Universidad Central "Marta Abreu" de Las Villas
Facultad de Matemática, Física y Computación
Centro de Estudios de Informática
Dpto. Inteligencia Artificial

Trabajo de Diploma

Cronos (v1.0): Sistema para la generación automática de horarios docentes

Autores:

Neysi Parada Gamboa

Luis Angel Medina Castillo

Tutores:

Dra. Yailén Martínez Jiménez

MSc. Beatríz María Méndez Hernández

Santa Clara

2015

Dictamen

Los que suscriben, Neysi Parada Gamboa y Luis Angel Medina Castillo, hacemos constar que el trabajo titulado CRONOS (v 1.0): Sistema para la generación automática de horarios docentes fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de los estudios de la especialidad de Licenciatura en Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____	_____
Neysi Parada Gamboa	Luis Angel Medina Castillo

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____	_____
Firma del Tutor	Firma del Jefe del Laboratorio

«Si automatizas un procedimiento desastroso, obtienes un procedimiento desastrosamente automatizado.»

Rod Michael

Dedicatoria

A mi má, Deysi, si hay algo que sé hacer bien es por ella, y cuando llega la recompensa por un esfuerzo no puedo dejar de recordar su cercanía, complicidad, devoción, ... su ejemplo. Que esta sea la recompensa a tantos años de entrega, desvelos, apoyo. Estamos juntas, te quiero.

A mi abuela, que ya no está, pero desde donde esté espera lo mejor de mi.

A Luis, por aguantarme y respetarme siempre.

A Julio, por enseñarme a dar los primeros pasos, dejarme caer y ayudarme a levantarme.

Neysi

Dedicatoria

Dedico esta tesis a mi madre que aunque no estuvo conmigo durante 2 años por estar cumpliendo misión, siempre me ayudó y estuvo atenta a mí, como solo las madres saben hacer.

A mi abuela, que no está con nosotros pero que siempre la tendré en mi corazón.

Luis Angel

Agradecimientos

Un agradecimiento singular debo a Beatríz y Yailén, quienes como tutoras de esta tesis, me han orientado, apoyado y corregido en este proyecto con un interés y una entrega que han sobrepasado, con mucho, todas las expectativas que, como estudiante, deposité en ellas.

A mis profesores, a los más exigentes por hacerme más fuerte y tolerante, a todos porque marcaron cada etapa de nuestro camino universitario.
A Bea, la amiga.

A mi negro, Luis Angel, por acompañarme siempre en silencio, mientras tenía tiempo libre y por ser parte fundamental en este proyecto.

A Alia, por estar aquí bien dentro, y por ayudarme hasta cuando no sabía lo que hacía.

A Claudia, por confiar en mí todas sus decisiones.
A Lisvandy, por su sinceridad, cruel a veces.
A Lázaro, por enseñarme a insistir , aunque no siempre funcione.
A Ana Li, Erick y Juan Luis, por estar siempre un año delante.
A mis amigos del pre, que aún de lejos se preocuparon por mí.

A Google, por saberlo todo.

A los que empezaron y no se graduaron, por enseñarme a decidir cuándo es el momento perfecto para cambiar el rumbo.

A todos aquellos que no creyeron en mí, a los que esperaban mi fracaso en cada paso que daba hacia la culminación de mis estudios, a todos aquellos que apostaban a que me rendiría a medio camino, a todos los que supusieron que no lo lograría, a todos ellos también les dedico esta tesis.

Y a mí claro, por lograrlo.

Neysi

Agradecimientos

Agradezco a todos los que me han ayudado en el transcurso de estos 5 años...

A mi hermanita, que siempre la estoy molestando pero es lo que más aprecio y quiero.

A mi hermano que por motivos internacionalistas no está presente, el cual es mi ejemplo a seguir y siempre lo tendré como mi luz inspiradora.

No quiero dejar de mencionar a mi gente, los CAP, los cuales me han aguantado durante estos 5 años y nos hemos divertido como solo los universitarios saben hacer.

A mis tutoras que aunque muchas veces quedé mal con ellas por estar de fiesta las quiero y las estimo.

Ahora no por ser la última es de menos importancia , a ella le debo que me esté graduando, a Neysi quien ha estado arriba de mí todo el tiempo para la culminación de la tesis y por fin hemos terminado esta etapa con mucho sacrificio y horas en las cuales se imponía priorizar este trabajo de diploma y declinar gustos que nos podíamos dar.

A todos, Gracias!!

Luis Angel

Resumen

Los horarios docentes son un elemento fundamental de cualquier centro educacional. Constituyen el elemento organizativo de las actividades escolares y juegan un papel importante en la labor docente y educativa de la institución. Tradicionalmente, el proceso de elaboración de horarios docentes se ha considerado una tarea complicada, debido a la gran cantidad de información y condiciones que deben tenerse en cuenta.

Actualmente en la Universidad "Marta Abreu" de las Villas la gestión y confección de los horarios docentes se realiza de forma manual. Deben cumplirse una gran cantidad de restricciones, por lo que la confección semanal de los horarios es una tarea sumamente difícil.

Este trabajo consiste en la creación de un sistema automatizado para la gestión de los horarios docentes en la Facultad de MFC, en respuesta a la necesidad de mejorar y perfeccionar la manera en que actualmente se lleva a cabo esta tarea. La utilización de un sistema automatizado para generar los horarios docentes significaría una mejora considerable en este proceso en cuanto a eficiencia, tiempo y organización, contribuyendo favorablemente al desarrollo del proceso docente y educativo de la universidad.

Abstract

Time schedules are a fundamental element in the organization of any educational center. They constitute the organizational element of school activities and performan an important role in the teaching and educational work. Traditionally the elaboration process of teaching schedules has been considered a complex task due to the great quantity of information and constrains to have into account.

At present at the Central University "Marta Abreu" of Las Villas, the confection process of teaching schedules comes true manually. There are big quantities of restrictions, for this reason, the weekly confection of schedules becomes a very hard and time consuming task.

This work consists of the creation of an automatized system for the elaboration process of teaching schedules in the Faculty of MFC, in answer to the need of getting a better way than the present which takes effect in this task. The utilization of an automated system to generate the teaching schedules would mean a considerable improvement in this process with efficiency, time and organization, favorably contributing to the development of the teaching and educational process of the university.

Índice general

Introducción **1**

Capítulo I. Fundamentación Teórica. **5**

1.1. Problemas de secuenciación de tareas 6
1.2. Problemas de generación de horarios 7
1.2.1. Tipos de restricciones 10
1.2.2. Métodos para resolver estos problemas 11
1.3. Aprendizaje Reforzado 15
1.3.1. Elementos del Aprendizaje Reforzado 16
1.3.2. Q-Learning 18
1.4. Base de datos 19
1.4.1. Sistema Gestor de Base de Datos (SGBD) 20
1.4.2. MySQL 21
1.4.3. Mapeo Objeto/Relacional (ORM) 22
1.4.4. Hibernate 22
1.5. Conclusiones parciales 24

Capítulo II: Arquitectura del sistema. **26**

2.1. Descripción del problema 27
2.2. Aplicación web 29
2.2.1. Herramientas 30
2.2.2. Diseño del sitio web 31
2.2.3. Mapa de navegación 32
2.3. Diseño de la base de datos 33
2.3.1. Diseño conceptual 33
2.3.2. Diseño lógico 34
2.3.3. Diseño físico 36
2.4. Ingeniería de software del sistema 37
2.4.1. Diagrama de Casos de Uso 38
2.4.2. Diagrama de Componentes 38
2.4.3. Diagrama de Clases 39
2.5. Conclusiones parciales 41

Capítulo III: Algoritmo de generación automática de horarios **42**
3.1. Restricciones 43
3.1.1. Fuertes 45
3.1.2. Suaves 46
3.2. Algoritmo 46
3.2.1. Entrada y Salida 47
3.2.2. Carrera y año inicial 48
3.2.3. Entorno de trabajo 48
3.2.4. Matriz de Q-valores 48
3.2.5. Estrategia de selección 49
3.2.6. Calcular recompensa 49
3.2.7. Actualizar Q-valores 49
3.3. Validación 49
3.4. Conclusiones parciales 51

Conclusiones **52**

Recomendaciones **53**

Bibliografía **54**

Índice de figuras

1.1. Problema de generación de horarios para universidades. 10
1.2. Métodos para resolver problemas de generación de horarios. . . . 15
1.3. Aprendizaje Reforzado 16
1.4. Pseudocódigo del algoritmo Q-Learning 19
1.5. Subsistema de un SGBD. 21
1.6. Estructura de MySQL. 22
1.7. Arquitectura de Hibernate. 24
2.8. Deficiencias en la confección manual de horarios. 29
2.9. Mapa del Sitio Web 32
2.10. Diagrama Entidad/Relación. 35
2.11. Diagrama de Casos de Uso del sistema. 38
2.12. Diagrama de componentes del sistema. 39
2.13. Diagrama de Clases del Sistema. 40
3.14. Información necesaria para generar horarios 47
3.15. Fragmento de P4 generado para 1ero CC-Grupo#1 48

Índice de tablas

2.1. Tablas de la Base de Datos Horario. 34
3.2. Cumplimiento de las restricciones blandas por año. 50
3.3. Preferencias personales que se cumplieron para cada profesor de primero y tercer año. 50

Introducción

Desde el inicio de la humanidad el hombre se ha propuesto hacer un uso racional y eficiente de los recursos, especialmente del tiempo. Existen distintos mecanismos de planificación donde predominan las planificaciones a corto plazo, tiempo durante el cual se desarrolla habitual o regularmente una acción o se realiza una actividad, esto se conoce también como horario. En la arista social todos los elementos están íntimamente relacionados con la planificación, pero indiscutiblemente el proceso docente-educativo es uno de los primeros beneficiados, siendo la planificación del horario docente uno de los procesos más importantes.

Los horarios docentes son un elemento fundamental en la organización de cualquier centro educacional. Constituyen el elemento organizativo de las actividades docentes y permiten una visión general de la distribución cuantitativa y cualitativa de los recursos humanos y materiales. Tradicionalmente el proceso de elaboración de horarios docentes se ha considerado una tarea complicada, realizándose de forma manual. En el mercado actual del software existen algunas herramientas de carácter general, dirigidas principalmente a la gestión académica; pero en la mayoría de los casos, no se ajustan a las restricciones reales de los distintos centros educacionales, siendo esta una de las razones por la cual su uso no se ha generalizado (Corrales and Pérez, 1976).

El desafío del problema de la planificación de horarios se ve acrecentado en las instituciones educativas, ya que en ellas el problema se presenta al menos dos veces al año, cuando ocurre un cambio en el periodo escolar. En este tipo de instituciones el problema general consiste en proponer una planificación de los recursos escolares (alumnos, profesores, aulas, equipo educativo, etc.) que cumpla con una serie de restricciones y obtenga una solución aceptable en un tiempo moderado. Como tal, esto no es una tarea trivial, ya que debido al gran número de restricciones impuestas, estos problemas normalmente se vuelven consumidores de tiempo y generalmente son clasificados como NP-Completos (Garey et al., 1976), ya que en ellos existe una proporción directa entre el tiempo y la calidad de la solución, motivo por el cual han sido objeto de estudio de diversos investigadores.

La literatura científica recoge con el nombre de generación de horarios a la variedad de problemas de Optimización Combinatoria que sirven para modelar de forma general los procesos organizativos de los centros de enseñanza. La comple-

jidad de los mismos es elevada, lo que ocasiona que muchos de los trabajos en el tema se desarrollen tomando datos y problemas creados artificialmente (Murray et al., 2007).

El problema de la planificación de horario consiste básicamente en asignar una serie de actividades o eventos en periodos de tiempo, bajo ciertas restricciones, de manera tal que se satisfagan lo mejor posible un conjunto de objetivos (Wren, 1996).

Actualmente en la Universidad Central "Marta Abreu" de las Villas, la gestión y confección de los horarios docentes se realiza de forma manual. Los horarios son elaborados por la dirección de cada una de las facultades en estrecha relación con los distintos departamentos docentes que, a nivel central, diseñan y confeccionan los planes de estudio de las asignaturas, así como el cronograma de actividades y exámenes. Durante la planificación docente se manejan grandes volúmenes de información y son numerosas las restricciones a cumplir, por ejemplo: la disponibilidad de los locales, las preferencias y afectaciones de los profesores, las características de las asignaturas y de cada uno de los grupos, entre otras. La confección semanal de los horarios docentes es una tarea sumamente difícil que consume mucho tiempo y generalmente requiere de varias versiones y revisiones para obtener un horario congruente y óptimo en la medida de lo posible. Lo anterior implica que el proceso de planificación del horario docente sea complejo, propenso a errores, vulnerable a cambios de última hora y su confección de forma manual se torna en ocasiones impracticable.

Debido a lo antes expuesto se plantea el siguiente ***problema de investigación:***

En la Facultad de Matemática, Física y Computación (MFC) la confección de los horarios docentes es realizada por los Profesores Principales de Año (PPA). Para la confección de este horario el PPA analiza los P1 de cada asignatura y las restricciones que plantean los profesores en dicho documento y basándose en el gráfico docente confeccionan el horario. Esta tarea resulta muy engorrosa y le ocupa a dicho profesor una gran cantidad de tiempo y trabajo. Por lo tanto sería muy ventajoso contar con una herramienta capaz de generar dicho horario de forma automática que se rigiera por los P1s y tuviese en cuenta todas las restricciones, tanto personales como administrativas.

Para dar solución al problema de investigación se plantea como ***objetivo ge-***

neral de esta tesis:

- Implementar una aplicación para la generación automática de los horarios docentes de la facultad de MFC de la UCLV utilizando Aprendizaje Reforzado.

Este objetivo se desglosa en los siguientes ***objetivos específicos***:

1. Analizar el comportamiento de los principales algoritmos utilizados para resolver problemas de generación de horario.
2. Diseñar e implementar una página web para captar las restricciones y la planificación por semana de las asignaturas.
3. Diseñar una base de datos que guarde la información requerida en la página web.
4. Implementar un algoritmo usando Aprendizaje Reforzado que utilice los datos almacenados en la base de datos y genere de forma automática el horario de la facultad.
5. Evaluar el desempeño del algoritmo de acuerdo a las restricciones blandas (conformidad de los profesores) con que cumple.

Para dar solución al problema de investigación y a los objetivos específicos de esta tesis se dará respuesta a las siguientes ***preguntas de investigación***:

- ¿Cómo se manejan las restricciones del problema en los principales algoritmos que aparecen en la literatura?
- ¿Es el algoritmo Q-Learning eficiente para la generación automática de horarios?
- ¿Será la herramienta propuesta capaz de generar un horario que cumpla con todas las restricciones de la facultad?

Justificación de la investigación.

