se establece adecuadamente el valor del parámetro C, se puede obtener el valor de L_{max} que permita alcanzar el equilibrio con el nivel de exploración deseado.

Ahora bien, el valor proporcionado por L_{max} corresponde al número máximo de iteraciones a ser realizadas en el nivel más bajo de la temperatura, es decir, en T_F. Sin embargo, al principio cuando $T_i = T_1$ es suficiente con realizar un intento, pues debido a la naturaleza del algoritmo, al final de la primera iteración se aceptará cualquier solución sin importar que tan "buena" o "mala" sea. Por lo tanto, es deseable establecer una fórmula para calcular los valores de L_i desde L_1 hasta L_k para los respectivos valores de T_i. Dichos valores deben ir creciendo a medida que T_i disminuye, hasta alcanzar L_{max} cuando $T_i = T_F$.

Continuando con la función geométrica empleada en el esquema de enfriamiento, y tomando en cuenta que la función (3.10) es aplicada repetidas veces en cada iteración del algoritmo, se obtiene:

$$T_F = \alpha^n T_1 \tag{3.19}$$

Donde n es el número de iteraciones realizadas para llegar de T_1 a T_F . De manera similar, se propone una función análoga para incrementar la longitud de las cadenas de Markov:

$$L_{i+1} = \beta L_i \quad , \ \beta > 1 \tag{3.20}$$

Donde L_i y L_{i+1} son las longitudes de la cadena de Markov para T_i y T_{i+1}, respectivamente, y β es el factor de incremento progresivo en la longitud de las cadenas de Markov. De manera similar a lo realizado con las temperaturas, desde L_1 hasta L_{max} se obtiene:

$$L_{max} = \beta^n L_1 \tag{3.21}$$

Es decir, L_i se incrementa n veces desde T_1 hasta T_F.

Para obtener cuál es el valor de n, se despeja de la ecuación (3.19), obteniendo:

$$n = \frac{\ln T_1 - \ln T_F}{\ln \alpha} \tag{3.22}$$

Finalmente, de las ecuaciones (3.21) y (3.22) se obtiene el factor de incremento β en las cadenas de Markov:

$$\beta = \exp \frac{\ln L_{\max} - \ln L_1}{n} \tag{3.23}$$

Con lo anterior, es posible determinar las diferentes longitudes de L_i, para $1 < i < max$, con los respectivos valores de T_i. Nótese que esto se realiza sin necesidad de hacer demasiadas pruebas experimentales, debido a que únicamente es necesario realizar las operaciones indicadas en las fórmulas anteriores, para obtener el valor deseado del parámetro requerido.

3.5. Descripción de RSS (Recocido Simulado Sintonizado)

Con los métodos para determinar las temperaturas inicial y final, así como la longitud de las cadenas de Markov para cada iteración del algoritmo, se obtiene el algoritmo Recocido Simulado Sintonizado (RSS).

En la figura 3-2 se presenta el diagrama de flujo del algoritmo RSS. Al principio se establecen la temperatura inicial, la longitud inicial de la cadena de Markov y la solución inicial generada de manera aleatoria. La variable L representa el número de intentos realizados en el ciclo interno, acotado al final del ciclo por la longitud de la cadena de Markov L_i, determinada para esa iteración. El desempeño del algoritmo es muy similar al estándar presentado en la Figura 3-1, salvo que se realizan menos iteraciones en busca de la mejor solución para cada temperatura. Sin embargo, cuando los parámetros ya están sintonizados mediante el método presentado en las secciones 2.2 y 2.3, su desempeño resulta más eficiente. La razón de esto se explica en la sección 3.6.

En capítulos siguientes se ilustra la implementación realizada con diversas instancias del problema, tanto con las proporcionadas por el *benchmark* del PATAT como con casos reales tomados de instituciones públicas, donde podrá apreciarse la mejora en cuanto al tiempo de ejecución del algoritmo RSS respecto al RS, sin sacrificar calidad en las soluciones encontradas.

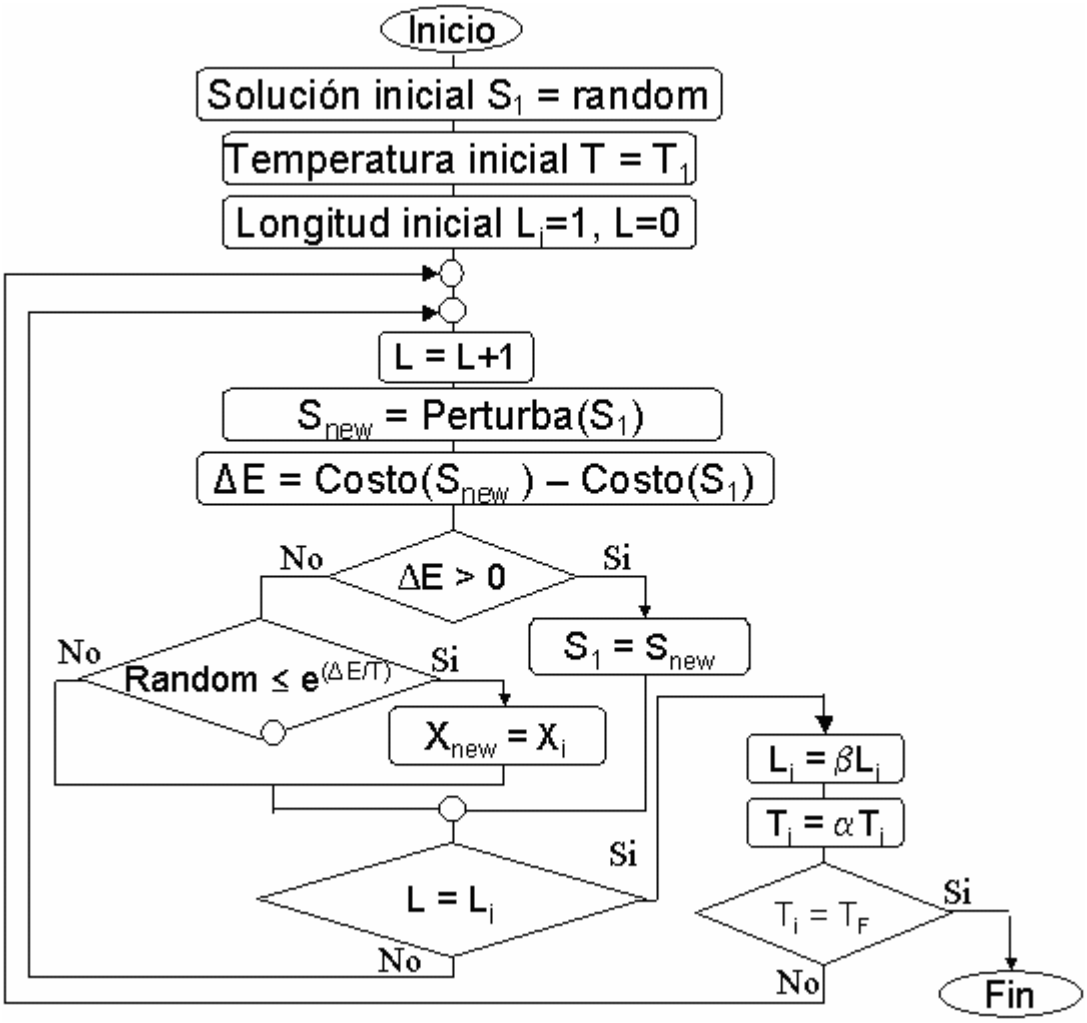

Figura 3-2. Diagrama de flujo del RSS para Maximización

3.6. Análisis de eficiencia

Con el método de sintonización explicado anteriormente, se obtiene un mejor desempeño pues el tiempo de ejecución del algoritmo RSS se reduce casi a la mitad respecto al RS, como se puede observar en las pruebas realizadas al implementar el algoritmo. La razón de ésta reducción en tiempo se explica a continuación.

En el algoritmo RS, el ciclo externo se ejecuta cierto número *m* de veces. A su vez, el ciclo interno se ejecuta otro número *n* de veces por cada ejecución del externo.

Es decir, el número total de iteraciones del algoritmo RS, sea I_{RS}, está dado por:

$$I_{RS} = n + n + n + \ldots + n = m \times n \qquad (3.24)$$

En la Tabla 3-1 se ilustra lo anterior.

Tabla 3-1. Representación de las iteraciones realizadas en el algoritmo RS

Iteración	No. Veces que se ejecuta el ciclo interno
1	n
2	n
3	n
...	...
m	n
total	m x n

En el algoritmo RSS el ciclo externo se sigue ejecutando m veces, no así el ciclo interno pues la longitud de la cadena de Markov no es fija sino aumenta progresiva y paulatinamente en función de la ecuación geométrica (3.25).

$$L_{i+1} = \beta L_i \tag{3.25}$$

Donde β es el factor de incremento en la longitud de la cadena de Markov. En la Tabla 3-2 se ilustra lo anterior.

Tabla 3-2. Representación de las iteraciones realizadas en el algoritmo RSS

Iteración	No. Veces que se ejecuta el ciclo interno
1	1
2	β
3	β^2
...	...
m	β^{m-1}
total	m x n

Es decir, el número total de iteraciones I_{RSS} del algoritmo RSS, está dado por la suma:

$$I_{RSS} = 1 + \beta + \beta^2 + \ldots + \beta^{m-1} \tag{3.26}$$

La ecuación (3.26) también puede representarse mediante la siguiente sumatoria:

$$I_{RSS} = \sum_{i=0}^{m-1} \beta^i \tag{3.27}$$

Donde (3.27) es una serie conocida, cuya fórmula está dada por [Stewart, 2002]:

$$\sum_{n=0}^{N} x^n = \frac{1 - x^{N+1}}{1 - x} \quad , \quad x \neq 1 \tag{3.28}$$

Por lo tanto, puede llegarse a:

$$I_{RSS} = \sum_{i=0}^{m-1} \beta^i = \frac{1 - \beta^m}{1 - \beta} \tag{3.29}$$

Como se observa, el valor de la serie (3.29) está en función de β y m, donde m es el número de iteraciones del ciclo externo del algoritmo, y $1 \leq \beta \leq 2$ (Ver ecuaciones (3.20) y (3.23)). Por lo tanto, para estimar la complejidad analizaremos el mejor caso, el peor caso y el caso promedio.

En el mejor caso, cuando se realiza el número mínimo de iteraciones, $\beta \to 1$ (el menor valor que puede tomar β es el más cercano a uno y el mayor es el más cercano a 2). Por lo tanto, el número de iteraciones en el mejor caso está dado por:

$$\lim_{\beta \to 1} \frac{1-\beta^m}{1-\beta} = \frac{\lim_{\beta \to 1} -1-\beta^m}{\lim_{\beta \to 1} -1-\beta} = \frac{-m\beta^{m-1}}{-1} = m\beta^{m-1}\Big|^1 = m \tag{3.30}$$

Es decir, en el mejor de los casos, cuando β toma un valor muy cercano a 1, el ciclo interno del algoritmo se ejecuta m veces (una ejecución por cada iteración del ciclo externo). Esto porque $1^2 = 1^x = 1$ para cualquier valor de x.

En el peor caso, $\beta \to 2$, el número de iteraciones es:

$$\lim_{\beta \to 2} \frac{1-\beta^m}{1-\beta} = \frac{1-2^m}{1-2} = 2^m - 1 \tag{3.31}$$

Por lo tanto, el número promedio de iteraciones está dado por:

$$I_{RSS} = (m + 2^m - 1)/2 \tag{3.32}$$

De donde se observa que cuando $m * n > I_{RSS}$, el tiempo de ejecución del algoritmo RSS es mejor que el tiempo que se tarda RS.

De lo anterior, se concluye que cuando se cumple (3.32), el tiempo disminuye.

$$\frac{m+2^m-1}{2} \quad < \quad m*n \tag{3.33}$$

$$n > \frac{m+2^m-1}{2m} \tag{3.34}$$

$$n \leq \frac{m+2^m-1}{2m} \tag{3.35}$$

Por lo tanto, si ocurre (3.33), el tiempo de RSS es mejor respecto al de RS. En caso contrario, cuando sucede (3.34), el tiempo de RSS no es menor respecto al de RS.

Lo anterior explica por qué en las ejecuciones del algoritmo el tiempo se reduce aproximadamente a la mitad cuando se aplica el método para sintonizar la Cadena de Markov, sin que la longitud máxima exceda cierto parámetro. Además, al aplicar la sintonización de temperaturas, el tiempo de ejecución se reduce aún más con el precio de perder un pequeño porcentaje en calidad.

La reducción en tiempo y variación en calidad se puede apreciar mejor con los resultados obtenidos al realizar la implementación, lo cual se describe en los siguientes capítulos.

3.7. Resumen del capítulo

En este capítulo se ha presentado el algoritmo RS clásico, así como las modificaciones realizadas para mejorar su desempeño. Como ha podido observarse, el problema de la sintonización de parámetros no resulta trivial e incluso en la literatura se han publicado diversos intentos por encontrar mejores maneras de establecer los valores adecuados.

Con el método que se ha presentado aquí se soluciona una de las principales desventajas consideradas propias de la metaheurística de Recocido Simulado: el tiempo de desempeño. Lo más relevante es que esto se logra plantear de manera general, independientemente del problema que se desee resolver. El buen desempeño del método expuesto se puede observar en los siguientes capítulos en los que se presenta la aplicación del mismo a un problema genérico de carácter teórico y a un problema particular de una institución educativa de nivel superior.

Capítulo 4. Aplicación de RS y RSS al problema del PATAT

Debido a la gran variedad de restricciones que se pueden presentar en el problema ETT, dependiendo del tipo de institución a la que corresponda, ha surgido la inquietud de homogeneizar, hasta donde sea posible, la especificación del problema. En este capítulo se presenta un caso general propuesto por el PATAT. Se presenta la aplicación del método planteado en el capítulo 3, sección 3.4 a 3-6, para resolver las instancias publicadas por el PATAT, así como las soluciones obtenidas para el mismo con los algoritmos RS y RSS, implementados en este trabajo.

4.1. Descripción del PATAT

El PATAT es una organización internacional dedicada a la investigación del problema Timetabling automatizado, la cual organiza un foro bi-anual al que asisten investigadores y especialistas de dicho problema. Las conferencias que ahí se realizan sirven como punto central para el intercambio de ideas en una comunidad internacional de investigadores sobre diversos aspectos de la generación de horarios asistida por computadora.

El PATAT forma parte del ASAP (*Automated Scheduling, Optimization and Planning*) un grupo cuyo objetivo principal es dirigir investigaciones en modelos, heurísticas y algoritmos para producir automáticamente soluciones de alta calidad para una variedad de problemas de optimización y calendarización en el mundo real, tales como Scheduling, Timetabling, logística, asignación de espacios y ajuste de inventarios [PATAT, 2005].

Los temas de las conferencias incluyen:

- Timetabling deportivo.
- Educational Timetabling.
- Timetabling en transporte.
- Timetabling personal.
- Timetabling en sistemas distribuidos.
- Estudio de casos.
- Paquetes comerciales.
- Timetabling interactivo vs. automático.
- Actualización de horarios.
- Relaciones con otros Problemas de Scheduling.
- Áreas de investigación en Timetabling, tales como:
    - Métodos basados en restricciones.
    - Computación evolutiva.
    - Inteligencia artificial.
    - Sistemas basados en conocimiento.
    - Investigación de Operaciones.
    - Programación matemática.
    - Recocido Simulado.
    - Coloreo de Grafos.
    - Sistemas expertos.
    - Heurísticas de Búsqueda.
    - Búsqueda local.
    - Búsqueda tabú.
    - Algoritmos genéticos.

Con el fin de asegurar la confiabilidad de los resultados obtenidos en este trabajo de tesis, se utilizan instancias del *benchmark* del PATAT para realizar las pruebas y el análisis comparativo presentado en este capítulo.

4.2. Definición del problema

El problema particular en cuestión consiste de un conjunto E de eventos[13] que deben ser programados en 45 períodos de tiempo [14] (5 días de 9 intervalos cada uno), un conjunto A de salones en los cuales pueden efectuarse los eventos, un conjunto S de estudiantes que asisten a los eventos, y un conjunto F de características[15], satisfechas por los salones y requeridas para los eventos.

El problema del PATAT considera que cada estudiante asiste a cierto número de eventos y cada salón tiene una capacidad determinada. Un horario factible es aquél en el cual todos

[13] Se considera evento a cualquier asignatura, taller, curso, etc., que deba ser calendarizado.
[14] Un período es un intervalo de tiempo disponible para impartir un evento.
[15] Las características son aquellos elementos que se requieren para ciertos eventos. Pueden ser elementos audiovisuales, condiciones especiales, etc.

los eventos han sido asignados en un período de tiempo y en un salón de tal forma que se satisfacen las siguientes restricciones duras:

- Ningún estudiante asiste a más de un evento a la vez.
- El salón es lo suficientemente grande para todos los estudiantes asistentes y satisface todas las características requeridas por el evento.
- Solo hay un evento en cada salón, por cada período de tiempo.

Además, es deseable que se satisfagan las siguientes restricciones suaves:

- Un estudiante tiene un evento programado en el último período del día.
- Un estudiante tiene más de dos clases consecutivas.
- Un estudiante tiene una única clase en un día.

Por lo tanto, las soluciones obtenidas con el algoritmo RS implementado deben satisfacer todas las restricciones duras anteriores y minimizar lo más que se pueda el número de restricciones suaves insatisfechas.

4.3. Instancias del problema

En [Socha, 2002] se emplea un generador de instancias para producir casos con diferentes características, para distintos valores de los parámetros dados (número de estudiantes, de eventos, de salones y de características). Todas las instancias producidas tienen una solución perfecta, es decir, es posible encontrar al menos una solución en la que ninguna restricción (duras o suaves) sea violada.

En la Tabla 4-1 se presentan las características de las instancias proporcionadas por el PATAT y la Tabla 4-2 contiene los criterios para clasificar dichas instancias, según Socha.

Tabla 4-1. Benchmark del PATAT

Caso	Eventos	Salones	Características	Estudiantes
C01	400	10	10	200
C02	400	10	10	200
C03	400	10	10	200
C04	400	10	5	300
C05	350	10	10	300
C06	350	10	5	300
C07	350	10	5	350
C08	400	10	5	250
C09	440	11	6	220
C10	400	10	5	200
C11	400	10	6	220
C12	400	10	5	200
C13	400	10	6	250
C14	350	10	5	350
C15	350	10	10	300
C16	440	11	6	220
C17	350	10	10	300
C18	400	10	10	200
C19	400	10	5	300
C20	350	10	5	300

Tabla 4-2. Clasificación de instancias publicada por Socha [Idem].

Caso	Eventos	Salones	Características	Estudiantes
small	100	5	5	80
medium	400	10	5	200
large	400	10	10	400

Para las instancias de la competencia del PATAT, cada caso CompetitionX se ha renombrado como Cx, es decir, Competition01 se presenta como C01, Competition02 como C02, y así sucesivamente. En el caso de las instancias de Socha, los cinco casos small posteriormente se renombran como S1, S2, S3, S4 y S5, respectivamente. Los casos medium son M1, M2, M3, M4 y M5. Y los casos large son L1 y L2.

4.4. Modelación matemática del problema

Para formular el modelo matemático del problema ETT del PATAT, se requiere definir formalmente los elementos descritos en la sección 4.3.

Se cuenta con los siguientes elementos y conjuntos de elementos:

- n Eventos: $E = \{e_1, e_2, \dots, e_n\}$
- m Estudiantes: $U = \{u_1, u_2, \dots, u_m\}$
- 45 Períodos: $P = \{p_1, p_2, \dots, p_{45}\}$
- r Aulas o salones: $A = \{a_1, a_2, \dots, a_r\}$
- r Capacidades de aulas: $C = \{c_1, c_2, \dots, c_r\}$
- t Características: $F = \{f_1, f_2, \dots, f_t\}$

También se cuenta con 3 matrices binarias, que contienen lo siguiente:

- Eventos a los que asiste cada estudiante: D_{nxm}

$$D_{nxm} = \begin{bmatrix} d_{11} & d_{12} \dots d_{1n} \\ d_{21} & d_{22} \dots d_{2n} \\ \dots & \dots \dots \dots \\ d_{m1} & d_{m2} \dots d_{mn} \end{bmatrix}$$

Donde $d_{il} = 1$ si el estudiante l asiste al evento i, y 0 si no.

- Características que poseen los salones: S_{txr}

$$S_{txr} = \begin{bmatrix} s_{11} & s_{12} \dots s_{1t} \\ s_{21} & s_{22} \dots s_{2t} \\ \dots & \dots \dots \dots \\ s_{r1} & s_{r2} \dots s_{rt} \end{bmatrix}$$

Donde $s_{jf} = 1$ si el salón j posee la característica f, y 0 si no.

- Características que requieren los eventos: Q_{txn}

$$Q_{txn} = \begin{bmatrix} q_{11} & q_{12} \dots q_{1t} \\ q_{21} & q_{22} \dots q_{2t} \\ \dots & \dots \dots \dots \\ q_{n1} & q_{n2} \dots q_{nt} \end{bmatrix}$$

Donde $q_{if} = 1$ si el evento i requiere la característica f, y 0 si no.

Para modelar matemáticamente las restricciones planteadas, sea la variable x_{ijk} la representación del evento *i*, asignado al salón *j* en el período *k*, cuyo valor será 1 cuando exista tal asignación y 0 en caso contrario. Es decir, cuando la variable $x_{231} = 1$ significa que el evento 2 se imparte en el salón 3, período 1.

Es decir, tal como se plantea en [Gómez, 2005], el problema consiste en minimizar:

$$\begin{aligned} & z = Scv \\ & sujeto \quad a \\ & Hcv \end{aligned} \tag{4.1}$$

Donde S_{CV} es el número de restricciones suaves violadas y H_{CV} es el número de restricciones duras violadas.

Considerando lo anterior, el problema de asignación de horarios estará resuelto cuando se satisfagan todas aquellas restricciones duras planteadas y se cumplan el mayor número posible de restricciones suaves.

4.4.1. Restricciones duras

Las restricciones duras del problema ETT del PATAT son las siguientes:

1. ***Ningún estudiante asiste a más de un evento a la vez.***

   Es decir que, en cualquier período *k*, a lo más se debe programar un evento $i \in Q_e$. Donde Q_e es el conjunto de eventos que incluye al evento *e* y a todos aquellos eventos que se encuentran en conflicto con el evento *e*. Esta restricción se expresa en (4.2)

$$\sum_{i \in Q_e} x_{ijk} \leq 1 \qquad j = 1,...,r \quad k = 1,...,45 \quad e = 1,...,n \tag{4.2}$$

   Donde la variable *i* corresponde al evento, *j* al salón y *k* al período.

2. ***El salón es lo suficientemente grande para todos los estudiantes asistentes y satisface todas las características requeridas por el evento.***

   Esta restricción puede ser dividida en dos:

   - La capacidad del salón debe ser lo suficientemente grande para albergar a todos los asistentes, y
   - El salón debe satisfacer todas las características requeridas por el evento.

   La primera restricción, dicha de otra forma es: Para cada uno de los salones *j* en los que se programa el evento *i*, la capacidad del salón *j* (c_j) debe ser mayor o igual al número de asistentes al evento *i*. Es decir, sea b_i el número de asistentes para el evento *i*,

$$b_i = \sum_{l=1}^{m} d_{li} \quad i = 1,...,n \tag{4.3}$$

Donde la variable *i* corresponde al evento, y la variable *l* a los estudiantes, d_{li} es 1 si el estudiante *l* asiste al evento *i*.

La primera parte de la restricción 2 puede expresarse como:

$$\forall x_{ijk} = 1, \quad b_i \le c_j, \qquad i = 1,...,n \quad j = 1,...,r \quad k = 1,...,45 \tag{4.4}$$

La segunda parte de dicha restricción puede ser interpretada como sigue: para cada uno de los *k* periodos, el salón *j* debe cumplir todos los requerimientos de cualquier evento *i* programado en dicho salón. Esta restricción se expresa como:

$$x_{ijk} = 1 \quad \Rightarrow \quad \forall f \; q_{if} \le s_{jf}, \qquad i = 1,...n \quad j = 1,...,r \quad k = 1,...,45 \tag{4.5}$$

Donde q_{if} representa la característica *f* asociada al evento *i*, y s_{jf} representa la característica *f* satisfecha por el salón *j*.

3. Solo hay un evento en cada salón, por cada período de tiempo.

Esto significa que en un período determinado *k*, un evento *i* cualquiera sólo puede ser programado a lo más en un salón *j*. Esta restricción es expresada en:

$$\sum_{j=1}^{r} x_{ijk} \le 1 \quad i = 1,...,n \quad k = 1,...,45 \tag{4.6}$$

Además, existe otra restricción dura implícita en la definición del problema pues resulta indispensable que se cumpla lo siguiente:

- ***Todos los eventos deben estar programados en algún período.***

Lo anterior significa que considerando los 45 períodos, todos los eventos *i* deben estar programados exactamente 1 vez. Es decir:

$$\sum_{k=1}^{45} \sum_{j=1}^{r} x_{ijk} = 1 \quad i = 1,...,n \tag{4.7}$$

4.4.2. Restricciones suaves

Las restricciones suaves son las siguientes:

1. Un estudiante no debe tener un evento programado en el último período del día.
Para poder especificar esta restricción, se requiere definir un conjunto *V* que contiene a los elementos correspondientes al último período del día:

$$V = \{k \mid k \bmod NUMPER = 0\}$$

Donde *k* = 1, 2, ... , 45 y *NUMPER* = Número de períodos por día, en este caso 9.

- Por lo tanto, la restricción se puede interpretar como sigue: no se debe programar algún evento *i*, programado en un salón cualquiera *j*, en un periodo $k \in$ V. La restricción es mostrada en:

$$\forall k \in V \quad \sum_{i=1}^{n} \sum_{j=1}^{r} x_{ijk} = 0 \tag{4.8}$$

2. Un estudiante no debe tener más de dos clases consecutivas.

Es decir, no es deseable que un estudiante tenga programadas 3 o más clases consecutivas. El hecho de que un participante *l* tenga programados más de dos eventos *j* en *k* periodos consecutivos aparece en (4.9).

Sea S_l el conjunto de eventos a los que debe asistir el estudiante *l*.

$$\forall\, v \in V \qquad \sum_{k=v-NUMPER+1}^{v-2} \sum_{i \in S_l} \sum_{j=1}^{r} x_{ij(k+1)} x_{ij(k+2)} x_{ij(k+3)} = 0 \qquad l = 1,2,...,m \tag{4.9}$$

3. Un estudiante no debe tener una única clase en un día.

Todos los estudiantes deben tener programado cero o más de un evento $i \in S_l$ en un día particular. La restricción está representada en:

$$\forall v \in V \qquad \sum_{k=v-NUMPER+1}^{v-2} \sum_{i \in S_l} \sum_{j=1}^{r} x_{ijk} \neq 1 \qquad l = 1,...,m \tag{4.10}$$

4.5. Implementación y resultados obtenidos

La implementación del algoritmo para resolver el problema ETT del PATAT parte de una solución factible inicial, proporcionada por un algoritmo determinístico propio (del tipo Ramificación y acotamiento) que para encontrar dicha solución considera el evento "más restringido", el salón "más restringido" y la actualización de disponibilidades cada vez que se realiza una asignación. La existencia de esta solución factible está asegurada por la especificación de las instancias del problema. En la Figura 4-1 se presenta el pseudocódigo del algoritmo programado.

Para elegir el evento más restringido se utiliza el concepto de saturación que consiste en lo siguiente: los eventos se ordenan de mayor a menor según su nivel de restricción. Un evento e_1 es más restringido que otro e_2 cuando el número de períodos disponibles para que el evento e_1 sea asignado sea menor que el de e_2. En caso de ocurrir un empate, el segundo criterio a considerar es la disponibilidad de lugares.

Un lugar se define como una combinación hora-salón y se considera disponible para un evento dado cuando el salón se encuentre desocupado a la hora indicada, y además, su asignación a dicho evento no viole alguna de las restricciones duras del problema. El evento E1 estará más restringido que E2 si tiene menos lugares disponibles. En caso de prevalecer el empate después de aplicar ambos criterios, se considera más restringido aquél evento que tenga un mayor número de eventos en conflicto con él mismo, siendo los eventos en conflicto aquellos pares de eventos que no pueden ser programados al mismo tiempo debido a que tienen uno o más estudiantes en común. Si el empate continúa, se considera el número de estudiantes que asisten a cada evento y se considera como mayor al evento que tenga mayor número de estudiantes. En caso de que aún después de aplicar todos los criterios anteriores el empate persista, ambos eventos se consideran con el mismo nivel de restricción y se ordenan según su aparición.

```
1 Inicio
2 Itera = 0, ban_factible = False
3 Lista_conflictos = NULL
4 Hacer
5  Contador = 0
6  L = todos los eventos;
7  Si Itera = 0
8    Asignar todos los eventos que tengan sólo un salón válido
9    Actualizar lista L de eventos E
10    Mientras Lista_conflictos <> Ø
11     Tomar primer elemento de Lista_conflictos
12      Asignar salón y período a ese evento
13      Actualizar la lista L
14      Lista_conflictos = Lista_conflictos – E
15    Fin Mientras
16    Mientras Contador < número_de_eventos
17      Ordenar eventos sin asignar según su grado de "restricción"
18      Seleccionar el evento mayor E
19      L = L – E
20      Si existen lugares disponibles para asignar el evento conservando
21      la factibilidad
22          Elegir el más restringido
23      Si no
24          ban_factible = False
25          Asignar al evento un salón y un periodo disponible (aunque
26          viole restricciones duras)
27          Lista_conflictos = Lista_conflictos + E
28      Finsi
29      Contador = Contador + 1
30    Fin mientras
31    Itera = Itera + 1
32 Mientras Itera < 10 y ban_factible == False
33 Fin
```

Figura 4-1. Pseudocódigo de la solución factible inicial

El proceso para seleccionar el salón más restringido parte de la selección del evento más restringido y consiste en buscar primero los lugares (combinación hora-salón) candidatos para asignar a ese evento. Para cada posible lugar, se calcula el número de eventos que se pueden asignar allí (sin contar el actual ni los que han sido previamente asignados) sin que se viole alguna de las restricciones duras del problema. Aquél lugar que tenga el menor número de eventos candidatos a ser asignados, será el lugar seleccionado para el evento en cuestión. En caso de haber un empate se queda con el primero que encontró. Una vez realizadas las dos selecciones anteriores, y después de asignar un evento a un salón y una hora particular, se actualiza el número de períodos y lugares disponibles para las demás asignaciones.

En el algoritmo para encontrar una solución factible inicial, *L* es la lista de eventos que aún no han sido asignados. *Lista_conflictos* es una lista donde se tienen eventos que no pudieron asignarse sin violar ninguna restricción dura. Un período es un intervalo de tiempo disponible para programar un evento.

La tenencia es un número aleatorio entre 1 y 20, que corresponde al número de iteraciones. El criterio de paro es encontrar el óptimo (0 duras y 0 suaves) o en su defecto, realizar un

número máximo de iteraciones (400, en este caso) sin que mejore la restricción global. Todos estos valores se obtuvieron después de realizar varias pruebas en la ejecución y observar el comportamiento de la solución obtenida.

4.5.1. Implementación de Recocido Simulado (RS)

En la implementación de Recocido Simulado para el problema ETT del PATAT, se consideran como parámetros importantes: el espacio de búsqueda sobre el cual trabajar, el criterio de vecindad para encontrar nuevas soluciones, la función de costo para decidir la bondad de una solución, así como el esquema de enfriamiento que contempla la temperatura inicial, la temperatura final y el factor de decremento.

El espacio de búsqueda está enfocado a las soluciones factibles del problema. Una vez que se obtiene una solución inicial factible, el algoritmo trabaja únicamente con aquellas asignaciones que permanezcan dentro de la región factible del problema. Esto se realiza utilizando el algoritmo descrito en la sección 4.5.

El criterio de vecindad que se aplica para generar las nuevas soluciones consiste en seleccionar aleatoriamente dos períodos y dos salones, si al intercambiar la asignación de los eventos correspondientes a esos lugares (salón-período) no se genera una infactibilidad, se toman esos períodos y se realiza el intercambio. De lo contrario se vuelven a generar dos períodos y dos salones de forma aleatoria, hasta encontrar una combinación que proporcione un intercambio factible.

La función de costo para evaluar la bondad o energía de una solución consiste en sumar el costo de las restricciones suaves más las restricciones duras violadas. El costo asignado a cada restricción dura es de 1000 y el de cada suave es de 1. Sin embargo, en realidad este costo se reduce al número de restricciones suaves violadas, debido a la naturaleza del espacio de búsqueda descrito previamente en el capítulo 3, sección 3.3.

