

Organización de horarios en Educación Primaria. Utilización de Técnicas de Inteligencia Artificial.

Juan Peiró Esteban
Trabajo de Fin de Máster
Big Data and Visual Analytics

Director: Daniel Puerto García

19 de septiembre de 2019

Índice general

1. Introducción y estado del arte	5
2. Modelo de horarios	9
2.1. 2.1. Elementos genéricos del modelo.	9
2.1.1. Índices principales.	9
2.1.2. Elementos base.	10
2.2. Reglas del modelo.	13
2.2.1. Reglas a asegurar.	13
2.2.2. Reglas de docencia.	13
2.2.3. Reglas de horario.	14
2.2.4. Reglas de control de calidad.	14
2.2.5. Reglas de docencia.	15
2.3. Modelo genérico.	15
2.3.1. Planteamiento matemático.	15
2.3.2. Ejemplo de centro de estudio.	22
3. Algoritmo: planteamiento, programación y resolución.	31
3.1. Principios generales de la programación orientada a objetos.	31
3.2. Planteamiento del algoritmo.	33
3.3. Programación del algoritmo. Aplicación de POO.	36
3.4. Resultados.	44
4. Visualización de Horarios: Tableau	47
4.1. Introducción a Tableau y objetivos generales.	48
4.2. Archivos de entrada.	50
4.3. Consideraciones generales.	51
4.4. Visualizaciones.	53

Abstract

This research's objective is to build an ensemble of tools which allows timetable organization for schools. A model of generic characteristics is settled, as it is intended to be applied for a great variety of types of schools, whilst user is provided some control on the strength of the requirements on docency. The tool plays two main roles. First of all, it obtains a suitable solution for timetable mathematic problem via a metaheuristic algorithm over a mixed integer problem programmed with CPLEX Java API Libraries. The Java part is written under Object Oriented Programming concepts to ensure correct data flow, easing new developments and future changes in the model. Finally, it visually represents the generated solution via Tableau Software. The Tableau part also checks the suitability of the solution with the requirements of the model and analyzes some aspects of the timetable detailing by teacher and group.

Resumen

El objetivo de esta investigación es construir una unión de herramientas para organizar horarios en colegios. Se establece un modelo genérico, ya que el objetivo es aplicarlo a una gran variedad de tipos de colegios, mientras que el usuario posee cierto control en la intensidad de requerimientos de docencia. La herramienta juega dos papeles principales. En primer lugar, obtiene una solución factible para el problema matemático de horarios vía un algoritmo metaheurístico sobre un problema entero mixto programado con la CPLEX API para Java. La parte en Java está escrita bajo preceptos de Programación Orientada a Objetos para asegurar el correcto flujo de datos, facilitando nuevos desarrollos y futuros cambios en el modelo. En segundo lugar, representa visualmente la solución generada vía Tableau Software. Esta parte también comprueba la corrección de la solución respecto a los requerimientos del modelo y analiza algunos aspectos del horario detallando por profesor y grupo.

Capítulo 1

Introducción y estado del arte

Comencemos con un comentario acerca del estado del arte de la resolución de horarios en centros educativos. En primer lugar, hablaremos de las características generales de los modelos tratados. Varios puntos nos resultarán de interés:

- La mayoría de los modelos propuestos en la literatura son referidos a centros de Educación Secundaria, siendo el caso de los dos artículos referentes a organización de horarios adjuntos en la bibliografía^{[1][2]}.

La principal diferencia entre estos institutos tratados generalmente y los colegios que modelaremos concretamente a lo largo de este trabajo radica principalmente en la capacitación de enseñar materias de los profesores:

- En el caso de los institutos pueden enseñar únicamente un reducido número de tipos de materias.
 - En el caso de los colegios las especializaciones son más generales y cada profesor puede enseñar una mayor variedad de materias.
- Otra diferencia relevante reside en la función de los tutores:
 - En el caso de los modelos para Educación Secundaria (instituto), los tutores generalmente imparten una única asignatura en el grupo aparte de la Tutoría, debido a la capacitación de los profesores en secundaria de enseñar un determinado tipo de asignatura. Debido a este hecho, en muchos casos los modelos no consideran tutorización de grupos. Este es el caso de los dos artículos de organización de horarios citados^{[1][2]}.
 - En el caso de los modelos para Educación Primaria (colegio) como el que desarrollaremos a lo largo de esta memoria, el tutor se suele ver asignado en su propio grupo tutorizado a lo largo de múltiples asignaturas aparte de la Tutoría, debido a la capacitación de los profesores en primaria de enseñar múltiples tipos de asignaturas.
 - Prácticamente todos los modelos de organización de horarios en centros educativos dan la posibilidad de docencia compartida, pudiendo enseñar las lecciones requeridas para una asignatura concreta en un grupo concreto entre dos profesores diferentes. Esto es una situación que a nivel de colegios en Educación Primaria no se presenta, por lo que deberemos diferenciar nuestro modelo por esa característica. De nuevo, este es el caso de los dos artículos de organización de horarios referenciados en la bibliografía^{[1][2]}.

Por lo tanto, a lo largo de este trabajo nos centraremos en desarrollar un modelo para Educación Primaria (colegios) que implique:

- Profesores con capacitaciones de docencia de múltiples asignaturas.
- Seguimiento pormenorizado de los grupos tutorizados por parte de los tutores, asignándoles un gran número de lecciones dentro de su propio grupo.

- Docencia única de cada asignatura por parte de un único profesor no compartida con otros docentes.

Para obtener un modelo con las citadas características, nos basaremos en el modelo empleado en el Trabajo de Fin de Máster adjunto en referencias^[3] del cuál es extensión o continuación el actual. Sobre este modelo llevaremos a cabo ciertos cambios y/o evoluciones para mejorar su adaptación a la situación genérica de un centro de Educación Primaria que acabamos de describir, de tal manera que el modelo sea aplicable a una variedad mayor de centros, así como para asemejar su expresión matemática en mayor medida a su posterior programación computacional. En este sentido:

- Generalizaremos el modelo eliminando las peculiaridades de sincronización entre las lecciones de Religión y Valores en un mismo hueco por grado así como los desdobles establecidos entre grupos utilizadas en el artículo de referencia^[3], de tal manera que sea aplicable a una mayor variedad de centros.
- Para asegurar el seguimiento pormenorizado de los grupos por parte de sus respectivos tutores, anteriormente^[3] empleábamos una condición referida a una cota máxima de número de lecciones fuera de su propio grupo tutorizado para cada tutor. Sin embargo, esto no garantiza que se asignen un gran número de lecciones dentro del propio grupo tutorizado, debido al hecho de que consideramos la posibilidad de que los distintos profesores se encuentren libres en los diferentes huecos. Para mejorar esta condición y dar mayor seguridad a la solución obtenida por el algoritmo, estableceremos un porcentaje mínimo objetivo de las lecciones asignadas entre todos los grupos que deben ser obligatoriamente asignadas dentro del grupo tutorizado para cada uno de los tutores considerados.
- En este sentido, para asegurarnos de que ningún profesor se encuentre en el caso citado en el anterior punto y se vea prácticamente sin lecciones asignadas y incluso libre para la totalidad del horario, estableceremos un porcentaje mínimo objetivo de las lecciones máximas disponibles semanales a ser asignadas de manera obligatoria para todo profesor.