Con la aplicación del algoritmo Q-Learning al problema de la generación de horarios de nuestra facultad, se pretende obtener una solución automatizada donde se reduzca el tiempo y la ocurrencia de errores que trae consigo la asignación manual de los horarios de clases por parte de los PPA, y se satisfagan al máximo las necesidades de los usuarios. Lo anterior significa un avance tecnológico y organizativo cuyo propósito es el de minimizar los solapamientos entre asignaturas y horas, evitando extensos intervalos entre clases y bajando los niveles de insatisfacción de los docentes y los estudiantes.

Esta investigación nace de la necesidad de romper con los esquemas paradigmáticos de la antigua concepción de la asignación de clases y concienciar a la comunidad universitaria de que la implantación de un software que genere el horario automáticamente sería una herramienta que serviría como guía para llevar a cabo de manera formal el proceso académico-administrativo, porque servirá de patrón base para planear, organizar, ejecutar y consolidar el debido procedimiento de la asignación de horarios de clases y aulas en todas las facultades de la universidad.

La tesis está estructurada en tres capítulos, en el capítulo 1 se realiza un estudio de los problemas de secuenciación de tareas, específicamente generación de horarios, además de los diferentes métodos que existen para resolverlos, también se hace una análisis de los diversos objetivos que pueden ser optimizados para este tipo de problemas. Por último, se hace un estudio del aprendizaje reforzado haciendo énfasis en el algoritmo Q-Learning y sus variantes para resolver problemas de Generación de horarios. Seguidamente, en el capítulo 2 se hace una breve descripción del problema a resolver, se describen las herramientas utilizadas y los diagramas de Ingeniería de Software correspondientes a la aplicación desarrollada. Posteriormente, en el capítulo 3 se explica como fueron tratadas y categorizadas las restricciones para el problema, se describen las partes esenciales del algoritmo y por último se hace una validación del mismo. El documento culmina con las conclusiones y las recomendaciones de los autores.

Capítulo I

Capítulo I . Fundamentación Teórica

Este capítulo aborda todo lo referente a la base teórica que fundamenta esta investigación. Se hace una profunda revisión de la bibliografía para determinar las técnicas de optimización que han sido utilizadas para resolver problemas de secuenciación de tareas del tipo de generación de horarios. Además, se hace un análisis de los diversos objetivos que pueden ser optimizados para este tipo de problemas. Se exponen los principales aspectos del Aprendizaje Reforzado haciendo énfasis en el algoritmo Q-Learning y sus variantes para resolver problemas de generación de horarios.

1.1. Problemas de secuenciación de tareas

Los problemas de secuenciación de tareas están presentes en todas aquellas situaciones que implican la ejecución de acciones que requieren recursos cuya disponibilidad está limitada y por tanto deben asignarse de modo eficiente. Problemas de este tipo aparecen en numerosos contextos, entre los que se encuentra la generación automatizada de horarios.

Los problemas de secuenciación son típicamente clasificados como NP-Completos (Chahal and De Werra, 1989), lo que significa que encontrar una solución óptima se torna imposible si no se usa un algoritmo esencialmente enumerativo, y el tiempo de cómputo se incrementa exponencialmente de acuerdo al tamaño del problema (Shen, 2002; Jiménez, 2012).

El enfoque clásico para resolver un problema de secuenciación consiste en construir un modelo matemático, comúnmente basado en Programación Mixta en Enteros y luego aplicar un algoritmo de búsqueda como el algoritmo de Ramificación y Acotamiento para encontrar la solución óptima(Méndez et al., 2006). Sin embargo, a medida que el número de recursos disponibles en el problema incrementa, o el número de operaciones a ser planificadas crece, o se incrementan las restricciones del problema, este enfoque no será capaz de encontrar la solución óptima en un tiempo computacional razonable (Ruiz et al., 2008). Es aquí donde los métodos heurísticos se convierten en el centro de atención, pues son capaces de encontrar buenas soluciones de forma eficiente (Zhang and Lau, 2005).

1.2. Problemas de generación de horarios

El problema de la generación automática de horarios es clásico en la computación. Reconocido en inglés como Timetabling, se define como: "la asignación, sujeta a restricciones, de un grupo de recursos a objetos ubicados en tiempo y espacio, de tal manera que se satisfagan un conjunto de objetivos deseados" (Wren, 1996), y aparece en diversos ámbitos incluyendo: la planificación de transporte, programación de eventos deportivos, horarios docentes y laborales, *nurse rostering*, entre otros (Qu, 2002).

Visto como un problema de secuenciación de tareas, los problemas de generación de horarios constan de un determinado conjunto de operaciones que deben ser ejecutadas. Cada una de estas operaciones requiere utilizar un determinado tipo de recurso para llevar a cabo su objetivo. Existe una serie de restricciones que deben ser satisfechas simultáneamente por las asignaciones realizadas para que estas sean consideradas factibles y por ende, una solución del problema. De esta forma, el objetivo es hallar un instante de inicio y un instante de finalización para cada operación, satisfaciendo el conjunto de restricciones correspondiente, y optimizando una determinada función objetivo (Nandhini and Kanmani, 2009).

Los problemas de generación de horarios educacionales (como también se les conoce) se clasifican, según (Schaerf, 1999), en:

• Generación de horarios de escuela (*School Course Timetabing*): Planificación semanal para todas las clases de una escuela secundaria, evitando reservar diferentes grupos de estudiantes y profesores en un mismo turno.

• Generación de horarios para instituciones de educación superior o universidades (*University Course Timetabling*): Programación de todas las clases de un conjunto de módulos de estudios universitarios, minimizando los solapamientos de clases que tienen los grupos y evitando reservas dobles de profesores.

• Generación de horarios de exámenes (*Examination Timetabling*): Programación de los exámenes de un conjunto de cursos universitarios evitando la superposición de exámenes y separando en tiempo tanto como sea posible la ocurrencia de un nuevo examen para los estudiantes.

En (Carter and Laporte, 1998) los autores descomponen el problema de la generación de horarios en cinco subproblemas:

1. Horarios de cursos (*course timetabling*).
2. Horarios para grupo-profesor (*class-teacher timetabling*).
3. Asignación de estudiantes a grupos (*student scheduling*).
4. Asignación de profesores a grupos (*teacher assignment*).
5. Asignación de locales a actividades (*classroom assignment*).

Si a los subgrupos definidos por (Carter and Laporte, 1998) se les suma la programación de exámenes de (Schaerf, 1999), totalizan seis subgrupos que abarcan todos los posibles subproblemas de generación de horarios educacionales.

Cada una de estas clasificaciones tiene características propias pero no escapan de tener similitudes, pues todas tratan de encontrar una distribución que asigne recursos a espacios de tiempo.

Horarios de cursos

El problema de la generación de horarios de cursos (*course timetabling*) surge cuando las universidades ofertan un conjunto de cursos conformado cada uno por un conjunto finito de horas clases pero sin un plan de estudio bien definido. Este problema se identifica porque los estudiantes se matriculan en las asignaturas que ellos estimen convenientes de acuerdo a determinados requisitos que les imponen las universidades, por ejemplo para cursar una asignatura X se debe tener cursada la asignatura Y. Las asignaturas son impartidas por varios profesores a distintas horas y el alumno decide a cual asistir (De Werra, 1985).

Horarios para grupo-profesor

El problema de la generación de horarios para grupo-profesor (*class-teacher timetabling*) se identifica por un grupo, compuesto por un conjunto de estudiantes, que siguen el mismo programa docente donde $G = \{g_1, ..., g_m\}$ es un conjunto de grupos, $P = \{p_1, ..., p_n\}$ un conjunto de profesores y R una matriz de restricciones mxn donde r_{ij} es el número de clases que involucran al grupo g_i y al profesor p_j. Siempre se asume que todas las clases tienes el mismo periodo de duración, entonces teniendo un conjunto de periodos o turnos $T = \{t_1, ..., t_k\}$, el problema será asignar cada clase a algún turno de manera tal que ningún profesor esté involucrado en más de un turno a la vez (De Werra, 1985).

Asignación de estudiantes a grupos

La asignación de estudiantes a grupos (*student scheduling*), resulta útil en sistemas que tengan gran número de asignaturas optativas o lectivas, de manera que se trata de crear grupos con la mejor afinidad posible lo cual desarrolla la motivación entre los estudiantes y facilita la planificación de las actividades (Echevarría Cartaya, 2014)

Asignación de profesores a grupos

El objetivo de este sub-problema es asignar profesores a los grupos considerando sus preferencias. Algunos investigadores han discutido el problema de asignación de profesores en la literatura. Uno de los primeros documentos fue escrito por (Andrew and Collins, 1971) el problema es cómo hacer las asignaciones del profesor de forma tal que sean altamente efectivas y que cumplan la mayor cantidad de preferencias del profesor, y a la misma vez asegurar que todos los grupos estén provistos de personal y que ningún profesor se cargue de trabajo excesivamente (Gunawan, 2009).

Asignación de locales a actividades

Los grupos tienen que ser asignados a locales y periodos de tiempo específicos. Para simplificar, la asignación de grupos a los periodos de tiempo normalmente se hace antes de la asignación de los grupos a las aulas (Carter, 1989; Glassey and Mizrach, 1986; Gosselin and Truchon, 1986). Algunos de los problemas de asignación de aulas pueden ser considerados problemas fáciles aunque otros pueden ser difíciles de resolver (Carter and Tovey, 1992).

Horarios de exámenes

El problema de generación de horarios de exámenes requiere la planificación de un número dado de exámenes (uno para cada asignatura) dentro de un tiempo determinado. Este tipo de problema es similar al problema de generación de horarios de cursos, y es difícil hacer una distinción clara entre estos dos problemas. De hecho, algún problema específico puede formularse de las dos maneras, como un problema de generación de horarios de examen y como uno de generación de horario de curso. No obstante, es posible definir algunas de las diferencias entre ellos. Los horarios de exámenes tienen las características siguientes (diferentes a las del problema de generación de horarios de curso) (Schaerf, 1999):

- Hay sólo un examen para cada asignatura.
- La condición de conflicto es generalmente estricta, por lo que un estudiante no puede faltar a un examen.
- Hay tipos diferentes de restricciones, por ejemplo, cada estudiante puede tener a lo sumo un examen por día y no puede tener demasiados exámenes consecutivos.
- El tiempo para hacer un examen puede variar, en contraste con el problema de generación de horarios de cursos donde el tiempo de duración de cada turno es fijo.
- Puede haber más de un examen por aula.

Los problemas de generación de horarios de la vida real siempre contienen una combinación de los subproblemas descritos anteriormente, aunque no todos los subproblemas pueden ser relevantes para una situación en particular. Para describir un problema de este tipo específicamente, es necesario saber en detalle, cuáles de estos subproblemas se resolverán y en qué orden se deben solucionar. Por algunas universidades o escuelas se permite un poco de flexibilidad en esta clasificación, mientras otros obedecen las reglas estrictamente, por ejemplo el problema de generación de horarios para nuestra universidad solo está compuesto por cinco subproblemas, como vemos en la Figura 1.1.

Figura 1.1: Problema de generación de horarios para universidades.

1.2.1. Tipos de restricciones

El problema de la generación de horarios es muy complejo e involucra a todas las instituciones donde sea necesaria la construcción de horarios. Uno de los factores que hacen este problema tan complejo es la necesidad de satisfacer una serie de restricciones. Estas restricciones se clasifican en restricciones duras o fuertes y restricciones suaves o blandas; las primeras deben cumplirse, la violación de alguna, origina un horario no válido y bajo ninguna circunstancia deben ser infringidas, además son utilizadas para encontrar soluciones factibles. Por otro lado, las restricciones suaves denotan preferencias del usuario (políticas flexibles) y se desea que se cumplan en la medida de lo posible. La violación de algunas de ellas seguirá produciendo un horario factible, pero de menor calidad que si se cumplieran todas. Las soluciones que no violan las restricciones fuertes son las

llamadas soluciones factibles del problema. Las restricciones suaves no son esenciales (Nandhini and Kanmani, 2009).

Como se mencionara anteriormente, cada institución puede tener su propio conjunto de restricciones pero, existen restricciones que son compartidas por cualquier horario educacional. Por ejemplo:

- Ningún grupo de estudiantes puede tener más de una actividad docente en el mismo espacio de tiempo.
- Ningún profesor puede tener más de una actividad al mismo tiempo.
- Ningún grupo docente o profesor puede tener más actividades que espacios de tiempo disponibles para recibirlas o impartirlas respectivamente.
- No puede haber más de una actividad asignada a un mismo local en un mismo espacio de tiempo.

Este problema ha sido objeto de numerosas investigaciones desarrolladas por la comunidad científica internacional. Sus características (complejidad), hacen necesaria una revisión previa para elegir las herramientas adecuadas para la aplicación en un centro determinado, lo cual es precisamente el objetivo de este trabajo.

1.2.2. Métodos para resolver estos problemas

Los métodos heurísticos se utilizan para tratar de encontrar soluciones a problemas para los que no existe un algoritmo que converja a la solución ni exista una fórmula explícita que la encuentre. Según (Hooker, 1995) "...un método heurístico es un procedimiento para resolver un problema complejo de optimización mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución de manera eficiente..."

Sobre la base de esta definición, y apoyándose en el hecho de que no se puede asegurar que existe el mejor método heurístico capaz de obtener los mejores resultados para cualquier problema de optimización, lo cual se conoce como Teorema *No Free Lunch* (Whitley and Watson, 2005), en (Wolpert and Macready, 1997) se emplean diferentes métodos para tratar de resolver diferentes instancias del problema de generación de horarios.

Cuando los problemas son demasiado complejos, puede ser que los métodos heurísticos tradicionales no sean capaces de encontrar soluciones a ellos, por esto surgen las meta- heurísticas:

¨...una clase de métodos aproximados que están diseñados para resolver problemas complejos de optimización, en los que los heurísticos clásicos no son efectivos. Las meta-heurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos...¨ (Osman and Kelly, 1996).

A continuación, se describen algunas de las heurísticas y meta-heurísticas que existen para resolver problemas de generación de horarios educacionales. Se aprecian diferencias, puntos de contacto y combinaciones entre ellas, siempre con el objetivo de mejorar las soluciones o reducir el costo computacional de obtenerlas.

Una de las primeras técnicas heurísticas que se utilizaron para la solución de problemas de generación de horarios fue el coloreado de grafos (De Werra, 1996). La técnica original consiste en encontrar una coloración adecuada para cada nodo del grafo, de forma tal que no exista ningún par de nodos adyacentes con el mismo color. Estableciendo una analogía con el problema que dio origen a esta técnica, un problema de generación de horarios puede modelarse de la siguiente forma:

Cada nodo representa un grupo y cada arista significa que esos dos grupos están involucrados en una restricción por lo que no se pueden colorear del mismo color. Los colores pueden ser los espacios de tiempo o los locales disponibles. Sin embargo esta representación no permite modelar el amplio conjunto de restricciones que se encuentran normalmente en un problema de generación de horarios académicos (De Werra, 1996).

No obstante a esto se han reportado casos exitosos utilizando el coloreado de grafos cuando se generan horarios sencillos semanalmente (Koyuncu and Seçir, 2007) o para encontrar la menor cantidad de locales necesarios reduciendo el número de colores disponibles (Selim, 1988).