La temperatura inicial del algoritmo se obtiene con la fórmula:

$$T_I = 470n + e \qquad (4.11)$$

Donde n es el número de estudiantes y e es el número de eventos del problema. Esta fórmula se obtuvo considerando las tres restricciones duras y las tres restricciones suaves dadas en la definición del problema. La temperatura final considerada fue $T_F = 0.01$, por ser un valor lo suficientemente pequeño. En cada iteración, la temperatura se decrementa geométricamente según la función (3.1), con $\alpha = 0.95$. Para el criterio de equilibrio se considera un número máximo de iteraciones igual a 10,000. Estos parámetros fueron seleccionados después de realizar varias pruebas y observar que el mejor desempeño se obtenía con éstos valores.

En [Burke, 2003] se realiza una comparación entre las heurísticas Búsqueda Tabú (HH), el Algoritmo Hormiga (ANT) y un método de Búsqueda Local (RRLS). La Tabla 4-3 presenta los resultados obtenidos en dicho trabajo, los valores de la tabla corresponden al número de restricciones violadas. En la columna HH se listan los mejores valores obtenidos, tomados

de 5 ejecuciones realizadas con la heurística de Búsqueda Tabú implementada. En la columna RRLS la palabra INFAC significa que la solución encontrada por el algoritmo de Búsqueda Local no satisface todas las restricciones duras del problema, en 40 ejecuciones.

Como puede observarse en la Tabla 4-3, la mayoría de los resultados que se publican en dicho trabajo son superados por la primera ejecución de cada instancia con el algoritmo RS presentado en el presente trabajo. El mejor resultado para cada caso se resalta en negritas.

```
1 Inicio
2  Leer archivo
3  Generar solución factible inicial
4  costoIni = cuenta violaciones()
5  mejorCosto = costoIni
6  temper = 470*num_students + num_eventos
7  TEMPFIN = 0.01
8  ALFA = 0.09
9   Mientras (temper > TEMPFIN)
10   Iter = 0
11   Si (mejorCosto > costoIni)
12    Reemplazar mejorSolución por soluciónActual
13    Reemplazar mejorCosto por costoIni
14   Fin si
15   Mientras (iter < MAXNUMITER)
16    Buscar otra soluciónActual que sea factible
17    costoNew = cuenta violaciones
18    costoDif = costoNew – costoIni
19    Si (costoNew <= 0)
20          costoIni = costoNew
21    sino
22     si (random() < exp(-costoDif/temper))
23          costoIni = costoNew
24     sino
25          Retomar solución anterior
26     Fin si
27    Fin si
28    iter = iter + 1
29   Fin Mientras
30    temper = temper * ALFA
31 Fin Mientras
32 Imprimir la mejor solución
33 Fin
```

Figura 4-2. Pseudocódigo del algoritmo RS

En la Tabla 4-3 se exponen los resultados obtenidos empleando el algoritmo cuyo pseudocódigo se presenta en la Figura 4-2, ejecutado en el sistema operativo Windows XP, en una computadora portátil Toshiba Satellite, con un procesador Intel Celeron a 2.4 GHz con 512 MB de Memoria RAM.

Tabla 4-3. Número de restricciones violadas por RS en las instancias propuestas por Burke.

Caso	HH	RRLS	ANT	RS
S1	**1**	8	**1**	**1**
S2	**2**	11	3	10
S3	**0**	8	1	1
S4	**1**	**7**	**1**	**1**
S5	**0**	**5**	**0**	82
M1	146	199	195	**126**
M2	173	202.5	184	**161**
M3	267	INFAC	248	**149**
M4	169	177.5	164.5	**105**
M5	303	INFAC	219.5	**72**
L1	1166	INFAC	851.5	**753**
L2	*	I*	*	**870**

* No se encontraron resultados publicados

En la Tabla 4-4 se presentan los resultados reportados en el concurso internacional PATAT 2004 (lugares del 1º al 9º), así como los obtenidos con el algoritmo RS implementado en el presente trabajo con α=0.95. Los números internos de la tabla corresponden al número de restricciones suaves violadas por el algoritmo implementado en cada caso.

Tabla 4-4. Comparación de los resultados obtenidos de RS con los del concurso PATAT 2004

Caso	1º	2º	3º	4º	5º	6º	7º	8º	9º	RS
C01	45	61	85	63	132	148	178	211	257	153
C02	25	39	42	46	92	101	103	128	112	157
C03	65	77	84	96	170	162	156	213	266	198
C04	115	160	119	166	265	350	399	408	441	400
C05	102	161	77	203	257	412	336	312	299	504
C06	13	42	6	92	133	246	246	169	209	187
C07	44	52	12	118	177	228	225	281	99	289
C08	29	54	32	66	134	125	210	214	194	37
C09	17	50	184	51	139	126	154	164	175	130
C10	61	72	90	81	148	147	153	222	308	125
C11	44	53	73	65	135	144	169	196	273	153
C12	107	110	79	119	290	182	219	282	242	231
C13	78	109	91	160	251	192	248	315	364	198
C14	52	93	36	197	230	316	267	345	156	232
C15	24	62	27	114	140	209	235	185	95	321
C16	22	34	300	38	114	121	132	185	171	113
C17	86	114	79	212	186	327	313	409	148	418
C18	31	38	39	40	87	98	107	153	117	95
C19	44	128	86	185	256	325	309	334	414	172
C20	7	26	0	17	94	185	185	149	113	176

El criterio que emplea el PATAT para determinar al ganador del concurso se basa en la fórmula dada por:

$$F_i = (x_i - b_i)/(w_i - b_i) \qquad (4.1)$$

Donde i es la instancia del problema, $1 < i < 20$; x es el número de restricciones suaves violadas por el participante; b es el número de restricciones suaves violadas en la mejor solución obtenida por todos los algoritmos y w es el número de restricciones suaves violadas en la peor solución de los algoritmos participantes.

Para cada instancia *i* del problema se calcula F_i (un valor real entre 0 y 1) y el valor *F* del algoritmo será la suma de las F_i obtenidas. El ganador será el algoritmo que obtenga la menor suma de las F_i.

En cuanto a la calidad de las soluciones obtenidas, el RS implementado en el presente trabajo quedaría en 7º lugar, como puede observarse en la Tabla 4-5, la cual contiene los cálculos realizados en la aplicación del criterio de evaluación. Las columnas ***w*** y ***b*** contienen los valores w_i y b_i de la fórmula (4.1). Las siguientes columnas corresponden a los mejores resultados reportados en el concurso, lugares del 1º al 9º, listados en ese orden. Las celdas internas contienen los valores que resultan de aplicar la fórmula dada en (4.1) para cada instancia F_i del problema. La última fila de la tabla contiene los lugares que se obtienen después de aplicar la sumatoria indicada en dicha fórmula para cada algoritmo participante.

Tabla 4-5. Aplicación del criterio de evaluación del PATAT.

Caso	**w**	**b**	**1º**	**2º**	**3º**	**4º**	**5º**	**6º**	**7º**	**8º**	**9º**	**RS**
C01	257	45	-	0.08	0.19	0.08	0.41	0.49	0.63	0.78	1.00	0.51
C02	157	25	-	0.11	0.13	0.16	0.51	0.58	0.59	0.78	0.66	1.00
C03	266	65	-	0.06	0.09	0.15	0.52	0.48	0.45	0.74	1.00	0.66
C04	441	115	-	0.14	0.01	0.16	0.46	0.72	0.87	0.90	1.00	0.87
C05	504	77	0.06	0.20	-	0.30	0.42	0.78	0.61	0.55	0.52	1.00
C06	246	6	0.03	0.15	-	0.36	0.53	1.00	1.00	0.68	0.85	0.75
C07	289	12	0.12	0.14	-	0.38	0.60	0.78	0.77	0.97	0.31	1.00
C08	214	29	-	0.14	0.02	0.20	0.57	0.52	0.98	1.00	0.89	0.04
C09	184	17	-	0.20	1.00	0.20	0.73	0.65	0.82	0.88	0.95	0.68
C10	308	61	-	0.04	0.12	0.08	0.35	0.35	0.37	0.65	1.00	0.26
C11	273	44	-	0.04	0.13	0.09	0.40	0.44	0.55	0.66	1.00	0.48
C12	290	79	0.13	0.15	-	0.19	1.00	0.49	0.66	0.96	0.77	0.72
C13	364	78	-	0.11	0.05	0.29	0.60	0.40	0.59	0.83	1.00	0.42
C14	345	36	0.05	0.18	-	0.52	0.63	0.91	0.75	1.00	0.39	0.63
C15	321	24	-	0.13	0.01	0.30	0.39	0.62	0.71	0.54	0.24	1.00
C16	300	22	-	0.04	1.00	0.06	0.33	0.36	0.40	0.59	0.54	0.33
C17	418	79	0.02	0.10	-	0.39	0.32	0.73	0.69	0.97	0.20	1.00
C18	153	31	-	0.06	0.07	0.07	0.46	0.55	0.62	1.00	0.70	0.52
C19	414	44	-	0.23	0.11	0.38	0.57	0.76	0.72	0.78	1.00	0.35
C20	185	0	0.04	0.14	-	0.09	0.51	1.00	1.00	0.81	0.61	0.95
Σ			0.446	2.426	2.919	4.464	10.3	12.6	13.78	16.08	14.63	13.18
Lugar			**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**9**	**7**

En la Tabla 4-6 se muestran los resultados obtenidos con el algoritmo RS implementado en el presente trabajo, para las veinte instancias del PATAT, con diferentes valores de α. Los mejores valores obtenidos en cuanto la a calidad se resaltan con fuente en negritas. En cuanto al tiempo, los mejores valores se obtienen con $\alpha = 0.75$.

Tabla 4-6. Resultados obtenidos con el RS para las instancias del PATAT.

	No. Restricciones suaves violadas					*Tiempo (segs)*				
Caso	α=0.95	α=0.9	α=0.85	α=0.8	α=0.75	α=0.95	α=0.9	α=0.85	α=0.8	α=0.75
C01	**153**	203	208	179	219	4334	2109	1373	987	**786**
C02	157	151	**150**	170	201	4231	2024	1428	1008	**730**
C03	**198**	233	269	271	281	4398	2072	1376	1025	**739**
C04	**400**	463	467	448	564	4906	2478	1725	1068	**852**
C05	**504**	511	520	554	643	4380	2328	1552	1091	**787**
C06	**187**	245	264	324	271	4435	2214	1525	1096	**771**
C07	**289**	294	325	438	426	4834	2396	1625	1066	**871**
C08	**37**	64	89	93	119	4462	2255	1506	1042	**795**
C09	130	**126**	131	167	153	4841	2358	1589	1171	**871**
C10	**125**	169	188	194	169	4306	2190	1439	1007	**753**
C11	**153**	158	186	185	240	4253	2201	1466	1002	**892**
C12	231	**229**	276	282	280	4006	2129	1362	927	**807**
C13	**198**	217	242	259	272	4446	2410	1408	1014	**897**
C14	**232**	250	278	337	317	4431	2308	1452	1086	**889**
C15	321	**310**	319	364	360	4013	2127	1349	995	**821**
C16	**113**	137	131	163	140	4698	3037	1551	1158	**917**
C17	**418**	427	420	465	474	4479	2269	1388	1045	**847**
C18	**95**	134	142	141	171	4971	2207	1364	1052	**840**
C19	172	**147**	187	220	248	5675	2640	1638	1213	**911**
C20	**176**	196	258	297	307	4465	2175	1475	1051	**899**

En la Tabla 4-6 se observa que para valores de α más cercanos a 1, el tiempo de ejecución del algoritmo aumenta drásticamente. Por ejemplo, para α=0.95 todas las instancias tomaron más de 4000 segundos de ejecución, tiempo que se reduce a la mitad por cada disminución en 0.05 del valor de α. En cuanto a la calidad de los resultados, se observa una ligera mejora para diferentes valores de α; sin embargo, es notorio que la columna correspondiente a α=0.95 es la que reporta la mayoría de los mejores resultados.

4.5.2. Implementación de Recocido Simulado Sintonizado (RSS)

Con el algoritmo presentado anteriormente (Figura 4-2) se obtienen buenas soluciones aunque el tiempo que consume en la ejecución podría mejorarse. Por lo tanto, se emplea el método de sintonización, propuesto en [Sanvicente, 2004].

Retomando lo definido en las ecuaciones (3.3) y (3.4), puede observarse que ambos deterioros (máximo y mínimo) están en función del esquema de vecindad definido para cada problema, sin embargo, su obtención se realiza de manera general: considerando los casos "peor" y "mejor", respectivamente, que pueden presentarse entre dos soluciones vecinas.

En este caso, el criterio de vecindad consiste de intercambiar la programación de dos eventos (salón y período), por lo que, el deterioro máximo es producido por el máximo número de estudiantes que asisten a un evento (*max_num_estXevto*), y como esto puede afectar simultáneamente a 5 de las 6 restricciones totales, entonces se obtiene (4.2).

$$\Delta Z_{Vmax} = 5 * max\_num\_estXevto \tag{4.2}$$

De manera análoga, el deterioro mínimo es nulo, pues es posible que al realizar un intercambio, los eventos afectados queden en condiciones similares, por lo que no se produce cambio alguno, es decir:

$$\Delta Z_{Vmin} = 0 \tag{4.3}$$

Ahora bien, $P_A(\Delta Z_{Vmax})$ y $P_A(\Delta Z_{Vmin})$ corresponden a la probabilidad de aceptar la solución producida por el máximo y mínimo deterioro, respectivamente. En este caso, se considera deseable una precisión de 0.95 y 0.05, respectivamente.

Por lo tanto, de la ecuación (3.7) se obtiene la fórmula para T_1 y de la ecuación (3.9) se obtiene T_F:

$$T_1 = \frac{-\Delta Z_{V\max}}{\ln\left(P_A(\Delta Z_{V\max})\right)} \tag{4.4a}$$

$$T_1 = \frac{-5 * Max\_num\_estXevt}{\ln(0.95)} \tag{4.4b}$$

$$T_F = \frac{-\Delta Z_{V\min}}{\ln\left(P_A(\Delta Z_{V\min})\right)} \tag{4.5a}$$

$$T_F = \frac{-0}{\ln(0.05)} = 0 \tag{4.5b}$$

La ecuación (4.5b) establece que la T_F debe ser cero. Sin embargo, se sabe que un valor T_F igual a cero resulta indeseable pues conduce a invertir demasiado tiempo al final de la ejecución del algoritmo. Por lo tanto, el valor de T_F se conserva en T_F =0.01, valor obtenido en la sintonización manual del algoritmo.

Por otro lado, para la sintonización del número de iteraciones realizadas en cada temperatura (cadena de Markov), se sustituyen en la ecuación (4.6) los valores obtenidos en (4.4b) y (4.5b), resultando un valor *n* diferente para cada instancia del problema, según se aprecia en:

$$n = \frac{\ln T_1 - \ln T_F}{\ln \alpha} \tag{4.6}$$

Para obtener el factor de incremento β en la longitud de las cadenas de Markov, se requiere considerar que la primera longitud de la cadena de Markov es $L_1 = 1$ y también es necesario calcular L_{max}. Para esto último es deseable un nivel de exploración del 95% de la vecindad, es decir, $C = 3$. Además, el tamaño de la vecindad está dado por el número total de eventos de la instancia (num_eventos) multiplicado por ese mismo número menos uno.

Por lo tanto, de (3.18) se obtiene:

$$L_{max} = C\,|V_{Si}| = 3 * num\_eventos*(num\_eventos - 1) \tag{4.7}$$

Por lo tanto, de las ecuaciones anteriores es posible obtener el valor de β sustituyendo en la ecuación (3.23), que para comodidad se repite a continuación:

$$\beta = \exp \frac{\ln L_{\max} - \ln L_1}{n} \tag{4.8}$$

Lo anterior (ecuaciones 4.2 a 4.8) se presenta en [Sanvicente, 2004] como un método general aplicable a cualquier problema que se desee resolver, únicamente calculando de manera analítica los parámetros T_I, T_F, n y β, con las fórmulas anteriores. Dicho método ha sido aplicado exitosamente en problemas tales como la determinación del diámetro de tuberías [Sanvicente, 2004] y en la solución de instancias 3-SAT [Froilán, 2005]. Sin embargo, para el problema ETT del PATAT se encontraron algunos inconvenientes, listados a continuación y que fueron superados con ligeras modificaciones al método:

- En la implementación del algoritmo RSS, se observó que en todas las instancias del problema del PATAT, el deterioro mínimo es 0, es decir, con el criterio de vecindad descrito en la sección 4.5.1, el mínimo cambio producido entre dos soluciones es nulo, lo cual impide aplicar la fórmula para calcular T_F pues es indeseable establecer $T_F = 0$, debido a que se gastaría demasiado tiempo al final del algoritmo. Debido a lo anterior, se mantuvo el valor de $T_F = 0.01$, estimado previamente.
- El valor que se obtiene al aplicar (4.7) resulta demasiado grande por lo que el tiempo que consume el algoritmo RSS puede superar al consumido por el algoritmo RS. Por lo tanto, cuando el valor de L_i (en la i-ésima iteración) ha superado las diez mil iteraciones, no se permite incrementar su valor. Las diez mil iteraciones fueron tomadas del número de iteraciones realizadas en el algoritmo RS implementado, cuya obtención de parámetros se realizó previamente por experimentación manual.

Con lo anterior, el pseudocódigo del algoritmo queda como se muestra en la Figura 4-3. Los cambios realizados en el algoritmo RSS respecto al algoritmo RS se resaltan con fuente en negrita. La principal diferencia se encuentra en el ciclo interno, en la línea 20, donde al agregar la condición (*iter* < *L1*) se reduce el número de iteraciones realizadas al principio del algoritmo, cuando no se requiere gran exploración dado que al final del ciclo interno, a temperaturas altas el algoritmo aceptará cualquier solución.

El pseudocódigo de la Figura 4-3 fue implementado por partes, primero sintonizando la Cadena de Markov (implementando las líneas 10, 11, 12, 13, 14 y 20) y manteniendo la Temperatura inicial dada en la ecuación (4.11) y posteriormente se incluyó también la sintonización de la Temperatura inicial (línea 6). Lo anterior se realizó de ésta manera para realizar un mejor análisis respecto a los resultados obtenidos.

```
1 Inicio
2  Leer archivo
3  Generar solución factible inicial
4  costoIni = cuenta violaciones
5  mejorCosto = costoIni
6  TEMPINI = -5 * maxNumStudents /(log(0.95))
7  TEMPFIN = 0.01
8  ALFA = 0.95
9  Iter = 0
10 C = 3
11 N = log(TEMPFIN/TEMPINI)/log(ALFA)
12 L1 = 1
13 LMAX = C * (num_eventos *(num_eventos – 1)
14 BETA = exp((log(LMAX)-log(L1))/N)
15 Mientras (temper > TEMPFIN)
16   Si (mejorCosto > costoIni)
17    mejorSolucion = solucionActual
18    mejorCosto = costoIni
19   Fin si
20   Mientras (iter < L1 || costoIni == 0){
21    Buscar solución factible aleatoria
22    Reemplazar solución
23    costoNew = cuenta violaciones
24    costoDif = costoNew – costoIni
25    Si (costoNew <= 0)
26          costoIni = costoNew
27    sino
28     si (random() < exp(-costoDif/temper))
29          costoIni = costoNew
30     sino
31          Retomar solución anterior
32     Fin si
33      Fin si
34    iter = iter + 1
35   Fin Mientras
36   Si L1 < 10000
37    L1 = Parte mayor entera de (L1 * BETA)
38   Finsi
39   temper = temper * ALFA
40 Fin Mientras
```

Figura 4-3. Pseudocódigo del algoritmo RSS

En la Figura 4-3, las líneas añadidas al realizar la sintonización de parámetros son: 10,11,12,13,14, 36, 37 y 38. En la línea 20 se modificó la condición del ciclo interno.

4.6. Resultados obtenidos

Para verificar el desempeño de los algoritmos, se realizó una ejecución del RSS con cada una de las 20 instancias del PATAT, mostradas en la Figura 4-1. En la Tabla 4-6 se presentan los resultados obtenidos después de realizar la sintonización de las Cadenas de Markov. En la tabla 4-7 se presentan los resultados obtenidos aplicando además la nueva temperatura inicial. Las mejores soluciones se resaltan con fuente en negrita.

En la Tabla 4-7, cuando únicamente se implementó la sintonización de las Cadenas de Markov, los mejores resultados se obtuvieron con α=0.95, a excepción del caso C08 en el que el mejor fue para α=0.90. Esto verifica la eficacia del algoritmo pues conforme $\alpha \rightarrow 1$ la calidad en la solución mejora. En cuanto al tiempo, los mejores valores fueron para α=0.75.

Tabla 4-7. Resultados obtenidos con el RSS (sólo Cadena de Markov).

	No. Restricciones suaves violadas					*Tiempo (segs)*				
Caso	α=0.95	α=0.90	α=0.85	α=0.80	α=0.75	α=0.95	α=0.90	α=0.85	α=0.80	α=0.75
C01	**139**	225	209	239	220	2426	1313	639	672	**346**
C02	**131**	145	155	171	198	2372	1147	620	615	**343**
C03	**232**	235	256	292	283	2369	1262	907	434	**329**
C04	**369**	461	456	463	534	2666	1368	931	590	**384**
C05	**530**	625	683	701	671	2479	1824	718	534	**336**
C06	**201**	233	268	317	302	2455	1148	944	566	**348**
C07	**332**	336	441	396	445	2597	1407	720	562	**362**
C08	81	**64**	87	95	113	2505	1197	765	600	**370**
C09	**109**	120	134	150	153	2692	1323	773	617	**413**
C10	**139**	183	199	187	190	2534	1323	685	489	**330**
C11	**143**	181	192	198	187	2461	1176	668	549	**365**
C12	**217**	254	257	287	286	2446	1776	840	427	**344**
C13	**205**	220	236	259	237	2554	1007	762	533	**404**
C14	**302**	343	326	376	436	3276	1209	724	658	**379**
C15	**287**	314	336	369	391	2663	1207	683	485	**354**
C16	**109**	110	115	125	138	2846	1076	833	664	**395**
C17	**418**	434	483	531	534	2528	1279	680	492	**348**
C18	**111**	127	131	130	152	2581	1103	729	470	**336**
C19	**153**	170	219	255	221	2804	1063	778	583	**408**
C20	**177**	217	252	333	333	2449	1220	681	538	**361**

En la Tabla 4-8, se implementó la sintonización de la temperatura inicial, además de las Cadenas de Markov. Los mejores resultados se obtuvieron con α=0.95 y los mejores tiempos con α=0.75.

Tabla 4-8. Resultados obtenidos con el RSS (incluyendo temperatura inicial).

	No. Restricciones suaves violadas					*Tiempo (segs)*				
Caso	α=0.95	α=0.90	α=0.85	α=0.80	α=0.75	α=0.95	α=0.90	α=0.85	α=0.80	α=0.75
C01	**157**	188	198	253	259	1833	860	478	328	**243**
C02	**136**	162	202	205	220	1696	766	470	372	**283**
C03	**224**	241	280	279	274	1966	791	526	310	**250**
C04	**433**	504	537	601	661	1991	901	601	424	**266**
C05	**616**	705	795	813	783	1835	852	503	357	**243**
C06	**167**	278	367	408	458	1825	822	472	276	**243**
C07	**335**	431	436	479	564	2309	1021	576	447	**344**
C08	**46**	81	98	107	148	1849	850	523	419	**244**
C09	**125**	134	130	166	172	1981	1118	576	360	**279**
C10	**147**	157	202	199	238	1775	877	534	325	**281**
C11	**148**	200	187	210	233	2044	785	507	365	**289**
C12	**223**	265	275	321	324	1882	783	495	404	**299**
C13	**185**	216	270	307	366	2046	799	514	425	**290**
C14	**302**	381	406	457	491	2007	832	601	434	**273**
C15	**305**	420	418	452	478	1772	861	570	381	**274**
C16	**121**	128	149	170	172	1907	1003	600	438	**287**
C17	**449**	464	512	527	626	1805	909	506	380	**242**
C18	**115**	135	134	145	183	1701	940	467	324	**228**
C19	156	214	222	273	298	1978	957	699	469	275
C20	184	272	367	365	439	1721	865	619	352	273

En el Capítulo 7 se puede observar una comparación más detallada en cuanto a la calidad y tiempo promedio obtenidos al realizar 5 ejecuciones de los algoritmos RS y RSS por cada instancia. Además, en el Apéndice B se muestran las ejecuciones de ambos algoritmos, con cada valor de α y para las diversas instancias del problema ETT del PATAT.

4.7. Conclusiones del capítulo

En este capítulo se presentó la implementación realizada de los algoritmos RS y RSS para las instancias del PATAT. Como puede observarse, el desempeño resulta sumamente eficiente en cuanto al tiempo, al reducirse casi a la mitad. En cuanto a la calidad de las soluciones generadas por ambos algoritmos, no se presenta gran variación, por lo que el RSS se considera un método aceptable para resolver el problema ETT, tanto por la calidad de los resultados obtenidos como por el tiempo que le toma encontrarlos.

En el Capítulo 5 se presenta la implementación del algoritmo RSS con un problema real de asignación de horarios: el caso de la Universidad Autónoma de Yucatán (UADY) y en el Capítulo 7 se realiza una comparación detallada de los resultados obtenidos.

Capítulo 5. Asignación de horarios en la UADY con RSS

Cada institución educativa cuenta con sus propias políticas y recursos que determinan la forma en la que se debe realizar la asignación de horarios para los eventos que ahí se imparten. En este capítulo se presenta un modelo general del problema de asignación de horarios que se presenta comúnmente en las instituciones de educación superior y se resuelve el caso particular de la UADY (Universidad Autónoma de Yucatán) con el algoritmo RSS presentado en el capítulo 4. En las siguientes secciones se describe el modelo general diseñado para resolver el problema, así como el modelo general de la base de datos diseñada para almacenar la información necesaria en la solución del problema de asignación de cargas académicas y asignación de horarios. También se describe cómo se realiza actualmente a asignación de horarios en la UADY, así como el procedimiento y consideraciones tomadas en cuenta para obtener la solución del problema.

5.1. Descripción del problema

La Universidad Autónoma de Yucatán, en su perfil académico, está dividida en 15 Facultades, una de las cuales es la Facultad de Matemáticas, institución de la cual se desea resolver el problema de la asignación de horarios. Actualmente, la Facultad de Matemáticas cuenta con aproximadamente 100 Maestros, 6 planes de estudio y aproximadamente 1000 alumnos.

Se ha dividido el problema de asignación de horarios de la Facultad de Matemáticas en dos partes: la asignación de cargas académicas y la asignación de horarios.

5.1.1. Asignación de cargas académicas

Actualmente la asignación de cargas académicas, que corresponde a la distribución de asignaturas entre profesores, se realiza de manera manual por los grupos académicos existentes, de tal forma que cada grupo se encarga de repartir entre sus miembros las asignaturas que corresponden al área académica de ese grupo. Sin embargo, se desconoce hasta el momento cuáles son los parámetros que cada grupo considera para realizar la asignación, pues no hay un criterio homogéneo definido para distribuir las asignaturas.

Dado lo anterior, en este trabajo se propone aplicar el método propuesto en [Gómez, 2005] para realizar la asignación de cargas. El método consiste en emplear un parámetro para ponderar el grado de preferencia que cada profesor tiene hacia las asignaturas que podría impartir, y en base a éstos distribuir las cargas académicas.

Para realizar lo anterior, se consideran como datos de entrada:

- La demanda de asignaturas que se deben impartir en el período escolar deseado.
- El número máximo de asignaturas y horas frente a grupo para cada profesor.
- Las preferencias que cada profesor tiene hacia las asignaturas que puede impartir.

Las restricciones duras que se requieren satisfacer son las siguientes:

- Todas las asignaturas deben ser impartidas, cada una por un profesor.
- A un profesor no se le deben asignar más asignaturas de las que debe impartir en un período escolar.
- A un profesor no se le deben asignar más horas de las que debe impartir en un período escolar.
- Todos los profesores activos deben impartir al menos una asignatura en el período escolar.

La restricción suave que se desea satisfacer es la siguiente:

- Un profesor debe impartir únicamente asignaturas que se encuentran dentro de sus preferencias.

5.1.2. Asignación de horarios

En el caso específico del problema ETT en la Facultad de Matemáticas de la UADY, se considera que la semana es de 5 días hábiles (Lunes a Viernes), cada uno de los cuales contiene 8 períodos de clase disponibles para programar las asignaturas. Además, se considera lo siguiente:

Datos de entrada

- Los catálogos (datos) de profesores, materias, grupos y materias por grupo para el período lectivo a considerar.
- La relación de asignaturas que imparte cada profesor en el semestre.
- La relación de asignaturas que no pueden ser programadas a la misma hora (porque tienen estudiantes en común).

- La duración en horas teóricas y/o prácticas de cada asignatura.
- Los salones disponibles para impartir clases.
- El horario disponible de los profesores para impartir clases.

Restricciones indispensables (duras)

- Un profesor sólo puede impartir una asignatura a la vez.
- No pueden programarse en el mismo horario dos asignaturas diferentes con estudiantes en común. Es decir, las materias que corresponden a un grupo no pueden programarse a la misma hora.
- Cada asignatura debe impartirse el número de veces y en días diferentes necesarios para cubrir las horas requeridas. Es decir, no se deben programar en sesiones continuas ni en un mismo día para una misma asignatura.
- Deben programarse todas las asignaturas requeridas en un semestre, con el número de horas (sesiones) establecidas para cada una.

Restricciones deseables (suaves)

- Los períodos 4 y 5 del día solo deben emplearse en caso necesario. Es decir, es deseable no utilizarlos para programar eventos.
- Es deseable respetar la disponibilidad de horarios de los profesores, para impartir sus clases.

5.2. Modelación matemática del problema

Para resolver el problema, es necesario realizar primero la asignación de cargas académicas para posteriormente realizar la asignación de horarios. A continuación se presenta la modelación matemática de ambas fases del problema.

5.2.1. Modelado de la Asignación de cargas académicas

Considerando las restricciones planteadas en la sección 5.1.1, para la asignación de cargas académicas se cuenta con [Gómez, 2005]:

- n profesores $T = \{t_1, t_2,\ldots,t_i,\ldots, t_n\}$
- m asignaturas a impartir $A = \{a_1, a_2,\ldots,a_j,\ldots, a_m\}$.
- Número de sesiones por materia $S = \{s_1, s_2,\ldots,s_i,\ldots s_m\}$
- Matriz de restricciones de tiempo y asignaturas para cada profesor $\mathbf{R}_{2xn}$
- Matriz de preferencias de cada una de las asignaturas para cada profesor $\mathbf{P}_{nxm}$.

Formalmente, el problema consiste en encontrar el valor de la variable binaria x_{ij}, la cual toma el valor 1 cuando la asignatura j es impartida por el profesor i, y toma el valor 0 en caso contrario, es decir cuando la asignatura j no es impartida por el profesor i.

La función objetivo que se desea optimizar consiste en maximizar la suma de las preferencias de cada una de las asignaturas que impartirá cada profesor; dicha función se expresa en (5.1).

$$z = \sum Scv$$

$$sujeto\ a \qquad (5.1)$$

$$Hcv = 0$$

Donde *Svc* = Restricciones suaves y *Hvc* = Restricciones duras violadas.

Las restricciones duras para el problema de asignación de cargas académicas en la UADY son las siguientes:

1. ***Todas las asignaturas deben ser impartidas, cada una por un profesor.***
   Es decir, cada una de las *m* materias debe ser asignada a uno y solo uno de los *n* profesores:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,...,m \qquad (5.2)$$

   Donde la variable *i* representa al profesor y *j* a la asignatura.

2. ***A un profesor no se le deben asignar más asignaturas de las que debe impartir en un período escolar.***
   Es decir, el número de asignaturas que imparte cada uno de los *n* profesores no debe exceder de la cantidad máxima de materias que puede impartir. Esto se representa en (5.3).

$$\sum_{j=1}^{m} x_{ij} \le r_{1i} \qquad (r_{1i} \in R), \quad i = 1,...,n \qquad (5.3)$$

   Donde la variable *i* representa al profesor, *j* a la asignatura y r_{1i} representa el máximo número de horas del *i*-ésimo profesor.