En otro aspecto diferente, trataremos de definir el modelo matemáticamente de tal manera que pueda relacionarse más fácilmente con su programación computacional. Esto implicará principalmente definir todas las combinaciones posibles de asignaturas, grupos, lecciones o docencias para posteriormente definir en otro objeto¹. Para ilustrar esto con un caso concreto, en el caso de asignaturas plantearemos todas las materias posibles a lo largo de la totalidad de los diferentes grupos. Sin embargo, sabemos que en la práctica no todas las materias son impartidas sobre la totalidad de los cursos, por lo que deberemos definir todas las combinaciones válidas tanto a nivel matemático como de programación.

Por otra parte, hemos de recalcar que la mayoría de trabajos acerca de organización de horarios y su resolución a partir de sistemas de ecuaciones y variables de decisión se quedan en una posición teórica, definiendo el problema matemático genérico a nivel abstracto. Sin embargo, no se encuentra una amplia variedad de artículos que concreten el modelo matemático para un centro concreto con determinados profesores, grupos, etc., ni tampoco es frecuente el análisis de soluciones de horarios concretas tras la ejecución del algoritmo. Por lo tanto, a lo largo de este trabajo se hará un fuerte énfasis en ello.

Una vez reflejadas las características generales de los modelos tratados en artículos sobre este campo (como es el caso de las referencias adjuntas^{[1][2]}) y las posibles diferencias que encontraremos en el modelo de organización desarrollado a lo largo de este trabajo, será el turno de hablar del algoritmo de resolución utilizado para este. En cuanto a la resolución de los modelos tratados a través de determinado algoritmo, el estado del arte actual viene caracterizado por:

¹En el Trabajo de Fin de Máster utilizado como referencia^[3] únicamente se definían las combinaciones válidas como objetos matemáticos, lo que dificultaba su traducción a lenguaje computacional

- Mayoría de los modelos emplean condiciones de optimalidad^{[1][2]}, bien sea buscada por algoritmos determinísticos o heurísticos. No existe una variedad de modelos que traten con condiciones suficientes que permitan únicamente comprobar la factibilidad de las soluciones.
- Mayoría de los modelos consideran el problema como una única etapa global^{[1][2]} con un enorme número de variables, a lo largo de la que itera el algoritmo escogido para su resolución. No existe una variedad de modelos que dividan el problema en varias etapas.

De nuevo, extenderemos el algoritmo metaheurístico empleado del Trabajo de Fin de Máster referenciado^[3], que nos permitirá:

- Considerar condiciones suficientes con control de la intensidad de requerimientos por parte del usuario.
- Considerar problema como dos subproblemas de etapas ciertamente independientes.
- La menor complejidad matemática del modelo permite un mayor número de ejecuciones por unidad de tiempo de este. Por ello, en lugar de limitarnos a resolver un único modelo para una única pareja de parámetros introducidos por el usuario, exploraremos diversos valores de estos dos parámetros tratando de obtener una solución válida para cada una de estas intensidades de requerimiento incrementándola hasta que el usuario se encuentre satisfecho.

En otro punto, la mayoría de los trabajos se centran en la descripción teórica de un algoritmo de resolución y en el análisis de los resultados obtenidos, pero no resulta frecuente una explicación de cómo programarlo de manera práctica. Por ello, tanto comentaremos el código en un apartado de la memoria como adjuntaremos el código en anexos al Trabajo de Fin de Máster.

Llegados a este punto, tendremos que plantearnos la manera práctica de programar el algoritmo. Para ello, evolucionaremos la programación llevada a cabo a través de las librerías de Cplex para Java en el anterior Trabajo de Fin de Máster^[3]. Previamente al desarrollo de este Trabajo de Fin de Máster, la programación en Java resultaba ciertamente precaria, presentando los siguientes problemas:

- Indefinición de clases para elementos importantes del código. Únicamente se encuentran implementadas para ciertos aspectos del modelo y su posterior resolución en Cplex.
- Falta de modularidad en las funciones. Se encuentran todas «apiladas» en archivos de tipo «.java» sin dividir correctamente por paquetes de optimización, y sin relacionar las clases con sus métodos correspondientes.
- Acceso directo a los elementos en Java sin pasar por métodos de tipo «getter/setter».

La solución a esto pasa por aplicar los principios de Programación Orientada a Objetos, según los preceptos definidos en el artículo adjunto^[4]. Nos basaremos principalmente en:

- Definición de clases para todos los elementos implicados en el modelo matemático y su resolución. Emplearemos atributos privados, métodos «builders», «getters» y «setters» para acceder a ellos con propiedad e integraremos los métodos relacionados con cada clase dentro de esta misma (especialmente los métodos de lectura de información de entrada del problema).
- Abstracción de métodos de escritura, para que no necesitemos instanciar un objeto de su clase para generar la salida de texto. La escritura de la solución global del problema implicará el paso como argumento de elementos de ambas etapas del subproblema. Por esta razón, se decide no definir las funciones de escritura dentro de ninguna clase relacionada con las soluciones de docencia o lección al problema creado en Cplex, sino definir las como un método abstracto dentro de una clase con este propósito concreto. De tal forma, no necesitaremos instanciar ningún objeto de esta clase concreta de escritura, sino que podremos acceder directamente a los métodos de escritura simplemente a través del identificador de la clase.

- Acceso de elementos a través de métodos del tipo «*getter/setter*» para asegurar el correcto acceso a los objetos.

Por otra parte, también tendremos que poner especial atención en el archivo de salida del proyecto en Java, debido a que esta será la información de entrada para el siguiente paso de análisis visual de los horarios a través de Tableau. En este sentido, nos aseguraremos de escribir una salida de texto que nos permita caracterizar el horario del centro tanto desde el punto de vista de los diferentes profesores como desde el punto de vista de los diversos grupos. En ese sentido, nos aseguraremos de que la salida se encuentre agregada al máximo nivel de detalle, mostrando todas las lecciones establecidas incluyendo huecos no asignados, libres o no disponibles, para los diversos profesores. De esta manera, podremos analizar el horario de los profesores a partir del archivo de salida concreto, mientras que al excluir esas asignaciones libres o no disponibles de los profesores podremos observar el horario desde el punto de vista de los grupos.