Esta estrategia puede resultar muy conveniente para generar horarios de exámenes debido a que estos tienen muchas menos restricciones. Los nodos constituirían los exámenes y las aristas relacionan a dos exámenes a los que debe asistir al menos un mismo estudiante (Abdullah, 2006).

Otra meta-heurística muy aplicada y trabajada por los investigadores en la comunidad internacional es la búsqueda tabú. Una descripción bien detallada de todas sus características se puede encontrar en el libro de (Glover and Laguna,

1999). El algoritmo de búsqueda tabú (TS por sus siglas en inglés) realiza una búsqueda exhaustiva en una vecindad o subconjunto de una vecindad, seleccionando el mejor movimiento de los que se consideraron, de acuerdo a una función de calidad. Para evitar atascos en óptimos locales se utiliza una lista con n soluciones previamente visitadas, las que estarán ahí durante un número determinado de movimientos (ésta lista da el nombre al método) (Gendreau and Potvin, 2005). Sobre este problema resalta el trabajo de (Kendall and Hussin, 2005), quienes incorporaron la búsqueda tabú como parte de una hiper-heurística para seleccionar heurísticas que resuelvan tanto problemas de horarios de exámenes como horarios de actividades lectivas.

Otro enfoque utilizado para resolver este tipo de problemas es la técnica de sistemas computacionales basados en restricciones (Freuder and Wallace, 1992). Los problemas de satisfacción de restricciones o *constraint satisfaction problems* por sus siglas en inglés CSP pueden ser formulados como CSP = (X, D, C) donde X es un conjunto finito de variables $X = x_1, x_2, ..., x_n$ (por ejemplo periodos o aulas para cada curso), D es un conjunto finito de valores del dominio $D = d_{1x}, d_{2x}, ..., d_{nx}$, los cuales las variables pueden tomar (por ejemplo posibles horas de inicio de los periodos o aulas disponibles); y C es un conjunto finito de restricciones, $C = c_1, ..., c_m$, donde las restricciones son relaciones entre subconjuntos de estas variables. De aquí que la solución final del CSP es una asignación de valores a cada variable de tal forma que todas las restricciones sean cumplidas (Müller, 2005; Zhang and Lau, 2005).

Se conoce de experiencias cubanas aplicando esta técnica. Tal es el caso del trabajo de (Tamayo et al., 2010) quienes proponen una alternativa para el proceso de planificación basado en CSP que tiene en cuenta no solo las restricciones explícitamente declaradas sino también un conjunto de criterios de prioridad (Chiong Molina, 1995) que otorgan mayor calidad a los horarios generados. Las restricciones están en la naturaleza de los problemas de generación de horarios lo que hace que modelar las instancias de un problema siguiendo esta meta-heurística, se vea muy natural.

Los algoritmos genéticos (AG) están basados en el proceso de evolución natural y usan un conjunto de términos de las ciencias biológicas para un mejor entendimiento. Para resolver problemas usando AG es necesario modelar los cromosomas, que serán soluciones candidatas, así como los operadores de cruce y mutación que se utilizarán. Otro componente fundamental de esta meta-heurística es la función

de aptitud o calidad, *fitness* en inglés, que es la encargada de evaluar cuán bueno es cada cromosoma.

Los problemas de generación de horarios y problemas de secuenciación de tareas figuran entre los más apropiados para resolver con esta popular técnica y una prueba de ello es la cantidad de autores que la han utilizado y continúan investigando sobre el tema, como es el caso de (Burke et al., 1994) y (Jat and Yang, 2009). Es innegable que es una buena opción para afrontar problemas de planificación aunque no carece de detractores por su carácter estocástico y tiempos elevados de ejecución.

Para aplicar optimización con enjambre de partículas (PSO por sus siglas en inglés *Particle Swarm Optimization*) en generación de horarios (problema con dominio discreto) se necesitan realizar algunos ajustes a su enfoque tradicional (orientado a problemas con dominio continuo). Según (Poli, 2008), quien realizara un estudio sobre las diferentes aplicaciones de PSO, de todas las aplicaciones revisadas solo el 3,5 % corresponden a problemas de optimización combinatoria. Por esto se puede decir que PSO no es muy popular a la hora de resolver problemas de este tipo. Sin embargo, se han reportado experiencias positivas utilizando esta técnica, tal es el trabajo de (Montero et al., 2011) quienes lo utilizan en lo que denominan la primera de dos fases de un algoritmo diseñado para generar horarios docentes que variarán dinámicamente al introducirles actividades durante el semestre. Otro resultado positivo fue el obtenido por (Matijaš et al., 2010), quienes, con relativamente bajo consumo de memoria, obtuvieron un algoritmo ajustable a diferentes tipos de instituciones que trabaja muy bien con instancias grandes del problema.

Como se reportara en (Socha et al., 2003), usando Recocido Simulado (SA por sus siglas en inglés), se pueden obtener buenas soluciones para instancias del problema de asignación de horarios de tamaño medio. Otras propuestas han sido enfocadas al uso de hiper-heurísticas con base en el SA o su hibridación con otras meta o hiper-heurística debido a que el carácter particular de cada instancia del problema, sobre el supuesto de las diferencias entre las instituciones que lo presenten, hacen que la selección de un método u otro sea crítica para obtener un buen tiempo de respuesta, menor espacio de almacenamiento y el valor real de la solución (Nandhini and Kanmani, 2009).

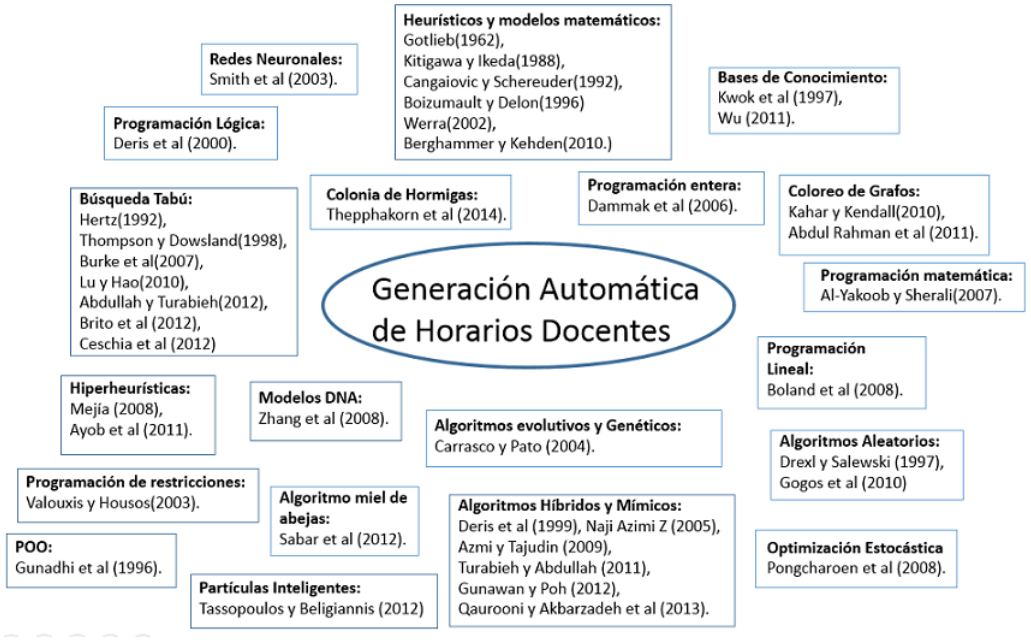


Figura 1.2: Métodos para resolver problemas de generación de horarios.

1.3. Aprendizaje Reforzado

El Aprendizaje Reforzado, como se menciona en (Kaelbling et al., 1996) , se remonta a los momentos iniciales de la cibernética y los trabajos en estadística, psicología, neurociencia y ciencias de la computación. Durante las últimas décadas también ha atraído el interés de la comunidad de inteligencia artificial. Aprendizaje Reforzado es aprender qué hacer (como asociar situaciones con acciones) de forma tal que se maximice una señal numérica de recompensa. Al aprendiz no se le dice qué acciones tomar, sino que debe descubrir qué acciones arrojan la mayor recompensa seleccionándolas. En los casos más interesantes las acciones no sólo afectan la recompensa inmediata sino también los escenarios siguientes y por tanto, las recompensas futuras. Estas dos características, búsqueda a 'prueba y error' y recompensa retardada son dos de las características más importantes que distinguen el aprendizaje reforzado.

En el modelo estándar del aprendizaje reforzado un agente está conectado con su ambiente vía percepción y acción, en cada interacción el agente percibe el estado actual s del ambiente y selecciona una acción a para cambiar dicho estado. Esta transición genera una señal r que es recibida por el agente, cuya tarea es aprender una política de selección de acciones en cada estado para recibir la mayor recompensa acumulada a largo plazo (Zhang, 1996).

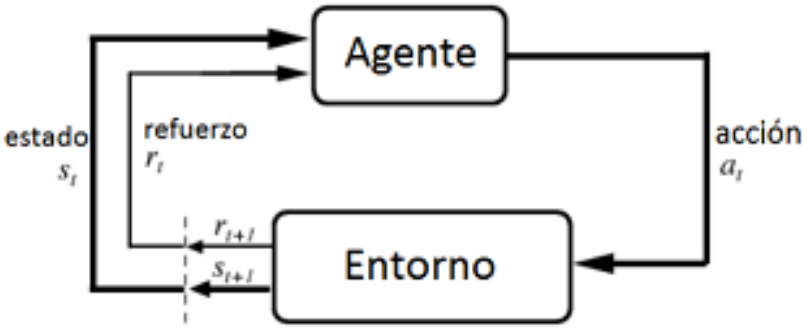


Figura 1.3: Aprendizaje Reforzado

En la figura 1.3 (Sutton and Barto, 1998)se describe la dinámica básica de este tipo de aprendizaje. En el momento t el agente se encuentra en el estado s_t y con un refuerzo r_t. La ejecución de la acción a_t resulta en un refuerzo r_{t+1} (puede ser una recompensa o castigo) y en un pasaje hacia el estado s_{t+1}.

Uno de los retos que se plantea el aprendizaje por refuerzo es el equilibrio entre la exploración y la explotación. El conflicto de intereses es claro: por un lado el agente prefiere llevar a cabo las acciones que se han ejecutado en el pasado y cuyas recompensas han demostrado ser altas; y por otro lado, es evidente que para descubrir este tipo de acciones debe probar acciones que no ha tomado aún. En otras palabras, el agente debe explotar lo que sabe hasta el momento para obtener recompensas altas, pero también debe explorar nuevas acciones para conseguir acciones que en el futuro le generen recompensas altas. Lo que se produce es un conflicto entre la recompensa de corto plazo y la de largo plazo. El dilema consiste en que ninguna de estas tareas puede ser ejecutada de forma exclusiva sin que la otra falle. Por lo tanto, el agente deberá balancear entre una tarea y la otra para obtener el mejor rendimiento, y para ello tendrá que ir probando una variedad de acciones nuevas y de forma progresiva ir inclinándose hacia aquellas que lo recompensen mejor.

1.3.1. Elementos del Aprendizaje Reforzado

Más allá del agente y del ambiente, se pueden identificar 4 subelementos de un sistema de aprendizaje por refuerzo: una política, una función de recompensa, una función de valor, y un modelo del ambiente.

Política

Una política define el comportamiento de un agente aprendiz en un momento dado. En otras palabras, una política es un mapeo entre un conjunto de estados S percibidos del ambiente y un conjunto de acciones A que deben ejecutarse en dichos estados. En algunos casos la política puede ser tan simple como una función o una tabla de entradas, y en otros casos puede ser un proceso complejo de búsqueda. La política es lo más importante de un agente que aprende por refuerzo, en el sentido de que es lo que determina su comportamiento.

Función de recompensa

Una función de recompensa R define la meta en un problema de aprendizaje por refuerzo. Esta función hace un mapeo de cada estado (o par estado-acción) percibido del ambiente a un único número $r \epsilon R$ llamado recompensa, indicando qué tan deseable es cada estado.

El objetivo principal de un agente que aprende por refuerzo es maximizar la recompensa total que recibe a largo plazo. La función de recompensa define qué eventos son buenos y malos para el agente. La función de recompensa no puede ser modificada por el agente, sin embargo puede servir como la base para modificar la política. Por ejemplo, si la acción seleccionada por la política es seguida por una recompensa baja, entonces la política puede cambiarse para que en el futuro nos indique una acción diferente cuando nos encontremos en la misma situación.

Función de valor

Una función de valor F nos indica qué acciones son buenas a largo plazo. En pocas palabras, el valor de un estado es la cantidad total de recompensa que un agente puede esperar acumular en el futuro, empezando en ese estado. Mientras que las recompensas son dadas directamente por el ambiente, los valores de los estados deben ser estimados y reestimados a partir de las observaciones que el agente haga a lo largo de su interacción con el ambiente. De hecho, el componente más importante de casi todos los algoritmos de aprendizaje por refuerzo es un método para estimar eficientemente los valores de los estados. Puede verse como una función que asigna a cada par estado-acción $(s_i; a_i)$ tal que $s_i \epsilon\ S$ y $a_i\ \epsilon\ A$, un número real $r \epsilon\ R$.

Modelo

El modelo imita el comportamiento del ambiente. Por ejemplo, dado un estado y una acción, el modelo podrá predecir el siguiente estado y recompensa resultantes. Los modelos son utilizados para planear, con esto nos referimos a cualquier forma que tengamos para decidir un curso de acción, al considerar situaciones futuras posibles antes de que sean experimentadas realmente.

Algunos investigadores intentan resolver el problema de aprendizaje por refuerzo sin tener modelo alguno, y en ese caso encuentran directamente un mapeo de los estados del ambiente a las acciones, a este enfoque se le conoce como libre de modelo. Otro enfoque consiste en construir primeramente un modelo del ambiente, el cual lo utilizan posteriormente para seleccionar las mejores acciones.

Recientemente se han propuesto algoritmos híbridos que utilizan ambos enfoques, dependiendo del estado actual del ambiente y el conocimiento acumulado del agente (Tamar et al., 2012).

1.3.2. Q-Learning

Un algoritmo muy conocido dentro del aprendizaje reforzado es el Q-Learning (QL), el cual se basa en aprender una función acción-valor que brinda la utilidad esperada de tomar una acción determinada en un estado específico. El centro del algoritmo es una simple actualización de valores, cada par (s, a) tiene un Q-valor asociado, cuando la acción a es seleccionada mientras el agente está en el estado s, el Q-valor para ese par estado-acción se actualiza basado en la recompensa recibida por el agente al tomar la acción. También se tiene en cuenta el mejor Q-valor para el próximo estado s', la regla de actualización completa es la siguiente:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma(max)_{a'}Q(s',a') - Q(s,a)]$$

En esta expresión, $\alpha \ \epsilon \ [0;1]$ representa la velocidad del aprendizaje y r la recompensa o penalización resultante de ejecutar la acción a en el estado s. La velocidad de aprendizaje α determina el grado por el cual el valor anterior es actualizado. Por ejemplo, si $\alpha = 0$, entonces no existe actualización, y si $\alpha = 1$, entonces el valor anterior es reemplazado por el nuevo estimado.