3. ***A un profesor no se le deben asignar más horas de las que debe impartir en un período escolar.***
   Es decir, para cada uno de los *n* profesores, la suma de las horas correspondientes a las sesiones de las materias asignadas no deben exceder al número máximo de sesiones del profesor. Ésta restricción está representada en (5.4).

$$\sum_{j=1}^{m} s_j x_{ij} \le r_{2i} \qquad (r_{2i} \in R, s_j \in S), \quad i = 1,...,n \qquad (5.4)$$

4. ***Todos los profesores activos deben impartir al menos una asignatura en el período escolar.***
   Es decir, para cada uno de los *n* profesores, el número de materias asignadas debe ser al menos 1. La ecuación (5.5) representa ésta restricción.

$$\sum_{j=1}^{m} x_{ij} \le 1 \qquad i = 1,...,n \qquad (5.5)$$

Y la única restricción suave es la siguiente:

1. ***Un profesor debe impartir únicamente asignaturas que se encuentran dentro de sus preferencias***

   Es decir, para cada uno de los *n* profesores, si la asignatura *j* le es asignada, ésta debe encontrarse dentro de sus preferencias P. La cual se encuentra representada por (5.6)

$$\sum_{j=1}^{m} p_{ij} x_{ij} \geq 0 \qquad (p_{ij} \in P), \quad i = 1,...,n \tag{5.6}$$

5.2.2. Modelado de la Asignación de horarios

Considerando las restricciones planteadas en la sección 5.1.2, para la asignación de horarios se cuenta con:

- *n* Materias: $E = \{e_1, e_2, \ldots, e_n\}$
  Donde cada e_i consta de s_i sesiones
- 40 Períodos (8 intervalos en 5 días de la semana): $P = \{p_1, p_2, \ldots, p_{40}\}$
- *m* Profesores: $T = \{t_1, t_2, \ldots, t_m\}$
- *r* Aulas: $A = \{a_1, a_2, \ldots, a_r\}$
- *s* Grupos: $G = \{g_1, g_2, \ldots, g_s\}$
- 2 Turnos: 0 y 1. Donde los primeros 20 períodos corresponden al turno 0 y los restantes al turno 1.
- Un vector de turnos por grupo $V = \{v_1, v_2, \ldots, v_s\}$
- Una matriz binaria de Materias por Grupo: D_{nxs}. Donde $d_{ij} = 1$ si el grupo *j* asiste a la materia *i*.
- Una matriz binaria de Materias por Maestro: C_{nxs}. Donde $c_{ij} = 1$ si el maestro *j* imparte la materia *i*.

Además, sea la variable x_{ijk} la representación de la materia *i* asignada al salón *j* en el período *k*, cuyo valor será 1 cuando exista tal asignación y 0 en caso contrario.

Las restricciones duras mencionadas anteriormente se expresan formalmente a continuación:

1. ***Un maestro sólo puede impartir una asignatura a la vez.***

   Es decir, para cada materia que imparte el profesor *l*, en cualquier período *k*, a lo más se debe programar una sesión de la materia *i*.
   Sea B_l el conjunto de materias que imparte el profesor *l*, donde

$$B_l = \{i \mid c_{il} = 1, \quad i = 1,...,n\}$$

   Por lo tanto, la restricción queda representada por:

$$\sum_{k=1}^{40} x_{ijk} \leq 1, \quad j = 1,...,r \quad i \in B_l, l = 1,...,m \tag{5.7}$$

2. ***En un salón sólo puede impartirse una asignatura a la vez.***

Es decir que, en un salón específico *j*, en cualquier período *k*, a lo más se debe programar un evento *i*. Esta restricción se expresa en (5.8).

$$\sum_{i=1}^{n} x_{ijk} \leq 1 \qquad j = 1,...,r \quad k = 1,...,45 \tag{5.8}$$

3. ***No pueden programarse en el mismo horario dos asignaturas diferentes con estudiantes en común.*** Es decir, las materias que corresponden a un mismo grupo no pueden programarse a la misma hora.

O bien, en cada período k a lo más puede haber programada una de todas las materias i a las que asiste un mismo grupo q.

Sea E_q el conjunto de materias a las que asiste el grupo *q*, donde

$$E_q = \{i \mid d_{iq} = 1, \quad i = 1,...,n\}$$

Por lo tanto, la restricción queda representada en (5.9).

$$\sum_{k=1}^{40} x_{ijk} \leq 1\,, \quad j = 1,...,r \quad i \in E_q\,, q = 1,...,s \tag{5.9}$$

4. ***Cada asignatura debe impartirse el número de veces necesario para cubrir las horas requeridas, en días diferentes.*** Es decir, se deben programar todas las sesiones de cada materia y no se deben programar sesiones continuas ni en un mismo día.

O bien, para cada materia *i* se deben programar s_i sesiones en *k* períodos discontinuos y correspondientes a días diferentes.

La primera parte de esta restricción queda representada en:

$$\sum_{k=1}^{40} x_{ijk} = s_i\,, \quad i = 1,...,n \quad j = 1,...,r \tag{5.10}$$

Para la segunda parte de la restricción, sea U el conjunto de períodos que se encuentran al final de cada turno del día,

$$U = \{k \mid (k \bmod 4)\ U\ (k \bmod 8)\,,\ k = 1,\ldots,40\}$$

La ecuación (5.11) representa esa restricción.

$$\sum_{k \notin U} x_{ijk} x_{ij(k+1)} = 0\,, \quad i = 1,...,n \quad j = 1,...,r \tag{5.11}$$

5. ***Cada asignatura debe ser programada en el turno correspondiente al grupo al que ésta pertenece.***

Es decir, todas las asignaturas *i* a las que asiste un grupo *q* deben programarse en el turno *v* que le corresponde a dicho grupo. La ecuación (5.12) representa ésta restricción.

$$\prod_{l=1}^{s_i} x_{ijk_l} = v_q\,, \quad d_{iq} = 1\,, \quad i = 1,...n \quad j = 1,...,r \quad k = 1,...,40 \tag{5.12}$$

Análogamente, la única restricción suave del problema es:

1. ***Los períodos último del primer turno y primero del segundo turno sólo deben ser empleados en caso necesario.***

Es decir, en los períodos 4, 8, 12, 16, 20, 21, 25, 29, 33, 37 no deben programarse eventos.

Para esto, definamos el conjunto W de períodos no deseados,

$$W = \{ k \mid k \bmod 4 = 0 \text{ y } 1 \leq k \leq 20\} U \{k+1 \mid k \bmod 4 = 0 \text{ y } 21 \leq k \leq 40 \}$$

Por lo tanto, la ecuación (5.13) representa la restricción anterior.

$$\sum_{k \in W} x_{ijk} = 0 , \quad i = 1,...,n \quad j = 1,...,r \tag{5.13}$$

Y considerando que el problema consiste en satisfacer todas las restricciones duras y la mayor cantidad de restricciones suaves posibles, el problema ETT de la UADY queda expresado formalmente como minimizar:

$$\begin{aligned} & z = Scv \\ & sujeto\ a \\ & Hcv \end{aligned} \tag{5.14}$$

Donde Scv en este caso es la única restricción suave del problema y Hcv son todas las restricciones duras del mismo.

5.3. Implementación de RSS

Ambos problemas, la asignación de cargas académicas y la asignación de horarios, fueron resueltos de manera independiente con su propia implementación del algoritmo RSS, considerando las restricciones establecidas en las secciones 5.1 y 5.2. En ambos casos se sigue el método descrito en el capítulo 4 que se implementó para resolver el problema del PATAT, cambiando únicamente la solución inicial generada, el criterio de vecindad empleado en cada caso y los parámetros del algoritmo, obtenidos mediante el método de sintonización descrito en el capítulo 3.

La implementación fue realizada en un sistema con una sencilla interfaz para permitir la ejecución del algoritmo. La programación del algoritmo y de los módulos indispensables para la utilización del sistema fue realizada en el lenguaje Java 1.2, utilizando la versión libre del software Jcreator.

El sistema desarrollado permite realizar una preasignación de cargas académicas, con lo cual se deja abierta la posibilidad de continuar realizando la distribución como hasta ahora se hace por lo grupos académicos, o bien, utilizar el método automatizado propuesto en este trabajo. En las siguientes secciones se describe el proceso realizado para resolver ambas etapas del problema.

5.3.1. Implementación de la asignación de cargas académicas

Para realizar la asignación de cargas académicas, primero se extraen los datos que se encuentran almacenados en la base de datos. Se asume que dicha base de datos ha sido previamente alimentada con la información necesaria. El algoritmo genera una solución inicial cercana a la factibilidad, es decir, se busca satisfacer las restricciones duras del problema listadas en la sección 5.1, pero esto en ocasiones no es posible, pues podría ser que las características del problema no permitan una solución factible (por ejemplo, si se cuenta con más asignaturas de las que todos los maestros juntos pueden impartir, de acuerdo a sus horas de trabajo). La Figura 5-1 ilustra la representación interna de una solución, que corresponde a un ejemplo para 2 profesores y 4 asignaturas. El pseudocódigo para obtener la solución inicial se presenta en la Figura 5-2.

La Figura 5-1 Indica que el profesor 1 imparte las asignaturas 1 y 3, y el profesor 2 imparte las asignaturas 2 y 4. Como puede observarse, todas las asignaturas son impartidas por algún profesor (en cada línea se encuentra exactamente un 1). Además, considerando que la carga máxima para cada profesor es de 4 asignaturas ó 20 horas semanales frente a grupo, ambas restricciones son satisfechas.

		Profesor	
		1	**2**
Materia	**1**	**1**	**0**
	2	**0**	**1**
	3	**1**	**0**
	4	**0**	**1**

Figura 5-1. Representación interna de una Carga académica

A partir de la solución inicial generada, se aplica el criterio de vecindad que consiste en modificar la asignación del profesor que imparte una asignatura. La asignatura a la que se le aplicará dicho criterio se elige de manera aleatoria. El objetivo del algoritmo consiste en maximizar el número de restricciones suaves, es decir, las preferencias de los profesores.

Para obtener el valor indicado de la temperatura inicial, se aplica nuevamente el método propuesto en [Sanvicente, 2004].

En este caso, según el criterio de vecindad empleado, el deterioro máximo producido entre cualesquiera dos soluciones vecinas es 100. Esto debido a que, lo peor que puede ocurrir es que a un profesor se le cambie una asignatura por la que tenía toda la preferencia (100) por otra que no desea impartir (0). Esto es 100 – 0 = 100.

$$\Delta Zv_{max} = 100 \tag{5.15}$$

```
1 Para cada materia
2   Para cada profesor
3     Si la materia se encuentra entre sus preferencias
4           Elegir a ese profesor
5     Si no se ha elegido a algún profesor
6           Elegir a un profesor de manera aleatoria
7     Hacer
8       Asignar al profesor
9       Si al asignar esa materia al profesor se excece el límite de
10      materias del profesor
11          Desasignar al profesor
12      Si con esa materia se excede el límite de horas del profesor
13          Desasignar al profesor
14      Elegir a otro profesor
15    Mientras la materia no tenga un profesor asignado
16   Fin Para
17 Fin Para
18 Mientras no haya un profesor asignado
19   Para cada profesor
20    Si ese profesor no tiene alguna materia
21          Quitarle una materia a un profesor que tenga más de una
22          Asignarla al profesor que no tenía
23    Fin si
24   Fin Para
25 Fin Mientras
```

Figura 5-2. Pseudocódigo para encontrar una Carga académica inicial

De manera similar, el deterioro mínimo producido es 0, siendo dos posibles casos: que a un profesor se le cambie una por la que tenía toda su preferencia (100) por otra que también tenía toda su preferencia (100), esto es: 100 – 100 = 0. O bien, que se le cambie una que no quería impartir por otra que tampoco quería impartir: 0 – 0 = 0.

$$\Delta Zv_{min} = 0 \tag{5.16}$$

Por lo tanto,

$$T_0 = \frac{-\Delta Z_{V\max}}{\ln\left(P_A(\Delta Z_{V\max})\right)} = \frac{-100}{\ln(0.95)} = 1949.57 \tag{5.17}$$

$$T_f = \frac{-\Delta Z_{V\min}}{\ln\left(P_A(\Delta Z_{V\min})\right)} = \frac{-0}{\ln(0.05)} = 0 \tag{5.18}$$

De esta manera se obtiene que el mejor valor para la temperatura inicial es $T_I = 1949.57$. Sin embargo, el valor $T_F = 0$ no es deseable, por lo que se emplea el valor $T_F = 0.01$.

Ahora, para sintonizar la longitud de la cadena de Markov se considera deseable un nivel de exploración del 95% de la vecindad, por lo que el valor de L_{max} está determinado por:

$$L_{max} = C\,|V_{Si}| = 3 * \texttt{n\_asignaturas*n\_profesores} \qquad (5.19)$$

El número de iteraciones del ciclo externo del algoritmo, dado por *n*, se expresa en:

$$n = \frac{\ln T_0 - \ln T_F}{\ln \alpha} = \frac{7.575 - (-4.605)}{\ln \alpha} = \frac{12.18}{\ln \alpha} \qquad (5.20)$$

De lo anterior, se obtiene el valor de β:

$$\beta = \exp \frac{\ln L_{max} - \ln L_1}{n} \qquad (5.21)$$

El pseudocódigo del algoritmo RSS implementado para encontrar la mejor asignación de cargas académicas se presenta en la Figura 5-3.

```
1 Lee Alfa
2 Calcula L1
3 Calcula Beta
4 Establece Temperatura Inicial TEMPINI
5 Establece Temperatura final TEMPFIN
6 matrizCargas = genera_Solucion_Inicial()
7 costoIni = evalua(matrizCargas)
8 Mientras no se alcance el equilibrio térmico (temper > TEMPFIN)
9     iter = 0
10    respalda_sol_anterior()
11    Mientras (iter < L1)
12          Elegir evento1 = random(n_materias)
13          Aplica el criterio de vecindad, swap(evento1)
14          costoNew = evalua_Nueva_solucion()
15          costoDif = costoNew – costoIni
16          si (costoDif > = 0)
17                Acepta la nueva solución, costoIni = costoNew
18          si no
19                si (r = random() <= exp(-costoDif/temper)
20                      costoIni = costoNew
21                si no
22                      Restaura_sol_Anterior()
23                fin si
24          fin si
25          iter=iter+1
26          Calcula L1 = L1 * Beta
27    Fin Mientras
28    calcula temper = temper * Alfa
29 Fin Mientras
30 Imprime_Mejor_Solucion
```

Figura 5-3. Pseudocódigo de la asignación de Cargas para la UADY

El sistema final fue desarrollado con José Luis Gómez Ramos [Gómez, 2005] y Federico Alonso Pecina [Alonso, 2005], como parte de un gran proyecto existente en la Cátedra de Optimización Combinatoria. En el manual de usuario presentado en el Apéndice C se puede encontrar la pantalla de ejecución del algoritmo y la explicación de su uso.

5.3.2. Implementación de la asignación de horarios

Para resolver la asignación de horarios con el algoritmo RS, se sigue el mismo proceso descrito en la sección anterior, con la diferencia de que la solución inicial consiste en la asignación de un período y un salón para cada sesión de la asignatura y el criterio de vecindad consiste en intercambiar dos materias asignadas.

Siguiendo con el ejemplo anterior y suponiendo que cada asignatura consta de 2 sesiones, se requiere programar en total 8 sesiones-materias, donde las sesiones 1 y 2 corresponden a la materia 1, las sesiones 3 y 4 a la materia 2, las sesiones 5 y 6 a la materia 3, y por último, las sesiones 7 y 8 a la materia 4. Supóngase además que las materias 1 y 2 corresponden a un grupo y las materias 3 y 4 a otro grupo. Si se cuenta con 2 salones, una posible distribución se representa en la Figura 5-4. En lo sucesivo, se denomina materia o asignatura a cada elemento sesión-materia para no generar confusiones.

		Sesión-Materia							
		1	2	3	4	5	6	7	8
Periodo	1	1	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0
	3	0	1	0	0	0	0	0	0
	4	0	0	0	1	0	0	0	0

Salón 2		Sesión-Materia							
		1	2	3	4	5	6	7	8
Periodo	1	0	0	0	0	0	0	1	0
	2	0	0	0	0	1	0	0	0
	3	0	0	0	0	0	0	1	0
	4	0	0	0	0	0	1	0	0

Figura 5-4. Representación interna de un Horario

Como puede observarse en el horario ilustrado en la Figura 5-4, las asignaturas impartidas por el mismo profesor, no se programan en períodos iguales y las sesiones correspondientes a la misma asignatura tampoco se programan simultáneamente. Además, no se programan en el mismo período las asignaturas que corresponden al mismo grupo.

Una vez generada la solución inicial, se ejecutan las iteraciones del algoritmo considerando como criterio de vecindad el intercambio del salón y el período de dos materias cualesquiera.

En este caso, el deterioro máximo que se puede producir al aplicar el criterio de vecindad se obtiene considerando las restricciones duras y suaves que pueden ser violadas al realizar el intercambio. Es decir, sea M el costo por violar una restricción dura y 1 el costo de violar una restricción suave, donde M representa un valor muy grande que penaliza la violación de una restricción dura. Al realizar el intercambio puede suceder:

1. Que los profesores afectados sólo impartían una asignatura en el período seleccionado y ahora cada uno imparta dos. (Deterioro: 2M)
2. Que se hayan puesto en el mismo horario dos asignaturas del mismo grupo. (Deterioro: 1M)

Por lo tanto,

$$\Delta Zv_{max} = 3M \tag{5.22}$$

De manera similar, el deterioro mínimo producido es 0, si se considera que el intercambio mantiene estable la cantidad de restricciones incumplidas. Es decir,

$$\Delta Zv_{min} = 0 \tag{5.23}$$

Por lo tanto, considerando M = 1000, los valores de T_1 y T_F están dados por 5.24 y 5.25.

$$T_1 = \frac{-\Delta Z_{V\max}}{\ln\left(P_A(\Delta Z_{V\max})\right)} = \frac{-3000}{\ln(0.95)} = 58{,}487.177 \tag{5.24}$$

$$T_F = \frac{-\Delta Z_{V\min}}{\ln\left(P_A(\Delta Z_{V\min})\right)} = \frac{-0}{\ln(0.05)} = 0 \tag{5.25}$$

Nuevamente, el valor $T_F = 0$ no es deseable, por lo que se emplea el valor $T_F = 0.01$.

Ahora, considerando deseable un nivel de exploración del 95% de la vecindad, el valor de L_{max} está dado por (5.26).

$$L_{max} = C\,|V_{Si}| = 3 * \texttt{n\_asignaturas*(n\_asignaturas-1)} \tag{5.26}$$

El valor de n está dado por:

$$n = \frac{\ln T_0 - \ln T_F}{\ln \alpha} = \frac{10.976 - (-4.605)}{\ln \alpha} = \frac{15.58}{\ln \alpha} \tag{5.27}$$

Con los datos anteriores y la fórmula dada en la ecuación 5.21 se obtiene el valor de β.

El pseudocódigo del algoritmo RRS implementado para encontrar la mejor asignación de horarios es el mismo que se presenta en la Figura 5-3, únicamente cambiando la representación de la solución y el criterio de vecindad.

5.4. Resultados obtenidos

El algoritmo RSS se implementó tanto para la asignación de cargas académicas como para la asignación de horarios. Los resultados se presentan en las secciones 5.4.1 y 5.4.2.

5.4.1. Asignación de Cargas Académicas con RSS

Se realizó la implementación del algoritmo RSS para obtener las cargas académicas de la UADY, sin embargo, como se comentó en la sección 5.1.1, actualmente la asignación de cargas en esa institución no se realiza con el método sugerido sino de forma manual con criterios no estandarizados. Por lo tanto, no se cuenta con datos históricos de preferencias de los profesores hacia las materias, que permitan evaluar la bondad de la solución obtenida con el algoritmo RSS.

Sin embargo, el sistema implementado permite realizar la asignación de cargas académicas de manera automatizada y de manera manual, para ésta última se cuenta con una pantalla

para introducir la asignación de cargas académicas como Preasignaciones (ver Manual de Usuario en el Apéndice C).

Al implementar el algoritmo RSS con instancias reales del problema de asignación de cargas académicas de la UADY, se esperaba obtener coincidencias al 100% debido a que las preferencias de los profesores fueron establecidas considerando precisamente la asignación manual de cargas académicas. Sin embargo, se obtuvieron los resultados mostrados en la Tabla 5-1. Como se observa, no se obtuvo una satisfacción al 100% en las preferencias, debido a que los datos con los que se ejecutó el problema admiten una única solución correcta, y el algoritmo no logra encontrar tal solución con los casos de prueba ejecutados. Se espera que con mayor información sobre las preferencias de los profesores se obtengan mejores soluciones.

Tabla 5-1. Asignación de cargas académicas en la UADY con RSS

Alfa	Errores en cargas académicas	Errores en preferencias	Tiempo (seg)	Porcentaje en calidad (duras)	Porcentaje en preferencias (suaves)
0.75	0	10	99	100%	92.36 %
0.85	0	8	147	100%	93.89%
0.95	0	8	598	100%	93.89%

La tabla 5-1 fue realizada con los resultados obtenidos para el problema real de la UADY en el que se tienen 62 profesores activos y 117 materias para asignar entre ellos. En la primera columna de la tabla se despliegan los diferentes valores del parámetro α al resolver las instancias del problema de asignación de cargas académicas. En la segunda columna se despliega el número de restricciones duras no cumplidas en la solución final que obtuvo el algoritmo RSS para cada ejecución; como se observa, para todos los valores de α se alcanza la factibilidad del problema. La tercera columna corresponde a los errores encontrados en las restricciones suaves del problema, que corresponden a la satisfacción de preferencias de los profesores. Las columnas 5 y 6 de la Tabla 5-1 corresponden a los porcentajes de restricciones duras y suaves cumplidas, respectivamente.

Con el fin de complementar la evaluación del desempeño del algoritmo RSS se realizó la ejecución de las instancias de la UJAT, en la cual se tienen 369 materias para asignar entre los 102 profesores que se encuentran en la planta docente de dicha Universidad.

En [Gómez, 2005] se resolvió la asignación de cargas académicas empleando Algoritmos Genéticos (AG). En este documento se realiza una comparación de la calidad de los resultados obtenidos con RSS y los resultados obtenidos con AG. En cuanto al tiempo de ejecución no fue posible realizar la comparación pues en [Gómez, 2005] no se reportan los tiempos de ejecución del AG implementado. Los resultados que se obtuvieron con AG se presentan en la Tabla 5-2 y los resultados de RSS en la Tabla 5-3.

Se realizaron ejecuciones de RSS para distintos valores de α, los cuales se indican en la primera columna de la Tabla 5-3. La segunda columna contiene el número de errores que se encontraron al realizar la asignación de cargas académicas; cabe señalar que éstos errores se deben principalmente a que ningún profesor indicó preferencia para algunas materias,

por lo cual, independientemente del profesor al que se hayan asignado dichas materias, se detecta un error en sus asignaciones. En la instancia del problema real de la UJAT se detectaron 12 materias sin preferencia, por lo que 12 es el número mínimo de errores posibles en la asignación. La tercera columna contiene el porcentaje de concordancias encontrado con las preferencias de los profesores hacia las materias que se les asignaron, éste porcentaje se obtuvo considerando el número total de materias por asignar y el número de errores encontrado en la asignación. La última columna indica el tiempo que le tomó al algoritmo encontrar la asignación de cargas académicas, el cual se obtuvo directamente del algoritmo tomando la hora del sistema operativo al inicio y al final de la ejecución.

Tabla 5-2. Resultados obtenidos con AG para la asignación de cargas académicas en la UJAT

Prueba	Iteraciones sin cambio	Asignaturas no programadas	Errores en cargas académicas	Errores en preferencias	Porcentaje de preferencias
1	1500	0	2	64	82.66 %
2	2000	0	2	56	84.82 %
3	3000	0	0	42	88.62 %
4	4000	0	0	25	93.22 %
5	5000	0	0	23	93.77 %

Tabla 5-3. Resultados obtenidos con RSS para la asignación de cargas académicas en la UJAT

Valor de α	Asignaturas no programadas	Errores en cargas académicas	Errores en preferencias	Concordancia de preferencias	Tiempo de ejecución
0.75	0	0	69	81.3 %	603 seg.
0.85	0	0	38	89.7 %	984 seg.
0.95	0	0	27	92.68 %	3,595 seg.

Como puede observarse, la calidad en las soluciones fue mejor con RSS, para todos los casos: con AG el mejor resultado se obtuvo en la prueba 5, y fue de 23 errores en las preferencias, lo cual equivale a un 93.77% de concordancia en la asignación, con las preferencias de los profesores, mientras que con RSS se lograron obtener 27 errores, es decir, un 92.68 % de concordancia.

En [Gómez, 2005] también se reportan la concordancia de los resultados obtenidos con AG respecto a la solución manual obtenida por la administración de la universidad. Sin embargo, no se reportan los resultados que se obtuvieron de forma manual, por lo que no fue posible hacer esa comparación con los resultados que se obtuvieron con RSS.

5.4.2. Asignación de Horarios con RSS

La implementación de la asignación de horarios con RSS se realizó para las instancias de la UADY correspondientes al período escolar Febrero-Julio2005. Se cuenta con 117 asignaturas a impartir en el semestre, cada una con cierto número de sesiones, en total son 346 sesiones para programar; 20 salones; 40 períodos de clase, 8 períodos en cada uno de los 5 días de la semana; 20 grupos de alumnos y 6 planes de estudio o carreras. Para

realizar la asignación de horarios fue necesario considerar las restricciones mencionadas en la sección 5.1.2 y modeladas en 5.2.2. La Tabla 5-4 muestra los resultados obtenidos.

Tabla 5-4. Resultados obtenidos con el algoritmo RSS para la asignación de horarios

Valor de α	Restricciones duras violadas	Restricciones suaves violadas	Porcentaje de factibilidad	Porcentaje de preferencias	Tiempo de ejecución
0.75	0	7	100 %	99.95%	2776 seg
0.85	0	5	100 %	99.94%	5805 seg
0.95	0	0	100 %	100%	19937 seg

En la Tabla 5-4, la primera columna corresponde a los valores de α para cada caso de prueba, la segunda y tercera columnas (restricciones duras violadas y restricciones suaves violadas) engloban los conflictos listados en la sección 5.1.2. En este caso, con α=0.95 el algoritmo logró eliminar las restricciones duras y suaves del problema, obteniendo así el 100 % de factibilidad. El tiempo de ejecución se obtuvo directamente del algoritmo, tomando los tiempos del sistema al inicio y al final de la ejecución.

La implementación de RSS para resolver el problema de ETT con las instancias de la UJAT no fue realizada debido a que las políticas de ambas universidades (UADY y UJAT) difieren considerablemente, por lo que no se consideró apropiado realizar la comparación de los resultados obtenidos al no tener parámetros homogéneos de comparación.

5.5. Conclusiones

En este capítulo se abordó el problema real que se presenta en la Facultad de Matemáticas de la Universidad Autónoma de Yucatán. Se describió la forma en la que actualmente se realiza la asignación de cargas académicas y la asignación de horarios, y con base en las restricciones del problema se realizó la modelación matemática del mismo que posteriormente fue empleada para implementar la solución utilizando el algoritmo RS.

Los resultados obtenidos son los deseados pues el método implementado permite obtener una buena solución en poco tiempo.

El siguiente capítulo presenta una arquitectura general propuesta para la solución del problema Timetabling en instituciones educativas. Posteriormente se realiza el análisis y la comparación de los resultados obtenidos tanto para el problema real de la UADY, como para el problema general del PATAT.

Capítulo 6. Análisis y comparación de resultados con RS y RSS

Una vez presentada la implementación de los algoritmos RS y RSS con datos teóricos y prácticos, en este capítulo se realiza la comparación de los resultados obtenidos con ambos métodos en la solución del problema ETT del PATAT y de la UADY. Para el problema ETT del PATAT se realizaron dos etapas: la sintonización de las cadenas de Markov y la sintonización de temperaturas.

6.1. Comparación de resultados con las instancias del PATAT

Para comprobar el buen desempeño de los algoritmos, se realizó la ejecución del RSS con las primeras 6 instancias del PATAT, considerando que el 30% de la población es una muestra de tamaño suficientemente grande.

En la Tabla 6-1 se presentan los resultados obtenidos con el algoritmo RS, cuando aún no se implementaba la sintonización de parámetros. En la Tabla 6-2 se presentan los resultados obtenidos después de realizar la sintonización de las Cadenas de Markov (RSSM). En la tabla 6-3 se presentan los resultados obtenidos aplicando además la nueva temperatura inicial (RSST). En esta sección se hace referencia a los algoritmos implementados usando las siglas RSSM y RSST, para diferencias las 2 etapas de sintonización implementadas. Posteriormente, se hará referencia con las siglas RSS al algoritmo RSST. Como puede observarse, la calidad en las soluciones del primer caso prácticamente no varía respecto al segundo, en el tercero existen más variaciones, sin embargo, el tiempo se reduce considerablemente en los casos 2 y 3.

Para realizar una comparación más objetiva, y debido al tiempo que toma realizar cada ejecución, fueron seleccionadas de manera aleatoria 9 de las 20 instancias del problema, lo cual corresponde al 45% de la población, una muestra representativa del problema [Milton, 2003], para las cuales se realizaron cinco ejecuciones, los resultados aquí presentados corresponden al promedio obtenido para cada instancia del problema.

Se realizaron cinco ejecuciones debido a que por limitaciones de tiempo no fue posible realizar más repeticiones, además de que se observó que el promedio de ejecuciones tiende a estabilizarse en ese punto, es decir, con 5 ejecuciones el promedio es muy similar al que se obtiene con 6 ejecuciones, lo cual puede observarse en la Tabla 6-1.

Tabla 6-1. Promedios obtenidos con 2, 3, 4 y 5 ejecuciones, con RS y α=0.75

Caso	*Promedio 2 ejecuciones*	*Promedio 3 ejecuciones*	*Promedio 4 ejecuciones*	*Promedio 5 ejecuciones*	*Promedio 6 ejecuciones*
C01	220.00	214.00	209.25	208.80	208.00
C03	274.50	273.00	268.50	266.40	267.00
C04	519.50	522.00	518.00	524.60	523.83
C06	305.50	299.67	295.75	293.80	293.50
C09	149.50	144.33	148.00	152.60	152.83
C11	214.50	218.67	218.00	217.20	218.00
C12	268.00	272.00	277.25	277.20	277.67
C15	349.00	351.33	364.50	366.00	371.50
C19	260.50	256.33	253.00	244.00	246.00

Además, el número óptimo de corridas dado por el teorema de Tchebycheff está dado por [Azarang, 1996]:

$$n = \frac{m^2}{\alpha} \tag{6.1}$$

Donde α es la probabilidad de error permitida y 1/m es el número de desviaciones estándar permitido sobre la media de la distribución a simular.

Por lo tanto, con una probabilidad de error permitida de 0.05 y una desviación estándar de 2, se obtiene el número óptimo de corridas:

$$n = \frac{0.5^2}{0.05} = \frac{0.25}{0.05} = 5 \tag{6.2}$$

A continuación se presentan los resultados obtenidos con cada algoritmo, para los 3 valores de α elegidos, seleccionados con el fin de cubrir el rango en el que se encuentran los mejores valores.

En las Tablas 6-2 a 6-4 se observa que los mejores resultados se obtienen con α = 0.95, sin embargo, en ese caso la ejecución del algoritmo resulta ser la más tardada pues para todas las instancias se consume más de 4000 segundos. El mejor tiempo se obtiene con α = 0.75, aunque la calidad en las soluciones en todas las instancias resulta la peor, pues presenta el mayor número de restricciones suaves violadas.

Tabla 6-2. Resultados obtenidos con el RS (sin sintonización).

	No. Restricciones suaves violadas			*Tiempo (segs)*		
Caso	α=0.95	α=0.85	α=0.75	α=0.95	α=0.85	α=0.75
C01	**169**	189	209	4278	1386	771
C03	**221**	253	266	4315	1523	769
C04	**389**	483	525	4937	1580	867
C06	**187**	261	294	4435	1489	790
C09	**111**	140	152	4707	1571	901
C11	**155**	189	217	4448	1477	835
C12	**222**	278	277	4224	1340	765
C15	**289**	323	366	4320	1403	804
C19	**178**	200	244	5270	1613	898

Tabla 6-3. Resultados obtenidos con el RSSM (sólo Cadena de Markov sintonizada).

	No. Restricciones suaves violadas			*Tiempo (segs)*		
Caso	α=0.95	α=0.85	α=0.75	α=0.95	α=0.85	α=0.75
C01	**171**	204	209	2510	646	346
C03	**229**	265	271	2443	707	331
C04	**398**	472	539	2814	770	382
C06	**196**	264	330	2534	701	340
C09	**107**	140	146	2807	784	397
C11	**153**	200	195	2753	680	361
C12	**214**	251	288	2472	689	330
C15	**268**	356	408	2533	682	347
C19	**164**	205	241	2927	765	406

Tabla 6-4. Resultados obtenidos con el RSST (incluyendo temperatura inicial sintonizada).

	No. Restricciones suaves violadas			*Tiempo (segs)*		
Caso	α=0.95	α=0.85	α=0.75	α=0.95	α=0.85	α=0.75
C01	**168**	195	243	1879	483	242
C03	**226**	319	288	1885	509	241
C04	**419**	607	671	2113	567	263
C06	**195**	335	450	1931	471	245
C09	**112**	137	178	2069	588	277
C11	**158**	190	226	1904	523	298
C12	**220**	297	319	2066	477	300
C15	**321**	438	467	1869	540	297
C19	**156**	230	342	2010	634	290

En las Tablas 6-2 a 6-4 se pueden observar las gráficas comparativas del desempeño de los tres algoritmos, en cuanto a calidad y tiempo. Para todos los casos, se observa el mismo comportamiento: cuando α tiende a 1, la calidad de las soluciones mejora y el tiempo empeora, y cuando α es más pequeña, el tiempo mejora y la calidad empeora. En los tres casos, las siglas RS corresponden al algoritmo Recocido Simulado simple, RSSM al algoritmo Recocido Simulado en el que únicamente se sintoniza la Cadena de Markov y RSST corresponde al algoritmo en el que se sintoniza también la temperatura.