Por último, hablaremos de la última parte que afrontaremos a lo largo de la memoria de este Trabajo de Fin de Máster, la creación de una visualización interactiva con el usuario que permita el análisis de las soluciones directamente generadas con las librerías de Cplex para Java. Llevaremos a cabo esta visualización a través del software de Tableau, en concreto a través de la versión Tableau Desktop. Nuestro objetivo es crear una herramienta en la que simplemente sustituyendo en el libro de Tableau la solución empleada anteriormente por la nueva solución generada a través de Cplex y Java podamos analizar este nuevo horario generado, de tal manera que no tengamos que volver a construir una y otra vez el mismo libro cada vez que obtengamos una nueva solución. En líneas generales, crearemos una visualización que se encargue de:

- Representar el horario generado para cualquier profesor o grupo.
- Representar el horario generado para cualquier tutor y relacionarlo con el horario de su grupo tutorizado.
- Representar el horario generado para cualquier grupo y relacionarlo con el horario de su tutor.
- Analizar en profundidad el horario de cualquier profesor, caracterizando sus horas asignadas por diversos aspectos.
- Analizar en profundidad el horario de cualquier grupo, caracterizando sus horas asignadas por diversos aspectos.
- Asegurar la corrección de asignaciones de docencias y lecciones, así como el cumplimiento de las reglas que requiere un modelo de organización de horarios.

Capítulo 2

Modelo de horarios

Esta sección 2.1 de la memoria del Trabajo de Fin de Máster tiene como objetivo explicar la lógica del modelo de organización de horarios de colegio empleada, así como establecer el modelo matemático, antes de plantearnos tanto su programación y solución como el análisis de los resultados obtenidos. Para ello, definiremos los diversos elementos que entran en juego en la construcción de una estructura de asignaciones de profesores a determinadas asignaturas en grupos concretos de ese colegio durante los huecos del horario semanal definidos. También será necesario definir la serie de reglas que han de cumplir para que el horario generado no resulte incorrecto y por lo tanto no sea aplicable para regir de manera real la actividad docente del colegio, distinguiendo entre aquellas reglas obligatorias y aquellas destinadas al control de calidad de las soluciones. Por último, también plantearemos el modelo matemático a resolver posteriormente a través de la aplicación en Java con la API de CPLEX.

2.1. 2.1. Elementos genéricos del modelo.

Hablemos en primer lugar de los principales agentes que entran en juego a la hora de establecer una docencia organizada en un centro de estudios a nivel semanal. Este asunto será tratado a lo largo de la subsección 2.1.1. En líneas generales, organizar un horario para un colegio implica establecer una serie de asignaciones entre profesores y grupos en huecos concretos del horario para impartir determinadas lecciones de una materia concreta que se repitan a lo largo de las diferentes semanas de las que consta un curso lectivo. Por lo tanto, esta semana será el periodo básico que deberemos organizar, asegurando que se cumplan una serie de reglas para que el horario sea factible y pueda ser puesto en práctica de esta forma. Esta segunda parte será posteriormente abordada en la sección 2.2.

2.1.1. Índices principales.

Definamos las dimensiones principales del problema, que nos darán una idea de las dimensiones del problema aplicado al centro concreto. Deberemos exponer cualquier elemento que pueda entrar en juego en el establecimiento de una lección, considerada como la asignación de un profesor a la enseñanza de una materia concreta para un conjunto de alumnos caracterizado por un grado y letra determinados durante un día y sesión dados. En este sentido, deberemos de caracterizar:

- Profesores: elementos que son capaces de impartir lecciones, conformado por la lista de personal del centro en cuestión.
- Materia: campos concretos del conocimiento concreta impartidos en el centro a lo largo de los diversos diversos grados en los que se organiza la vida estudiantil. Ejemplos de materias podrían ser Lengua o Matemáticas, sin concretar sobre qué grado o letra son enseñados.
- Grado: niveles de enseñanza en los que se dividen las distintas etapas de la vida estudiantil, por los que los alumnos van avanzando curso estudiantil tras curso estudiantil una vez han superado las anteriores etapas.

- Letra: subdivisiones de cada uno de los niveles de enseñanza a los que corresponden los cursos, dividiendo al total de los alumnos en un determinado curso en diferentes clases representadas con una letra determinada. Este nivel de agrupación de los alumnos será el encargado de recibir las lecciones.
- Día: marca el puesto de una lección concreta a lo largo de los diferentes días lectivos de la semana.
- Sesión: marca el puesto de una lección concreta dentro de un día concreto de la semana, en función de si se lleva a cabo en una hora más temprana o más tardía.

Observamos por lo tanto cómo el número principal de dimensiones del problema es 6.

Por último y para finalizar esta subsección, comentaremos el hecho de que deberemos conocer el número total de cada uno de estos elementos para poder plantear el modelo. De cara tanto a la descripción teórica del modelo matemático de este como a la posterior programación y resolución de los horarios, estableceremos un alias para cada uno de los elementos mencionados anteriormente. Estos alias nos servirán posteriormente para interpretar a través de Tableau la solución generada con la API de Cplex para Java, ya que para este optimizador todas las variables que le introduzcamos son *mudas*, valores numéricos con una serie de restricciones bajo una lista de condiciones que definiremos posteriormente.

2.1.2. Elementos base.

En la anterior subsección 2.1.2 definimos los 6 índices base a recorrer en este problema: profesores, materias, grados, letras, días y sesiones. Ahora bien, nos interesará agrupar estos índices en diversas estructuras compuestas para facilitar el posterior planteamiento matemático en la sección 2.3 así como su programación, resolución y análisis en futuros capítulos. Los elementos base del modelo a definir en función de estos son:

- Hueco: lista de espacios en el horario en los que se puede asignar una determinada lección. Estará dividido en:
 - Día: primera subdivisión del hueco, indicando en qué día lectivo de la semana se ubica el hueco.
 - Sesión: segunda subdivisión del hueco, marcando la posición dentro de un día concreto de la lección. En principio y para nuestro modelo genérico, todos los días dispondrán del mismo número de sesiones¹, y cada sesión tendrá una duración constante, marcando la duración uniforme de todos los huecos².
- Grupo: lista de conjuntos de alumnos en las que el centro divide a sus estudiantes, recibiendo las lecciones requeridas. Cada uno de los grupos vendrá caracterizado por:
 - Grado: nivel educativo en el que se encuadra el grupo.
 - Letra: subdivisión del nivel educativo a la que corresponde el grupo de alumnos concreto en ese curso. No todos los niveles educativos o grados tienen el mismo número de alumnos dentro de un centro, por lo que el número de letras en las que se subdivide cada grado podrá variar. De esta forma, a la hora de definir los grupos a partir de las combinaciones de grados y letras definidas como índices principales, deberemos caracterizar qué letras dentro de cada grado se encuentran permitidas para conformar los diferentes grupos.

¹Si decidiésemos modelar un planteamiento que considere diferente número de sesiones por día, podríamos realizar una sencilla evolución del modelo marcando qué combinaciones de día y sesión marcan un hueco factible y cuáles no.