Normalmente se utiliza un valor pequeño para la velocidad de aprendizaje, por ejemplo, $\alpha = 0{,}1$. El factor de descuento (parámetro γ) toma un valor entre 0 y 1, si está cercano a 0 entonces el agente tiende a considerar sólo la recompensa

inmediata, si está cercano a 1 el agente considerará la recompensa futura como más importante(Jiménez, 2012).

El algoritmo Q-Learning es usado por los agentes para aprender de la experiencia, donde cada episodio es equivalente a una sesión de entrenamiento. En cada iteración el agente explora el ambiente y obtiene señales numéricas hasta alcanzar el estado objetivo. El propósito del entrenamiento es incrementar el conocimiento del agente, representado a través de los Q-valores. A mayor entrenamiento, mejores serán los valores que el agente puede utilizar para comportarse de una forma más óptima.

Inicializar los Q-values de forma arbitraria

Para cada iteración

  Inicializar s

  Para cada paso de la iteración

    Escoger a de s

    Tomar la acción a, observar el siguiente estado s´ y r

    Actualizar los Q-valores,

    $\mathbf{Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]}$

    $\mathbf{s \leftarrow s'}$

  fin ciclo interior

fin ciclo exterior

Figura 1.4: Pseudocódigo del algoritmo Q-Learning

1.4. Base de datos

Una base de datos es un “almacén” que permite guardar grandes cantidades de información de forma organizada para que luego se pueda encontrar y utilizar fácilmente. Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos. Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una

parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

Entre las principales características de los sistemas de base de datos se pueden mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

1.4.1. Sistema Gestor de Base de Datos (SGBD)

Un Sistemas Gestor de Base de Datos (en inglés *DataBase Management System*) es un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Entre los diferentes tipos de SGBD, se pueden encontrar los siguientes:

- MySQL: es una base de datos con licencia GPL basada en un servidor. Se caracteriza por su rapidez. No es recomendable usar para grandes volúmenes de datos.
- PostgreSql y Oracle: Son sistemas de base de datos poderosos. Administra muy bien grandes cantidades de datos, y suelen ser utilizadas en intranets y sistemas de gran calibre.
- Access: Es una base de datos desarrollada por Microsoft. Esta base de datos, debe ser creada bajo el programa Access, el cual crea un archivo .mdb con la estructura ya explicada.

– Microsoft SQL Server: es una base de datos más potente que Access desarrollada por Microsoft. Se utiliza para manejar grandes volúmenes de informaciones.

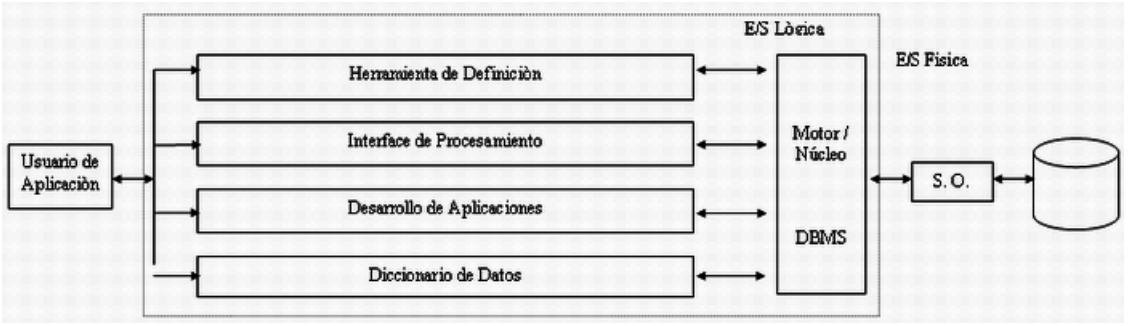


Figura 1.5: Subsistema de un SGBD.

1.4.2. MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. Soporta gran cantidad de tipos de datos para los campos de las tablas. Dispone de API's para diferentes lenguajes, gran portabilidad entre sistemas, además soporta hasta 32 índices por tabla. Gestiona usuarios y contraseñas, manteniendo un buen nivel de seguridad en los datos. Es una de las herramientas más utilizadas y posee infinidad de bibliotecas y otras herramientas que permiten su uso a través de múltiples lenguajes de programación.

La siguiente figura es una visión abstracta de la arquitectura lógica de MySQL. En la figura 1.6 se hace una división entre los componentes que conforman el servidor, las aplicaciones cliente que lo utilizan y las partes del sistema operativo en las que se basa el almacenamiento físico.

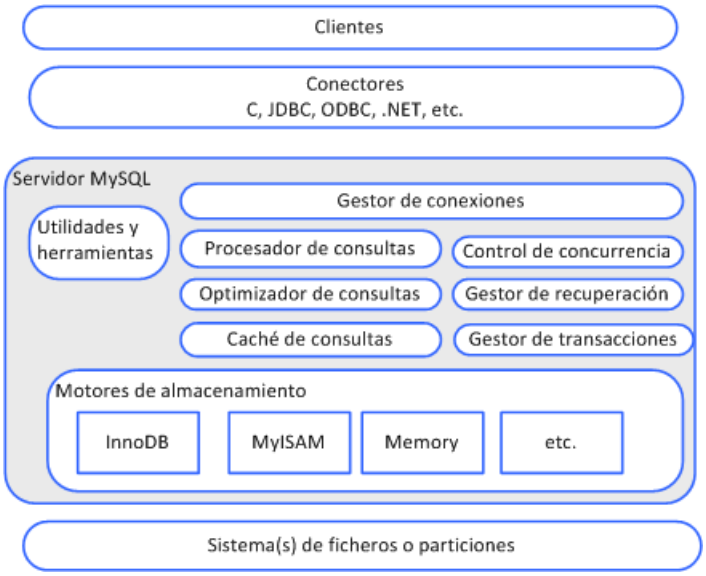


Figura 1.6: Estructura de MySQL.

1.4.3. Mapeo Objeto/Relacional (ORM)

Object-Relational mapping, o lo que es lo mismo, mapeo objeto-relacional, es un modelo de programación que consiste en la transformación de las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador. El ORM tiene una capa intermedia que abstrae al programador de la base de datos y lo centra en el desarrollo de la aplicación. Permite crear, modificar, recuperar y eliminar objetos persistentes, además permite navegar por las asociaciones entre objetos y luego actualizarlos al finalizar una transacción, realizando todas estas labores a través de un lenguaje de alto nivel orientado a objetos. Algunas de las ventajas de un ORM es su facilidad y velocidad de uso, además la abstracción de la base de datos usada y la seguridad de la capa de acceso a datos contra ataques.

Casi todos los lenguajes de alto nivel actualmente disponen de alguna solución de este tipo, una de las más conocidas es Hibernate para Java, pero existen muchas más:

Java: Hibernate, iBatis, Ebean.

NET: Entity Framework, nHibernate.

PHP: Doctrine, Propel, ROcks, Torpor.

1.4.4. Hibernate

Usar JDBC (*Java Data Base Connectivity*) es complejo y muy dependiente de la estructura de los datos. Sería más natural y mucho más sencillo trabajar

directamente con objetos, pero es imposible con las bases de datos relacionales, la mejor opción entonces es utilizar un motor de persistencia, que es el componente software encargado de traducir entre objetos y registros. Un motor de persistencia de código abierto es Hibernate, que permite hacer cosas como poder guardar un objeto en la base de datos simplemente con *session.save(miObjeto)* o borrarlo con *session.delete(miObjeto)*.

Hibernate es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los *beans* de las entidades que permiten establecer estas relaciones. Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la Programación Orientada a Objetos (POO). Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por SQL como lenguaje. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

En cualquier aplicación que use Hibernate aparecen cuatro objetos básicos: *Configuration*, *SessionFactory*, *Session y Transaction*. El primero contiene la información necesaria para conectarse a la base de datos. *SessionFactory* es una fábrica de *Sessions*, un objeto *Configuration* es capaz de crear una *SessionFactory* ya que tiene toda la información necesaria. Normalmente, una aplicación sólo tiene un *SessionFactory*, que es el objeto mediante el cual se abren nuevas sesiones de Hibernate. El tercero es la principal interfaz entre la aplicación Java e Hibernate, es la que mantiene las conversaciones entre la aplicación y la base de datos. Permite añadir, modificar y borrar objetos en la base de datos y por último *Transaction,* que como su nombre indica, se encarga de la transaccionalidad y permite definir unidades de trabajo.

Para insertar objetos en la base de datos se usa el método *save(Object objeto)* de *Session*, para insertar o actualizar si ya existe *saveOrUpdate(Object objeto)*, para borrar *delete(Object objeto)* y para cargar un objeto desde la base de datos *get(String clase, Tipo id)* o *load(String clase, Tipo id)* que devuelven el objeto de la clase indicada por el primer parámetro y con identificador el segundo parámetro si es que existe en la base de datos. Para hacer búsquedas en la base de datos podemos usar *find(String consulta)* que devuelve una lista con los resultados, o *iterate(String consulta)* que devuelve un iterador, o bien crear un objeto *Query* usando *createQuery(String consulta)* y llamar más tarde al método *find()* del *Query*. La consulta en todos los casos estará escrita en un lenguaje similar a SQL propio de Hibernate llamado HQL (*HibernateQueryLanguage*).

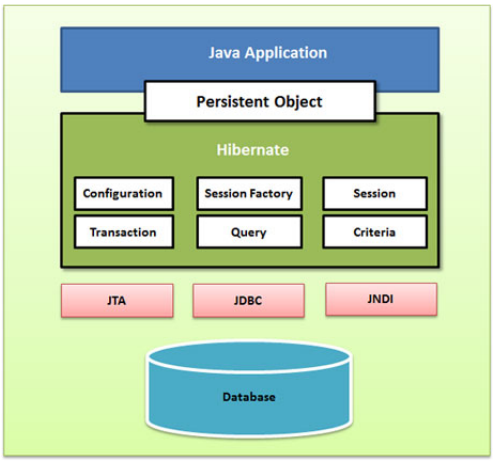


Figura 1.7: Arquitectura de Hibernate.

1.5. Conclusiones parciales

En este capítulo se abordaron distintos temas de interés relacionados con la investigación que se lleva a cabo en este trabajo. De esta forma se arribaron a las siguientes conclusiones parciales:

- El problema de la generación automática de horarios se define como: la asignación, sujeta a restricciones, de un grupo de recursos a objetos ubicados en tiempo y espacio, de tal manera que se satisfagan un conjunto de objetivos deseados. Se clasifican en 6 subproblemas: horarios de cursos, horarios para grupo-profesor, asignación de estudiantes a grupos, asignación de profesores a grupos, asignación de locales a actividades y horarios de exámenes.

- Uno de los factores que hacen este problema tan complejo es la necesidad de satisfacer una serie de restricciones las cuales pueden ser fuertes o blandas;

las primeras deben cumplirse, la violación de alguna, origina un horario no factible y bajo ninguna circunstancia deben ser infringidas. Por otro lado, las restricciones blandas denotan preferencias del usuario y se desea que se cumplan en la medida de lo posible. La violación de algunas de ellas seguirá produciendo un horario factible, pero de menor calidad que si se cumplieran todas.

- Existen heurísticas y meta-heurísticas para resolver problemas de generación de horarios. Entre ellas se aprecian diferencias, puntos de contacto y combinaciones, siempre con el objetivo de mejorar las soluciones o reducir el costo computacional de obtenerlas.

- Aprendizaje Reforzado es aprender qué hacer de forma tal que se maximice una señal numérica de recompensa. Al aprendiz no se le dice que acciones tomar, sino que debe descubrir que acciones arrojan la mayor recompensa seleccionándolas.

- Un algoritmo muy conocido dentro del aprendizaje reforzado es el Q-Learning (QL), el cual se basa en aprender una función acción-valor que brinda la utilidad esperada de tomar una acción determinada en un estado específico.

Capítulo II

Capítulo II: Arquitectura del sistema

En el presente capítulo se describe el problema real de cómo se realiza la confección de horarios actualmente en la Facultad de Matemática – Física – Computación y además se caracterizan las principales herramientas utilizadas para la implementación del sistema desarrollado, el cual consta de un servicio web para la recopilación de los datos necesarios que serán almacenados en una base de datos y luego utilizados por una aplicación de escritorio que será la rectora en el proceso de generación automática del horario docente.

2.1. Descripción del problema

La Facultad de Matemática – Física – Computación imparte actualmente en total cuatro carreras en pregrado: Licenciatura en Ciencia de la Computación, Ingeniería Informática, Licenciatura en Física y Licenciatura en Matemática, y se pronostica que asuma una nueva carrera, Licenciatura en Ciencia de la Información . Las primeras tres carreras tienen una duración de 10 semestres, mientras que la última, tiene una duración de 8 semestres. Cabe destacar que existe una malla curricular en el plan de estudio de cada carrera, la cual determina el orden en que los alumnos deben tomar las distintas asignaturas. De este modo, es posible caracterizar cada asignatura por el semestre en que se ubica dentro del mencionado plan de estudio, conociendo así información útil a la hora de definir qué asignaturas no pueden ser dictadas en forma simultánea en un mismo turno .

Cada semestre se dictan en promedio 126 asignaturas, las que tienen un número variable de frecuencias. Las asignaturas tienen varias formas de docencia, por ejemplo: conferencias, clases prácticas, laboratorios, seminarios, talleres, entre otras. Las frecuencias de las asignaturas dependen de la cantidad de horas clases con que cuentan las mismas. Las clases se imparten de lunes a viernes en turnos de 90 minutos de duración. Cada día se compone de 6 turnos de clases como máximo.

Un curso se divide en 2 sesiones de clases según el año docente: sesión matutina y sesión vespertina. La Facultad cuenta, hasta el momento, con 11 aulas, las cuales son compartidas por todos los grupos de cada año docente de todas las carreras impartidas, además tiene 3 laboratorios de Computación de los cuales 2 son

compartidos por las carreras Ciencia de la Computación e Ingeniería Informática, y el restante es compartido entre Matemática y Física y además cuenta con un laboratorio experimental de Física. Cabe destacar que las aulas son el recurso escaso de la Facultad.

Actualmente, en la Facultad de MFC la confección del horario docente es una tarea engorrosa, consumidora de tiempo y que además implica a un gran número de profesores. Todo el proceso comienza cuando al terminar cada semestre el Jefe de cada Departamento solicita el P1 de cada asignatura al profesor que impartirá clases de dicha asignatura en el próximo semestre y se hacen llegar al Vice Decano Docente (VDD) estos documentos para su total aprobación. Estos documentos incluyen entre otras cosas: las frecuencias de las asignaturas durante las semanas hábiles del semestre, las formas de enseñanza, el sistema de evaluación, la bibliografía a utilizar, los medios y locales a utilizar y las preferencias o afectaciones del profesor. Posteriormente se les emite a los Jefes de Año, encargados de confeccionar el P4 correspondiente a su año, teniendo en cuenta indicadores concretos. Todos los coordinadores se reúnen, en un primer momento, y acuerdan la necesidad de horas laboratorios para cada año y cada asignatura, evitando una coincidencia en la utilización de los mismos. Luego colocan en los espacios libres las otras frecuencias de las asignaturas de cada año por semana. Por último se discute en la Junta de Año antes de comenzar el semestre y se aprueba el horario, el cual no debe ser modificado a no ser por una causa de fuerza mayor.