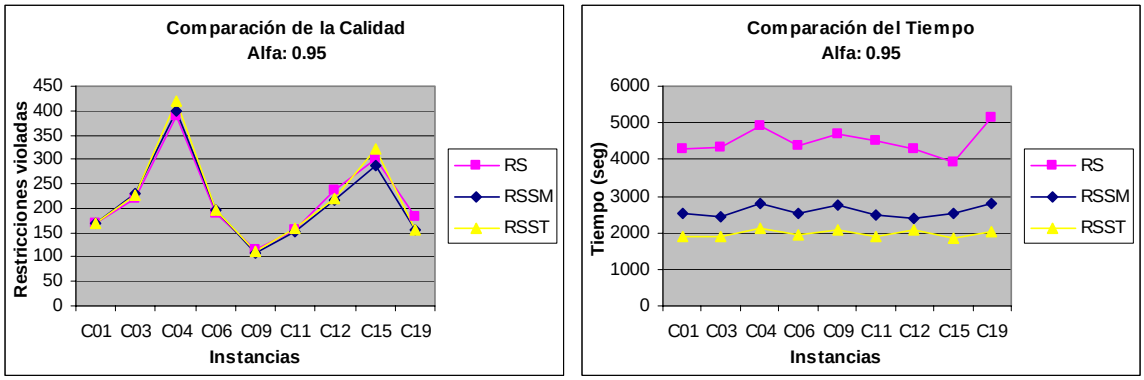

Figura 6-1. Resultados obtenidos en calidad y tiempo para el PATAT con α = 0.95

En la Figura 6-1, la gráfica de la izquierda muestra la calidad de las soluciones encontradas para 6 instancias del PATAT. Las líneas de los 3 algoritmos difiere notoriamente en la instancia C05, donde se desempeña mejor el RSST. En la gráfica de la derecha se presenta la comparación en cuanto al tiempo, donde se observa que RSST consume el menor tiempo, le sigue RSSM y finalmente, RS consume la mayor cantidad de tiempo.

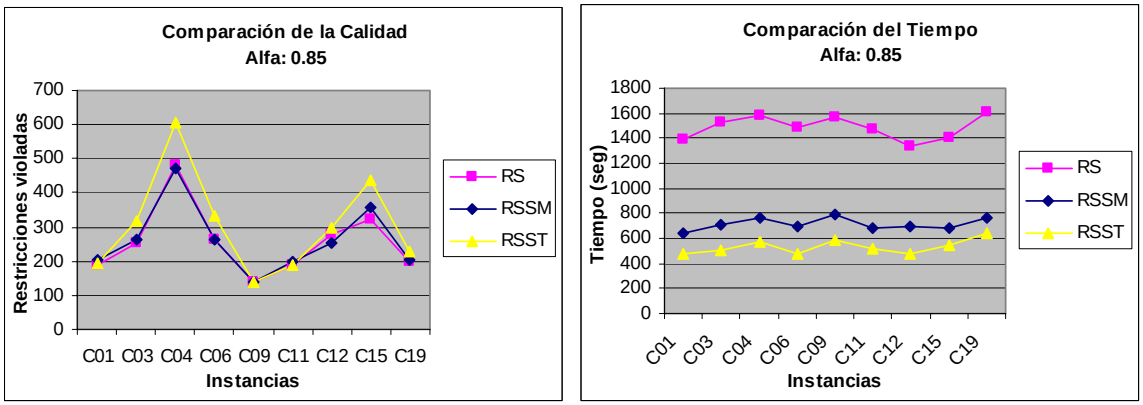

Figura 6-2. Resultados obtenidos en calidad y tiempo para el PATAT con α = 0.85

En la Figura 6-2, la gráfica de la izquierda, que corresponde a la calidad de las soluciones encontradas, la línea correspondiente a RSST está por encima de las otras dos, lo cual significa que es la que viola el mayor número de restricciones suaves; en este caso, para la instancia C05 se desempeña mejor el RS y en las demás instancias no hay una gran diferencia. En la gráfica de la derecha puede observarse que nuevamente RSST consume el menor tiempo, le sigue RSSM y finalmente, RS consume la mayor cantidad de tiempo. La Figura 6-3, con α=0.75, presenta un comportamiento muy similar a la Figura 6-2 (α=0.85).

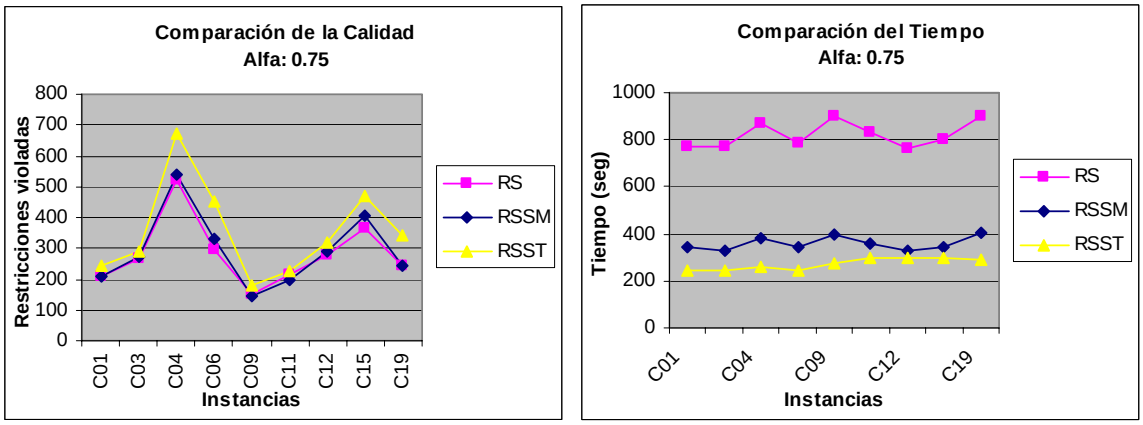

Figura 6-3. Resultados obtenidos en calidad y tiempo para el PATAT con α = 0.75

En términos de porcentajes, la comparación realizada entre los algoritmos RS y RSSM es la que se presenta en la Tabla 6-5. En las columnas correspondientes a la calidad, cuando el porcentaje es negativo significa que fue mejor el algoritmo RS. En cuanto al tiempo, el porcentaje representa cuánto se mejora, pues en todos los casos se invierte menos tiempo con el algoritmo RSSM.

Tabla 6-5. Comparación entre los algoritmos RS y RSSM en términos de porcentajes

	Calidad			***Tiempo***		
Caso	**0.95**	**0.85**	**0.75**	**0.95**	**0.85**	**0.75**
C01	-1%	-8%	0%	41%	53%	55%
C03	-4%	-5%	-2%	43%	54%	57%
C04	-2%	2%	-3%	43%	51%	56%
C06	-5%	-1%	-12%	43%	53%	57%
C09	4%	-1%	4%	40%	50%	56%
C11	1%	-6%	10%	38%	54%	57%
C12	4%	10%	-4%	41%	49%	57%
C15	7%	-10%	-12%	41%	51%	57%
C19	8%	-3%	1%	44%	53%	55%
Promedio	**1%**	**-2%**	**-2%**	**42%**	**52%**	**56%**

La Tabla 6-5 indica que con α = 0.95, en promedio se gana un 1% en la calidad de las soluciones, y además se gana un 42% en cuanto al tiempo de ejecución, por lo que en general, se obtiene una mejora del 43%. Con α = 0.85, en promedio se pierde un 2% en la calidad de las soluciones, y se gana un 52% en cuanto al tiempo de ejecución, obteniendo un promedio general del 50% de mejora. Y con α = 0.75, en promedio se pierde un 2% en la calidad de las soluciones, y se gana un 56% en cuanto al tiempo de ejecución, por lo que en general, se obtiene una mejora del 54%. Según éste análisis, el mejor parámetro es α=0.75, aunque en la práctica es necesario considerar cuánto se está dispuesto a sacrificar en calidad, contra lo que se puede ganar en tiempo.

La Tabla 6-6 presenta la comparación realizada entre los algoritmos RS y RSS. Los porcentajes se interpretan de manera similar a los de la Tabla 6-5. De igual manera, en la Tabla 6-7 se presenta la comparación respectiva entre los algoritmos RSSM y RSS.

Tabla 6-6. Comparación entre los algoritmos RS y RSS en términos de porcentajes

	Calidad			***Tiempo***		
Caso	**0.95**	**0.85**	**0.75**	**0.95**	**0.85**	**0.75**
C01	-1%	-8%	0%	41%	53%	55%
C03	-4%	-5%	-2%	43%	54%	57%
C04	-2%	2%	-3%	43%	51%	56%
C06	-3%	-1%	-12%	42%	53%	57%
C09	4%	-1%	4%	40%	50%	56%
C11	1%	-6%	10%	38%	54%	57%
C12	4%	10%	-4%	41%	49%	57%
C15	7%	-10%	-12%	41%	51%	57%
C19	8%	-3%	1%	44%	53%	55%
Promedio	**1%**	**-2%**	**-2%**	**42%**	**52%**	**56%**

Tabla 6-7. Comparación entre los algoritmos RSSM y RSS en términos de porcentajes

	Calidad			***Tiempo***		
Caso	**0.95**	**0.85**	**0.75**	**0.95**	**0.85**	**0.75**
C01	2%	4%	-16%	25%	25%	30%
C03	1%	-20%	-6%	23%	28%	27%
C04	-5%	-29%	-25%	25%	26%	31%
C06	1%	-27%	-36%	24%	33%	28%
C09	-1%	-18%	-21%	26%	25%	30%
C11	-1%	-23%	-22%	31%	23%	18%
C12	-2%	-24%	-26%	16%	31%	9%
C15	-1%	-23%	-26%	26%	21%	14%
C19	-1%	-22%	-24%	31%	17%	29%
Promedio	**-1%**	**-20%**	**-22%**	**25%**	**25%**	**24%**

6.2. Comparación de resultados en la UADY y la UJAT

La asignación de cargas académicas en la UADY es realizada de manera manual, por los diferentes grupos académicos existentes. Cada grupo académico reparte entre sus maestros integrantes las materias que le corresponde a su área académica. De manera que la asignación de cargas académicas toma de 1 a 2 semanas realizarla, pues cada grupo organiza sus propias reuniones en los días más convenientes para sus integrantes. Con el método de asignación de cargas propuesto, se espera realizar una asignación de materias más acorde con las preferencias de cada profesor y en un tiempo menor al que se requiere con el método manual. En este caso no resulta objetivo hacer una comparación de los resultados obtenidos con el algoritmo RSS en la asignación de cargas académicas, pues no se cuenta con la información necesaria requerida por el algoritmo para funcionar de manera eficiente, como se explicó previamente en la sección 5.4.1.

En la implementación con los datos de la UJAT se obtuvieron resultados de calidad similar (98.17%) a los reportados en [Gomez, 2005], con la ventaja de que el tiempo de ejecución de RSS es menor que el de AGDF. En la Tabla 6-7 se presenta una comparación de los resultados obtenidos al resolver la asignación de cargas académicas con RSS y con AGDF.

Tabla 6-8. Comparación del desempeño de RSS con AGDF para la asignación de cargas en la UJAT

	RSS		AGDF		
Caso	**Errores en cargas académicas**	**Errores en preferencias**	**Errores en cargas académicas**	**Errores en preferencias**	**Diferencia en porcentajes**
Mejores resultados	0	27	0	23	1.83 %
Peores resultados	2	38	2	64	7.04 %

En la Tabla 6-8 se observa que en los mejores resultados obtenidos con ambos algoritmos, la calidad de las soluciones encontradas con AGDF fue mejor que las que reportó RSS, en un 1.83%. Este porcentaje se obtuvo considerando las 369 materias que fueron asignadas. En el caso de los peores resultados hay una diferencia del 7.04% de los resultados de RSS sobre los de AGDF, además de que con el primero no hubo errores en las cargas académicas y con el segundo se presentaron 2 errores.

En cuanto a la asignación de horarios en la UADY con el algoritmo RSS, los resultados que se obtuvieron fueron comparados con la solución manual realizada por la administración (ver Apéndice D). La Tabla 6-9 presenta una comparación de los resultados obtenidos con ambos métodos: manual y automático con α=0.75.

Tabla 6-9. Comparación del desempeño de RSS con la forma manual de asignar horarios en la UADY

RSS			Manual		
Violaciones duras	**Violaciones suaves**	**Tiempo de ejecución**	**Violaciones duras**	**Violaciones suaves**	**Tiempo de ejecución**
0	0	5h 32’ 17”	18	28	2 a 3 semanas

Las violaciones duras corresponden a las restricciones físicas en cuanto a tiempo y espacio listadas en la sección 5.1.2 y las violaciones suaves son aquellas restricciones deseables de cumplirse, que también se listan en dicha sección. En el caso de la forma manual, las restricciones duras deben ser cero para considerar que el horario se encuentra listo, sin embargo, en el anexo D correspondiente a la solución manual realizada por la administración, se observan períodos en los que no se emplean los horarios establecidos (se emplean períodos no definidos, tales como1.2, 1.3, 1718 y 1719), por lo que se éstos casos se han considerado restricciones duras violadas.

Capítulo 7. Arquitectura General para la Asignación de Horarios

En los capítulos anteriores se presenta un enfoque de solución para problemas particulares, debido a las políticas propias de cada institución. En este capítulo se presenta un diseño general que se toma como base para resolver el problema de asignación de horarios en cualquier institución educativa.

7.1. Antecedentes

Cada institución educativa cuenta con sus propias políticas y en consecuencia, con su propia manera de asignar los horarios y salones para sus eventos. Sin embargo, en el desarrollo del presente trabajo se realiza un diseño general que engloba los casos de tres instituciones: el ITESM Campus Cuernavaca, la UADY y la UJAT (Universidad Juárez Autónoma de Tabasco). La primera es una institución de carácter privado y las dos últimas de carácter público.

Realizar un modelo general del problema fue posible gracias a que existen factores comunes en las instituciones, por ejemplo, los catálogos de datos que se requieren, el hecho de que cada estudiante asiste a cierto número de eventos, cada salón tiene una capacidad determinada, etc. Sin embargo, la solución del problema para cada institución requiere de una codificación muy personalizada que considere las políticas propias de la misma.

Por lo tanto, en este trabajo se realiza un modelado general aplicable a las tres instituciones, sin embargo, la solución implementada que se presenta en este trabajo corresponde a las políticas propias del problema de la UADY.

Cabe señalar que el trabajo realizado en el presente capítulo fue hecho en colaboración con José Luis Gómez Ramos [Gómez, 2005] y Federico Alonso Pecina[Alonso, 2005].

El modelo general fue hecho pensando en una arquitectura Web que funcione de la forma ilustrada en la Figura 7-1. Hasta el momento, el sistema Web se encuentra parcialmente implementado, permitiendo únicamente consultar los catálogos y listados principales de la base de datos: profesores, materias, grupos y aulas.

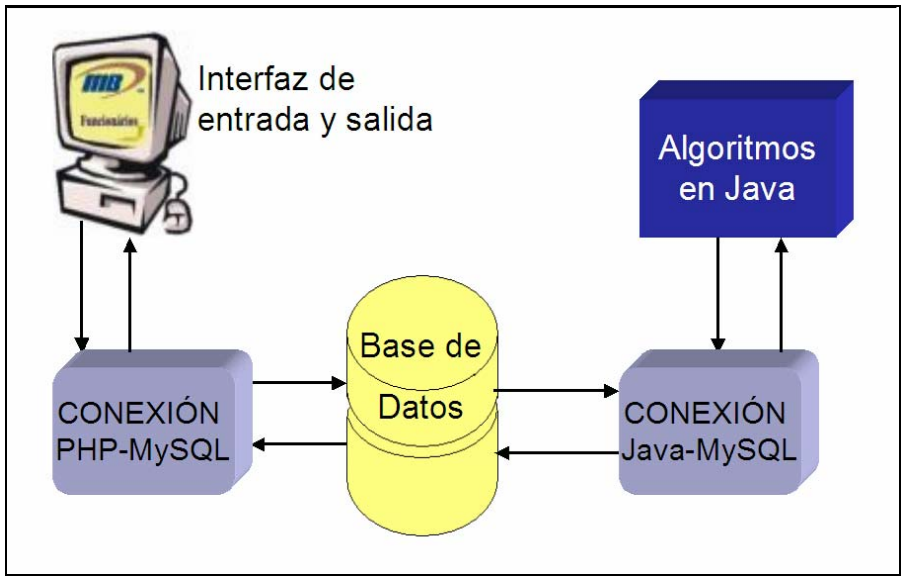

Figura 7-1 Arquitectura general del sistema de asignación de horarios

En el diagrama de la Figura 7-1, el acceso remoto a través de Internet permite manejar los catálogos necesarios en el sistema, tales como materias, grupos, profesores, etc., los cuales se pueden consultar con más detalle en la sección 5.3. La ejecución del algoritmo que resuelve el problema ETT para la institución deseada, en este caso la UADY, se realiza directamente en la máquina servidor, esto debido principalmente a cuestiones de seguridad y velocidad en la ejecución. El servidor en el que se implementó el sistema corre bajo el Sistema operativo *Windows 2000 Server*, con dos procesadores a 500 MHz, disco duro de 15 GB y 512 MB de memoria RAM.

El diseño conceptual del sistema general se presenta en la Figura 7-2.

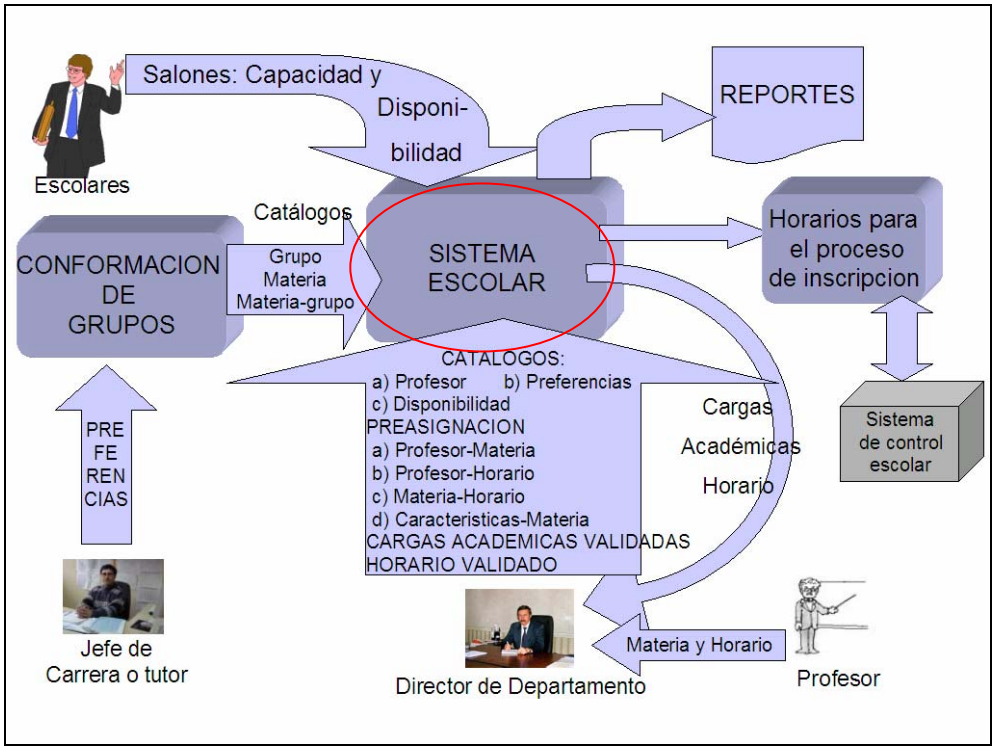

Figura 7-2. Diseño conceptual genérico del problema Educational Timetabling

El diagrama de la Figura 7-2 representa el flujo de datos de entradas y salidas que se realizan antes o durante el proceso de asignación de horarios. El sistema desarrollado en el presente trabajo está centrado en el cuadro central "Sistema escolar", resaltado en un óvalo de color rojo, el sistema se presenta detalladamente en la Figura 7-3 en función de las entradas y salidas que recibe y genera, respectivamente.

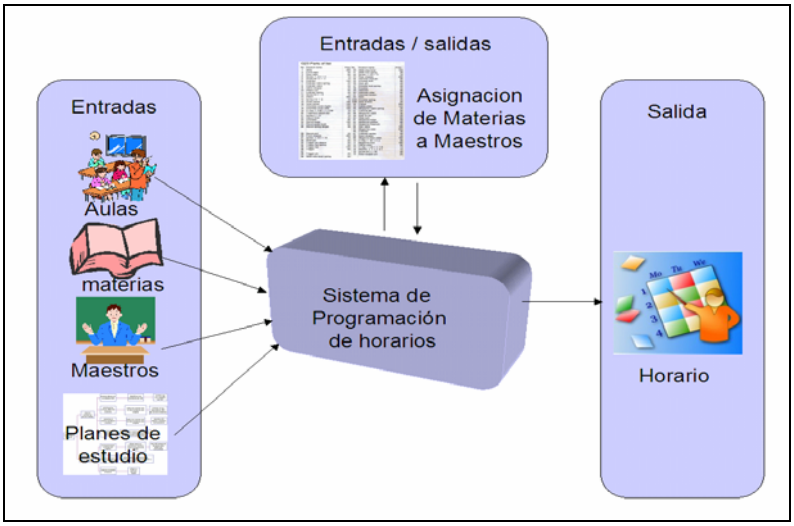

Figura 7-3. Entradas y Salidas del sistema

En el cuadro central superior de la Figura 7-3 (Entradas/Salidas), se consideran posibles entradas y salidas a la asignación materia-maestro pues en algunas instituciones esto se realiza como parte del sistema de programación de horarios y en otras se considera dato de entrada, como es el caso de la UADY. Sin embargo, el sistema desarrollado en este trabajo permite ambas posibilidades.

7.2. Modelo general de la base de datos

Se ha realizado un modelo general para que el algoritmo implementado pueda ser utilizado en cualquier institución cuyas políticas para la asignación de horarios sean similares a las consideradas en el presente trabajo. Para lo anterior, se ha considerado la base de datos cuyas tablas y campos se listan en el Apéndice A. El diagrama correspondiente al modelo general se presenta en la Figura 7-4. El diseño de la base de datos se realizó de manera general con el propósito de poder resolver la asignación de horarios de cualquier otra institución cuyas políticas sean similares a las que se consideran en la FMAT de la UADY, para lo cual únicamente se requiere introducir los datos a la base de datos y ejecutar el algoritmo.

Las tablas que se emplean en el algoritmo RSS implementado para el problema ETT de la UADY, son las siguientes:

- Tablas requeridas para la ejecución del algoritmo
    1. Aula
    2. Categoría
    3. Demanda (Se puede generar automáticamente)
    4. Disponibilidad_profesor
    5. Grado_academico
    6. Grupo
    7. Horario
    8. Materia
    9. Materia_plan
    10. Materia_grupo
    11. Período_lectivo
    12. Plan
    13. Preasignacion (es opcional)
    14. Preferencias_materia_profesor
    15. Profesor
    16. Turno
    17. Grupo-materia

- El algoritmo modifica
    1. Asignación
    2. Carga_académica_previa
    3. Preasignación

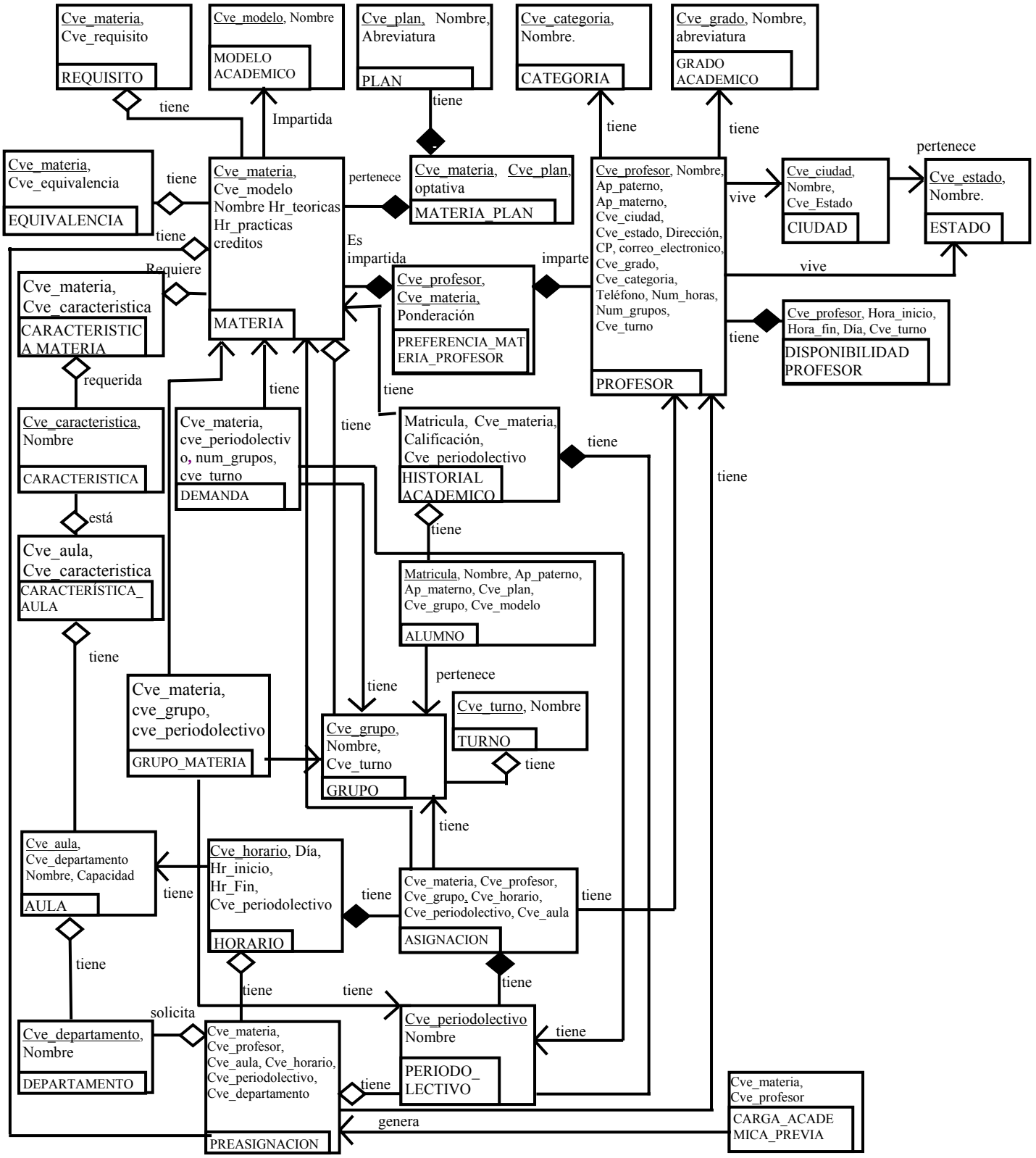

Figura 7-4. Diagrama E-R de la Base de Datos

7.3. Consideraciones del Modelo de la Base de Datos

El modelo de la Figura 7-4 fue realizado con el fin de poder englobar varios casos de problemas reales que se presentan en instituciones de nivel superior, específicamente se verificó que cumplen los requisitos y soportan la información necesaria para resolver el caso de la UADY, la UJAT y el ITESM Campus Cuernavaca. Este modelo fue diseñado considerando lo siguiente:

- Existen ciertos Profesores, Alumnos, Aulas y Materias en las cuales está centrado el proceso de generación de horarios.
- Las aulas y/o profesores cuentan con ciertas características especiales que a su vez son requeridas por algunas materias.
- Los profesores pertenecen a ciertas ciudades y estados, en el caso de las instituciones con diferentes campus.
- Los profesores son contratados con diferentes categorías y grados académicos, lo cual puede influir en la asignación de materias.
- Los estudiantes, al igual que las materias, pertenecen a ciertos grupos, que pueden ser formados de diferente manera, según la institución de la que se trate.
- El horario se genera en cada período lectivo, el cual puede ser un año, semestre, trimestre, cuatrimestre, etc.
- Puede realizarse una asignación previa de materia-profesor mediante un sub-sistema que se ejecuta previamente a la generación del horario general, o bien, introducir directamente los datos de la asignación realizada por la institución.
- En el caso de la realización automática de las cargas académicas materia-profesor, se requiere especificar las preferencias de los profesores hacia las materias, donde un mayor número significa mayor preferencia.
- Las materias pueden tener ciertas materias requisitos que el alumno debe cursar previamente a la materia en cuestión, al igual que puede tener nombres diferentes para diversos planes, lo cual aquí se ha denominado equivalencia de materias o materias equivalentes.
- Se cuenta con un historial académico para verificar que un alumno haya cursado las asignaturas que son requisitos de alguna otra.
- En cada período lectivo se genera cierta demanda de materias, es decir, un número de grupos requeridos para cada una así como el horario deseado para impartirlas.
- Es posible realizar preasignaciones, es decir, forzar a que el horario generado respete que una materia se imparta en cierto horario, en cierto salón o por cierto profesor.
- Es posible personalizar los horarios (horas por día de la semana) disponibles para la asignación de materias, esto se hace en la tabla llamada Horario.
- El horario generado ha sido denominado Asignación.

7.4. Implementación

La implementación se realizó en un servidor Apache 2.0.52 utilizando el lenguaje php 5.0. Se realizaron pruebas de ejecución en las plataformas Windows XP y Windows 2000. Hasta el momento el sistema se ha realizado de manera parcial, únicamente permite acceder a algunos catálogos de manera remota para poder introducir y consultar información básica. La Figura 7-4 presenta la pantalla de ingreso al sistema, en la Figura 7-5 se presenta la opción de Añadir a un profesor y permite ver el menú del sistema.

En este trabajo se han puesto únicamente las bases para una implementación más completa, por lo que la realización del sistema en sí se propone como trabajo futuro. La parte implementada puede verse con más detalle en el Apéndice E.