²De nuevo esto podría modelarse con una ligera evolución del modelo actual, que presenta un coeficiente de asignación de una lección a un hueco con valor 1 en este modelo. En casos de duración variable del hueco, el coeficiente de la asignación de lección en hueco podría adaptarse en función de la mayor o menor duración del hueco

- Tutor: profesor encargado del seguimiento de un grupo de alumnos concreto. Se encarga de la impartición de la asignatura de Tutoría dentro del propio grupo, y es asignado un porcentaje elevado de sus horas disponibles dentro de este propio grupo, todo ello encaminado al seguimiento pormenorizado de los diferentes alumnos del grupo a la hora de afrontar las evaluaciones.

No permitiremos huecos en los horarios de los grupos, por lo que cada uno de estos deberá tener en cada hueco asignada una lección a una asignatura concreta con un profesor determinado.

- Profesores: serán los elementos encargados de impartir las lecciones. Cada profesor de la lista de índices anteriormente definida deberá ser caracterizado a través de diferentes elementos:
 - Disponibilidad máxima semanal: número que marca el límite máximo de lecciones que es capaz de impartir un profesor durante una semana.
 - Grupo tutorizado: determinados profesores se encargan de la tutorización de cierto grupo, teniendo que impartir su asignatura de Tutoría y debiendo impartir un número elevado de lecciones dentro de este grupo para llevar un seguimiento pormenorizado y poder participar de manera más activa en la evaluación de los alumnos de ese grupo. Deberemos por lo tanto conocer para cada profesor si es tutor, y en caso afirmativo, el grupo al que tutoriza.
 - Disponibilidad para vigilar recreos: en función del carácter de su contrato (media jornada, jornada completa, contrato de personal del equipo directivo...) ciertos profesores se encargarán de la vigilancia de periodos de recreos de los alumnos. Por ello, para cada profesor deberemos conocer si se encuentra disponible para vigilar recreos, ya que en caso afirmativo, deberemos descontar un determinado número de lecciones de la disponibilidad semanal destinada a lecciones para reservarlos directamente como periodos de recreos vigilados.
 - Cualificación para impartir materias: en función de la especialización de cada uno de los profesores, estos serán capaces de impartir únicamente determinadas materias de la lista total disponible. Por lo tanto, para cada profesor deberá caracterizarse si la docencia de cada una de las diferentes materias se encuentra permitida. Cada profesor podrá enseñar una serie de materias en función de su especialidad.
 - Disponibilidad en los huecos del horario: en función del tipo de contrato del docente anteriormente comentado, este se encontrará disponible para impartir lecciones solo en determinados huecos. Por lo tanto, para cada profesor deberemos caracterizar su disponibilidad en cada uno de los huecos que conforman el horario semanal.

Durante un hueco concreto, un profesor podrá tener tres tipos de asignaciones diferentes: a una asignatura concreta (implicando una materia enseñada dentro de un grupo permitido), libre y no disponible.

- Asignaturas: unidad de docencia básica, consistiendo en la concreción de una materia determinada para un determinado grupo permitido.
 - Materia: tipo de conocimiento impartido a lo largo de las lecciones de la asignatura, dentro de la lista de materias disponibles definidas en los índices principales.
 - Grado: nivel educativo del grado al que corresponde el grupo al que se asigna la asignatura.
 - Letra: identificador del grupo concreto dentro del nivel educativo al que se le asigna la asignatura determinada. Como hemos comentado anteriormente, no todos los grupos se encontrarán permitidos, por lo que las combinaciones de grados y letras deberán dar lugar. De esta forma, evitaremos asignar materias a grupos inexistentes.
 - Necesidad de docencia: no todas las materias son impartidas en cada uno de los grupos, por lo que para cada combinación de materia y grupo (grado y letra permitidos) deberemos de describir si la asignatura tiene lugar realmente.

- Necesidad semanal de lecciones: número que marca el número de lecciones que requiere cada asignatura en el periodo de una semana que estamos organizando.
- Límite mínimo y máximo diario de lecciones: número mínimo y máximo de lecciones que pueden asignarse a esa asignatura determinada en cada día concreto de los que se organiza en el horario. El número máximo y mínimo de lecciones diarias de cada asignatura se define de tal manera que las lecciones de esta se esparzan lo máximo a lo largo de los diversos días del horario, posibilitando que dos o más lecciones de una misma asignatura no coincidan en una mismo día si resulta posible. De esta forma:
 - Asignaturas con un número de lecciones requeridas semanalmente menor que el número de días disponibles. Serán asignadas un número entre 0 y 1 lecciones diarias, de tal manera que no existirá ningún día con dos lecciones asignadas para esa asignatura.
 - Asignaturas con un número de lecciones requeridas semanalmente exactamente igual al número de días disponibles. Será asignada exactamente 1 lección diaria, estableciendo tanto el mínimo como el máximo a un valor de 1. De esta manera, todos los días tendrán exactamente 1 lección diaria asignada, de tal manera que no existirá ningún día con dos lecciones asignadas.
 - Asignaturas con un número de lecciones requeridas semanalmente mayor que el número de días disponibles. Será asignado un mínimo de 1 lección diaria, para que en todos los días exista al menos una lección de la asignatura y se alivie la concentración de estas lecciones en los días que deban repetir lecciones de la determinada asignatura. El número máximo de lecciones dependerá de la relación entre el número de lecciones requeridas semanales y el número de días existentes. Si el número requerido de lecciones semanal no llega al doble del número de días, estableceremos un número máximo de dos lecciones diarias para la asignatura, de tal manera que no se repita una asignatura más de dos veces dentro de un mismo día para cada grupo. En otro caso, si el número requerido de lecciones semanales duplica el número de días pero no llega a triplicarlo, asignaríamos un número máximo de lecciones diarias de 3, y así sucesivamente.

Al no permitir huecos en los horarios de los diferentes grupos, deberemos asegurarnos que el total de lecciones requeridas por todas las materias que requieren docencia en un grupo concreto, o visto de otra forma, las materias que requieren docencia, sumen un total de lecciones igual al número de huecos disponibles en el horario semanal.

Los elementos compuestos definidos en esta subsección 2.1.2 también presentarán un alias, que vendrá dado como la combinación de los índices primarios definidos en la subsección 2.1.2. De esta forma, identificaremos los diferentes elementos compuestos como:

- Profesores: directamente por su índice de profesor.
- Grupos: combinación de un índice de grado y otro de letra, siempre y cuando den lugar a un grupo permitido.
- Asignaturas: combinación de una materia, un grado y una letra. Además, esta combinación de grado y letra ha de dar lugar a un grupo factible, y la combinación de materia y grupo ha de representar una docencia necesaria.
- Huecos: combinación de un día y una sesión.

Introduciremos por último en este final de la sección 2.1 el concepto de docencia y horario.