Considerando los antecedentes planteados anteriormente, la generación de la programación de horarios se transforma en una tarea en extremo compleja y que consume una enorme cantidad de recursos. Por este motivo, en este trabajo se busca que los requerimientos impuestos por la Facultad sean apropiadamente estructurados mediante la formulación de un modelo de programación basado en aprendizaje reforzado. La resolución de este modelo entregará la programación de horarios y asignación de aulas y profesores a grupos docentes de manera óptima de acuerdo a una determinada función objetivo.

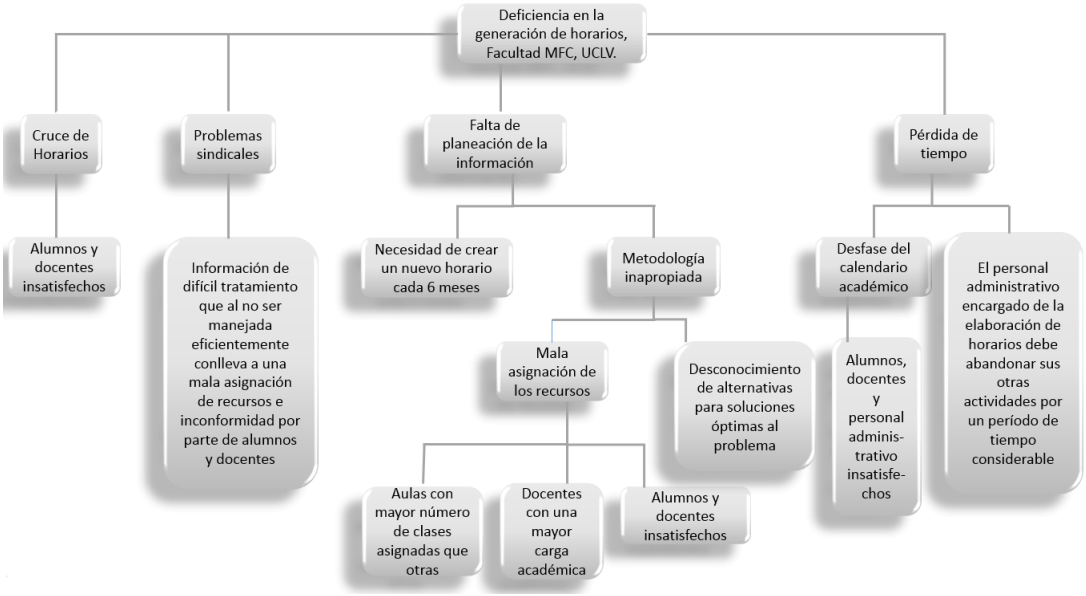


Figura 2.8: Deficiencias en la confección manual de horarios.

2.2. Aplicación web

Una de las ventajas que tienen las páginas web con respecto a otros medios de publicación, es la interactividad, ya que cuando el usuario ingresa a la página web, inmediatamente puede interactuar con ella, puede visitar lo que más le interese, dejar su opinión, comunicarse por medio del correo electrónico o simplemente ingresar información. Otra ventaja de las páginas web es la accesibilidad, ya que se encuentran disponible para todos, cualquier persona que esté interesada en una página web la puede visitar, independientemente del lugar en donde se encuentre ubicado el usuario, ésta es una razón por la cual se podría decir que "una publicación web es esencialmente democrática, ya que ofrece información muy especializada"(Piwonka and Arata, 1996).

Debido a todas las ventajas expuestas anteriormente se decidió utilizar un pequeño sitio web con el objetivo de captar la información necesaria para la generación de los horarios. El sitio Formulario-P1 va dirigido especialmente a los profesores principales de cada año que serán los encargados de introducir datos como, la información personal de cada profesor, todas sus afectaciones y las características de la asignatura que imparte dicho profesor, incluyendo el P1 de la misma. Esta información será almacenada en una base de datos para ser procesada por la aplicación de escritorio y así lograr la generación automática de los horarios de la facultad.

2.2.1. Herramientas

En el desarrollo de aplicaciones web se tienen herramientas para el diseño, maquetación, programación y depuración. Todas son muy importantes, y por ello se debe elegir la más adecuada de acuerdo a las necesidades y capacidades. Para el desarrollo del sitio Formulario-P1 se eligieron como herramientas principales para la programación el lenguaje HTML, para estructurar y presentar el contenido en la web, PHP para la creación de contenidos dinámicos dentro de la web y Java Script para crear diferentes efectos e interactuar con los usuarios.

HTML (*Hyper Text Markup Language*) es un lenguaje de marcado usado para estructurar y presentar el contenido en la web. HTML 5 es el nombre que se usa para referirse a la quinta versión de este lenguaje, el cual es el resultado de agrupar las especificaciones relacionadas al desarrollo de páginas web: HTML 4 , XHTML 1 , DOM nivel 2 (*Document Object Model*), e integrar algunos elementos de CSS nivel 2. HTML 5 describe la estructura y el contenido de la web usando solo texto y lo complementa con objetos tales como imágenes, flash y otros. Su estructura se compone de etiquetas entre las cuales van insertados los diferentes elementos que componen la página como son los bloques de texto, scripts y la ruta a la ubicación de archivos externos como imágenes y otros archivos multimedia. Al navegador cargar dichos archivos representa todos los elementos en ella de forma adecuada. HTML 5 surge como una evolución lógica de las versiones anteriores y por la necesidad de lograr los siguientes objetivos:

- Lograr que la información, y la forma de presentarla estén lo más separadas posible.
- Resumir, simplificar y hacer más sencillo el código utilizado.
- Definir un lenguaje que haga las páginas compatibles con todos los navegadores web, incluyendo los de los teléfonos móviles y otros dispositivos modernos usados en la actualidad para navegar en Internet.
- Eliminar restricciones que hagan el código más popular y asequible.

PHP, se define como un lenguaje de programación para la rápida creación de contenidos dinámicos dentro de sitios web, como son los foros, blogs, sistemas de noticias, formularios, entre otros. También, crea aplicaciones gráficas independientes del navegador y aplicaciones para servidores. Es un lenguaje *script* dentro del HTML. La principal función del PHP es permitir la interacción de la página

web con el usuario, además de procesar la información de formularios, generar páginas con contenidos dinámicos, y enviar y recibir *cookies*. Puede ser utilizado en cualesquiera de los principales sistemas operativos del mercado y soporta la mayoría de servidores web que existen en la actualidad.
Otra de las características más potentes y destacables de PHP es su soporte para una gran cantidad de bases de datos.Tiene la capacidad de expandirse potencialmente por su gran cantidad de módulos; con bibliotecas de funciones externas; es de código abierto y por lo tanto asequible para todo el que lo desee. El PHP en la actualidad ha sido muy solicitado por los programadores, principalmente por su potencia, alto rendimiento, fácil aprendizaje, escasez de consumo de recursos, y lo más importante, su gratuidad.

Java Script es un lenguaje de programación web, el cual brinda funciones muy potentes que hace que una página web pueda ser dinámica, rápida y puede mejorar la vista de la interfaz de la misma. Es un lenguaje interpretado, es decir, no requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos, además es orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Tiene la ventaja de ser incorporado en cualquier página web y puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado. Tiene como principal característica ser un lenguaje independiente de la plataforma. Con él se pueden crear diferentes efectos e interactuar con los usuarios.

Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas, entre otros. Todos los navegadores modernos interpretan el código Java Script integrado en las páginas web. Es una excelente solución para poner en práctica la validación de datos de un formulario en el lado del cliente.

2.2.2. Diseño del sitio web

El diseño web es una actividad que exige prestar atención a múltiples factores: las características del público al que va dirigido el sitio, sus recursos de *hardware* y *software*, las posibilidades que tiene la red para transmitir datos, los recursos técnicos, financieros y temporales disponibles y los objetivos de los realizadores del sitio, entre otros. Un diseño web efectivo no sólo debe contemplar cada uno de estos aspectos en el proceso de desarrollo, sino que debe estar abierto a sus posibles modificaciones: pueden aparecer o desaparecer tecnologías, los gustos de

los usuarios pueden modificarse o los objetivos de los responsables del sitio pueden redefinirse por un cambio de sus políticas o intereses.

El sitio web "Formulario-P1" se crea teniendo en cuenta las características del problema que se tiene que resolver con él, es decir, la necesidad de captar un conjunto de datos que serán proporcionados por un grupo de usuarios situados en momentos y lugares diferentes. Está estructurado linealmente, es decir, una página lleva a otra mediante enlace, por lo que la navegación resulta ser bien sencilla. Desde la página de inicio, los usuarios tendrán que autentificarse para acceder al contenido del sitio y tener la posibilidad de introducir todos los datos necesarios para la confección de los horarios en la página "P1" y desde la página "Eliminar" se podrá borrar cualquier profesor de la base de datos y toda la información que a él respecte.

2.2.3. Mapa de navegación

Los mapas del sitio aportan al usuario una visión general de los contenidos. Se pueden organizar a través de un diagrama jerárquico o utilizando metáforas geográficas. Ayudan a orientarse con facilidad dentro del sitio, ver su contenido y comprender la estructura de navegación. En la figura 2.9 se muestra el mapa de navegación del sitio en cuestión.

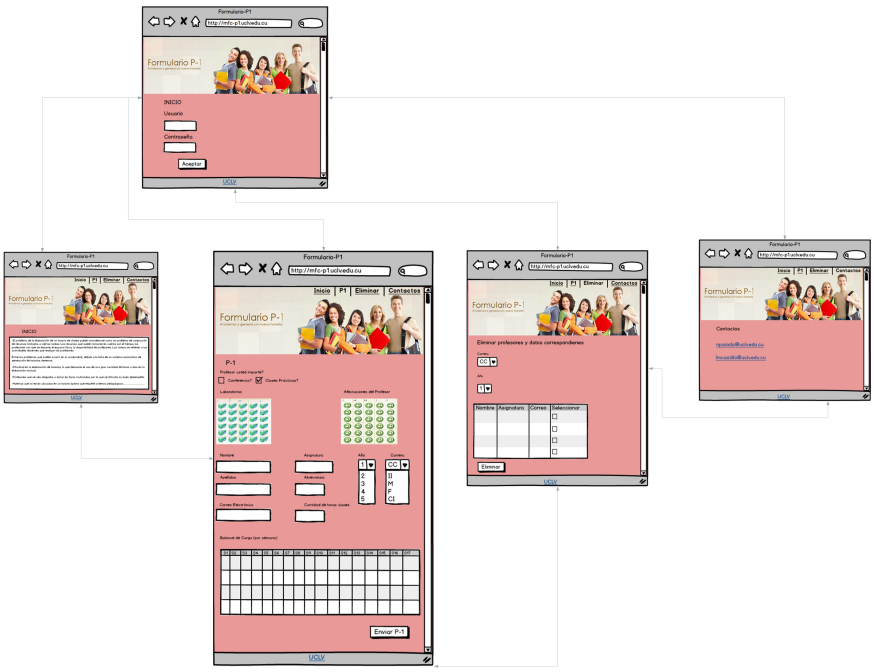


Figura 2.9: Mapa del Sitio Web

2.3. Diseño de la base de datos

El diseño de una base de datos consiste en definir la estructura de los datos que debe tener un sistema de información determinado. Para ello se suelen seguir por regla general ciertas fases en el proceso de diseño, definiendo para ello el modelo conceptual, el lógico y el físico.

2.3.1. Diseño conceptual

En el diseño conceptual se hace una descripción de alto nivel de la estructura de la base de datos, independientemente del SGBD que se vaya a utilizar para manipularla. Su objetivo es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar dicha información.
Por tanto, conceptualmente la base de datos ¨Horario¨ se diseña para dar solución al problema descrito en el epígrafe 2.1 la cual almacena parte de la información necesaria para la gestión de horarios docentes de la Facultad de MFC y maneja entidades como son: profesores (que imparten asignaturas), aulas (en las que se imparten clases), grupos de estudiantes, asignaturas que se imparten, afectaciones semanales para cada profesor, y laboratorios semanales para cada asignatura. La base de datos consta de 39 tablas, a continuación se resumen las características de cada una:

Tabla	Descripción	Campos
Profesor	Almacena información referente a los profesores que imparten las diferentes asignaturas.	Nombre, Apellidos, Asignatura(que imparte), Correo,Carrera,Año, imparteC(conferencias), imparteCp(clases prácticas).
Asignatura	Almacena información referente a las asignaturas que se imparten en un curso.	Nombre,Carrera,Año, abrev(abreviatura), cantHC(horas clases), s11,s21,s31,s51....s175,s176 (Planificación por semana).
Grupo	Almacena información referente a los todos los grupos que reciben las clases.	Numero,Carrera,Año, aulaC (aula donde recibe las conferencias), aulaCp (aula donde recibe las clases prácticas).

Aula	Almacena información referente a las aulas, donde se imparten las clases. Son compartidas por todos los grupos de la facultad.	numero, tipo(si es de conferencias o de clases prácticas).
AfectSem(i) $1 \leq i \leq 17$	Almacena los días y turnos que un profesor está afectado para impartir clases.(Existen 17 tablas de este tipo, una para cada semana de curso).	nombreProf,nombreAsig, L1,L2,L3,M1,M2....V4,V5,V6 (días y turnos semanales).
LabSem(i) $1 \leq i \leq 17$	Almacena los días y turnos que cada asignatura hará uso de laboratorios de computación. (Existen 17 tablas de este tipo, una para cada semana de curso).	nombreAsig, año,carrera, L1,L2,L3,M1,M2....V4,V5,V6 (días y turnos semanales).
Usuario	Almacena la infomación necesaria para autentificarse en el sitio web.	usuario,contraseña,tipo

Tabla 2.1: Tablas de la Base de Datos Horario.

2.3.2. Diseño lógico

El diseño lógico parte del resultado del diseño conceptual y da como resultado una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. El diseño lógico depende del tipo de SGBD que se vaya a utilizar, se adapta a la tecnología que se debe emplear, pero no depende del producto concreto. En el caso de bases de datos relacionales, el diseño lógico consiste en definir las tablas que existirán, las relaciones entre ellas, normalizarlas, entre otras.