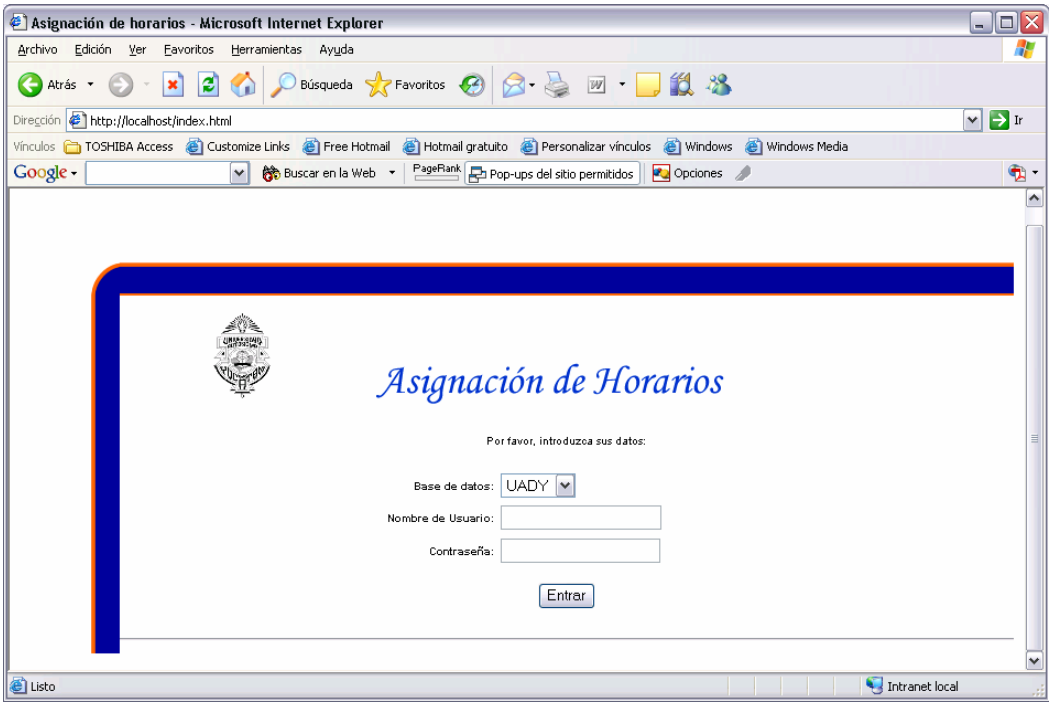

Figura 7-5. Pantalla de acceso al Sistema en Web

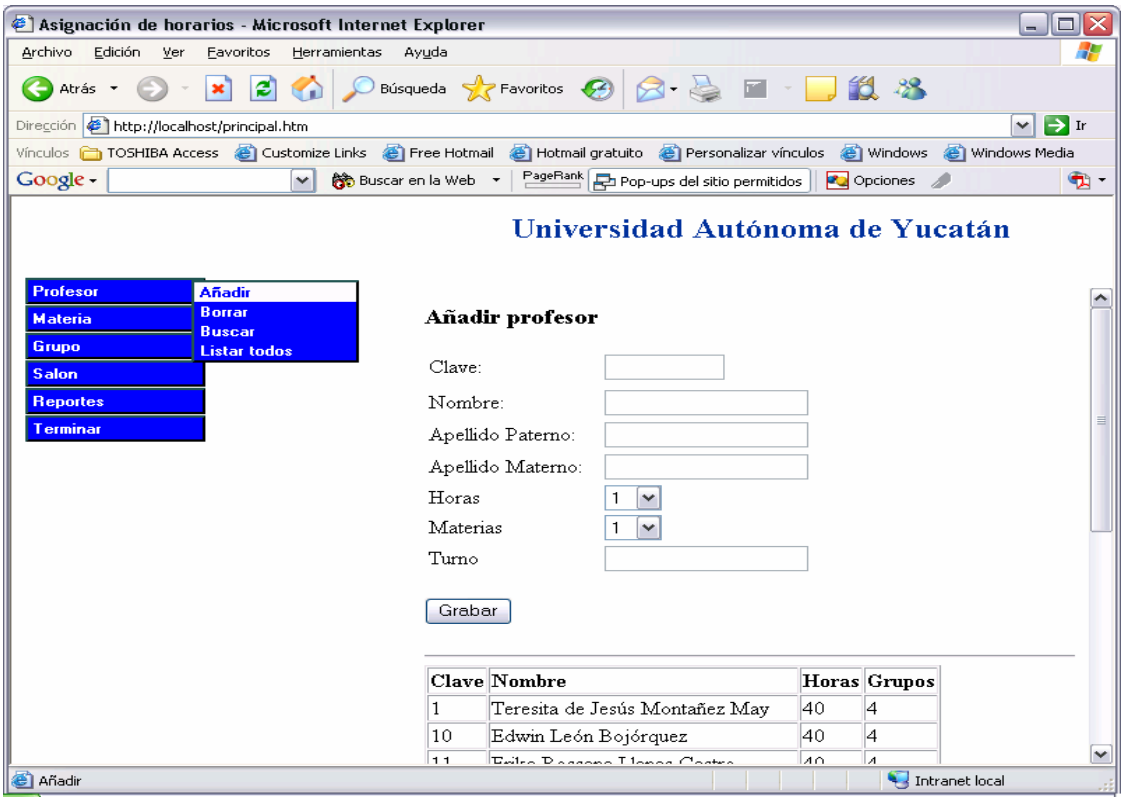

Figura 7-6 Menú del Sistema y Altas de Profesor

7.5. Resumen

En este capítulo se presenta un modelo general para resolver el problema TTE en instituciones Educativas, lo cual ha servido como pilar básico en la implementación del sistema realizado con el presente trabajo. El siguiente capítulo presenta el análisis y la comparación de los resultados obtenidos en este trabajo.

Capítulo 8.
Conclusiones y trabajo futuro

La realización de ésta tesis permitió conocer y adentrarse en el área de la optimización combinatoria así como conocer la gran variedad de aplicaciones que ésta tiene en problemas reales cotidianos.

Después de examinar algunas de las heurísticas que han sido empleadas para resolver problemas difíciles, se eligió el algoritmo Recocido Simulado (RS) para solucionar el problema de asignación de horarios y salones conocido como Educational Timetabling (ETT). La selección se realizó después de observar el desempeño de ésta heurística en los concursos realizados por el PATAT durante los últimos años y por la calidad de los resultados obtenidos al realizar codificación de una primera versión del algoritmo RS.

Se implementó una mejora al algoritmo RS con el fin de disminuir el tiempo de ejecución, realizando la sintonización de los parámetros de manera analítica, sin requerir demasiado tiempo de experimentación, con lo cual se disminuyó considerablemente el tiempo de ejecución. En el presente trabajo se denominó RSS (Recocido Simulado Sintonizado) al nuevo algoritmo.

Una de las contribuciones importantes de este trabajo de tesis consiste en realizar un análisis del método de sintonización de parámetros propuesto en [Sanvicente, 2004], en cuanto a la longitud de las Cadenas de Markov (CM), para encontrar la explicación lógica de la reducción de tiempo, lo cual condujo a concluir que dicha reducción tiene ciertos factores de restricción, los cuales se describen detalladamente en la sección 3.6.

También se propone un método automatizado de realizar la asignación de cargas académicas, pues actualmente ésta se realiza de manera manual, sin políticas claramente establecidas para realizar éste proceso.

Para el problema del PATAT, se obtuvieron soluciones con calidad similar a los resultados reportados en el concurso PATAT'2004, tal como se demuestra en el capítulo 6, sección 6.1, donde el algoritmo RSS se implementa en 2 etapas: sintonizando sólo las CM y sintonizando también la temperatura inicial.

Para el problema real de la UADY se diseñó e implementó un sistema de software que permite gestionar los datos necesarios para ejecutar correctamente el algoritmo y logra resolver eficaz y eficientemente el problema de asignación de horarios y/o cargas académicas de la institución. También se implementó parcialmente la interfaz bajo una arquitectura web que permite acceder remotamente a la información contenida en algunas tablas de la base de datos. La implementación completa no se realizó por cuestiones de tiempo, por lo cual se propone como trabajo futuro.

En el caso de las instancias del PATAT, el desempeño del algoritmo RSS obtiene resultados mejores que las del 7º lugar reportados en el concurso del PATAT realizado en el año 2004, por lo que, el algoritmo RSS quedaría en el 7º lugar, en cuanto a la calidad. Para al UJAT, la calidad de los resultados obtenidos con RSS son mejores que los que se reportan en la asignación de cargas académicas utilizando AGDF.

Con el trabajo realizado, se considera que el tiempo que le toma a la Facultad de Matemáticas de la Universidad Autónoma de Yucatán, realizar manualmente la asignación de horarios se verá reducido considerablemente, pues con la elaboración manual esto toma algunas semanas mientras que con la automatización implementada tomará unos cuantos minutos (menos de 30), o quizás, unas pocas horas (alrededor de 2 horas), dependiendo del valor del parámetro α empleado en la función geométrica de enfriamiento.

En resumen, se logró lo siguiente:

- Implementación exitosa del algoritmo Recocido Simulado Sintonizado, para el problema Educational Timetabling, tanto para el caso teórico del PATAT, como el caso práctico de la UADY.
- Exploración del método analítico de sintonización de parámetros, para explicar la reducción del tiempo de ejecución del algoritmo RSS.
- Resultados con calidad mejor que los del 7º lugar del concurso PATAT'2004.
- Propuesta de un método automático para realizar la asignación de cargas académicas en la UADY.
- Diseño e implementación de un sistema de software para resolver el problema ETT de la UADY, ejecutando el algoritmo RSS.
- Diseño de una arquitectura y base de datos general que proporcionan las bases para resolver el problema ETT de cualquier institución educativa.

Como trabajo futuro se proponen los siguientes puntos:

- Implementar una modificación al algoritmo RSS, denominada recalentamiento, que consiste en aumentar la temperatura al final del proceso, con el fin de buscar resultados con mejor calidad.
- Evaluar diferentes criterios de vecindad que puedan ser aplicados al problema ETT, como puede ser el intercambio de 2 ó más eventos.
- Desarrollar e implementar una paralelización del algoritmo RSS.
- Implementar una hibridación del algoritmo implementado, con el fin de mejorar la calidad de los resultados obtenidos con el algoritmo RSS.
- Realizar pruebas de hipótesis para probar que el desempeño de RSS es mejor que el de RS, con algunos niveles de confianza deseados.
- Mejorar la interfaz del SACAHO (Sistema de Asignación de Cargas Académicas y Horarios) implementado para ejecutar el algoritmo RSS.
- Terminar el sistema que está bajo la arquitectura Web, contemplando aspectos que corresponden al área de ingeniería de software (tales como la usabilidad y las características del diseño de la interfaz) y seguridad.
- Incluir una base de datos general que permita seleccionar los datos de la institución con los que se desea trabajar en el sistema, con el fin de resolver el problema ETT para diferentes campus de una misma institución.
- Añadir al algoritmo RSS implementado para la resolución del problema ETT de la UADY las características correspondientes al Modelo Educativo Académico, que se encuentra próximo a entrar en vigor, en el que existen materias compartidas por estudiantes de diferentes planes académicos.
- Considerar los datos históricos de períodos lectivos anteriores (años, semestres, trimestres o cuatrimestres) en la asignación de cargas académicas cuando no se cuente con preferencias de los profesores hacia algunas materias.
- Considerar el nuevo modelo académico a punto de entrar en vigencia en la Facultad de matemáticas, para las asignaturas que serán compartidas entre diferentes grupos de estudiantes.

Por todo lo anterior, los objetivos planteados al inicio de la tesis han sido alcanzados en su totalidad, encontrando soluciones para las instancias del PATAT con mejor calidad, en algunos casos, que los resultados reportados en el concurso del año 2004, y contribuyendo a resolver el problema real de la UADY.

Referencias Bibliográficas

[Aarts, 1989] Aarts, Emile y Korst, Jan, (1989). *Simulated Annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*. John Wiley & Sons, Great Britain, 272pp.

[Abba, 2001] Abbas, A. M. y Tsang, E. P. K. (2001). *Constraint-Based Timetabling a case study*. ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'01). 1165. Pags. 67-72.

[Abramson, 1992] Abramson D. y Abela J. (1992). *A parallel genetic algorithm for solving the School Timetabling Problem*. Technical report. Division of Information Technology, C.S.I.R.O. University of Edinburgh.

[Abramson, 2001] Abramson, D. (2001) *Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms*. Management Science, Vol. 37(1), 98-113.

[Alonso, 2002] Alonso, Federico. (2002). *Programación de horarios con coloreo de grafos*. Tesis de maestría. Instituto Tecnológico de Ciudad Madero. México.

[Alonso, 2005] Alonso, Federico. (2005). *Propuesta de Tesis doctoral en proceso*. ITESM, Campus Cuernavaca. México.

[Alvarez, 2000] Alvarez, Juan. (2000). *Un algoritmo Branch and Bound para el problema de Job Shop Scheduling*. Tesis de maestría, Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Cuernavaca. México.

[Andreu, 2000] Andreu, Juan. *Desarrollo, implementación y prueba de un algoritmo de reconstrucción de objetos a partir de una representación axonométrica, utilizando técnicas de optimización*. Tesis de Licenciatura. Facultad de Informática, Universidad Politécnica de Valencia. España.

[Appleby, 1960] Appleby, J. S., Blake, D. V., y Newman, E. A., (1960). *Techniques for roducing school timetables on a computer and their application to other scheduling problems*. The Computer Journal 3, 237–245.

[Azarang, 1996] Azarang, Mohamad y García, Eduardo. *Simulación y análisis de modelos estocásticos*. McGraw Hill. México, 1996.

[Bilbro, 1992] Bilbro, Griff; Snyder, Wesley; Garnier, Stephen y Gault, James. (1992). Mean Field Annealing: A Formalism for Constructing GNC-Like Algorithms. IEEE Transactions on Neural Networks. Vol. 3, Num 1.

[Blum, 2003] Blum, Christian y Roli, Andrea. (2003). *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison*. ACM Computing Surveys, Vol. 35, No. 3, pp. 268–308.

[Brelaz, 1979] Brelaz, D. (1979). *New methods to color the vertices of a graph.* Communications of ACM. 22:251—256.

[Burke, 1994] Burke, E.K.; Elliman, D.G. y Weare, R. (1994) *A University Timetabling System based on Graph Colouring and Constraint Manipulation*. Journal of Research on Computing in Education. Vol. 27, Iss. 1, pp. 1-18.

[Burke, 1997] Burke, Edmund; Kingston, Jeff; Jackson, Kirk y Weare, Rupert. (1997). *Automated University Timetabling: The State of the Art.* The Computer Journal 40 (9) 565-571.

[Burke, 2002] Burke, Edmund y Petrovic, Sanja. (2002). *Recent Research Directions in Automated Timetabling*. European Journal of Operational Research – EJOR, Vol. 140(2), 266-280.

[Burke, 2003] Burke, E.K.; Kendall, G. y Soubeiga, E., (2003). *A Tabu-Search Hyperheuristic for Timetabling and Rostering*. Journal of Heuristics, 9: 451–470.

[Burke, 2003a] Burke, Edmund; Bykov, Yuri; Newall, James y Petrovic, Sanja. (2003). *A Time-Predefined Approach to Course Timetabling*, Yugoslav Journal of Operational Research (YUJOR), Vol 13, No. 2, pages 139-151.

[Carter, 1998] Carter, Michael y Laporte, Gilbert. (1998). *Recent Developments in Practical Course Timetabling*. En Burke E., Carter M. (eds.): The Practice and Theory of Automated Timetabling II: Selected Papers (PATAT'97). Lecture Notes in Computer Science, Vol. 1408, Springer-Verlag, Berlin, Heidelberg, New York, 3-19.

[Castillo, 2002] Castillo, Enrique; Conejo, Antonio J.; Pedregal, Pablo; García, Ricardo y Alguacil, Natalia. (2002). *Building and Solving Mathematical Programming Models in Engineering and Science*, Pure and Applied Mathematics Series, Wiley, New York.

[Cerny, 1983] Cerny, V. (1983). *Minimization of Continuous Functions by Simulated Annealing*. Research Institute for Theoretical Physics, University of Helsinki, preprint No. HU-TFT-84-51.

[Cheng, 2003] Cheng, Eddie; Kruk, Serge and Lipman, (2003). Mark. *On the Multicommodity Flow Formulations for the Student Schedulign Problem*. Congressus Numerantium 160, pages 177-181.

[Colorni, 1998] A. Colorni, M. Dorigo, and V. Maniezzo. (1997). *Metaheuristics for high-school Timetabling*. Computational Optimization and Applications Journal, Vol. 3 No. 9 pages 277-298.

[Cooper, 1995] Cooper, T.B. and Kingston, JH. (1995) *The Complexity of Timetable Construction Problems*, in the Practice and Theory of Automated Timetabling, ed. E.K. Burke and P Ross, Springer-Verlag (Lecture Notes in Computer Science), Springer Lecture Notes in Computer Science 1153, pages 283-295.

[De Causmaecker, 2002] De Causmaecker, Pl; Demeester, P.; Lu Y.; Vanden Berghe, G. (2002). *Using Web Standards for Timetabling*, Proceedings of the Fourth International Conference on Practice and Theory of Automated Timetabling. Gent, Belgium. Vol. 1, pp. 238-257.

[Díaz, 1996] Díaz, A., Glover, F., Ghaziri, H.M., Gonzalez, J.L., Laguna, M, Moscato, P. y Tseng, F.T. (1996). *Optimización Heurística y Redes Neuronales*. Ed. Paraninfo, Madrid.

[Dorigo, 1992] Dorigo, M. (1992). *Optimization, learning and natural algorithms*. Tesis doctoral, DEI, Politecnico di Milano, Italia. pp. 140.

[Dowsland, 2001] Dowsland, Cathryn y Díaz, Belarmino. *Diseño de Heurísticas y Fundamentos del Recocido Simulado*. (2001). Inteligencia Artificial. Revista iberoamericana de Inteligencia Artificial, No. 20. pp. 34-52.

[Dueck, 1990] Dueck G. and Sheuer T, 1990, *Threshold Accepting : A general purpose algorithm appearing superior to simulated Annealing*. Journal of Computational Physics 90, 161-175.

[Elmohamed, 1997] Elmohamed, Saleh y Fox, Geoffrey. (1997). *A comparison of Annealing Techniques for Academic Course Scheduling*. Selected papers from the Second International Conference on Practice and Theory of Automated Timetabling II. Ed. Edmund Burke and Mike Carter, Lecture Notes in Computer Science, Springer. Pages: 92 - 114.

[Faber, 1998]. Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. *Representing school Timetabling in a disjunctive logic programming language*. In Proceedings of the 13th Workshop on Logic Programming (WLP '98), 1998.

[Fallahi, 2004] Abdellah El Fallahi. (2004). *Entrenamiento de Redes Neuronales*. Tesis doctoral del Departamento de Estadística e Investigación Operativa, Universidad de Valencia. España.

[Feo, 1995] T. Feo, M. Resende. (1995). *Greedy randomized adaptative search*. Journal of Global Optimization 6 109–133.

[Fogel, 1962] Fogel, L. J. (1962). *Toward inductive inference automata*. In Proceedings of the International Federation for Information Processing Congress. Munich, 395–399.

[Frausto, 2000] Frausto Solís J, Sánchez Ante G., (2000). *Fundamentos de Lógica Computacional*, Editorial Trillas. México.

[Froilán, 2005] Imperial-Valenzuela, Froilán; Frausto, Juan y Sanvicente, Héctor. (2005). *Solving SAT Problems with TA Algorithms Using Constant and Dynamic Markov Chains Length*. AAIM 2005: 281-290.

[Fernandes, 1999] Fernandes C., Caldeira J., Melicio F., Rosa A., (1998). *High school weekly Timetabling by evolutionary algorithm*. Proceedings of the 1999 ACM symposium on Applied computing. ACM Press. Pages: 344 - 350.

[GehoWeb, 2005] GehoWeb. *Generación de Horarios de la Universidad de Vigo de España*. Sitio Web disponible en: http://www.ei.uvigo.es/. Fecha de consulta: Agosto 2005.

[Glover, 1989] F. Glover, (1989). *Tabu search—Part I*. ORSA J. Comput., vol. 1, no. 3, pp. 190–206,

[Glover, 1997] Glover, Fred y Laguna, Manuel. (1997). *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA.

[Goldberg, 1989] Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

[Gómez, 2005] Gómez R., Jose Luis. *Algoritmos Genéticos con Diversidad Forzada para la Resolución del Problema de Timetabling Educativo*. Tesis de Maestría. Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Cuernavaca. Agosto, 2005.

[Gti, 2004] GTI. *Software comercial para horarios* disponible en http://www.planningsoftwarelimited.co.uk/excellence.htm. Fecha de consulta: Junio, 2004.

[Guerequeta, 2000] Guerequeta, Rosa y Vallecillo , Antonio. (2000). *Técnicas de Diseño de Algoritmos*. 2ª. Ed. Servicio de Publicaciones de la Universidad de Málaga. España.

[Hajek, 1988] B. Hajek. (1988). *Cooling schedules for optimal annealing*. Mathematics of Operations Research, 13(2):311-329.

[Hansen, 1998] Hansen, Pierre y Mladenovic, Nenad. (1998). *An introduction to VNS*. S. Voss, S. Martello, I.H. Osman, C. Roucairol (Eds.), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, Boston.

[Hansen, 2001] Hansen, Pierre y Mladenovic, Nenad. (2001). *Variable neighborhood search*. Principles and applications. European Journal of Operational Research, 130, pp. 449-467.

[Hertz, 1991] A. Hertz, *Tabu Search For Large Scale Timetabling Problem*, European Journal of Operational Research, Vol. 31, N° 1, 1991, pp. 39-47.

[Hertz, 1992] A. Hertz, (1992). *Finding a feasible course schedule using Tabu Search*, Discrete Applied Mathematics, 35, pp. 255-270.

[Hertz, 1992a] A. Hertz, (1992). *Tabu Search for large scale Timetabling problems*. European Journal or Operational Research, 54, pp. 39-47.

[HEUR, 2005] Red HEUR. *Investigación y desarrollo en Optimización*. Sitio Web disponible en: http://heur.uv.es/ventana/descripcion.php. Fecha de consulta: Agosto, 2005.

[Holland, 1992] J. H. Holland. (1992). *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, Massachusetts, second edition.

[Ingber, 1993a] Ingber, L., (1993). *Adaptive simulated annealing (ASA): Lessons learned*. Control and Cybernetics. Vol. 25 pp. 33.

[Ingber, 1993b] L. Ingber. *Simulated Annealing: Practice versus Theory*. J. Mathl. Comput. Modelling 18(1993) pp. 29-57.

[Ingber, 2005] Ingber, L. Sitio Web disponible en http://www.ingber.com/ Fecha de consulta: Agosto, 2005.

[Kern, 2004] Kern, Stefan; D. Müller, Sibylle; Hansen, Nikolaus; Büche, Dirk; Ocenasek, Jiri y Koumoutsakos, Petros. (2004). *Learning Probability Distributions in Continuous Evolutionary Algorithms – A Comparative Review*. Natural Computing 3(1): 77-112.

[Kingston, 2001] Kingston J.H. (2001). *Modelling Timetabling Problems with STTL*. Practice and Theory of Automated Timetabling III, selected papers from proceedings of the Third International Conference on Practice and Theory of Automated Timetabling, Konstanz 2000. Springer-Verlag, vol. 2079, pp. 309-321.

[Kirkpatrick, 1983] Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. (1983). *Optimization by Simulated Annealing*, Science, Vol 220, Number 4598, pages 671-680.

[Kostuch, 2003] Kostuch, P.A. (2003). *Timetabling Competition - SA-based Heuristic*. Selected papers from proceedings of the Third International Conference on Practice and Theory of Automated Timetabling. Springer-Verlag, vol. 2079, pages 1-14.

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, USA.

[Lantiv, 2004] *Lantiv Timetable*. Software comercial disponible en http://www.lantiv.com/6/automatic.asp. Fecha de consulta: Mayo, 2004.

[Larrañaga, 2002] Larrañaga, Pedro; Lozano, J.A. (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publisher. Boston.

[Lierenski, 2003] Legierski, Wojciech y Widawski, Rafal. (2003). *System of Automated Timetabling*. Proceedings of the 25th International Conference Information Technology Interfaces ITI 2003, pp. 495-500.

[Lundy, 1986] M. Lundy y A. Mees. (1986). *Convergence of an annealing algorithm.* Mathematical Programmming, Vol. 34: 111-124.

[Mahdi 2003] Mahdi O. y Zainuddin R. (2003). *Using a genetic Algorithm optimizer tool to generate good quality timetables*. Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on Volume 3. Pages: 1300 - 1303.

[Marte, 2000] Marte, Michael. (2000). *Towards Constraint-Based Grammar School Timetabling*. In E. Burke and W. Erben, editors. Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, pages 222-224.

[Marte, 2003] Marte, Michael. (2003). *Models and Algorithms for School Timetabling – A constraint-Programming Approach*. Doctoral thesis, Institut für Informatik der Universität Manchen.

[Martí, 2003] Martí, Rafael. (2003). *Procedimientos Metaheurísticos en Optimización Combinatoria*. Matemátiques 1(1), pages 3-62.

[MECE, 2005] *Sistema para la generación de Horarios para IES* (Institutos de Educación Secundaria).Sitio Web del Ministerio de Educación y Ciencia de España, disponible en http://www.cnice.mec.es. Fecha de consulta: Marzo 2005.

[Melicio, 1999] Melicio, F., Caldeira, P. y Rosa, A. (1999). *Solving the Timetabling problem with Simulated Annealing*. Proceedings of the First International Conference on Enterprise Information Systems (ICEIS'99), 272-279.

[Metropolis, 1953] Metropolis, N; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H. y Teller, E. (1953). *Equation of state calculation by fast computing machines*. Journal of Chemistry Physics, 21: 1087-1091.

[Milton, 2003] Milton, J. Susan y Arnold Jesse. (2003). Probabilidad y Estadística con aplicaciones para ingeniería y ciencias computacionales. 4ª ed. McGraw Hill Interamericana. México.

[Mimosa, 2004] Mimosa. *Software comercial para horarios*, disponible en http://www.lantiv.com/6/automatic.asp. Fecha de consulta: Junio, 2004.

[Paecher, 1994] Paecher B., Cumming A., Luchian H. y Periuc M. (1994). *Two solutions to the General Timetable Problem using Evolutionary Methods*. In Proceedings of the IEEE World Congress in Computational Intelligence. Pages 300-305.

[Papadimitrou, 1982] Papadimitrou, Christos; Steiglitz, Kenneth. (1982). *Combinatorial Optimization, Algorithms and Complexity*. Dover Publications Inc., Nueva York, EEUU.

[Parada, 2005] Parada, Victor. *Optimos: Servicio de Optimización On-Line*. Sitio web de la Universidad de Santiago de Chile disponible en http://www.optimos.usach.cl/ Fecha de consulta: 7 de marzo de 2005.

[PATAT, 2005] *Practice and Theory of Automated Timetabling*. Sitio web disponible en http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml. Fecha de consulta: 25 de Agosto de 2005.

[Pinedo, 2002] M. Pinedo. (2002). *Scheduling: Theory, Algorithms and Systems*. 2nd. Ed. Prentice Hall, New York.

[Ross, 1994] Peter Ross, Dave Corne. (1994). *Applications of Genetic Algorithms*. AISB Quarterly 89, Ed. T.C. Fogarty, 23–30.

[Rossi-Doria, 2002] Rossi-Doria, O., Sampels, M., Birattari, M., Chiarandini, M., Dorigo, M., Gambardella, L.M., Knowles, J., Manfrin, M., Mastrolilli, M., Paechter, B., Paquete, L., Stützle T. (2003) *A comparison of the performance of different metaheuristics on the Timetabling problem*. Practice and Theory of Automated Timetabling IV: 4th International Conference, PATAT 2002, Selected Revised Papers. Springer. LNCS 2740. pp. 329-351.

[Sanvicente, 2004] Sanvicente-Sanchez, Hector y Frausto, Juan. (2004). *Method to Establish the Cooling Scheme in Simulated Annealing Like Algorithms*. International Conference, Assis, Italia. ICCSA'2004. LNCS Vol. 3095. 755-763.

[SIGEH, 2005] SIGEH. Sistema Generador de Horarios de la Facultad de Ciencias de la Universidad Autónoma de Baja California. Sitio Web disponible en http://lcc.ens.uabc.mx:8080/~sgh/index.jsp. Fecha de consulta: Marzo 2005.

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

[Schaerf, 2001] Schaerf, Andrea and Di Gaspero, Luca. (2001). *Local search techniques for educational Timetabling problems*. Proceeding of the 6th International Symposium on Operational Research in Slovenia (SOR-'01), Preddvor, Slovenia. pp.13-23.

[Smith, 1975] Smith, Graham. *On Maintenace of the Opportunity List for Class-Teacher Timetable Problems*. Communications of the ACM. April, 1975. Vol. 18 Num 4. pp 203-208.

[Socha, 2002] Socha, K.; Knowles, J.; Sampels, M. *A MAX-MIN Ant System for the University Timetabling Problem*. In Proceedings of the 3rd International Workshop on Ant Alghorithms, ANTS 2002, Lecture Notes in Computer Science, Vol. 2463, Springer, pp. 1-13. 2002.

[Stewart, 2002] Stewart, James. Cálculo. *Trascendentes Tempranas*. 4ª ed. Ed. Thomson Learning. México, 2002

[Taha, 1995] Taha, Hamdy. *Investigación de Operaciones*. Alfaomega Grupo Editor. México, 1995.

[Timetabler, 2004] Timetabler. Software comercial para horarios disponible en http://www.timetabler.com/. Fecha de consulta: Junio, 2004.

[Voudouris, 1995] C. Voudouris and E.P.K. Tsang. (1995). *Guided Local Search*. Technical Report CSM-247, Department of Computer Science, University of Essex.

[Werra, 1985] Werra, D. de. 1985. *An introduction to Timetabling*. European Journal of Operational Research, 19: 151-162.

[Wright, 1996] M. Wright, (1996). *School Timetabling Using Heuristic Search*. Journal of the Operational Research Society, Vol. 47, N° 3, 1996, pp. 347-357.

[Yu, 2002] Yu, Enzhe; Sunga, Ki Seok. (2002). *A Genetic Algorithm for a University Weekly Courses Timetabling Problem*. International transactions in Operational Research. 9. pp 703 – 717.

Apéndices

A. Definición de Campos de la Base de Datos

1. Alumno

Catálogo de alumnos. Almacena información personal de los alumnos de la institución.

Nombre del Campo	Tipo de Campo	Descripción
Matricula	varchar(10)	Matricula registrada del alumno. Es clave única.
Nombre	varchar(30)	Nombre(s) del alumno
Ap_paterno	varchar(30)	Apellido paterno del alumno
Ap_materno	varchar(30)	Apellido materno del alumno
Cve_plan	int(3)	Clave del plan académico inscrito (Tabla Plan)
Cve_grupo	int(4)	Clave del grupo al que pertenece (Tabla Grupo)
Cve_modelo	int(5)	Clave del modelo educativo al que pertenece (Tabla Modelo_academico)

2. Asignación

Es la tabla en la que se almacenan los datos de la solución del problema, es decir, al final contendrá los datos del horario final encontrado.

Nombre del Campo	Tipo de Campo	Descripción
Cve_periodolectivo	int(5)	Clave del período lectivo al que corresponde la asignación (Tabla Periodo_lectivo)
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Cve_profesor	varchar(10)	Clave del profesor (Tabla Profesor)
Cve_horario	int(3)	Clave del horario (Tabla Horario)
Cve_aula	varchar(5)	Clave del aula (Tabla Aula)
Cve_grupo	int(4)	Clave del grupo al que pertenece la materia (Tabla Grupo)

3. Aula

Catálogo de aulas con las que cuenta la institución.

Nombre del Campo	Tipo de Campo	Descripción
Cve_aula	varchar(5)	Clave del aula
Nombre	varchar(30)	Nombre con que se conoce al aula
Capacidad	int(3)	Número de alumnos que puede albergar el aula
Cve_departamento	int(3) NULL	Clave del departamento responsable del aula (Tabla Departamento)

4. Característica

Catálogo de características diversas existentes. Puede almacenar características de las aulas, los profesores, las asignaturas, etc.

Nombre del Campo	Tipo de Campo	Descripción
Cve_caracteristica	int(3) *	Clave de la característica
Nombre	varchar(30)	Nombre o descripción de la característica

5. Caracteristica_aula

Contiene la relación de aulas con sus respectivas características.

Nombre del Campo	Tipo de Campo	Descripción
Cve_aula	varchar(5)	Clave del aula (Tabla Aula)
Cve_caracteristica	int(5)	Clave de la característica (Tabla característica)

6. Caracteristica_materia

Contiene la relación de materias con sus respectivas características.

Nombre del Campo	Tipo de Campo	Descripción
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Cve_caracteristica	int(5)	Clave de la característica (Tabla característica)

7. Caracteristica_profesor

Contiene la relación de profesores con sus respectivas características.

Nombre del Campo	Tipo de Campo	Descripción
Cve_profesor	varchar(10)	Clave del profesor (Tabla Profesor)
Cve_caracteristica	int(5)	Clave de la característica (Tabla característica)

8. Carga_academica_previa

Contiene la asignación de materias que impartirán los profesores en el periodo lectivo.

Nombre del Campo	Tipo de Campo	Descripción
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Cve_profesor	varchar(10)	Clave del profesor (Tabla Profesor)

9. Categoría

Catálogo de categorías en las que se encuentran los profesores

Nombre del Campo	Tipo de Campo	Descripción
Cve_categoria	int(5)	Clave de la categoría
Nombre	varchar(50)	Nombre o descripción de la categoría

10. Ciudad

Catálogo de ciudades en las que radican los profesores

Nombre del Campo	Tipo de Campo	Descripción
Cve_ciudad	int(5) *	Clave de la ciudad
Cve_estado	int(5)	Clave del estado (Tabla Estado)
Nombre	varchar(30)	Nombre de la ciudad

11. Demanda

Contiene las materias que se deben impartir en el período lectivo correspondiente.

Nombre del Campo	Tipo de Campo	Descripción
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Cve_periodolectivo	int(5)	Clave del periodo lectivo (Tabla Periodo_lectivo)
Num_grupos	int(2)	Numero de grupos que se requiere

12. Departamento

Catálogo de departamentos existentes en la institución

Nombre del Campo	Tipo de Campo	Descripción
Cve_departamento	int(3)	Clave del departamento
Nombre	varchar(30)	Nombre o descripción del departamento

13. Disponibilidad_profesor

Horario de disponibilidad del profesor, para impartir clases. Un profesor puede tener varios registros, si entra y sale varias veces de la institución.

Nombre del Campo	Tipo de Campo	Descripción
Cve_profesor	varchar(10)	Clave del profesor (Tabla Profesor)
Hr_inicio	time	Hora de entrada
Hr_fin	time	Hora de salida
Dia	varchar(10)	Día de la semana
Cve_turno	int(1)	Clave del turno (Tabla Turno)

14. Equivalencia

Permite manejar la equivalencia entre materias con diferente nombre. Ambas claves corresponden a la tabla Materia.