- Docencia: asignación de un profesor a una asignatura concreta. Deberemos de vigilar diversos aspectos como son la capacidad del profesor de impartir la materia de la que trata la asignatura, el carácter permitido o no del grupo y el carácter requerido o no de la materia para ese grupo. Consideraremos una docencia como la asignación de un profesor al completo de requerimientos semanales de una asignatura.

- **Horario:** asignación de un profesor en un hueco concreto para una asignatura determinada. Debemos de vigilar diversos aspectos, añadiendo vigilar la disponibilidad del profesor en el hueco determinado a las condiciones de capacidad de enseñanza, existencia del grupo y requerimiento de la materia enunciados en el anterior punto.

Estos dos conceptos marcarán una categorización de las reglas del modelo en dos grupos en la sección 2.2, lo que a la hora de plantear el modelo concreto en la sección 2.3 tendrá tremendas implicaciones. Marcará las etapas en las que dividiremos el problema global, quedando dividido en dos subproblemas, respectivamente de docencia y de horarios. De esta manera, podremos pasar a definir una lección como la asignación de un profesor a una asignatura concreta en un hueco determinado del horario semanal, reduciendo sensiblemente la complejidad de la definición establecida en la anterior subsección 2.1.1. Todas las lecciones tendrán la misma duración, debido al hecho de que todos los huecos presentan una duración uniforme.

2.2. Reglas del modelo.

Una vez hemos definido las entidades fundamentales, siendo estas profesores, grupos, asignaturas y huecos, deberemos desgranar las condiciones y relaciones que han de establecerse para poder asignar correctamente las lecciones y de esta manera dar lugar a un modelo que efectivamente recoja los requerimientos generales de un centro de Educación Primaria típico. Dividiremos estos requerimientos en dos tipos en función de su carácter: reglas a asegurar y control de la calidad de las soluciones.

2.2.1. Reglas a asegurar.

En esta primera subsección pasaremos a describir las reglas de obligado cumplimiento para que el horario pueda ser puesto en funcionamiento por el centro sin problemas. Un aspecto fundamental de la docencia en colegios de Educación Primaria es que las asignaturas se enseñan en bloque por un único profesor, asignando las lecciones de esa materia a lo largo de los huecos del horario a un único docente. Esto marcará profundamente el carácter del modelo. Por lo tanto y tal como hemos introducido en el inicio de la sección 2.1, clasificaremos las restricciones en función de si afectan a la asignación entre profesores y asignaturas, caso de las reglas de docencia, o aquellas que rigen la asignación de lecciones concretas entre profesores y asignaturas en huecos determinados, caso de las reglas de horario.

2.2.2. Reglas de docencia.

Nos encargaremos de asegurar que:

- Ningún profesor supera su disponibilidad semanal máxima, asignándosele la docencia de una serie de asignaturas cuyo número total de lecciones semanales requeridas resulte menor o igual que el límite máximo establecido para ese profesor.
- Todas las asignaturas asignan en bloque su docencia a un único profesor, de manera que no quede ninguna asignatura sin asignar su docencia ni ninguna asignatura con dos profesores impartiendo docencia en ella ³.
- Todos los grupos asignan la docencia de su asignatura de Tutoría a su tutor.

³Esto nos llevará a la hora de definir el modelo matemático en la 2.3.1 a definir no solamente variables de asignación de lecciones concretas sino también otras que se encarguen de modelar la asignación de docencias de asignaturas concretas.

2.2.3. Reglas de horario.

Nos encargaremos de asegurar que:

- Todas las asignaturas reciben un número de lecciones cada semana entre todos los profesores posibles y en todos los huecos del horario exactamente igual al número de lecciones semanales requeridas para la asignatura.
- Todos los grupos tienen exactamente una lección asignada por hueco entre todas las asignaturas posibles, de tal manera que no existan huecos libres ni duplicados para los grupos que impidan la puesta en práctica del horario al introducir duplicidades en el horario de los grupos.
- Todos los profesores tienen como máximo una lección asignada por hueco entre todas las asignaturas posibles, siendo el caso de hueco asignado para una única lección asignada y el de hueco libre para ninguna lección asignada. De esta manera, se evitan duplicidades en el horario del profesor que impidan la puesta en práctica del horario.
- Todas las asignaturas reciben un número de diario de lecciones comprendido entre los límites mínimo y máximo definidos para esa asignatura.

2.2.4. Reglas de control de calidad.

El conjunto de las reglas correspondientes a la subsección 2.2.1 garantizan un horario factible que pueda ser puesto en práctica en el centro de Educación Primaria en cuestión. Ahora bien, si solo tuviésemos en cuenta esa serie de reglas a asegurar, podríamos obtener horarios con particularidades tan notorias que los llevarían a ser infactibles en la práctica del centro educativo aunque cumplan las reglas básicas. Esto se centra en dos aspectos.

En primer lugar, pueden existir soluciones en las que el grueso de las lecciones se asigne a una serie de profesores sin que superen su disponibilidad mínima semanal de tal manera que el resto de profesores queden prácticamente sin asignar o incluso en algunos casos directamente sin docencia asignada. Como podemos imaginarnos, todos los docentes que compartan una misma disponibilidad semanal máxima han de impartir un número similar de lecciones cada semana⁴.

En segundo lugar, deberemos de tener en cuenta que hasta el momento la única función definida para el tutor de un grupo es la asignación de la asignatura de Tutoría, por lo que podríamos obtener soluciones en las que un tutor impartiese un pequeño número de lecciones en su grupo tutorizado o incluso que directamente solo tenga asignada la lección de Tutoría semanal con su grupo. Esto, aunque cumple las reglas básicas del modelo general, provoca que las soluciones obtenidas no puedan ser puestas en práctica, ya que de esta manera los tutores no podrían llevar a cabo un seguimiento cercano de su grupo tutorizado tal y como se ha definido en sus funciones.

Por ello, trataremos de desarrollar una serie de reglas que modelen estos dos aspectos. Tanto el límite mínimo de lecciones que deseemos asignar semanalmente a cada profesor como el límite mínimo de estas en su grupo tutorizado en el caso concreto de los tutores son valores que pueden variar enormemente de modelo a modelo, e incluso dentro de un mismo modelo establecido para un centro concreto de año a año, en función del número de profesores disponibles en plantilla. Además, en función de la manera de regirse internamente, al colegio le puede interesar mayor uniformidad en el número de lecciones asignadas o puede resultarle más indiferente debido a que el centro emplea a sus docentes en otras funciones que puedan llevarse a cabo en cualquier hueco, como la elaboración de material docente o la organización de actividades complementarias. De igual forma, en función del centro también podrá interesarles llevar un seguimiento de los grupos más centrado en el tutor de estos, teniendo este el papel

⁴Este número no ha de ser exactamente igual, ya que la diferencia de huecos libres es asignada a los profesores para que estén de guardia en ese hueco. De esta manera, en el caso de que algún profesor falte a clase en un hueco determinado, el profesor que se encuentre libre podrá hacerse cargo de la guardia durante la duración de la clase.

principal en la evaluación, o más repartido entre diversos profesores, aumentando el número de opiniones diferentes a la hora de evaluar globalmente a un alumno.