El diagrama entidad/relación de la base de datos para manejar toda esta información se puede ver en la figura 2.10:

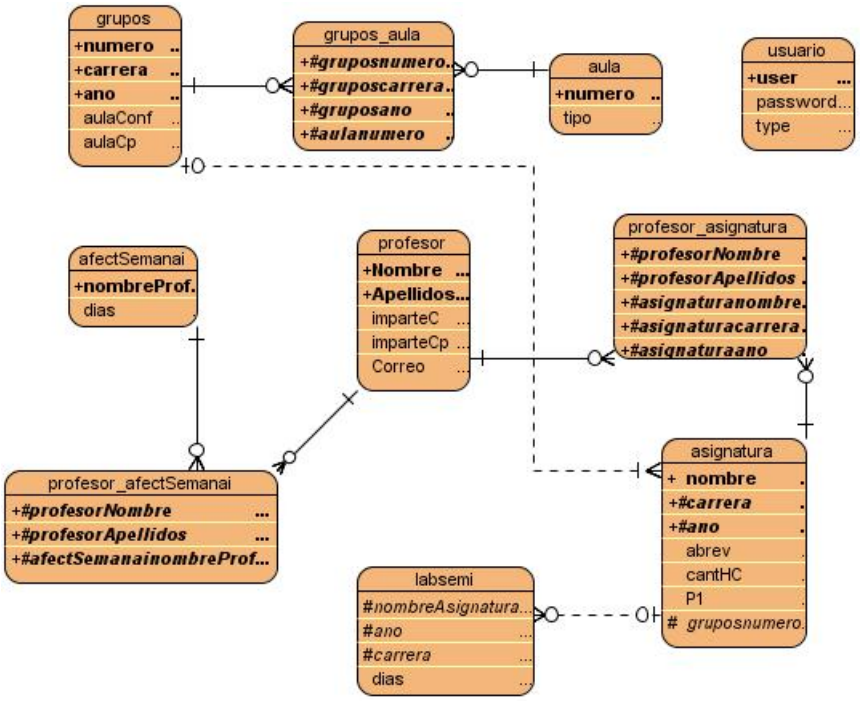


Figura 2.10: Diagrama Entidad/Relación.

Como se muestra existen tablas para representar cada una de estas entidades del mundo real. Además están relacionadas entre ellas de modo que, por ejemplo, un profesor tiene asociada una determinada asignatura, y lo mismo con las demás tablas. Cada tabla posee una serie de campos que representan valores que se quieren almacenar para cada entidad. Por ejemplo, un profesor posee los siguientes atributos que se traducen en los campos correspondientes para almacenar su información: Nombre , Apellidos, Correo y Tipo de docencia que imparte (Conferencias (imparteC) o Clases Prácticas (imparteCp)).

Cada tabla debe incluir una columna o conjunto de columnas que identifiquen inequívocamente cada fila almacenada en la tabla, esta información recibe el nombre de clave principal (o primaria) simple de la tabla , formada por un solo campo, o clave principal compuesta, cuando son necesarios dos o más campos de la tabla.

En el diagrama los campos marcados con "+" indican aquellos que son claves primarias. Por ejemplo, *número* es el identificador único de Aula, y *número*, *carrera* y *año* constituyen la clave primaria compuesta de la entidad Grupo.

Los campos marcados con "#" son claves foráneas o claves externas. Indican campos que van a almacenar claves primarias de otras tablas de modo que se puedan relacionar con la tabla actual. Por ejemplo, en la tabla Asignatura el campo

carrera está marcado como "+#" porque en él se almacenará el valor de la clave primaria de la tabla de Grupos que identifica al grupo que recibe esa asignatura y además es llave primaria porque a su vez identifica a cada una de las asignaturas.

Una relación es una asociación establecida entre campos comunes de dos tablas. existen tres tipos de relaciones: Uno a uno: cada registro de la Tabla A sólo puede tener un registro coincidente en la Tabla B, y viceversa. Uno a varios: Es el tipo de relación más común. En este tipo de relación, un registro de la Tabla A puede tener muchos registros coincidentes en la Tabla B, pero un registro de la Tabla B sólo tiene un registro coincidente en la Tabla A y el último tipo de relación es Varios a varios: un registro de la Tabla A puede tener muchos registros coincidentes en la Tabla B, y viceversa. Este tipo de relación sólo es posible si se define una tercera tabla (denominada tabla de unión) cuya clave principal consta de dos campos : las claves primarias de las Tablas A y B. Una relación de varios a varios no es sino dos relaciones de uno a varios con una tercera tabla.

En el diagrama existen 3 relaciones varios a varios entre las tablas Profesor y Asignatura, entre Profesor y AfectSem(i), y entre Grupo y Aula, nótese que efectivamente para cada caso se define una tercera tabla. También existe una relación uno a varios entre las entidades Asignatura y LabSem(i), debido a que una asignatura puede tener muchos registros en la tabla de laboratorios o no tener ninguno, (en caso de que alguna asignatura no tenga la necesidad de usar los laboratorios), pero un laboratorio solo puede tener un registro coincidente en la tabla asignaturas.

2.3.3. Diseño físico

El diseño físico parte del lógico y da como resultado una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Aquí el objetivo es conseguir una mayor eficiencia, y se tienen en cuenta aspectos concretos del SGBD sobre el que se vaya a implementar. Por regla general esto es transparente para el usuario, aunque conocer cómo se implementa ayuda a optimizar el rendimiento y la escalabilidad del sistema.

Para implementar esta base de datos se utilizó MySQL como SGBD, el cual tiene como característica más notable el separar el motor de almacenamiento (que se encarga de los detalles de entrada-salida y representación de la información en memoria secundaria) del resto de los componentes de la arquitectura. MySQL so-

porta varios motores de almacenamiento que tratan con distintos tipos de tabla, el motor predeterminado en este servidor Apache es InnoDB.

InnoDB es un mecanismo de almacenamiento transaccional. Esto aumenta la seguridad de los datos. Realiza bloqueo a nivel de filas y lecturas no bloqueantes MVCC tipo Oracle que aumentan la concurrencia y las prestaciones. Bloquea a nivel de fila (tupla), y proporciona concurrencia multiusuario para lectura. Organiza los datos en disco para optimizar el uso de claves primarias. Mantiene la integridad de datos mediante restricciones de llave foránea, que se aplican en *insert*, *update* y *delete*. InnoDB almacena los datos agrupados para reducir el flujo de entrada/salida de consultas habituales basadas en claves primarias. Se pueden combinar tablas InnoDB con tablas de otros mecanismos de almacenamiento, incluso en la misma consulta.

En cuanto a los métodos utilizados para acceder eficientemente a los datos se utilizó el patrón *Data Access Object* (DAO), que pretende principalmente independizar la aplicación de la forma de acceder a la base de datos, usando Hibernate como API de acceso.

Pasemos ahora a explicar cada uno de los métodos del patrón DAO:

T create(): Crea un nuevo objeto de la entidad.

void saveOrUpdate(T entity) : Inserta o actualiza un objeto de una entidad en la base de datos.

T get(ID id) : Obtiene un objeto de una entidad desde la base de datos en función de la clave primaria.

void delete(ID id) : Borra un objeto de una entidad de la base de datos en función de la clave primaria.

List<T> findAll() : Obtiene una lista con todos los objetos de una entidad de la base de datos.

2.4. Ingeniería de software del sistema

La Ingeniería de Software es la aplicación práctica del conocimiento científico en el diseño y construcción de software y la documentación asociada requerida para desarrollarlos, operarlos y mantenerlos.

2.4.1. Diagrama de Casos de Uso

Los casos de uso son una técnica para especificar el comportamiento de un sistema: todo sistema de software ofrece a su entorno o a aquellos que lo usan una serie de servicios. Un caso de uso es una forma de expresar cómo alguien o algo externo a un sistema lo usa. Cuando se menciona " alguien o algo" se hace referencia a que los sistemas son usados no sólo por personas, sino también por otros sistemas de *hardware* y *software*.

En la figura 2.11 se pueden observar los casos de uso del sistema para la generación automática de los horarios docentes.

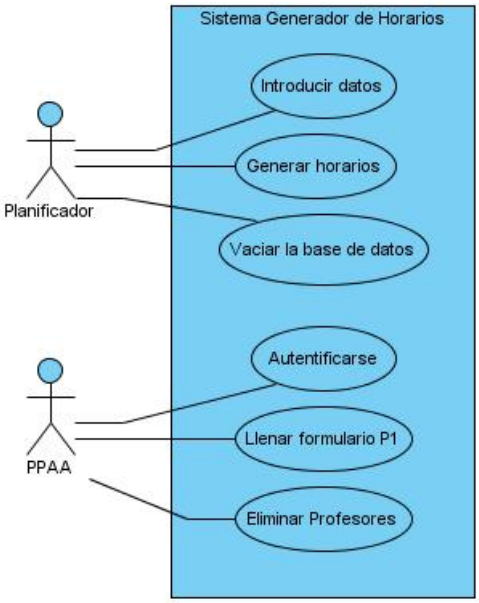


Figura 2.11: Diagrama de Casos de Uso del sistema.

2.4.2. Diagrama de Componentes

Un diagrama de componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Se utilizan para modelar la vista estática de un sistema. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Uno de los usos principales es que puede servir para ver que componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

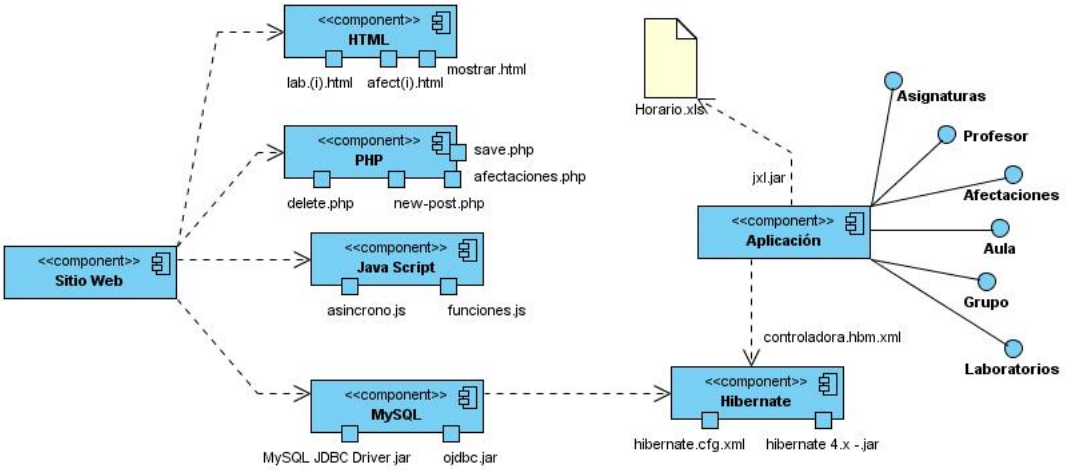


Figura 2.12: Diagrama de componentes del sistema.

En el diagrama se muestran los componentes necesarios para la realización del sistema Cronos v1.0. Como primer componente se tiene el sitio web , el cual se desarrolló a partir de lenguajes de programación, como son HTML, PHP y Java Script, se muestran también las dependencias entre estos componentes y las funciones principales usadas para su correcto funcionamiento. Como componentes intermedios se tienen MySQL, gestor de base de datos usado e Hibernate, componente que permite enlazar o conectar la base de datos con la aplicación java desarrollada a través del conector de MySQL.
Hibernate está en la capa de la aplicación que se conecta a la base de datos (la capa de persistencia), así que necesita información de la conexión. La conexión se hace a través de un pool de conexiones JDBC que también se debe configurar. Es necesario crear y configurar el archivo del *framework* hibernate.cfg.xml y crear también el archivo controladora.hbm.xml que permite mapear la base de datos, además se utilizó la biblioteca hibernate 4.x-.jar, que contiene las clases principales para el uso de Hibernate. Otro de los componentes es la propia aplicación, la cual es completamente dependiente de todos los componentes anteriores y utiliza las diferentes interfaces provistas para el tratamiento de toda la información. Finalmente, la salida de la aplicación es un archivo Excel que contiene los horarios generados para cada grupo.

2.4.3. Diagrama de Clases

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones (incluyendo herencia, agregación, asociación, etc.). Los diagramas de clase son utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido

(diseño). El diagrama de clases de más alto nivel, será lógicamente un dibujo de los paquetes que componen el sistema.

En la figura 2.13 se muestran las interrelaciones de las clases y paquetes del sistema para la generación de horarios.

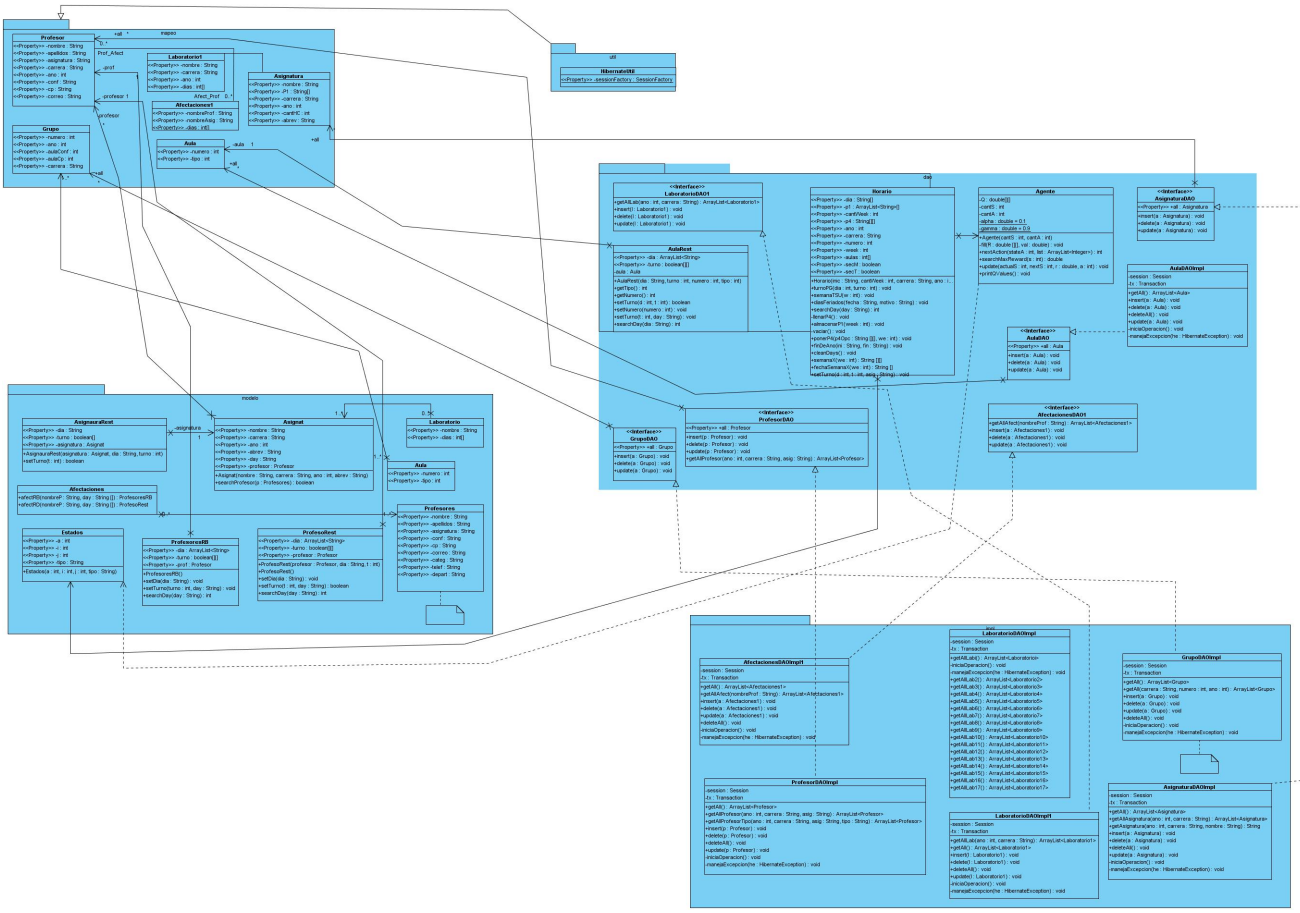


Figura 2.13: Diagrama de Clases del Sistema.