Nombre del Campo	Tipo de Campo	Descripción
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Cve_equivalencia	varchar(5)	Clave de la materia equivalente (Tabla Materia)

15. Estado

Catalogo de estados

Nombre del Campo	Tipo de Campo	Descripción
Cve_estado	int(5)	Clave del estado
Nombre	varchar(30)	Nombre del estado

16. Grado_academico

Catálogo de grados académicos de los profesores

Nombre del Campo	Tipo de Campo	Descripción
Cve_grado	int(3) *	Clave del grado académico
Nombre	varchar(70)	Nombre o descripción del grado académico
Abreviatura	varchar(5)	Abreviatura del grado académico

17. Grupo

Catálogo de grupos de alumnos

Nombre del Campo	Tipo de Campo	Descripción
Cve_grupo	int(4) *	Clave del grupo
Nombre	varchar(30)	Nombre o descripción del grupo
Cve_turno	int(1)	Clave del turno (Tabla Turno)

18. Grupo_materia

Relación de materias que lleva un grupo en cierto período lectivo

Nombre del Campo	Tipo de Campo	Descripción
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Cve_grupo	int(4)	Clave del grupo al que pertenece la materia (Tabla Grupo)
Cve_periodolectivo	int(5)	Clave del periodo lectivo (Tabla Periodo_lectivo)
Capacidad	int(3)	Número máximo de alumnos

19. Historial_academico

Historial académico de un alumno

Nombre del Campo	Tipo de Campo	Descripción
Matricula	varchar(10)	Clave del alumno (Tabla Alumno)
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Cve_periodolectivo	int(5)	Clave del periodo lectivo (Tabla Periodo_lectivo)
Calificacion	int(3)	Nota que obtuvo el alumno en la materia

20. Horario

Horarios disponibles para impartir las materias

Nombre del Campo	Tipo de Campo	Descripción
Cve_horario	int(10)	Clave del horario
Dia	varchar(10)	Día de la semana
Hr_inicio	Time	Hora de inicio
Hr_fin	Time	Hora de término
Cve_periodolectivo	int(5)	Clave del periodo lectivo (Tabla Periodo_lectivo)

21. Materia

Catalogo de materias que pueden ser impartidos en los diferentes planes educativos

Nombre del Campo	Tipo de Campo	Descripción
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Nombre	varchar(50)	Nombre de la materia
Hr_teoricas	float(2)	Número de horas teóricas requeridas a la semana
Hr_practicas	float(2)	Número de horas prácticas requeridas a la semana
Cve_modelo	varchar(5)	Clave del modelo educativo al que pertenece (Tabla Modelo_academico)
Creditos	int(3)	Créditos obtenidos por cursar esa materia
Cve_nivel	int(3)	Clave del nivel de la materia (Tabla Nivel_materia)

22. Materia_plan

Contiene la lista de planes, con todas las materias que pertenecen a cada uno

Nombre del Campo	Tipo de Campo	Descripción
Cve_plan	int(3) *	Clave del plan (Tabla Plan)
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Optativa	bool	Para saber si la materia es optativa o no en ese plan

23. Modelo_academico

Catálogo de modelos académicos

Nombre del Campo	Tipo de Campo	Descripción
Cve_modelo	int(5) *	Clave del modelo académico
Nombre	varchar(30)	Nombre o descripción del modelo académico

24. Nivel_materia

Catálogo de niveles en los que se encuentran las materias. Por ejemplo, preparatoria, profesional o posgrado.

Nombre del Campo	Tipo de Campo	Descripción
Cve_nivel	int(3)	Clave del nivel
Nombre	varchar(30)	Nombre o descripción del nivel

25. Periodo_lectivo

Catálogo de periodos lectivos.

Nombre del Campo	Tipo de Campo	Descripción
Cve_periodolectivo	int(5) *	Clave del periodo lectivo
Nombre	varchar(30)	Nombre o descripción del periodo lectivo

26. Plan

Catalogo de planes educativos existentes en la institución.

Nombre del Campo	Tipo de Campo	Descripción
Cve_plan	int(3) *	Clave del plan
Nombre	varchar(50)	Nombre o descripción del plan
Abreviatura	Varchar(5)	Abreviatura del plan

27. Preasignación

Contiene las preasignaciones que deben respetarse al construir el horario, en caso de haber.

Nombre del Campo	Tipo de Campo	Descripción
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Cve_profesor	varchar(10)	Clave del profesor (Tabla Profesor)
Cve_aula	varchar(5)	Clave del aula (Tabla Aula)
Cve_horario	int(3)	Clave del horario (Tabla Horario)
Cve_periodolectivo	int(5)	Clave del periodo lectivo (Tabla Periodo_lectivo)
Cve_departamento	int(3)	Clave del departamento responsable de la preasignación (Tabla Departamento)

28. Preferencias_materia_profesor

Contiene las preferencias de un profesor hacia las materias que puede él dar.

Nombre del Campo	Tipo de Campo	Descripción
Cve_profesor	varchar(10)	Clave del profesor (Tabla Profesor)
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Ponderacion	int(3)	El grado de preferencia que tiene el profesor. Mientras más alto sea el número es mayor la preferencia.

29. Profesor

Catálogo de profesores que se encuentran en la planta docente. Nota: Las características adicionales se pueden incluir en la tabla de Características y Caracterisitica_profesor.

Nombre del Campo	Tipo de Campo	Descripción
Cve_profesor	varchar(10)	Clave del profesor
Nombre	varchar(30)	Nombre(s) del profesor
Ap_paterno	varchar(30)	Apellido paterno del profesor
Ap_materno	varchar(30)	Apellido materno del profesor
Cve_ciudad	int(5)	Clave de la ciudad donde reside (Tabla Ciudad)
Cve_estado	int(5)	Clave del estado donde reside (Tabla Estado)
Direccion	varchar(80)	Dirección del Profesor
CP	int(5)	Código postal del profesor
Correo_electronico	varchar(80)	Correo electrónico del profesor
Cve_grado	int(3)	Clave del grado académico (Tabla Grado)
Cve_categoria	int(3)	Clave de la categoría (Tabla Categoría)
Telefono	varchar(20)	Teléfono del profesor
Num_horas	int(2)	Num. De horas que trabaja a la semana
Num_grupos	int(2)	Num. Máximo de grupos que debe atender. 0 si no está dando clases por alguna razón.
Cve_turno	Int(1)	Clave del turno (Tabla Turno)

30. Requisito

Permite saber cuáles materias (Cve_requisito) se requiere haber cursado previamente a la materia en cuestión (Cve_materia). Pueden haber varios registros por materia.

Nombre del Campo	Tipo de Campo	Descripción
Cve_materia	varchar(10)	Clave de la materia (Tabla Materia)
Cve_requisito	varchar(5)	Clave de la materia requerida (Tabla Materia)
Tipo	int(2)	Indica el tipo de precedencia

31. Turno

Catálogo de turnos laborales o educativos (matutino, vespertino, etc.).

Nombre del Campo	Tipo de Campo	Descripción
Cve_turno	int(1) *	Clave del turno
Nombre	varchar(20)	Nombre o descripción del turno de trabajo

***) Campo autoincrementable**

B. Ejecuciones de los algoritmos para el caso PATAT

Tabla B-1. Ejecuciones del algoritmo RS con α=0.75

	Corrida1		Corrida2		Corrida3		Corrida4		Corrida5		Promedio	
Caso	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***
C01	219	786	221	789	202	756	195	758	207	768	**208.8**	**771**
C03	281	739	268	768	270	756	255	775	258	806	**266.4**	**769**
C04	564	852	475	873	527	853	506	855	551	904	**524.6**	**867**
C06	271	771	340	796	288	792	284	785	286	804	**293.8**	**790**
C09	153	871	146	892	134	930	159	927	171	886	**152.6**	**901**
C11	240	892	189	799	227	868	216	814	214	801	**217.2**	**835**
C12	280	807	256	731	280	815	293	734	277	737	**277.2**	**765**
C15	360	821	338	790	356	796	404	829	372	786	**366**	**804**
C19	248	911	273	894	248	892	243	917	208	874	**244**	**898**

Tabla B-2. Ejecuciones del algoritmo RSSM (sólo cadena de Markov) con α=0.75

	Corrida1		Corrida2		Corrida3		Corrida4		Corrida5		Promedio	
Caso	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***
C01	220	346	201	343	216	354	194	343	214	343	**209**	**346**
C03	283	329	268	335	272	336	267	325	266	331	**271.2**	**331**
C04	534	384	535	384	539	383	509	377	576	381	**538.6**	**382**
C06	302	348	354	334	320	354	340	332	335	332	**330.2**	**340**
C09	153	413	144	397	152	404	144	386	140	385	**146.6**	**397**
C11	187	365	182	353	208	358	210	361	188	369	**195**	**361**
C12	286	344	295	326	287	330	307	325	266	323	**288.2**	**330**
C15	391	354	446	334	378	350	409	354	418	342	**408.4**	**347**
C19	221	408	238	404	270	410	224	396	255	412	**241.6**	**406**

Tabla B-3. Ejecuciones del algoritmo RSST con α=0.75

	Corrida1		Corrida2		Corrida3		Corrida4		Corrida5		Promedio	
Caso	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***
C01	259	243	236	244	230	238	249	243	241	243	**243**	**242**
C03	274	250	312	233	270	234	293	241	293	245	**288.4**	**241**
C04	661	266	635	261	692	260	721	254	646	273	**671**	**263**
C06	458	243	421	254	429	241	474	246	466	242	**449.6**	**245**
C09	172	279	185	268	183	278	194	294	160	268	**178.8**	**277**
C11	233	289	218	283	229	280	229	303	224	333	**226.6**	**298**
C12	324	299	352	272	327	332	301	282	292	314	**319.2**	**300**
C15	478	274	446	290	474	319	499	304	442	298	**467.8**	**297**
C19	469	298	280	265	336	306	304	283	324	296	**342.6**	**290**

Tabla B-4. Ejecuciones del algoritmo RS con α=0.85

	Corrida1		Corrida2		Corrida3		Corrida4		Corrida5		Promedio	
Caso	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***
C01	208	1373	196	1445	185	1404	180	1360	178	1347	**189.4**	**1386**
C03	269	1376	250	1400	242	1534	271	1931	234	1376	**253.2**	**1523**
C04	467	1725	439	1544	464	1638	532	1507	512	1487	**482.8**	**1580**
C06	264	1525	260	1488	238	1461	270	1395	274	1575	**261.2**	**1489**
C09	131	1589	141	1538	153	1645	137	1517	138	1568	**140**	**1571**
C11	186	1466	193	1422	183	1552	193	1456	193	1490	**189.6**	**1477**
C12	276	1362	302	1336	286	1357	291	1367	238	1280	**278.6**	**1340**
C15	319	1349	317	1444	316	1429	346	1433	319	1358	**323.4**	**1403**
C19	187	1638	185	1701	208	1591	207	1595	214	1539	**200.2**	**1613**

Tabla B-5. Ejecuciones del algoritmo RSSM (sólo cadena de Markov) con α=0.85

	Corrida1		Corrida2		Corrida3		Corrida4		Corrida5		Promedio	
Caso	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***
C01	209	639	187	643	215	657	206	655	205	635	**204.4**	**645.8**
C03	256	907	274	661	281	691	236	642	277	634	**264.8**	**707**
C04	456	931	502	732	505	734	430	738	467	714	**472**	**769.8**
C06	268	944	285	646	273	634	224	639	271	641	**264.2**	**700.8**
C09	134	773	157	763	165	730	148	795	100	863	**140.8**	**784.8**
C11	192	668	183	679	212	674	188	681	229	701	**200.8**	**680.6**
C12	257	840	245	648	259	655	264	652	234	651	**251.8**	**689.2**
C15	336	683	348	647	338	702	340	648	422	731	**356.8**	**682.2**
C19	219	778	217	763	182	752	201	771	208	761	**205.4**	**765**

Tabla B-6. Ejecuciones del algoritmo RSST con α=0.85

	Corrida1		Corrida2		Corrida3		Corrida4		Corrida5		Promedio	
Caso	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***
C01	198	478	202	483	191	472	193	499	193	484	**195.4**	**483.2**
C03	280	526	268	502	258	474	264	468	244	480	**318.8**	**509.4**
C04	537	601	567	589	538	571	609	582	590	580	**607.4**	**566.6**
C06	367	472	332	472	356	487	297	495	321	428	**334.6**	**470.8**
C09	130	576	145	576	140	549	132	669	138	573	**137**	**588.6**
C11	187	507	197	521	191	511	182	556	193	524	**190**	**523.8**
C12	275	495	318	479	328	469	299	472	267	471	**297.4**	**477.2**
C15	418	570	456	536	453	534	396	525	469	535	**438.4**	**540**
C19	222	699	233	584	248	710	223	599	225	581	**230.2**	**634.6**

Tabla B-7. Ejecuciones del algoritmo RS con α=0.95

	Corrida1		Corrida2		Corrida3		Corrida4		Corrida5		Promedio	
Caso	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***
C01	153	4334	169	4211	182	4273	168	4303	171	4267	**168.6**	**4277.6**
C03	198	4398	242	4627	213	4154	211	4216	243	4178	**221.4**	**4314.6**
C04	400	4906	412	5020	401	4892	346	4958	385	4907	**388.8**	**4936.6**
C06	187	4435	193	4435	210	4227	181	4158	180	4611	**190.2**	**4373.2**
C09	130	4306	114	5006	106	4744	98	4853	109	4631	**111.4**	**4707.9**
C11	153	4253	160	4742	158	4505	162	4397	142	4345	**155**	**4448.3**
C12	231	4006	244	4521	202	4259	225	4125	208	4213	**222**	**4224.9**
C15	321	3446	281	4394	297	4622	258	4525	288	4617	**289**	**4320.8**
C19	172	5431	194	4868	209	5544	162	5278	157	5231	**178.8**	**5270.4**

Tabla B-8. Ejecuciones del algoritmo RSSM (sólo cadena de Markov) con α=0.95

	Corrida1		Corrida2		Corrida3		Corrida4		Corrida5		Promedio	
Caso	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***
C01	139	2426	174	2444	182	2519	172	2494	187	2665	**170.8**	**2509.6**
C03	232	2369	236	2406	222	2429	227	2396	229	2617	**229.2**	**2443.4**
C04	369	2666	364	2709	471	2894	368	2913	416	2886	**397.6**	**2813.6**
C06	201	2455	175	2838	163	2509	199	2418	244	2448	**196.4**	**2533.6**
C09	109	2692	109	2783	93	3005	113	2692	112	2867	**107.2**	**2807.8**
C11	143	2461	160	2769	158	3074	156	2865	149	2600	**153.2**	**2753.8**
C12	217	2446	217	2361	202	2596	191	2438	243	2520	**214**	**2472.2**
C15	287	2663	286	2428	269	2613	247	2382	251	2583	**268**	**2533.8**
C19	153	2804	155	2773	186	3096	156	2975	173	2991	**164.6**	**2927.8**

Tabla B-9. Ejecuciones del algoritmo RSST con α=0.95

	Corrida1		Corrida2		Corrida3		Corrida4		Corrida5		Promedio	
Caso	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***	**viol**	***seg***
C01	157	1833	176	1820	177	1933	184	1794	147	2016	**168.2**	**1879.2**
C03	224	1966	242	1740	223	2014	220	1775	225	1930	**226.8**	**1885**
C04	433	1991	420	2001	432	2265	411	2148	401	2160	**419.4**	**2113**
C06	167	1825	172	1852	205	2014	236	1979	197	1986	**195.4**	**1931.2**
C09	125	1981	105	2144	120	1992	97	2007	117	2224	**112.8**	**2069.6**
C11	148	2044	161	1821	169	1964	151	1887	164	1808	**158.6**	**1904.8**
C12	223	1882	221	1666	244	1819	212	2874	200	2092	**220**	**2066.6**
C15	302	2007	291	1742	287	1910	334	1835	303	1854	**303.4**	**1869.6**
C19	156	1978	132	1965	138	1917	172	2144	182	2047	**156**	**2010.2**

C. Manual de usuario del sistema de Horarios para la UADY (*SACAHO*)

El sistema implementado de Asignación de Cargas Académicas y Horarios (SACAHO) ha sido diseñado para proporcionar funcionalidad a los algoritmos desarrollados en el presente trabajo de tesis.

El sistema está diseñado para funcionar con los datos de cualquier universidad, aunque los algoritmos que realizan la asignación de Cargas Académicas y la asignación de Horarios fue diseñado para satisfacer las políticas propias de la Universidad Autónoma de Yucatán. El sistema fue realizado en trabajo conjunto con José Luis Gómez Ramos [Gómez, 2005] y Federico Alonso Pecina [Alonso, 2005] como parte de un proyecto global de la Cátedra de Optimización Combinatoria del Instituto Tecnológico de Monterrey, Campus Cuernavaca.

Al iniciar el sistema, se puede observar la pantalla principal que es la que se ilustra en la Figura A-1.

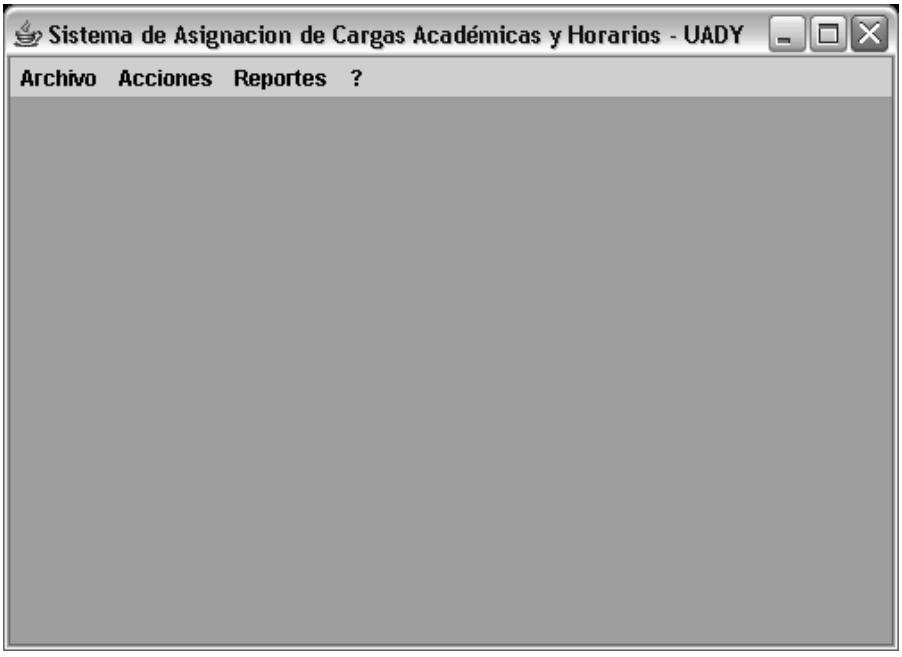

Figura A-1. Pantalla principal del SACAHO para la UADY

El menú principal se compone de 4 elementos: ***Archivo***, ***Acciones***, ***Reportes*** y ***?*** (Ayuda). El primero de ellos a su vez, contiene 2 opciones: ***catálogos***, donde se pueden capturar y visualizar los datos de los diferentes catálogos del sistema, y ***salir***, para abandonar el mismo.

Los catálogos disponibles son ***Profesores***, ***Materias***, ***Grupos***, ***Aulas***, ***Departamentos***, ***Horarios*** y ***Periodos Lectivos***. Y a su vez, para un Profesor es posible capturar sus ***Datos*** personales, sus ***Preferencias*** y su ***Disponibilidad*** de horario. En el caso de los Grupos, se cuenta con módulos para capturar los ***Datos*** del grupo y la ***Asignación de materias*** del mismo. Las Figuras A-2 y A-3 ilustran los menús descritos.

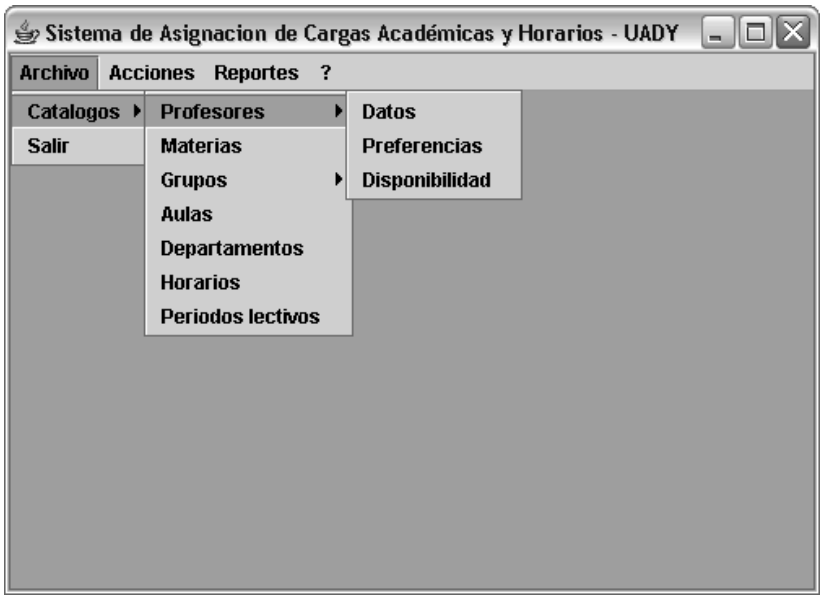

Figura A-2. Menú Catálogos – Profesores

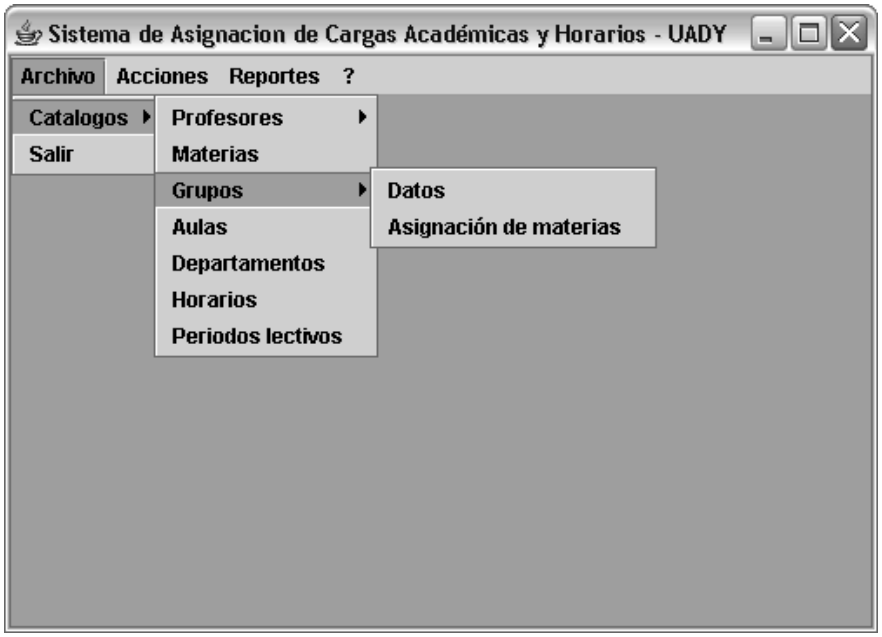

Figura A-3. Menú Catálogos – Grupos

Las pantallas han sido diseñadas para ser usadas de manera muy similar entre sí, de manera que al familiarizarse con el uso de alguna de ellas, no se tendrá dificultad en emplear las demás. Para cerrar una ventana es necesario hacer clic en la *X* que se encuentra en la parte superior derecha de cada una de ellas.

La pantalla para gestionar los ***Datos personales de los Profesores*** se ilustra en la Figura A-4. Inicialmente, el cursor se encuentra en el cuadro de texto esperando que el usuario introduzca una clave de profesor. Si la clave tecleada corresponde a un profesor que ya se encuentra en la base de datos, al presionar el botón <Verifica> se despliegan los datos del profesor; en caso contrario, se habilitan los cuadros para poder introducir y grabar los datos en el sistema. Como puede observarse, en dicha pantalla (Figura A-4), al igual que en todas las pantallas similares para gestión de catálogos, están integrados los módulos de Altas, Bajas y Modificaciones a los Datos de los profesores. Para poder eliminar a un profesor basta con localizarlo y presionar el botón <Eliminar>.

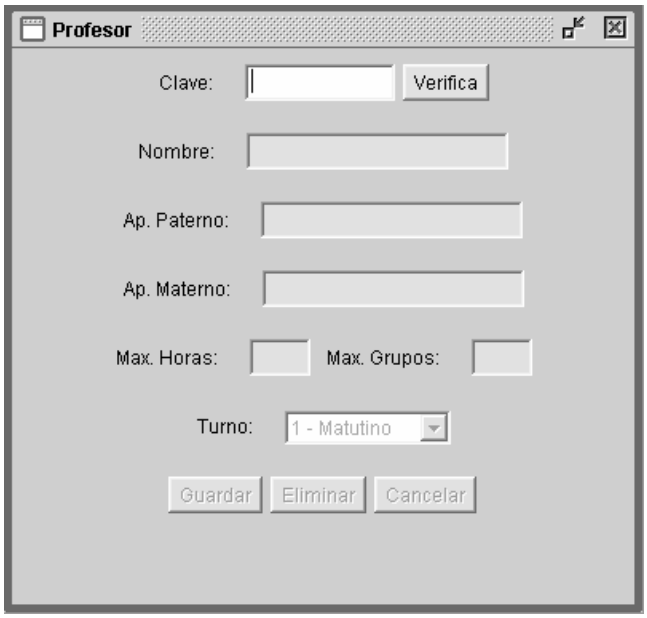

Figura A-4. Catálogo de Datos del Profesor

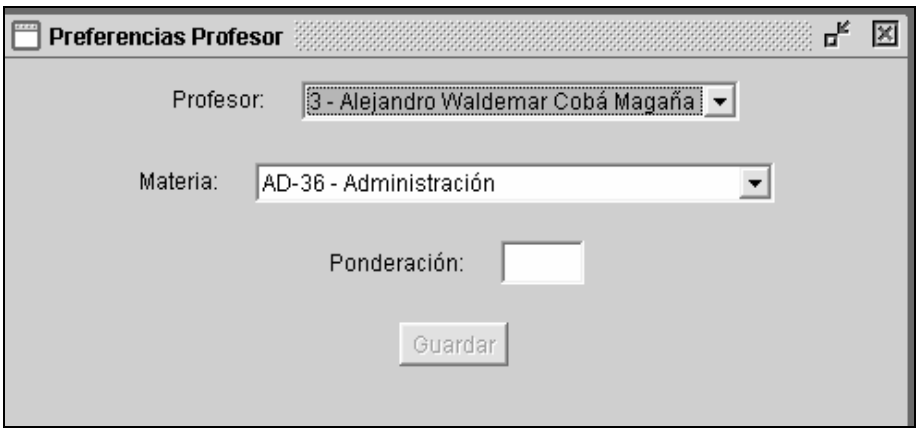

Figura A- 5. Preferencias de los profesores

Para capturar las ***Preferencias del profesor***, se cuenta con la pantalla ilustrada en la Figura A-5. Primero es necesario seleccionar en la primera lista desplegable al Profesor del que se desea capturar la preferencia, posteriormente en la segunda lista se debe seleccionar la materia deseada, enseguida se teclea el número que corresponde a la ponderación, mientras mayor sea éste número significa una mayor preferencia para impartir la materia en cuestión. Finalmente al oprimir el botón <Guardar> quedará almacenada la información. Este procedimiento se debe realizar para todas las materias para las que el profesor tenga cierta preferencia.

La ***Disponibilidad*** de los profesores se captura en la pantalla ilustrada en la Figura A-6. Primero se debe seleccionar el nombre del profesor para el que se desea capturar la disponibilidad, y posteriormente se debe hacer clic sobre todos aquellos horarios en los que dicho profesor se encuentre disponible, es decir, aquellos intervalos que se encuentren en su horario laboral. Finalmente se debe hacer clic sobre el botón <Guardar>. Los horarios seleccionados no se deshabilitan, pensando en que es común que varios profesores tengan el mismo horario; en caso de que esto no sea así, se puede oprimir el botón <Limpiar> antes de capturar la siguiente disponibilidad.

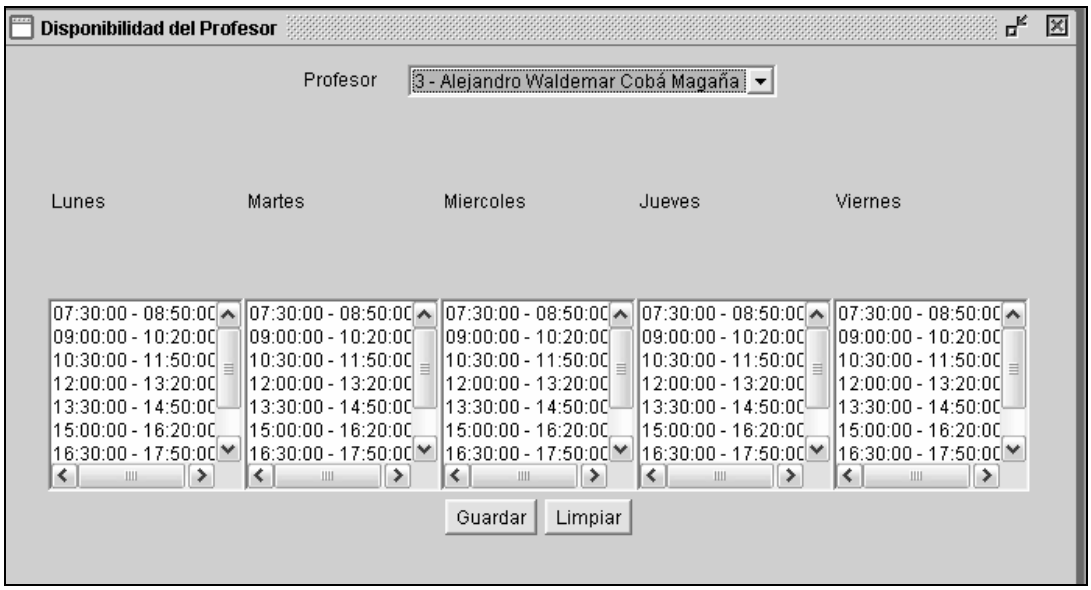

Figura A-6. Disponibilidad de los profesores

La pantalla para gestionar el ***Catálogo de Materias*** se ilustra en la Figura A-7. Al introducir la clave de una materia, si ésta se encuentra dada de alta en el sistema se visualizan los datos de la misma, en caso contrario se habilitan los cuadros de texto para introducir la información requerida. Para poder eliminar una materia basta con localizarla y presionar el botón <Eliminar>.

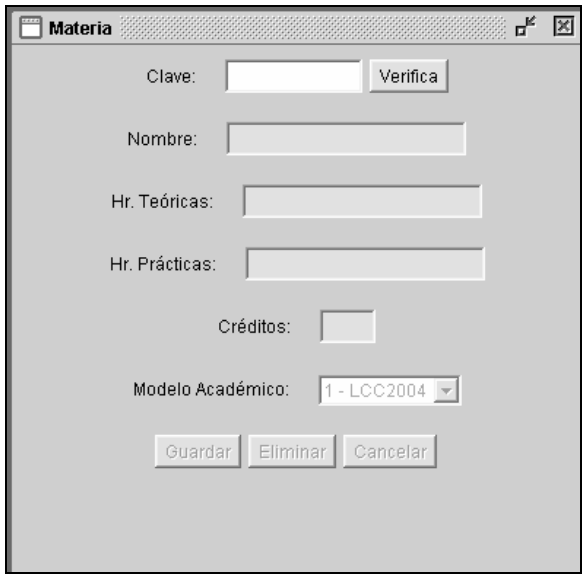

Figura A-7. Catálogo de Materias

El catálogo de ***Grupos*** se puede administrar desde la pantalla que presenta la Figura A-8. Una vez que se ha introducido la Clave del grupo, se presiona el botón <Verifica> y si dicha clave ya existe se despliegan los datos del grupo, en caso contrario se habilitan los

cuadros de texto para poder realizar la captura. Para poder eliminar un grupo basta con localizarlo y presionar el botón <Eliminar>.

Figura A-8. Catálogo de grupos

Para poder realizar la ***Asignación de materias*** que cursa un grupo, se cuenta con la pantalla representada en la Figura A-9. Primero es necesario seleccionar al grupo al que se desean capturar las materias, posteriormente se debe hacer clic en todas aquellas materias que correspondan al grupo, y finalmente se debe presionar el botón <Guardar>. Para capturar otro grupo con diferentes materias primero se debe presionar el botón <Limpiar>.