De esta manera, definiremos dos reglas de control de calidad. Según lo expuesto en el párrafo anterior, ambas afectan a la asignación entre profesores, grupos y asignaturas, sin llegar a requerir en ningún momento aspectos que afecten a los huecos del horario. Por ello, únicamente definiremos reglas de calidad dentro del apartado de docencia.

2.2.5. Reglas de docencia.

Nos encargaremos de asegurar que:

- Todos los profesores son asignados un número de lecciones mayor o igual a un porcentaje mínimo de su disponibilidad máxima semanal entre todas las asignaturas y huecos factibles. De esta forma, definiremos el ratio de asignación objetivo como el mínimo porcentaje de lecciones realmente asignadas para cada profesor respecto a su disponibilidad total semanal que nos permite considerar al profesor como parte de una solución válida.
- Todos los tutores son asignados dentro de su propio grupo tutorizado un número de lecciones mayor o igual a un porcentaje mínimo de su número de lecciones realmente asignadas entre todos los grupos existentes. De esta forma, definiremos el ratio de tutorización objetivo como el mínimo porcentaje de lecciones asignadas dentro del grupo tutorizado respecto al número de lecciones asignadas entre todos los grupos que nos permite considerar al profesor como parte de una solución válida.

Por el carácter concreto de estos requerimientos en cada centro expresado al inicio de esta subsección 2.2.4, el control de la intensidad de requerimiento de ambas reglas será dejado a la elección del usuario. De esta manera, la posterior exploración de diferentes modelos con porcentaje mínimo de disponibilidad asignada y porcentaje de asignación dentro del grupo variables darán lugar a un algoritmo de búsqueda de soluciones óptimas que permita caracterizar qué límite máximo de distribución de las lecciones entre profesores y de asignación de los tutores en sus grupos tutorizados es capaz de soportar el centro.

2.3. Modelo genérico.

Pasaremos en esta sección 2.3 a comentar el modelo matemático genérico, definiendo sus constantes definitorias del centro concreto, variables a definir y restricciones asociadas al problema de programación entera. Nos encargaremos de reflejar todos los aspectos definidos en las secciones 2.1 y 2.2 traduciendo a lenguaje matemático. Para hacer esta exposición más tangible, posteriormente definiremos los objetos matemáticos o constantes definitorias del modelo, para entender cómo puede concretarse ese modelo genérico a un centro concreto.

2.3.1. Planteamiento matemático.

Comenzaremos con la definición del planteamiento matemático en esta sección 2.3.1.

Objetos de constantes.

En primer lugar, a lo largo de esta subsección 2.3.1 definiremos las constantes matemáticas necesarias para caracterizar un centro genérico.

Comenzaremos con los objetos básicos del modelo, correspondientes a los índices fundamentales definidos en la subsección 2.1.1. Definiremos un número entero para cada objeto concreto representando su índice, de tal manera que el objeto completo el conjunto de estos índices:

- Profesores: sea el conjunto de profesores $P \subset \mathbb{Z}$. Definiremos un profesor como un índice $p \in \mathbb{Z}$ perteneciente al citado conjunto, $p \in P$.
- Materias: sea el conjunto de materias $M \subset \mathbb{Z}$. Definiremos una materia como un índice $m \in \mathbb{Z}$ perteneciente al citado conjunto, $m \in M$.
- Grados: sea el conjunto de grados $G \subset \mathbb{Z}$. Definiremos un grado como un índice $g \in \mathbb{Z}$ perteneciente al citado conjunto, $g \in G$.
- Letras: sea el conjunto de letras $L \subset \mathbb{Z}$. Definiremos una letra como un índice $l \in \mathbb{Z}$ perteneciente al citado conjunto, $l \in L$.
- Días: sea el conjunto de días $D \subset \mathbb{Z}$. Definiremos un día como un índice $d \in \mathbb{Z}$ perteneciente al citado conjunto, $d \in D$.
- Sesiones: sea el conjunto de sesiones $S \subset \mathbb{Z}$. Definiremos un profesor como un índice $s \in \mathbb{Z}$ perteneciente al citado conjunto, $s \in S$.

Además, tanto a la hora de concretar el modelo para un centro determinado como para programarlo estableceremos un alias para cada uno de estos elementos de tal manera que podamos identificarlo en el planteamiento y resolución del problema. Hablaremos de esto en la siguiente subsección 2.3.1.

Una vez definidas las estructuras básicas de modelo, podremos pasar a definir los objetos compuestos previamente caracterizados en la subsección 2.1.2. Definiremos tanto las combinaciones de índices válidas que dan lugar a estos objetos como sus constantes auxiliares, reflejando de esta manera el centro concreto.

- Huecos: como todos los días presentan la misma estructura de número de sesiones, podremos definir directamente un conjunto de dupla de índices en función de estos dos objetos previamente definidos, siendo este:

$$H = \{(d, s) \mid d \in D \cap s \in S\}$$

De esta manera, podremos identificar un hueco del horario como la dupla $h = (d, s) \in H$.

- Grupos: en principio, no todos los grados presentan la misma estructura de número de letras, pudiendo variar el número de estas para dividir el mayor o menor número de alumnos de ese nivel educativo en ese grado entre un número mayor o menor de grupos con diferentes letras. En resumen, en primer lugar definiremos un objeto que represente el total de las combinaciones para después representar cuáles son válidas y recoger únicamente esos grupos permitidos finalmente en un objeto.

- Combinaciones completas: sea el conjunto GR' de duplas de índices de los objetos previamente definidos para grados y letras, definido como:

$$GR' = \{(g, l) \mid g \in G \cap l \in L\}$$

Podremos representar un candidato a grupo válido como un elemento de este conjunto, $(g, l) \in GR'$. De esta forma, definiremos un array bidimensional de números binarios subindicado en G y L respectivamente, denominado EGR . Podremos identificar si una combinación de grado y letra $(g, l) \in GR'$ es válida en función del valor de su elemento asociado $egr_{g,l} \in \{0, 1\}$ perteneciente al objeto de validez de grupos, $egr_{g,l} \in EGR$. El grupo $(g, l) \in GR'$ será válido si su elemento asociado $egr_{g,l} = 1$, mientras que no será válido si $egr_{g,l} = 0$.

- Combinaciones de grupos permitidos: definiremos un subconjunto del total de combinaciones de grados y letras del anterior punto que recoja únicamente los grupos permitidos como:

$$GR = \{(g, l) \mid g \in G \cap l \in L \cap egr_{g,l} = 1\} \subset GR'$$

De esta manera, podremos definir un grupo válido como una dupla $(g, l) \in GR$.