La figura muestra todas las clases que intervienen en el sistema, separadas en 5 paquetes: modelo, útil, DAO, implementación y mapeo. Las clases contenidas en el primer paquete, son las encargadas de comprobar todas las restricciones, duras y blandas, y de aplicar el algoritmo Q-Learning que arrojará horarios válidos para la facultad. En el segundo paquete existe sólo una clase, HibernateUtil, que es la que crea e inicia sesiones de Hibernate. El paquete DAO, en el que se pueden encontrar las interfaces y clases que interactúan con la base de datos, las interfaces definen todos los métodos que se proporcionarán para acceder a la base de datos, éstos incluyen insertar, eliminar, actualizar, y recuperar información. En el cuarto paquete, el de implementación, están todas las clases que son necesarias para el acceso a los datos de forma real, ya que las interfaces no implementan los métodos. Por último el paquete mapeo, que contiene las clases encargadas en el mapeo de las clases java con las entidades de la base de datos.

2.5. Conclusiones parciales

En este capítulo se describió el problema que se pretende resolver con este trabajo. También se diseñó e implementó una aplicación web que es la encargada de captar los datos de los P1 mediante formularios y almacenarlos en una base de datos. La base de datos fue diseñada en MySQL y en este capítulo se mostró el diseño de la misma. Se implementó una aplicación de escritorio que se comunica con la base de datos mediante Hibernate y se expone toda la Ingeniería de Software de la misma.

Capítulo III

Capítulo III: Algoritmo de generación automática de horarios

En este capítulo se explican con detenimiento las restricciones, tanto fuertes como suaves, que fueron tomadas en cuenta para la generación automática del horario. También se hace una descripción del algoritmo y se muestra un pequeño pseudocódigo del mismo. Más adelante se introduce un pequeño ejemplo para ilustrar su comportamiento y para finalizar se hace una validación del algoritmo.

3.1. Restricciones

Uno de los principales elementos de los problemas de generación de horarios son las restricciones. De la manera en que las mismas son tratadas depende en gran medida la calidad del horario generado. A continuación, se describen los requerimientos (restricciones) impuestos por la Facultad para la planificación de los horarios docentes. Como se mencionó en el capítulo 1, estos requerimientos fueron categorizados en dos grupos: fuertes y suaves. Los requerimientos fuertes deben ser cumplidos obligatoriamente y los requerimientos suaves, que si bien no son obligatorios, representan condiciones deseables para la Facultad y los profesores del claustro. Los requerimientos suaves se incorporaron dentro de la función objetivo utilizada en el algoritmo, la cual trata de minimizar el número de veces en que no se cumplen estos requerimientos.

Existen requerimientos generales para toda la universidad y a los cuales la facultad, como parte de ella, tiene que responder, tal es el caso de:

- Cada curso tiene una fecha de inicio y de finalización.
- Cada año docente de cada carrera, empezando en la fecha que la universidad estipule, tiene que recibir docencia un determinado número de semanas.
- A los años pares, es decir, segundo y cuarto, se les imparte clases en la sesión de la tarde, a los restantes (primero, tercero y quinto), en la sesión de la mañana.

- La primera semana del curso se asigna como semana de familiarización para los años iniciales de cada carrera, por lo que no se planifican clases para ellos.

- Algunas semanas del curso se verán afectadas por otras actividades como son: trabajo socialmente útil (TSU), práctica laboral (PL), en las cuales no se planifica docencia. Estas semanas no necesariamente coinciden para todas las carreras ni años académicos. Así, para el año X_1 de la carrera Y_1 puede estar planificado el TSU en la semana k y para el año X_2 de la carrera Y_2 puede estar planificado en la semana r .

- Pueden existir otras actividades extracurriculares en el marco de un curso, como pueden ser: carnavales, forúms científicos, actos políticos o culturales, días destinados a preparación para la defensa, reuniones, días feriados, entre otras. En el caso de que alguna de estas actividades afecte un día de clases completo, tampoco se planificará clases. Si afecta una única sesión de clases, entonces se planificará clases para el resto del día, sin incluir los turnos afectados.

La notación para estas restricciones se define a continuación:

m =cantidad de aulas.

n =cantidad de asignaturas.

p =cantidad de profesores.

ds =días de la semana.

h =horas disponibles por día.

hh =número total de horas hábiles por semanas, obtenido de $ds{*}h$

q =cantidad total de semanas de un curso. $\forall\ m, n, p, q, ds\ \epsilon\ \ \mathbb{Z}$

t =cantidad de turnos correspondientes a un día de la semana.

L =matriz binaria de dimensión $((d-c)+1)$ x ds , donde cada elemento l_{ef} se define como

$$l_{ef} = \begin{cases} 1 \text{ si el día } e \text{ el turno } f \text{ está ocupado por un laboratorio} \\ 0 \text{ en otros casos} \end{cases}$$

$$X_{ijkl} = \begin{cases} 1 \text{ si el aula } i \text{ es asignada a la asignatura } j \text{ con el profesor } k \text{ en el turno } l, \\ 0 \text{ en otros casos} \end{cases}$$

$$y_{ij} = \begin{cases} 1 \text{ si hay examen de la asignatura } i \text{ en el día } j \\ 0 \text{ en otros casos} \end{cases}$$

donde $i \epsilon \{1, ..., m\}$, $j \epsilon \{1, ..., n\}$, $k \epsilon \{1, ..., p\}$, $l \epsilon \{1, ..., q\}$, $x \epsilon \{1, ..., ds\}$, $r \epsilon \{1, ..., h\}$.

3.1.1. Fuertes

Para que el horario generado se considere factible las restricciones fuertes deben cumplirse. En el caso de nuestro problema las restricciones que fueron tratadas como fuertes fueron las siguientes:

- En un aula , en un mismo día y turno horario, se puede impartir a lo sumo una clase.

$$\sum_{j=1}^{n} \sum_{k=1}^{p} X_{ijkl} \leq 1$$

- Toda asignatura j tiene p turnos semanales.

$$\sum_{i=1}^{m} \sum_{k=1}^{p} \sum_{l=1}^{q} X_{ijkl} = p_j$$

- Toda asignatura j debe tener asignado un profesor k y un aula i en un turno.

$$\sum_{i=1}^{m} \sum_{k=1}^{p} X_{ijkl} \leq 1$$

- Todo profesor k no puede dictar más de una asignatura j a la vez, y no puede tener asignada más de un aula i en un turno l.

$$\sum_{i=1}^{m} \sum_{j=1}^{n} X_{ijkl} \leq 1$$

- Los horarios de las asignaturas de un mismo semestre para una carrera y año específico no pueden coincidir en un mismo turno.

$$\sum_{k=1}^{p} \sum_{j=1}^{n} \sum_{i=1}^{m} X_{ijkl} \leq 1$$

- Cada asignatura posee un número variable de horas asignadas durante los ds días disponibles.

$$\sum_{i=1}^{m} \sum_{k=1}^{p} \sum_{x=1}^{ds} \sum_{r=1}^{h} X_{ijkl} = Total\ Horas\ Asignatura\,(n)$$

- No puede haber más horas clases planificadas que cantidad de horas disponibles en el semestre.

$$q * hh \geq Total\ Horas\ Asignatura$$

- En un mismo día no puede haber más de un examen planificado.

$$\sum_{i=0}^{n} y_{ij} \leq 1, \quad j = 1, ..., ds$$

3.1.2. Suaves

Para generar el horario, el algoritmo propuesto tomó como restricciones suaves las preferencias personales de los profesores, así como las diferentes actividades de las cuales los profesores tienen conocimiento por ser fijas durante todo el semestre.

Las restricciones suaves serán entonces los días o turnos en los que un profesor esté indispuesto para impartir clases, por cualquier motivo, o porque en su plan de trabajo tenga afectaciones, que si bien no es de obligatoria su asistencia o participación, sería deseable su presencia. Serán restricciones de este tipo entonces:

- Que el profesor esté indispuesto a dar clases en determinado turno, por razones personales bien justificadas.
- Que el profesor tenga preparación metodológica.
- Reuniones de departamento, del Sindicato o del PCC o la UJC.

Los planes de trabajo son confeccionados mensualmente por lo que, las actividades que en él se planifiquen, excluyendo las anteriormente mencionadas, no se tendrán en cuenta en la generación del horario docente y el profesor solo podrá asistir a las mismas si no tiene clases planificadas en ese espacio de tiempo.

3.2. Algoritmo

Este epígrafe se dedica a la descripción del algoritmo. A continuación se muestra el pseudocódigo del mismo.

Algoritmo 3.1 Generar Horario

```
Entrada : P1 e informaciones generales del semestre
Salida : P4 en un archivo Excel

 Seleccionar carrera y año inicial;
   Mientras (horario no completo) hacer
     crear entorno de trabajo;
     inicializar matriz de Q − valores;
       Para cada iteración hacer
         seleccionar la acción de mayor Q − valor;
         calcular la recompensa de esa acción;
         actualizar los Q − valores según la fórmula del algoritmo Q − Learning;
       Fin Para
    Fin Mientras
```

3.2.1. Entrada y Salida

En la base de datos Horario se encuentran almacenados los P1 proporcionados por el PPA. En esta base de datos no solo se encuentra la frecuencia y tipo de clase planificadas para las asignaturas, también están recogidas las restricciones suaves que fueron captadas al igual que los datos de las asignaturas mediante la página web diseñada para este fin. Además de los P1 de todas las asignaturas que van a estar almacenados en la base de datos el algoritmo necesita una serie de datos extra. Estos datos son por ejemplo: fecha de inicio del semestre, aulas disponibles, cantidad de grupos por carrera y año, días feriados, sesión de clases, entre otros.

A continuación en la figura 3.14 se muestran una serie de ventanas que son utilizadas para captar los datos necesarios antes de generar el horario.

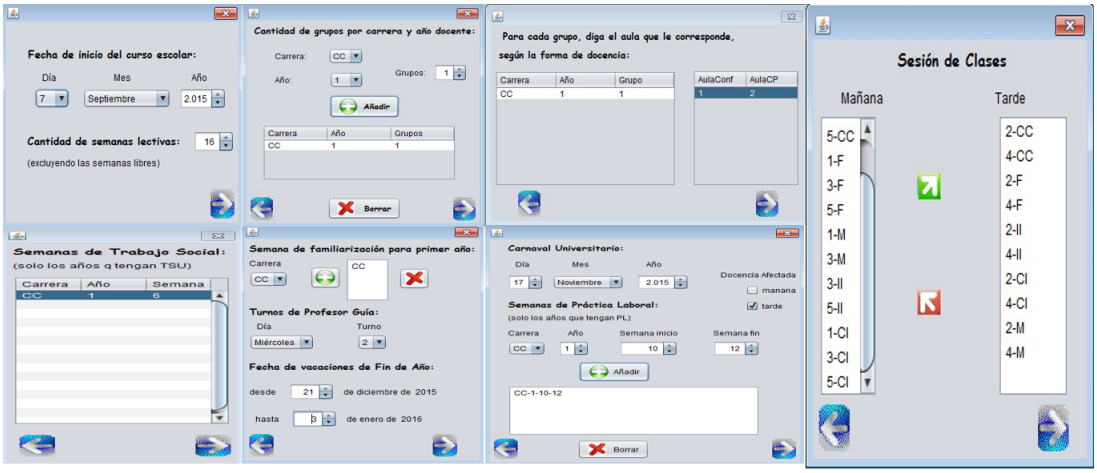
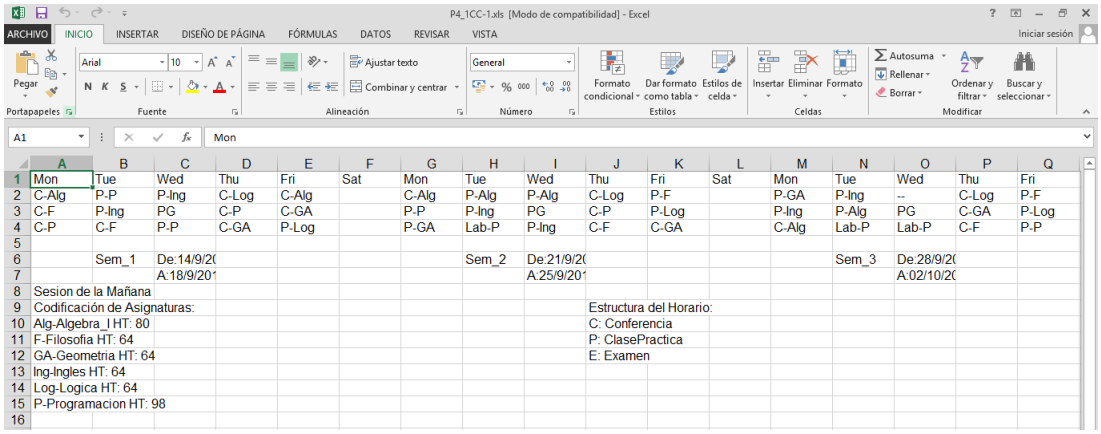


Figura 3.14: Información necesaria para generar horarios

Después de entrar los datos necesarios para ejecutar el algoritmo se genera el horario docente el cual tiene como salida un conjunto de documentos Excel con el horario de cada grupo. En la siguiente figura se muestra un fragmento del horario generado para el grupo 1 de primer año de Ciencia de la Computación.