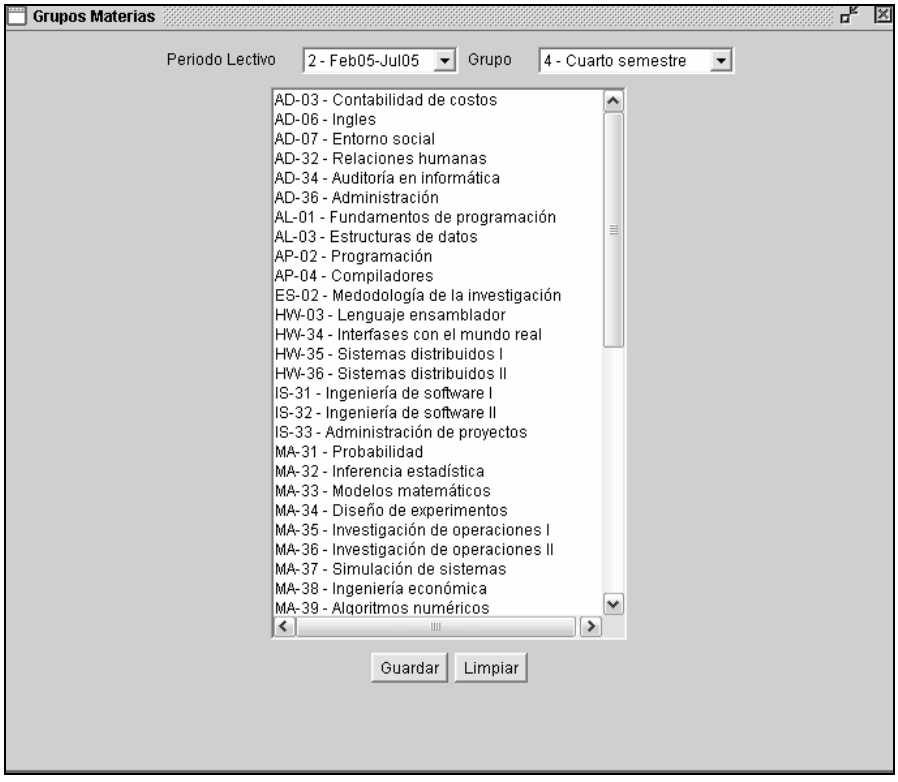

Figura A- 9. Asignación de materias a grupos

El catálogo de ***Aulas*** se puede administrar desde la pantalla que presenta la Figura A-10. Una vez que se ha introducido la Clave del aula, se presiona el botón <Verifica> y si dicha clave ya existe se despliegan los datos del aula; en caso contrario, se habilitan los cuadros de texto para poder realizar la captura. Para poder eliminar un aula basta con localizarlo y presionar el botón <Eliminar>.

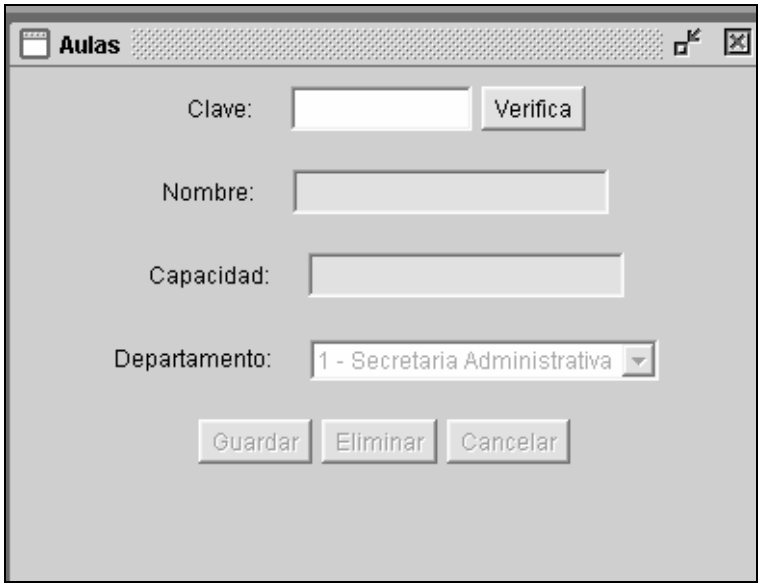

Figura A-10. Catálogo de Aulas

Para realizar la gestión de los ***Departamentos*** se cuenta con la pantalla ilustrada en la Figura A-11. Aquí únicamente es necesario introducir la clave del departamento para verificar si éste existe, en caso contrario se habilita el campo correspondiente al nombre del departamento para realizar la captura del dato.

Figura A-11. Catálogo de Departamentos

La gestión de ***Horarios*** se realiza de manera similar a los catálogos anteriores. En este caso, por Horario se entiende cada uno de los espacios disponibles para impartir clases. Un ejemplo es: "Lunes 7:00-8:30". La Figura A-12 ilustra la pantalla correspondiente. En este caso, el campo Día almacena el día de la semana y deben introducirse todos los días hábiles de la semana, cada uno con sus respectivos horarios disponibles. El campo Hr. Inicio corresponde a la hora en la que inicia el horario y el campo Hr. Fin a la hora en la que finaliza. La clave del período permite tener diferentes horarios para dos o más períodos lectivos, donde un período lectivo corresponde al lapso de tiempo en el que se cursan las materias, como puede ser un semestre, trimestre o año.

Figura A-12. Catálogo de Horarios

El último catálogo disponible es el de ***Períodos lectivos***. Aquí es donde deben introducirse los períodos lectivos cuando se desee trabajar con un nuevo horario. La Figura A-13 ilustra la pantalla de captura. Un ejemplo de un período lectivo es: "Septiembre-Enero 2005". El manejo de la pantalla es similar al de los catálogos anteriores.

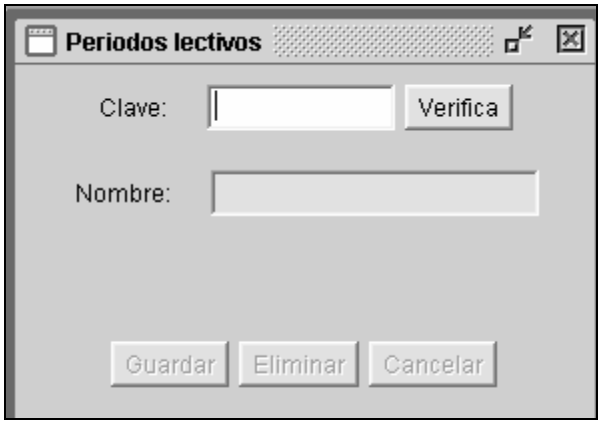

Figura A-13. Catálogo de Períodos Lectivos

El siguiente menú disponible corresponde al de ***Acciones*** y está representado en la Figura A-14. En este menú se pueden realizar dos opciones: administrar las ***Cargas Académicas*** o bien ***Calcular los horarios de clases***. Aquí, para poder realizar la asignación de horarios es necesario que la asignación de cargas ya esté realizada.

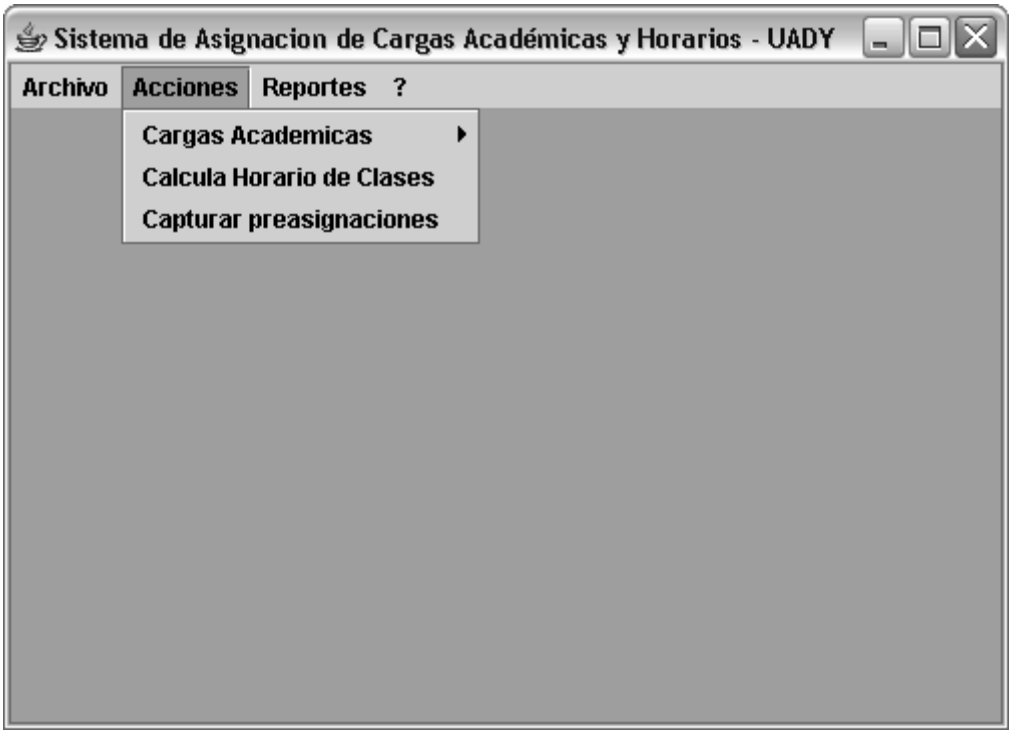

Figura A-14. Menú de Acciones

La pantalla para ***Calcular las Cargas Académicas*** está ilustrada en la Figura A-15 y corresponde al proceso que realiza la asignación Materia-Profesor. Aquí es necesario seleccionar el Período lectivo para el que se desea realizar el cálculo, esto se realiza en la lista desplegable de la parte superior de la pantalla, también se debe seleccionar el valor

deseado de α en la lista desplegable que se encuentra debajo del periodo lectivo y posteriormente se debe presionar el botón <Evalua>. Cabe mencionar que es aquí donde se ejecuta el algoritmo implementado, por lo que el tiempo de ejecución generalmente es tardado. Conforme el algoritmo se ejecuta, en los campos de la pantalla se van desplegando los resultados que va obteniendo.

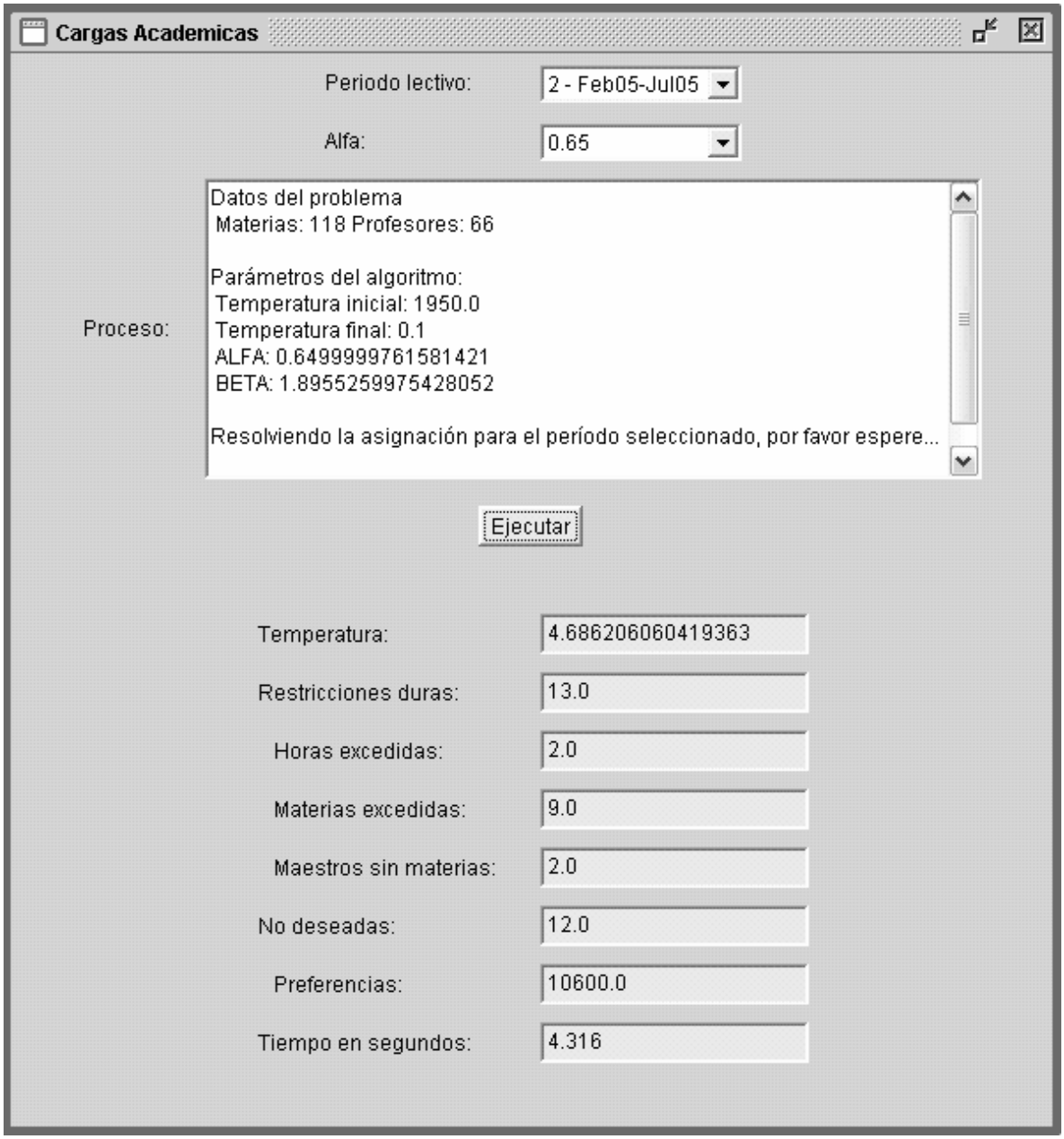

Figura A-15. Cargas académicas de los profesores

Una vez que se ha concluido la generación automática de cargas académicas, es necesario realizar la validación de las mismas. Para esto se encuentra el módulo de ***Validar Cargas Académicas*** ilustrado en la Figura A-16.

Con este módulo se debe tener especial cuidado pues las modificaciones manuales que se realicen deben ser responsabilidad del usuario del sistema. Una vez que se esté de acuerdo con la asignación de materias-profesor, se debe presionar el botón <Aceptar>, con lo cual esa asignación será tomada como la definitiva para el período lectivo seleccionado.

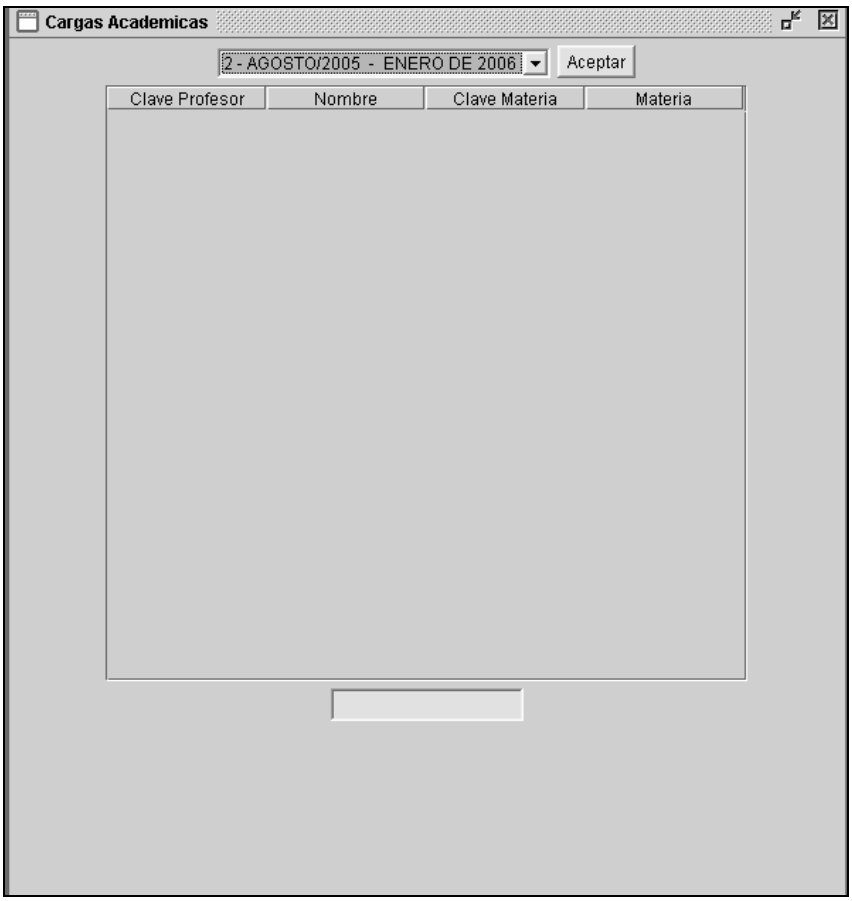

Figura A-16. Validación de cargas académicas

Una vez realizado lo anterior es posible realizar la ***Generación de Horarios***, lo cual se realiza en el módulo que se presenta en la Figura A-17. En esta pantalla es necesario seleccionar el período lectivo para el cual se desea generar el horario, después se debe presionar el botón <Evalua>, con lo cual da inicio el proceso realizado por el algoritmo.

El módulo de Generación de Horarios es similar al de Generación de Cargas Académicas ilustrado en la Figura A-15 que genera las cargas académicas. También consume una

cantidad considerable de tiempo y en los cuadros de texto se van desplegando los resultados obtenidos conforme avanza la ejecución del algoritmo. Para su utilización es necesario indicar el periodo lectivo para el que se desea resolver el horario y el valor de α con el que trabajará el algoritmo. Esto se hace en la parte superior de la pantalla.

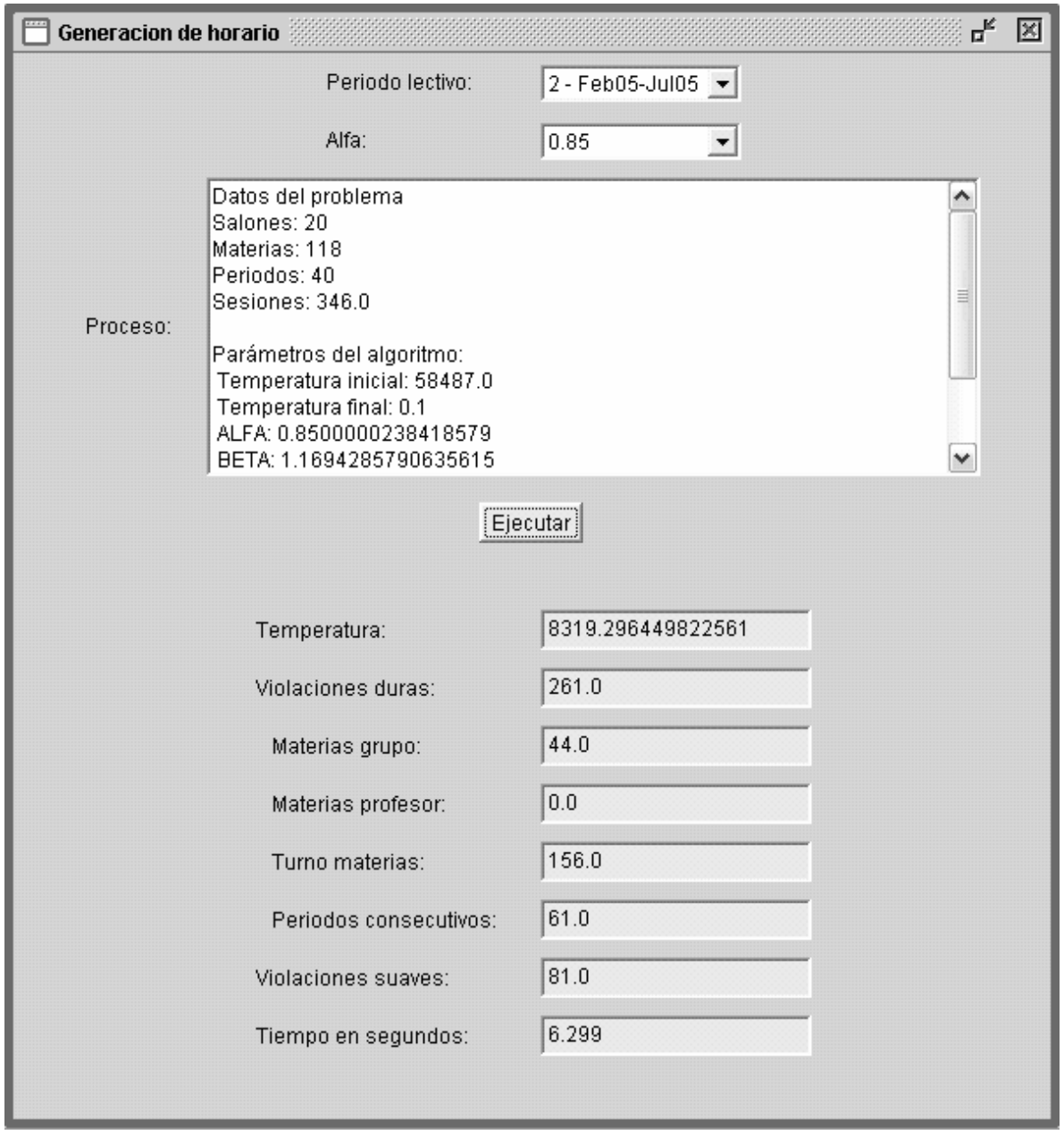

Figura A-17. Generación de horario con RSS

Para realizar la captura de ***Preasignaciones***, en caso de que se desee hacerlo, se cuenta con el módulo cuya pantalla se ilustra en la Figura A-18. Para introducir una preasignación, es necesario seleccionar primero la materia que se desea preasignar, posteriormente se deben elegir los datos para los que se desee realizar la preasignación, los cuales pueden ser uno o más de la lista de campos que se encuentran en la pantalla: Profesor, Aula u Horario. El dato del período es necesario para saber en qué período lectivo se desea realizar la preasignación. El campo Departamento es opcional y permite indicar cuál es el departamento que solicita la preasignación. En caso de que una materia tenga más de un aula u horario preasignado, se debe guardar una preasignación por cada uno de esos datos.

Figura A-18. Captura de Preasignaciones

El tercer menú disponible en el sistema es el de generación de ***Reportes*** y se puede observar en la Figura A-19. Aquí se cuenta con 7 reportes, los cuales se presentan en pantalla y también es posible enviarlos a imprimir físicamente. Los reportes disponibles son: ***Listado de Profesores Activos***, el cual presenta una lista de los profesores que se encuentran en el catálogo de profesores y que están activos, es decir, pueden tener asignados 1 grupo o más para impartir clases; ***Listado de Materias***, presenta el catálogo de materias existentes; ***Listado de Preferencias***, presenta las preferencias establecidas por profesor; ***Listado de Cargas para su revisión***, contiene el resultado preliminar del módulo de Generación de Cargas académicas, previamente a su revisión; ***Listado de inconsistencias***, donde se encuentran aquellas asignaturas para las que no fue proporcionada una preferencia para

ningún profesor; Listado de Cargas, que reporta las cargas académicas definitivas, después de haber sido validadas en el módulo correspondiente; y finalmente, el ***Listado de Grupos***, que presenta a los grupos dados de alta en el catálogo del sistema.

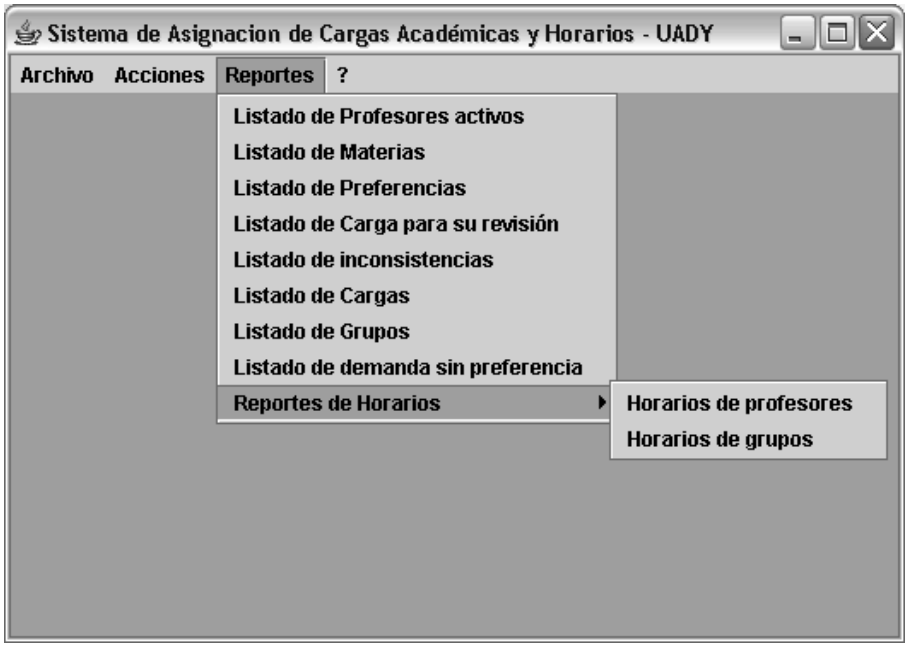

Figura A-19. Menú de Reportes del sistema

A continuación se presentan las Figuras A-20 a A-22 que ilustran algunos de los reportes anteriores. Si se desea enviar el reporte a la impresora, basta con hacer clic en el botón Imprimir, lo cual desplegará la pantalla de selección de impresora, la cual varía dependiendo del modelo de la impresora que esté instalada.

Figura A-20. Listado de profesores activos

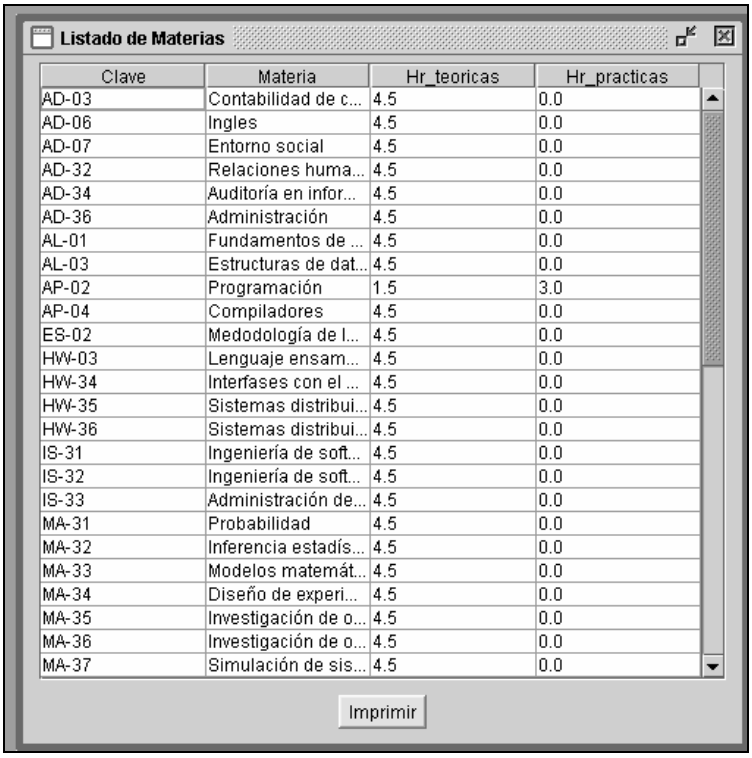

Figura A-21. Listado de materias

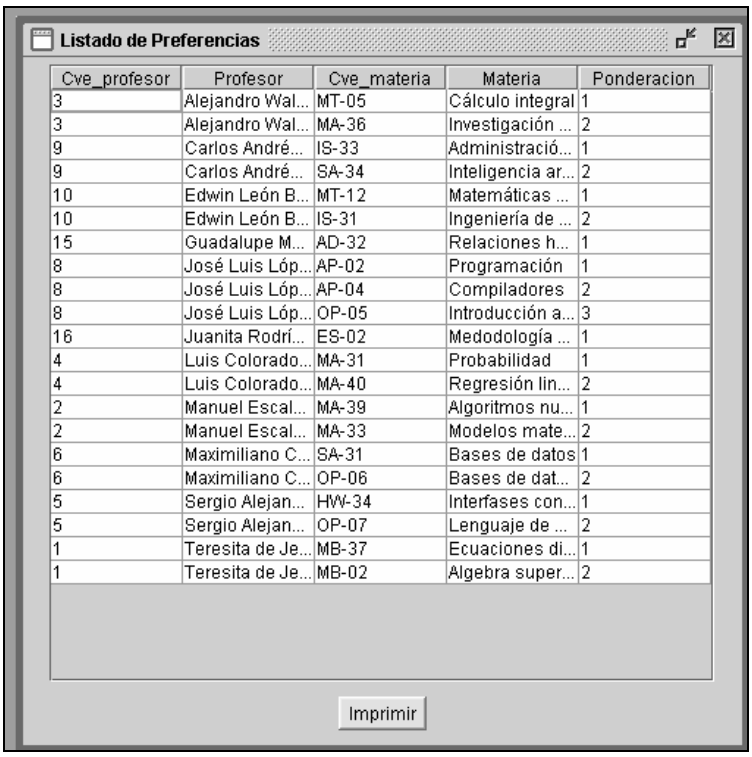

Figura A-22. Listado de preferencias de los profesores

Finalmente, el sistema presenta en el menú ***?*** la información de los desarrolladores del sistema. Este menú únicamente contiene una opción que corresponde al ***Acerca de...***, con información de los desarrolladores del sistema. La Figura A-23 ilustra lo anterior.

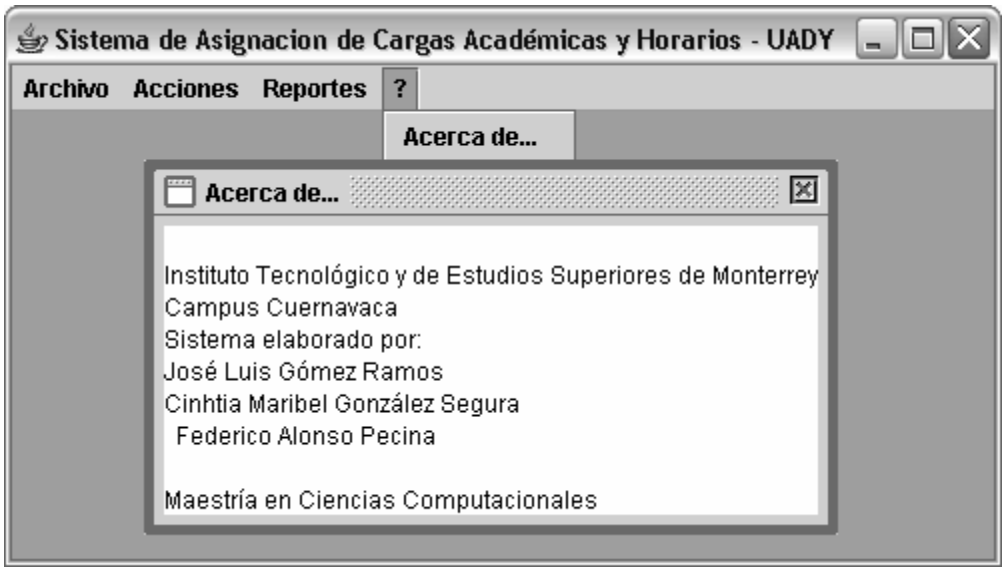

Figura A-23. Menú "?"