- Tutores de grupo: una vez definidas las combinaciones de grados y letras que dan lugar a grupos válidos, podremos pasar a definir sus tutores asociados. Para ello, definiremos un vector de índices subindicado GR , de nombre TGR . De esta manera, el tutor de un grupo $(g, l) \in GR$ podrá definirse como $tgr_{g,l} \in TGR$. Para que el índice del profesor sea un tutor correcto deberá de cumplirse que pertenezca al conjunto de constantes de tutores, $tgr_{g,l} \in T$.
- Asignaturas: en principio, no todas las materias son enseñadas en todos los grupos. En resumen, en primer lugar describiremos todas las combinaciones posibles entre materias y grupos, para posteriormente definir únicamente las combinaciones requeridas.
 - Combinaciones completas: sea el conjunto SB' de tuplas de tres elementos de los objetos previamente definidos para materias y grupos, definido como:

$$SB' = \{(m, g, l) \mid m \in M \cap (g, l) \in GR\}$$

Podremos representar una asignatura a ser candidata como una tupla de tres índices $(m, g, l) \in SB'$ perteneciente a este conjunto. De esta forma, definiremos una matriz de números binarios subindicada respectivamente en los conjuntos de profesores y grupos válidos, M y GR , llamado ESB . Una asignatura $(m, g, l) \in SB'$ de la lista de posibles estará requerida en función del valor de su objeto asociado $esb_{m,g,l} \in \{0, 1\}$ perteneciente al objeto de requerimiento de asignaturas, $esb_{m,g,l} \in ESB$. Una asignatura será requerida si su constante asociada $esb_{m,g,l} = 1$, mientras que no será requerida en caso de que $esb_{m,g,l} = 0$.

- Combinaciones de asignaturas permitidas: definiremos un subconjunto del total de combinaciones de materias y grupos del anterior punto de nombre SB que recoja únicamente las asignaturas requeridas como:

$$SB = \{(m, g, l) \mid m \in M \cap (g, l) \in GR \cap esb_{m,g,l} = 1\} \subset SB'$$

De esta manera, podremos definir una asignatura válida como una tupla de tres elementos perteneciente al correspondiente objeto, $(m, g, l) \in SB$. Observamos como no hemos tenido que expresar la condición de validez de la dupla de grado y letra, (g, l) , debido a que nos hemos asegurado de que pertenezca al conjunto de grupos válidos.

- Requerimientos semanales y diarios: cada asignatura tiene que tener su requerimiento en lecciones tanto a nivel semanal como a nivel de cotas máximas y mínimas diarias. Para ello, definiremos una matriz tridimensional de números enteros subindicada en M y en GR respectivamente para cada una de estas características, siendo WR el objeto de constantes de requerimientos semanales y $DMIN$ y $DMAX$ los objetos de constantes de requerimientos diarios mínimos y máximos respectivamente. De esta forma, para una asignatura válida $(m, g, l) \in SB$ sus características asociadas serán:
 - Requerimiento semanal: número de lecciones marcado por $wr_{m,g,l} \in \mathbb{Z}$ del objeto de constantes correspondiente, $wr_{m,g,l} \in WR$.
 - Requerimiento diario mínimo: número de lecciones diarias mínimas marcado por $dmin_{m,g,l} \in \mathbb{Z}$ del objeto de constantes correspondiente, $dmin_{m,g,l} \in DMIN$.
 - Requerimiento diario máximo: número de lecciones diarias máximas marcado por $dmax_{m,g,l} \in \mathbb{Z}$ del objeto de constantes correspondiente, $dmax_{m,g,l} \in DMAX$.
- Profesores: directamente mismo conjunto definido previamente, todos los índices $p \in P$ dan lugar a un profesor válido. Sus características de cara al modelo de organización de horarios serán:

- Disponibilidad máxima semanal: límite máximo de lecciones que pueden asignarse a un profesor en una semana. Definiremos un vector de números enteros subindicado en P , llamándolo WA' . De esta forma, la disponibilidad semanal de un profesor $p \in P$ vendrá dado por el número entero $wa'_p \in \mathbb{Z}$ perteneciente al objeto de constantes representando la disponibilidad máxima semanal, $wa'_p \in WA'$.
- Disponibilidad para vigilar recreos: la reserva o no de lecciones por parte de un profesor concreto marcará el cambio de valores del objeto de constantes de disponibilidad semanal A . De esta manera, definiremos un vector de número binarios subindicado en P , llamándolo R . Sus elementos $r_p \in R$ serán valores binarios $r_p \in \{0, 1\}$, de tal manera que un profesor $p \in P$ tendrá lecciones reservadas para vigilancia de recreo en el caso de que $r_p = 1$, mientras que en caso contrario no reservará lecciones en el caso de que $r_p = 0$. De esta manera, redefiniremos un nuevo objeto de disponibilidad semanal que tenga en cuenta la reserva de exactamente una lección semanal para los profesores que reservan disponibilidad para vigilancia de recreo. De esta forma, crearemos un vector de números enteros subindicado en P , llamándolo WA . Este objeto de constantes de disponibilidad se definirá en función del anterior de tal manera que la disponibilidad semanal actualizada $wa_p \in WA$ correspondiente a un profesor $p \in P$ valdrá:

$$wa_p = \begin{cases} wa'_p - 1, & r_p = 1 \\ wa'_p, & r_p = 0 \end{cases}$$

- Capacidad de docencia de materias: en función de si los profesores tienen la cualificación necesaria para enseñar un tipo concreto de materia, definiremos una matriz bidimensional de números binarios subindicada en P y en M respectivamente, llamándola Q . De esta manera, la capacidad de un profesor $p \in P$ para impartir una materia $m \in M$ vendrá dada por el número binario $q_{p,m} \in \{0, 1\}$ perteneciente a la matriz de constantes de capacidad de docencia, $q_{p,m} \in Q$. El profesor podrá enseñar la materia en el caso de que $q_{p,m} = 1$, mientras que en caso contrario no podrá enseñarla si $q_{p,m} = 0$.
- Disponibilidad en huecos semanales: en función de si los profesores se encuentran disponibles para impartir una lección en un hueco determinado, definiremos una matriz tridimensional de números binarios subindicado en P y en H respectivamente, llamándola A . De esta manera, la disponibilidad de un profesor $p \in P$ para impartir una lección en un hueco $(d, s) \in H$ vendrá dada por el número binario $a_{p,d,s} \in \{0, 1\}$ perteneciente a la matriz de constantes de disponibilidad de lecciones $a_{p,d,s} \in A$. El profesor podrá impartir la lección en el hueco en el caso de que $a_{p,d,s} = 1$, mientras que en caso contrario no podrá impartirla si $a_{p,d,s} = 0$.
- Carácter o no de tutor del profesor y grupo tutorizado en caso afirmativo: en función de si los profesores son tutores o no, definiremos un vector de números binarios subindicado en P llamado T' . De esta manera el carácter de tutor de un profesor $p \in P$ vendrá dado por el número $t'_p \in \{0, 1\}$ perteneciente al objeto de constantes de tutoría, $t'_p \in T$. De esta manera, un profesor será tutor en el caso de que $t'_p = 1$, mientras que si $t'_p = 0$ no lo será. En función a esto, definiremos dos nuevos objetos para resumir la información de aquellos profesores que sean tutores:
 - Lista de tutores: definiremos un subconjunto de la lista total de índices de profesores, de tal manera que cumplan la condición recién definida de tutorización. De esta manera, definiremos el subconjunto de tutores como $T = \{t \in \mathbb{Z} \mid t \in P \cap t'_p = 1\} \subset P$, pudiendo representar un tutor por un índice, $t \in T$.
 - Lista de grupos tutorizados: definiremos un vector de duplas de números enteros subindicado en la lista de tutores T , representando cada uno de estos elementos los índices respectivamente de grado y letra del grupo tutorizado, llamado GRT . Este elemento GRT será una reordenación o permutación del objeto GR de grupos válidos. De esta forma, el grupo tutorizado por un profesor $p \in P$ vendrá dado por la dupla de índices perteneciente al objeto