P4_1CC-1.xls [Modo de compatibilidad] - Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Mon	Tue	Wed	Thu	Fri	Sat	Mon	Tue	Wed	Thu	Fri	Sat	Mon	Tue	Wed	Thu	Fri
2	C-Alg	P-P	P-Ing	C-Log	C-Alg		C-Alg	P-Alg	P-Alg	C-Log	P-F		P-GA	P-Ing	--	C-Log	P-F
3	C-F	P-Ing	PG	C-P	C-GA		P-P	P-Ing	PG	C-P	P-Log		P-Ing	P-Alg	PG	C-GA	P-Log
4	C-P	C-F	P-P	C-GA	P-Log		P-GA	Lab-P	P-Ing	C-F	C-GA		C-Alg	Lab-P	Lab-P	C-F	P-P
5																	
6		Sem_1	De:14/9/2(					Sem_2	De:21/9/2(					Sem_3	De:28/9/2(		
7			A:18/9/201						A:25/9/201						A:02/10/2(		
8	Sesion de la Mañana																
9	Codificación de Asignaturas:									Estructura del Horario:							
10	Alg-Algebra_I HT: 80									C: Conferencia							
11	F-Filosofia HT: 64									P: ClasePractica							
12	GA-Geometria HT: 64									E: Examen							
13	Ing-Ingles HT: 64																
14	Log-Logica HT: 64																
15	P-Programacion HT: 98																
16																	

Figura 3.15: Fragmento de P4 generado para 1ero CC-Grupo#1

3.2.2. Carrera y año inicial

Después de captar los datos que se necesitan, el algoritmo decide por qué carrera y año va a comenzar a generar el horario. Esta decisión la toma el algoritmo de manera aleatoria si todos los grupos comienzan en la misma fecha, si esto no ocurre de esa manera entonces el algoritmo empieza a generar el horario por la carrera y año que primero comience las clases. En el primer semestre se escoge por quien comenzar aleatoriamente debido a que generalmente el curso siempre empieza el mismo día para para todo el mundo. Donde hay una diferencia marcada en el inicio del semestre es en el segundo, debido a que no todas las carreras y años tienen la misma cantidad de semanas en el primer semestre, además algunos años tienen la Práctica Laboral entre los dos semestres, entre otros factores.

3.2.3. Entorno de trabajo

El entorno o ambiente de trabajo es el conjunto de todos los recursos que deben tenerse en cuenta para la generación de los horarios. El término recursos no solo es utilizado para referirse a las aulas, laboratorios y asignaturas, con él también se hace referencia a los recursos humanos, en este caso particular, los profesores.

3.2.4. Matriz de Q-valores

En el algoritmo Q-Learning la matriz de los Q-valores está representada por una relación entre un par estado-acción y un valor (q-value), donde un estado es un mapeo entre un turno específico de la semana y la lista de posibles asignaturas (acciones) a dar en ese turno. Al principio esta matriz es inicializada con cero y

a medida que el algoritmo va avanzando, después de cada iteración, se actualiza la matriz utilizando la fórmula expuesta en el capítulo 1.

3.2.5. Estrategia de selección

Para la selección de la mejor acción se utilizó una estrategia ε-greedy. Es decir, se genera un valor aleatorio ε, si $\varepsilon < 0.2$ entonces se selecciona la acción de manera aleatoria, si no se selecciona la acción de mayor q-valor.

3.2.6. Calcular recompensa

La recompensa es uno de los principales elementos del Aprendizaje Reforzado y por ende, del algoritmo Q-Learning. En este problema en particular la recompensa de seleccionar una acción a en un estado s está dada por la cantidad de restricciones blandas que se cumplen al seleccionar esa acción entre el total de restricciones blandas que deben cumplirse.

3.2.7. Actualizar Q-valores

La actualización de los Q-valores se realiza de la forma tradicional según la fórmula de actualización del algoritmo Q-Learning que se describió en el capítulo 1. Se realizaron pruebas con distintas configuraciones de parámetros, pero finalmente se utilizó como factor de descuento (γ) 0.9 y como velocidad de aprendizaje (α) 0.1 al ser con ésta con la que mejor desempeño mostró el algoritmo.

3.3. Validación

Para evaluar el desempeño del algoritmo se tuvo en cuenta en qué medida el horario generado fue capaz de cumplir con las restricciones blandas, es decir, con las diferentes preferencias personales de los profesores. Para estas pruebas se generó el horario de los cinco años de la carrera Licenciatura en Ciencia de la Computación y la cantidad de grupos se corresponde con la cantidad existente en el curso 2014-2015.

En la siguiente tabla se muestra por año la cantidad de restricciones blandas que se cumplieron además del total de restricciones blandas por año. También en la tabla se expresa por año el por ciento de restricciones que se cumplieron.

Grupo	Cumple	Total	%
1-CC	276	279	98.92
2-CC	320	324	98.77
3-CC	425	433	98.15
4-CC	305	313	97.44
5-CC	256	258	99.22

Tabla 3.2: Cumplimiento de las restricciones blandas por año.

Como puede verse el por ciento de cumplimiento de las restricciones blandas es superior al 97 % en todos los casos. El por ciento en quinto año es mayor porque es solo un grupo y además no cuentan con una cantidad excesiva de horas clase. También el por ciento de restricciones blandas se calculó para los profesores, es decir, el por ciento de las preferencias personales que se cumplieron para cada profesor. En la siguiente tabla se muestran estos resultados para el claustro de profesores de primero y tercer año.

Prof'/Sem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Cumple	Total	%
Prof1-1CC	4	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	64	65	98.46
Prof2-1CC	0	1	0	0	0	2	0	1	0	0	0	0	0	0	0	0	4	4	100
Prof3-1CC	0	0	0	1	1	2	0	0	1	1	1	2	0	2	2	2	15	15	100
Prof4-1CC	3	2	1	3	3	2	1	1	2	2	2	2	1	2	2	2	30	30	96.76
Prof5-1CC	3	2	2	3	3	3	2	2	3	3	3	5	2	5	5	5	51	51	100
Prof6-1CC	7	8	7	7	7	7	7	7	7	7	7	7	7	7	7	7	112	113	99.12
Subtotal	17	18	14	18	18	20	14	15	17	17	17	20	14	20	20	20	276	276	
———	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	——	——	——
Prof1-3CC	4	5	4	4	0	4	0	2	2	0	2	0	2	2	4	4	45	45	100
Prof2-3CC	1	1	2	0	5	2	0	4	4	0	1	0	1	0	0	5	23	23	100
Prof3-3CC	1	0	3	1	1	2	0	1	1	2	1	2	0	2	2	2	22	23	95.65
Prof4-3CC	3	2	5	3	3	2	1	2	2	2	2	2	4	2	2	2	37	38	97.37
Prof5-3CC	3	2	2	3	3	3	2	3	3	5	3	5	2	5	5	5	51	51	100
Prof6-3CC	1	1	2	0	1	0	0	2	2	1	0	1	0	3	3	2	19	19	100
Prof7-3CC	1	3	2	4	2	2	1	2	2	1	2	1	0	2	3	3	31	32	96.88
Prof8-3CC	0	2	1	3	0	4	3	1	3	1	0	1	1	4	2	4	31	32	96.88
Prof9-3CC	2	1	3	3	1	4	1	1	4	1	0	1	4	2	3	3	35	35	100
Prof10-3CC	1	3	2	3	1	2	0	0	0	3	2	3	3	4	1	1	26	26	100
Subtotal	17	20	26	24	17	25	8	20	23	16	13	16	17	26	25	31	324	320	

Tabla 3.3: Preferencias personales que se cumplieron para cada profesor de primero y tercer año.

Como puede verse el por ciento de satisfacción de las restricciones de los profesores de primero y tercer año siempre se encuentra por encima del 95 %, en 9 de los 16 casos se cumplen para un 100 %, con lo cual se puede dar fe del buen funcionamiento del algoritmo.

3.4. Conclusiones parciales

En este capítulo se clasificaron las restricciones que se tomaron en cuenta para la generación automática del horario en duras y blandas y se formularon las mismas. También se describe el algoritmo y los principales pasos del mismo son descritos con detenimiento. Por último se evaluó el desempeño del algoritmo en cuanto a la satisfacción de los profesores y de las restricciones blandas en general. Los resultados obtenidos muestran que tanto por grupo como por profesores el por ciento de satisfacción por el cumplimiento de las restrcciones se encuentra por encima del 95 %.

Conclusiones

- Se detectaron las principales desventajas y ventajas de diferentes algoritmos utilizados para la generación de horarios para de esta forma retroalimentar el algoritmo propuesto.
- Se lograron captar los P1s de las diferentes asignaturas y las preferencias personales de los profesores mediante una página web.
- Se implementó un algoritmo basado en Aprendizaje Reforzado que genera de manera automática el horario de todas las carreras de la facultad.
- El algoritmo es capaz de cumplir las restricciones blandas (preferencias personales de los profesores) a casi un 98 % como promedio, demostrando así un buen desempeño.

Recomendaciones

1. Extender el algoritmo propuesto para que sea capaz de captar los requerimientos de las asignaturas de forma más genérica.
2. Paralelizar el algoritmo propuesto para lograr la generación de los horarios con más rapidez y menor consumo de memoria virtual.
3. Integrar el algoritmo propuesto a la herramienta desarrollada por el grupo de investigación para problemas de secuenciación de tareas.

Bibliografía

Abdullah, S., 2006. Heuristic approaches for university timetabling problems. Ph.D. thesis, Citeseer.

Andrew, G. M., Collins, R., 1971. Matching faculty to courses. College and University.

Burke, E., Elliman, D., Weare, R., 1994. A genetic algorithm based university timetabling system. In: Proceedings of the 2nd East-West International Conference on Computer Technologies in Education. Vol. 1. pp. 35–40.

Carter, M. W., 1989. A lagrangian relaxation approach to the classroom assignment problem. INFOR. 27 (2), 230–246.

Carter, M. W., Laporte, G., 1998. Recent developments in practical course timetabling. In: Practice and Theory of Automated Timetabling II. Springer, pp. 3–19.

Carter, M. W., Tovey, C. A., 1992. When is the classroom assignment problem hard? Operations Research 40 (1-supplement-1), S28–S39.

Chiong Molina, M. O., 1995. Higiene de la actividad docente. La Habana: Editorial Pueblo y Educación.

Corrales, D., Pérez, C., 1976. Hacia el perfeccionamiento del trabajo de dirección de la escuela. Ciudad de La Habana. Editorial Pueblo y Educación.

De Werra, D., 1985. An introduction to timetabling. European Journal of Operational Research 19 (2), 151–162.

De Werra, D., 1996. Extensions of coloring models for scheduling purposes. European Journal of Operational Research 92 (3), 474–492.

Echevarría Cartaya, Y., 2014. Sistema de apoyo a la planificación de horarios docentes. Ph.D. thesis, UCLV.

Freuder, E. C., Wallace, R. J., 1992. Partial constraint satisfaction. Artificial Intelligence 58 (1), 21–70.

Garey, M. R., Johnson, D. S., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Mathematics of operations research 1 (2), 117–129.

Gendreau, M., Potvin, J.-Y., 2005. Tabu search. In: Search methodologies. Springer, pp. 165–186.

Glassey, C. R., Mizrach, M., 1986. A decision support system for assigning classes to rooms. Interfaces 16 (5), 92–100.

Glover, F., Laguna, M., 1999. Tabu search. Springer.

Gosselin, K., Truchon, M., 1986. Allocation of classrooms by linear programming. Journal of the Operational Research Society, 561–569.

Gunawan, A., 2009. Modeling and heuristic solutions of university timetabling problems. Ph.D. thesis.

Hooker, J. N., 1995. Testing heuristics: We have it all wrong. Journal of Heuristics 1 (1), 33–42.

Jat, S. N., Yang, S., 2009. A guided search genetic algorithm for the university course timetabling problem. In: The 4th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2009), Dublin, Ireland: 180-191, 10-12 Aug 2009.

Jiménez, Y. M., 2012. A generic multi-agent reinforcement learning approach for scheduling problems.

Kaelbling, L. P., Littman, M. L., Moore, A. W., 1996. Reinforcement learning: A survey. Journal of artificial intelligence research, 237–285.

Kendall, G., Hussin, N. M., 2005. A tabu search hyper-heuristic approach to the examination timetabling problem at the mara university of technology. In: Practice and Theory of Automated Timetabling V. Springer, pp. 270–293.

Koyuncu, B., Seçir, M., 2007. Student time table by using graph coloring algorithm. In: 5th International Conference on Electrical and Electronics Engineering–ELECO.

Matijaš, V. D., Molnar, G., Čupić, M., Jakobović, D., Bašić, B. D., 2010. University course timetabling using aco: a case study on laboratory exercises. In: Knowledge-Based and Intelligent Information and Engineering Systems. Springer, pp. 100–110.

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjunkoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers & Chemical Engineering 30 (6), 913–946.

Montero, E., Riff, M.-C., Altamirano, L., 2011. A pso algorithm to solve a real course+ exam timetabling problem. In: International Conference on Swarm Intelligence. pp. 24–1.

Müller, T., 2005. Constraint-based timetabling.

Murray, K., Müller, T., Rudová, H., 2007. Modeling and solution of a complex university course timetabling problem. In: Practice and Theory of Automated Timetabling VI. Springer, pp. 189–209.

Nandhini, M., Kanmani, S., 2009. A survey of simulated annealing methodology for university course timetabling. International Journal of Recent Trends in Engineering 1 (2), 255–257.

Osman, I. H., Kelly, J. P., 1996. Meta-heuristics: theory and applications. Springer Science & Business Media.

Piwonka, A. B., Arata, M. P., 1996. Internet en acción. McGraw-Hill.

Qu, R., 2002. Case-based reasoning for course timetabling problems. Ph.D. thesis, University of Nottingham.

Ruiz, R., Şerifoğlu, F. S., Urlings, T., 2008. Modeling realistic hybrid flexible flowshop scheduling problems. Computers & Operations Research 35 (4), 1151–1175.

Schaerf, A., 1999. A survey of automated timetabling. Artificial intelligence review 13 (2), 87–127.

Selim, S. M., 1988. Split vertices in vertex colouring and their application in developing a solution to the faculty timetable problem. The Computer Journal 31 (1), 76–82.

Shen, W., 2002. Distributed manufacturing scheduling using intelligent agents. Intelligent Systems, IEEE 17 (1), 88–94.

Socha, K., Sampels, M., Manfrin, M., 2003. Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In: Applications of evolutionary computing. Springer, pp. 334–345.

Sutton, R. S., Barto, A. G., 1998. Reinforcement learning: an introduction. Neural Networks, IEEE Transactions on 9 (5), 1054–1054.

Tamar, A., Di Castro, D., Meir, R., 2012. Integrating a partial model into model free reinforcement learning. The Journal of Machine Learning Research 13 (1), 1927–1966.

Whitley, D., Watson, J. P., 2005. Complexity theory and the no free lunch theorem. In: Search Methodologies. Springer, pp. 317–339.

Wolpert, D. H., Macready, W. G., 1997. No free lunch theorems for optimization. Evolutionary Computation, IEEE Transactions on 1 (1), 67–82.

Wren, C., 1996. Reply to subsidies, additionality and financial constraints. Journal of Regional Science 36 (3), 501–508.

Zhang, L., Lau, S. K., 2005. Constructing university timetable using constraint satisfaction programming approach. In: Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on. Vol. 2. IEEE, pp. 55–60.

Zhang, W., 1996. Reinforcement learning for job-shop scheduling. Ph.D. thesis, Oregon State University.