D. Horarios obtenidos manualmente en la UADY (Febrero-Julio2005)

Asignatura	PE	S	#S	Nombre	Apellido	Tno	Lun	Mar	Mie	Jue	Vie
Optativa (Física computacional)	C	8	3	Aarón Abraham	Aguayo González		**3**		**3**		**3**
Auditoría en Informática	C	9	2	Carlos Javier	Aguilar Buenfil	**V**	**1**			**1**	
Gestión Tecnológica (Compartida)	IS	2	3	Carlos Javier	Aguilar Buenfil	**V**	**3**		**3**		
Gráficas por Computadora	C	7	3	Carlos Javier	Aguilar Buenfil	**V**		**2**	**1**		**1**
Matemáticas Discretas - Grupo B	C	2	3	Carlos Javier	Aguilar Buenfil	**V**		**3**		**3**	**3**
Microenseñanza II	E	8	2	Ángel Martín	Aguilar Riveroll			**1.2**			
Ayudantía curso Algebra Lineal II, Grupo A	AEM	4		José	Andueza Pech		**1**		**1**	**1**	
Desarrollo Curricular	E	8	2	Eddie	Aparicio Landa		**1**		**1**		
Introducción al análisis matemático	E	4	3	Eddie	Aparicio Landa		**2**	**2**		**2**	
Inferencia Estadística	P2	2	2	Jorge Armando	Argáez Sosa	**V**	**1718**				**1719**
Seminario de Metodología de Invest. (Compartida)	P1	1	2	Jorge Armando	Argáez Sosa	**V**	**1**		**1**		
Evaluación educativa	E	6	2	Beatriz	Ávila Ancona				**2**	**2**	
Teorías del aprendizaje	E	4	2	Beatriz Eugenia	Ávila Ancona				**3**		**2**
Ecuaciones de reacción-difusión	P2	2	2	Eric José	Ávila Vales			**4**			**4**
Modelación Matemática	M	8	3	Eric José	Ávila Vales		**3**		**3**	**3**	
Taller de Cálculo Integral, Grupo A1	C	2	1	Víctor	Bautista Ancona	**V**		**0**			
Taller de Cálculo Integral, Grupo B1	C	2	1	Víctor	Bautista Ancona	**V**	**0**				
Análisis Matemático II (Compartida)	M	6	3	Juan Antonio	Burgos Chablé		**3**	**3**		**3**	
Cálculo II	E	2	6	Juan Antonio	Burgos Chablé		**1**	**2**	**2**	**1.2**	**2**
Cálculo II	M	2	6	Juan Antonio	Burgos Chablé		**2**	**1**	**1.3**		**1.3**
Compiladores - Grupo B	C	4	3	Edgar	Cambranes Martínez			**2**	**2**		**2**
Optativa (HCI)	C	6	3	Edgar	Cambranes Martínez	**V**	**3**		**3**		**3**
Programación - Grupo A	C	2	3	Edgar	Cambranes Martínez	**V**		**2**		**2**	**2**
Cálculo y Análisis Vectorial	IIL	2		Miguel Ángel	Can Ek						
Matemáticas I	QI	2		Miguel Ángel	Can Ek						
Probabilidad y Estadística	QI	4		Miguel Ángel	Can Ek						
Taller de Cálculo II	A	2	2	Jorge Armando	Canul Matú		**4**		**4**		
Ingeniería Económica	C	7	3	Israel	Cauich Suaste	**V**	**1**	**1**		**1**	
Metodología de la Investigación - Grupo A	C	2	3	Israel	Cauich Suaste	**V**	**0**	**3**		**3**	

Asignatura	PE	S	#S	Nombre	Apellido	Tno	Lun	Mar	Mie	Jue	Vie
Metodología de la Investigación - Grupo B	C	2	3	Israel	Cauich Suaste	V	3	0	3		
Probabilidad	P2	2	2	Santanu	Chakraborty			1718		1718	
Probabilidad - Grupo A	C	4	3	Santanu	Chakraborty		1	1		1	
Matemática Actuarial II	A	4	3	María Auxilio Lucía	Chan García		2	2		2	
Optativa (Operación de seguros I)	A	4	3	María Auxilio Lucía	Chan García		4		4	4	
Álgebra Moderna II	M	6	3	Luis Celso	Chan Palomo		1		1		3
Álgebra Superior II	IC	2	3	Luis Celso	Chan Palomo	V		2	2		1
Optativa (Topología Algebraica)	M	8	3	Luis Celso	Chan Palomo		2	1		1	
Simulación de Sistemas	C	9	3	Martín	Chi Pérez	V	2	2		3	
Sistemas Distribuidos I	C	8	3	Martín	Chi Pérez	V		1	2		2
Técnicas de Muestreo	P1	1	2	Norma Roxana	Colonia Cabrera						1.2
Matemáticas Discretas - Grupo A	C	2	3	Luis Fernando	Curi Quintal	V	3		3		3
Teoría de Lenguajes de Programación	C	5	3	Luis Fernando	Curi Quintal		3	4		3	
Compiladores - Grupo A	C	4	3	Julio César	Díaz Mendoza			2	2		2
Investigación de Operaciones I	C	5	3	María Isabel	Díaz Ulloa		4		4		2
Teoría del Seguro	A	2	3	María Isabel	Díaz Ulloa		3	3		3	
Álgebra homológica (Optativa para LM)	P2	8	2	Javier Arturo	Díaz Vargas			3		2	
Introducción al álgebra conmutativa y a la geometría algebraica	P2	4	2	Javier Arturo	Díaz Vargas						
Teoría de campos de clases	P2	4	2	Javier Arturo	Díaz Vargas						
Teoría de números algebraicos	P2	4	2	Javier Arturo	Díaz Vargas						
Geometría de múltipes vistas	P2	3		Arturo	Espinosa Romero	V		1718		1718	
Optativa (Reconocimiento de patrones)	C	6	3	Arturo	Espinosa Romero	V	3		3		3
Programación	IC	2	3	Arturo	Espinosa Romero	V	2		1	2	
Ecuaciones Diferenciales II (Compartida)	M	4	3	Ángel Gabriel	Estrella González		2	3		3	
Comunicación en la enseñanza	E	2	2	Brenda	Gamboa Marrufo			4		4	
Cálculo Integral	IC	2	4	Lucia Belén	Gamboa Salazar	V	1	1			1.2
Cálculo Integral	IS	2	4	Lucia Belén	Gamboa Salazar	V	2	2		1.2	
Topología	M	6	3	Lucia Belén	Gamboa Salazar		2		2	1	
Análisis Matemático II (Compartida)	M	6	3	Gerardo	García Almeida		3	3		3	
Optativa (Análisis funcional)	M	8	3	Gerardo	García Almeida			2	2		2
Álgebra Superior II - Grupo B	C	2	3	María Elena	García Alvarez	V	1		1		1

Asignatura	PE	S	#S	Nombre	Apellido	Tno	Lun	Mar	Mie	Jue	Vie
Algoritmos Numéricos - Grupo A	C	4	3	María Elena	García Alvarez			**4**		**3**	**3**
Algoritmos Numéricos - Grupo B	C	4	3	María Elena	García Alvarez		**2**		**1**		**1**
Bases de Datos	C	6	3	Juan Francisco	Garcilazo Ortiz	**V**	**2**	**3**	**1**		
Programación	IS	2	3	Juan Francisco	Garcilazo Ortiz	**V**		**1**		**3**	**1**
Programación - Grupo B	C	2	3	Juan Francisco	Garcilazo Ortiz	**V**		**2**	**2**		**2**
Sistemas Distribuidos II	C	9	3	Jorge	Gómez Montalvo	**V**		**3**	**2**		**2**
Sistemas Operativos	C	5	3	Jorge	Gómez Montalvo		**1**		**2**	**2**	
Contabilidad	A	2	3	Gabriel	Góngora Biachi			**4**		**4**	**4**
Inferencia Estadística	P1	1	2	Carlos	Herrera Hoyos	**V**	**2**		**2**		
Regresión Lineal	C	6	3	Carlos	Herrera Hoyos	**V**	**1**	**1**		**1**	
Taller de Prácticas Profesionales	A	4	3	Carlos	Herrera Hoyos		**3**		**3**		**2**
Taller de Cálculo Integral	IC	2	1	Reymundo Ariel	Itzá Balam	**V**		**0**			
Taller de Cálculo Integral, Grupo B2	C	2	1	Reymundo Ariel	Itzá Balam	**V**				**0**	
Didáctica II	E	6	2	Martha Imelda	Jarero Kumul		**1**			**1**	
Taller de Formación Profesional	E	8	2	Martha Imelda	Jarero Kumul					**2**	**1**
Inferencia Estadística	C	5	3	Ma. Dióodora	Kantún Chim			**2**	**1**		**1**
Probabilidad - Grupo B	C	4	3	Ma. Dióodora	Kantún Chim		**3**	**1**		**3**	
Simulación Estadística	P1	3	2	Ma. Dióodora	Kantún Chim	**V**	**1**		**1**		
Álgebra Lineal II, Grupo B	AEM	4	3	Alejandro	Lara Rodríguez		**1**		**1**	**1**	
Optativa (Visión por computadora, optativa P2))	C	6	3	Ricardo	Legarda Sáenz		**3**		**3**		**3**
Álgebra lineal numérica (Optativa para LM)	P2	4	2	José	López Estrada			**1**			**1**
Análisis Numérico II	M	4	3	José	López Estrada		**3**		**3**		**3**
Taller de Cálculo II	M	2	2	José Luis	Maldonado Bazán		**4**		**4**		
Computación II	A	2	3	Alberto	Marín Hernández			**1**	**1**		**1**
Computación II	E	2	3	Alberto	Marín Hernández		**2**		**3**		**3**
Computación II	M	2	3	Alberto	Marín Hernández		**1**		**2**	**1**	
Ecuaciones Diferenciales - Grupo A	C	4	3	César Hernán	Mendiburu Silveira		**2**		**1**		**1**
Ecuaciones Diferenciales - Grupo B	C	4	3	César Hernán	Mendiburu Silveira		**1**		**3**	**1**	
Geometría Moderna	M	2	3	César Hernán	Mendiburu Silveira		**3**	**3**		**3**	
Administración de Archivos	C	5	3	Víctor	Menéndez Domínguez		**2**	**1**		**1**	
Ingeniería de Software II	C	7	3	Víctor	Menéndez Domínguez		**2**		**2**	**2**	
Administración de proyectos	C	8	3	Carlos Benito	Mojica Ruiz	**V**	**2**	**3**		**2**	

Asignatura	PE	S	#S	Nombre	Apellido	Tno	Lun	Mar	Mie	Jue	Vie
Ingeniería de Software I	C	6	3	Carlos Benito	Mojica Ruiz	**V**			**2**	**3**	**2**
Desarrollo de prototipos	IC	2	6	Luis Alberto	Muñoz Ubando	**V**	**-1**		**-1**		**-1**
Optativa (Fundamentos de robótica computacional)	C	6	3	Luis Alberto	Muñoz Ubando	**V**	**3**		**3**		**3**
Física II	IC	2	3	Gabriel	Murrieta Hernández		**3**	**3**		**3**	
Cálculo II	A	2	6	José Matías	Navarro Soza		**1**	**2**	**2**	**1.2**	**2**
Sistemas dinámicos (Optativa para LM)	P2	4	3	José Matías	Navarro Soza			**1**	**3**		**1**
Álgebra Superior II	E	2	3	María Guadalupe	Ordaz Arjona			**1**	**1**		**1**
Optativa (Aspectos didácticos de la demostración)	E	8	3	María Guadalupe	Ordaz Arjona			**4**	**4**		**4**
Inteligencia Artificial I	C	8	3	Alejandro	Pasos Ruiz	**V**	**1**		**1**	**1**	
Inteligencia Artificial II	C	9	3	Alejandro	Pasos Ruiz	**V**		**1**		**2**	**1**
Matemáticas discretas	IS	2	3	Alejandro	Pasos Ruiz	**V**		**3**	**2**		**3**
Diseño de experimentos	IQ	1		Josefina Irene	Peniche Ayora						
Diseños Experimentales	E	6	3	Josefina Irene	Peniche Ayora			**1**	**1**		**1**
Optativa (Álgebra Lineal Graduada)	P2	3	3	Ramón	Peniche Mena	**V**	**1718**		**1718**		
Optativa (Cálculo en Variedades)	M	4	3	Ramón	Peniche Mena			**2**		**2**	**2**
Optativa (Grupos de Lie)	M	8	3	Ramón	Peniche Mena		**1**		**1**		**1**
Taller de Cálculo II	E	2	2	Carlos Ariel	Pompeyo Gutiérrez		**4**		**4**		
Administración	C	5	3	Emilio	Rejón Herrera			**3**	**3**		**3**
Gestión Tecnológica (Compartida)	IS	2	3	Emilio	Rejón Herrera	**V**	**3**		**3**		
Estadística no paramétrica y datos categóricos	P1	3	2	Luis Alberto	Reyna Peraza			**2**		**2**	
Didactica de las Matemáticas II	E	8	4	Pilar	Rosado Ocaña		**2**		**2**	**1**	**2**
Geometría Moderna	E	2	3	Pilar	Rosado Ocaña		**3**	**3**		**3**	
Cálculo Integral - Grupo A	C	2	4	Felipe	Rosado Vázquez	**V**	**1**		**1.2**		**1**
Cálculo Integral - Grupo B	C	2	4	Felipe	Rosado Vázquez	**V**	**2**	**1**		**1.2**	
Ecuaciones Diferenciales II (Compartida)	M	4	3	Felipe	Rosado Vázquez		**2**	**3**		**3**	
Interfaces con el Mundo Real - Grupo A	C	4	3	Otilio	Santos Aguilar			**3**	**4**	**2**	
Interfaces con el Mundo Real - Grupo B	C	4	3	Otilio	Santos Aguilar		**4**	**4**			**3**
Optativa (Procesamiento de señales)	C	6	3	Otilio	Santos Aguilar	**V**	**3**		**3**		**3**
Control de Calidad	P1	1	2	Guadalupe	Siordia Montero	**V**		**1**		**1**	
Investigación de Operaciones	A	4	3	Guadalupe	Siordia Montero			**3**		**3**	**3**
Investigación de Operaciones II	C	6	3	Guadalupe	Siordia Montero	**V**		**2**		**2**	**1**

Asignatura	PE	S	#S	Nombre	Apellido	Tno	Lun	Mar	Mie	Jue	Vie
Computación y enseñanza de las matemáticas	E	4	3	Landy Elena	Sosa Moguel		**3**	**3**		**3**	
Desarrollo conceptual de las Matemáticas II	E	6	4	Landy Elena	Sosa Moguel		**2**	**2**	**3**		**2**
Álgebra Lineal II, Grupo A	AEM	4	3	Lucy del Carmen	Torres Sánchez		**1**		**1**	**1**	
Inferencia Estadística, Grupo B	AEM	4	3	Lucy del Carmen	Torres Sánchez			**1**	**2**		**1**
Álgebra Superior II	A	2	3	Irma Noemí	Trejo y Canché		**2**		**3**		**3**
Álgebra Superior II	IS	2	3	Irma Noemí	Trejo y Canché	**V**	**1**		**1**		**2**
Álgebra Superior II	M	2	3	Irma Noemí	Trejo y Canché			**2**		**2**	**2**
Álgebra Superior II - Grupo A	C	2	3	Irma Noemí	Trejo y Canché	**V**	**2**	**1**		**1**	
Diseños Experimentales	C	7	3	Felipe Reyes	Tuz Poot	**V**		**3**		**3**	**2**
Inferencia Estadística, Grupo A	AEM	4	3	Felipe Reyes	Tuz Poot			**1**	**2**		**2**
Taller de Cálculo Integral	IS	2	1	Miguel	Uh Zapata	**V**		**0**			
Taller de Cálculo Integral, Grupo A2	C	2	1	Miguel	Uh Zapata	**V**				**0**	
Optativa (Introducción a la investigación en matemática educativa)	E	8	3	Rocío	Uicab Ballote				**3**	**4**	**3**
Optativa (Taller de elaboración de materiales didácticos)	E	6	3	Rocío	Uicab Ballote		**3**	**3**		**3**	
Relaciones Humanas - Grupo A	C	4	2	Víctor Fernando	Villanueva Abuxapqui		**3**		**3**		
Relaciones Humanas - Grupo B	C	4	2	Víctor Fernando	Villanueva Abuxapqui			**3**		**2**	
Modelos matemáticos	C	8	3	Celia	Villanueva Novelo			**2**		**3**	**1**

Símbología:
PE = Programa Educativo o Carrera
S = Semestre
#S = Número de sesiones
Tno = Turno
A = Actuaría
C = Ciencias de la computación
E = Enseñanza de las matemáticas
FE = Facultad de Educación
IC = Ingeniería en computación
IIL =Ingeniería Industrial Logística (Ingeniería Química)
IS = Ingeniería de software
M = Matemáticas
Los números de horarios corresponden a los 8 períodos diarios, 4 en cada turno.

E. Manual de usuario del sistema en Web para la UADY

Para realizar el acceso mediante el Internet, es necesario introducir en una ventana del navegador la dirección del servidor en el que esté alojada la página. Una vez hecho esto se desplegará una página como la que se ilustra en la Figura B-1.

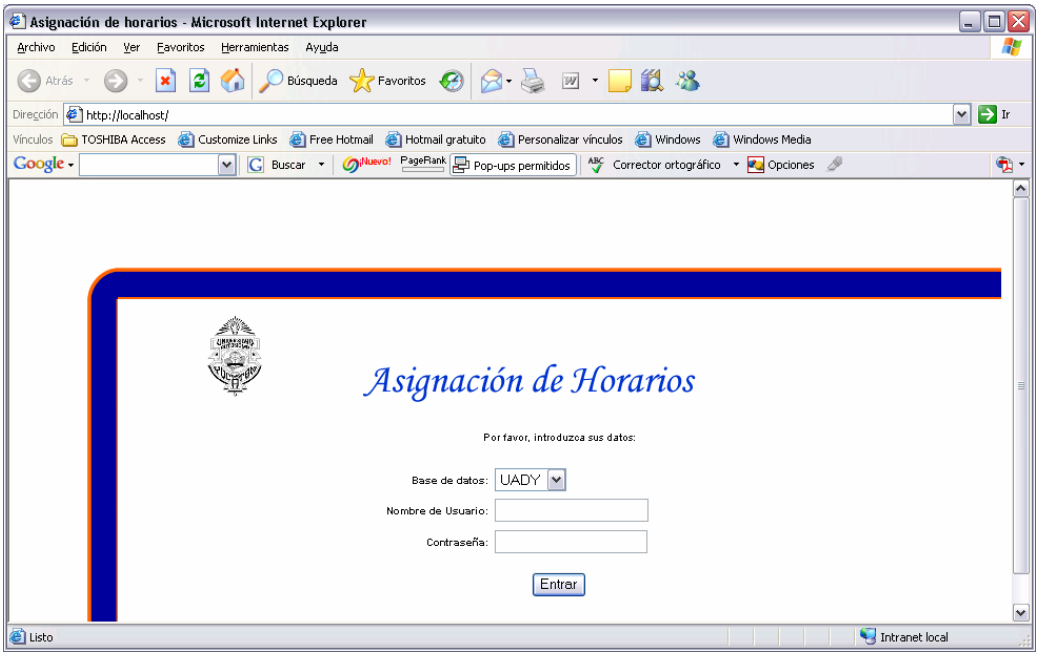

Figura B-1. Página inicial del sitio web

Por el momento, en la ventana anterior únicamente se presenta la interfaz final que se sugiere para el sistema, donde el usuario podrá elegir la base de datos que desea utilizar para leer la información. La implementación actual corresponde únicamente a una consulta: la de la base de datos que se encuentre instalada en el sistema.

Una vez seleccionada la base de datos, se debe presionar el botón <Entrar> lo cual conducirá a la pantalla ilustrada en la Figura B-2. En este ejemplo se presenta la información del Instituto Tecnológico de Monterrey, Campus Cuernavaca.

En la Figura B-2, del lado izquierdo se presenta el Menú Principal del sistema, que contiene los sub-menús de ***Añadir***, ***Borrar***, ***Buscar*** y ***Listar***, para cada una de las opciones del menú principal: ***Profesor***, ***Materia***, ***Grupo*** y ***Salón***. También se cuenta con el menú ***Reportes*** y por último la opción de ***Terminar***, para finalizar la sesión en el sistema.

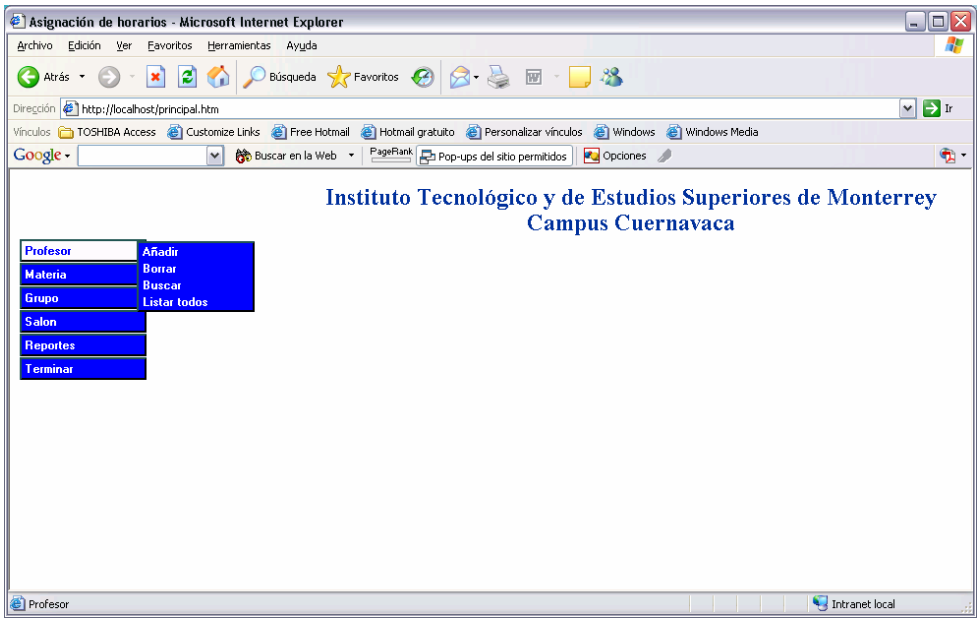

Figura B-2. Menú principal del sistema.

Si se desea ***Añadir un profesor***, el primer menú corresponde a esa opción y despliega una pantalla como la que se muestra en la Figura B-3. Aquí se despliegan los cuadros de texto para capturar la información de cada Profesor, y al final de la página se despliega una lista de los profesores existentes, para poder verificar inmediatamente los datos añadidos.

El segundo submenú corresponde a la ***Eliminación de un profesor*** y la pantalla correspondiente se presenta en la Figura B-4. Primero es necesario introducir la clave del profesor para realizar una búsqueda y en caso de encontrarlo, proceder a su eliminación.

El tercer submenú permite ***Buscar a un profesor*** y la pantalla correspondiente se presenta en la Figura B-5. Esta es muy similar a la de la Figura B-4, con la diferencia de que aquí no se elimina sino únicamente se despliegan los datos encontrados.

El cuarto y último submenú es el de ***Listar a todos los profesores*** y está ilustrado en la Figura B-6. Aquí es posible visualizar la información de todos los profesores que se encuentran almacenados en la base de datos.

Los menús de ***Materia***, ***Grupo*** y ***Salón*** son muy similares al del ***Profesor***, por lo que se omite la explicación, el uso de las pantallas es muy simple e idéntico al del catálogo del Profesor.

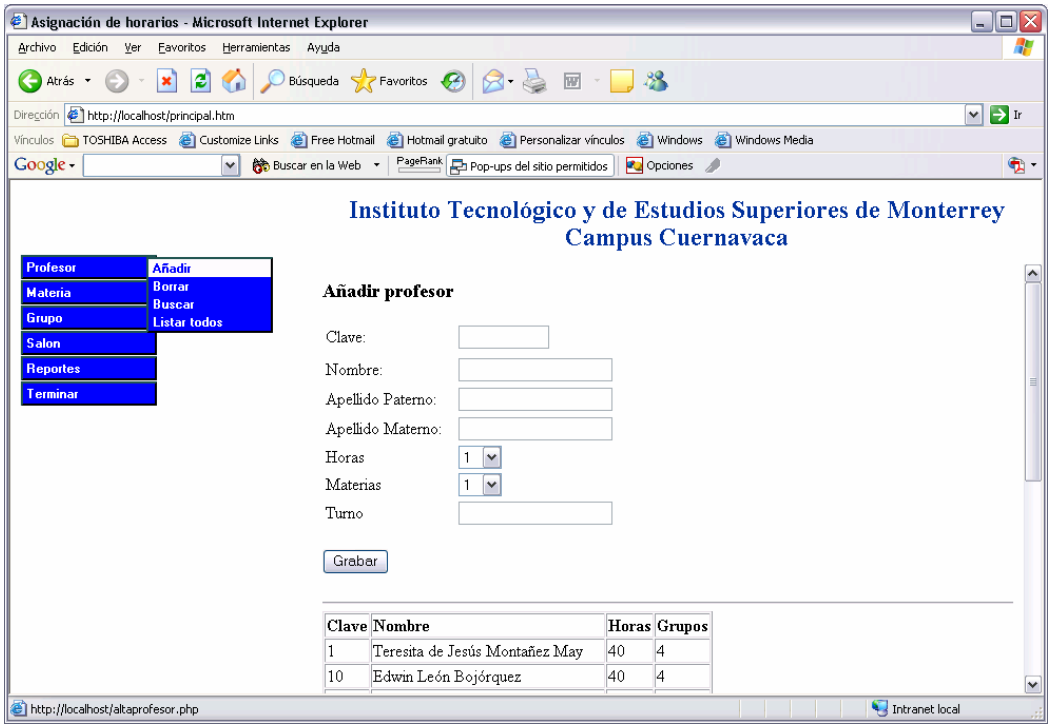

Figura B-3. Añadir a un Profesor

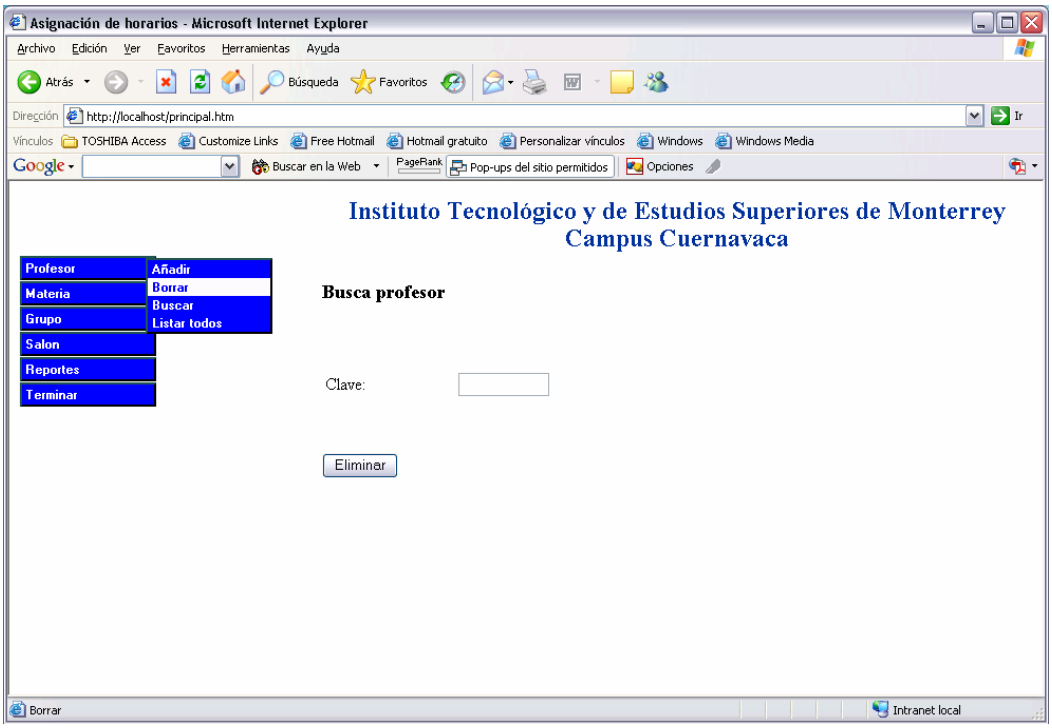

Figura B-4. Borrar a un profesor

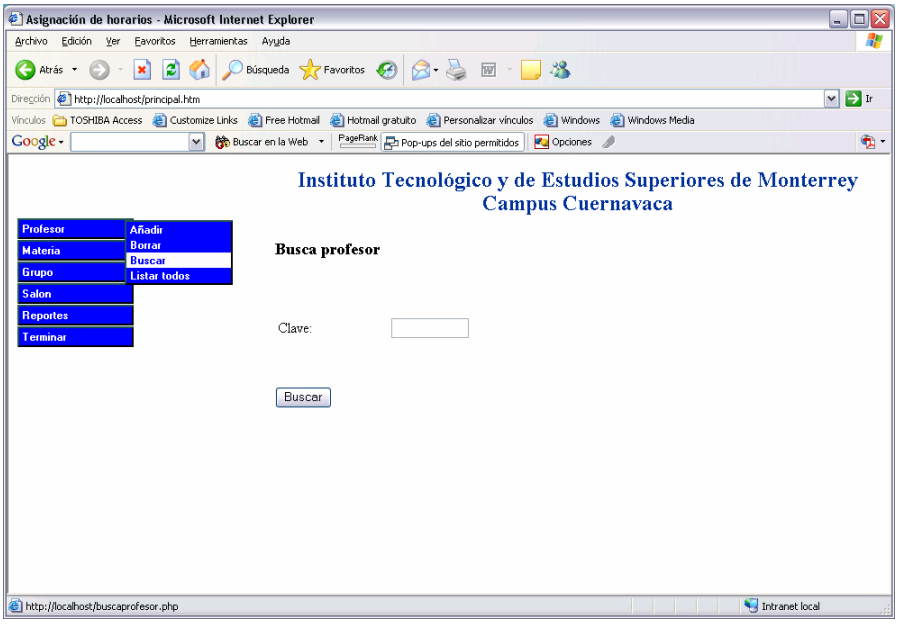

Figura B-5. Buscar a un profesor

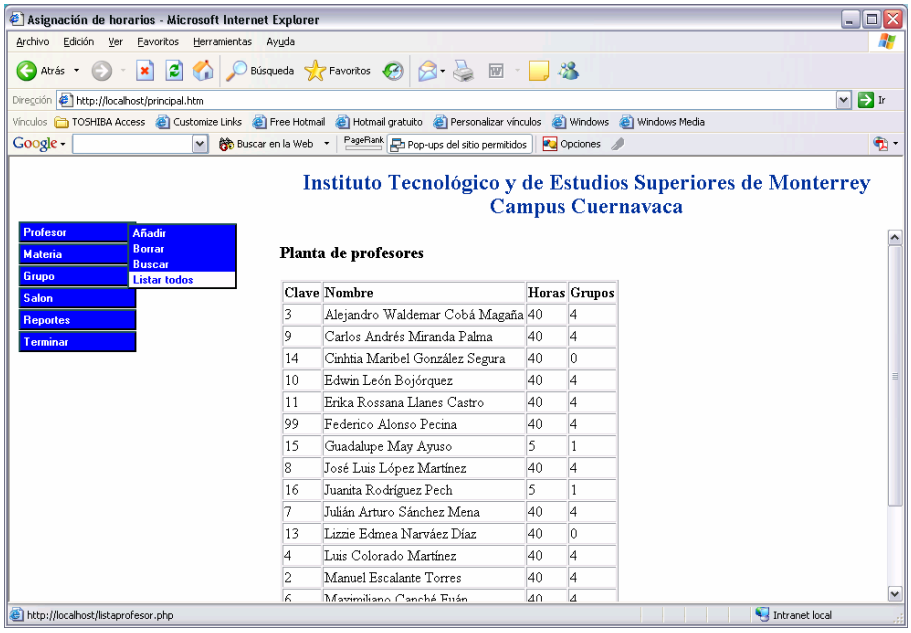

Figura B-6. Listar a todos los profesores

Posteriormente se encuentra el menú de ***Reportes***, el cual se ilustra en la Figura B-7. En éste es posible visualizar la información de los catálogos anteriores. El menú que se presenta corresponde al catálogo de Materias de los profesores. Es posible imprimir el reporte haciendo uso de la herramienta de impresión incluida en el visor de páginas web que se esté utilizando, de la misma manera en que se imprime cualquier página web.

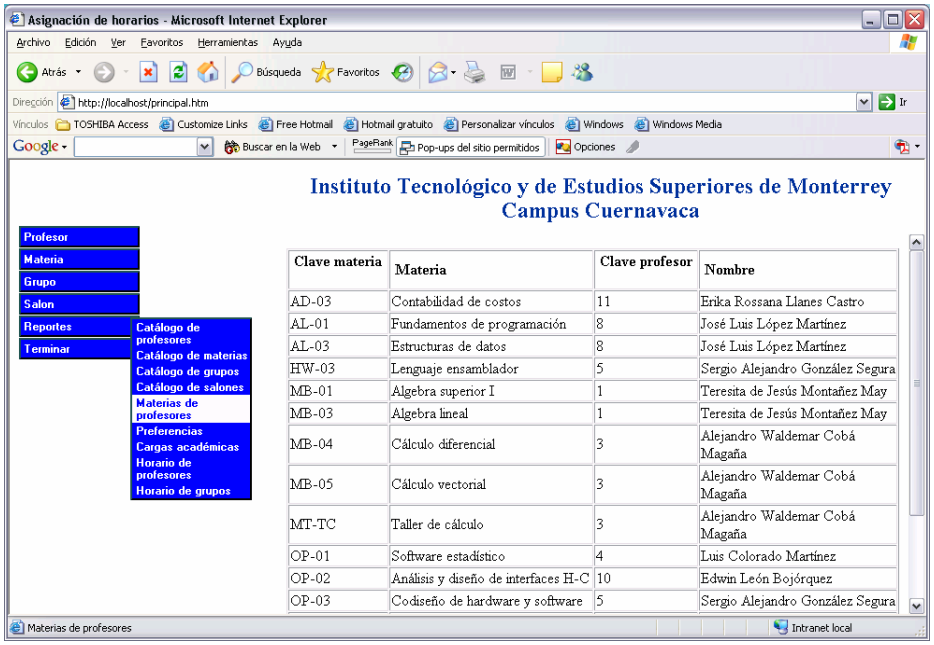

Figura B-7. Reportes del sistema.

Finalmente se cuenta con el menú ***Terminar***, en el que se tienen 2 opciones: el ***Acerca de…*** y ***Salir***. La opción ***Acerca de…*** contiene información de los autores del sistema, y en ***Salir*** se puede abandonar el mismo. La Figura B-8 ilustra lo anterior.

Figura B-8. Menú Terminar, visualizando la opción Acerca de...