de constantes de grupos tutorizados, $grt_p = (g_p, l_p) \in GRT$. Esta dupla de índices deberá conformar un grupo válido, por lo que además deberá de cumplirse que $(g_p, l_p) \in GR$.

Tras definir estos objetos estructurados de los que hemos hablado en la subsección 2.1.2 a partir de los índices fundamentales nombrados en la sección 2.1.1, nos quedará por último dar carácter matemático a los conceptos de docencia y horario definidos también a lo largo de la subsección 2.1.2.

- **Docencia:** definiremos el conjunto de todas las docencias posibles como la asignación de profesores a asignaturas permitidas cuya materia puedan impartir. De esta manera, el conjunto de todas las docencias permitidas podrá definirse como:

$$I = \{(p, m, g, l) \mid p \in P \cap (m, g, l) \in SB \cap q_{p,m} = 1\}$$

Veremos más adelante como este conjunto nos ayudará a modelar el efecto buscado de que los profesores asuman todas las lecciones requeridas semanalmente en bloque o no impartan la asignatura. Observamos como no hemos tenido que comprobar la validez de la tupla de tres elementos (m, g, l) debido al hecho de que nos hemos asegurado de que pertenezca al conjunto de asignaturas válidas A .

- **Lección:** definiremos el conjunto de todas las lecciones posibles como la concrección en un hueco determinado del horario de una docencia permitida entre un profesor y una asignatura, vigilando que el profesor se encuentre disponible en el hueco determinado. De esta manera, el conjunto de todas las lecciones permitidas podrá definirse como:

$$J = \{(p, m, g, l, d, s) \mid (p, m, g, l) \in I \cap (d, s) \in H \cap a_{p,d,s} = 1\}$$

Observamos como no hemos necesitado tener en cuenta la validez de la tupla de cuatro elementos (p, m, g, l) de docencia asignada, debido al hecho de que nos hemos asegurado de que pertenezca al conjunto de docencias permitidas I .

Variables de decisión.

En términos generales, nuestra organización del horario se va a basar en la asignación de lecciones en huecos concretos para las docencias completas de asignaturas asignadas a profesores. Por lo tanto, deberemos definir dos conjuntos de variables matemáticas binarias de decisión que nos permitan definir si cada una de las docencias y lecciones son establecidas o no.

- **Docencia:** para toda tupla permitida de docencia entre profesor y asignatura definiremos una variable de decisión. Su conjunto de variables de decisión Z como:

$$Z = \{z_{p,m,g,l} \in \{0,1\}, \forall (p,m,g,l) \in I\}$$

De tal manera que una docencia permitida $(p, m, g, l) \in I$ se verá asignada en función del valor de su variable de decisión binaria asociada $z_{p,m,g,l} \in Z$. La docencia se verá asignada si el valor de la variable de decisión tras la resolución del problema es $z_{p,m,g,l} = 1$, mientras que no se verá asignada en el caso de que $z_{p,m,g,l} = 0$.

- **Lección:** para toda tupla correspondiente a una asignación de una docencia permitida en un hueco determinado en el que el docente se encuentre disponible definiremos una variable de decisión. Su conjunto de variables de decisión X será:

$$X = \{x_{p,m,g,l,d,s} \in \{0,1\}, \forall (p,m,g,l,d,s) \in J\}$$

De tal manera que una docencia permitida $(p, m, g, l) \in I$ se verá asignada en función del valor de su variable de decisión binaria asociada $z_{p,m,g,l} \in Z$. La docencia se verá asignada si el valor de la variable de decisión tras la resolución del problema es $z_{p,m,g,l} = 1$, mientras que no se verá asignada en el caso de que $z_{p,m,g,l} = 0$.

Problema en dos etapas.

Tal y como hemos comentado en el capítulo 1, la mayoría de enfoques a la hora de plantear problemas de organización de horarios, bien sea a través de programación lineal entera tradicional o por algoritmos metaheurísticos, consiste en considerar el problema de manera global, considerando el total de variables y restricciones para intentar obtener una solución al problema real. Sin embargo, a lo largo de este trabajo se pretende cambiar el enfoque y plantear el problema global como dos etapas o subproblemas con gran independencia entre sí, de tal manera que organicemos en primer lugar una docencia factible a través de la primera etapa del modelo mientras que en la segunda utilicemos la solución de docencia obtenida para comprobar la factibilidad del horario. De esta manera, definiremos dos problemas independientes en dos apartados diferentes. Podremos traducir tanto todas las reglas a asegurar como destinadas a control de calidad en «hard requirements», restricciones de obligado cumplimiento. De esta manera, no existiran «soft requirements» o condiciones a optimizar. Por lo tanto, para ambas etapas del problema bastará con comprobar la factibilidad del problema, tal como se planteó en el capítulo 1. Pasaremos a traducir las reglas a seguir expresadas en a lo largo de la sección 2.2 a través de las constantes matemáticas y variables de decisión binarias decididas en las pasadas subsecciones 2.3.1 y 2.3.2 en restricciones matemáticas que conformen un problema cuya factibilidad comprobaremos al programarlo.

Etapla 1: Docencia. Iremos regla por regla, incluyendo tanto reglas de obligado cumplimiento como aquellas de control de calidad en el caso de las restricciones de docencia.

- (1) Ningún profesor supera su disponibilidad semanal máxima. El conjunto de lecciones requeridas por todas las asignaturas cuya docencia se le asigna no puede exceder el límite máximo recogido.

$$\sum_{(p=\bar{p}, m, g, l) \in I} wr_{m,g,l} z_{p,m,g,l} \leq wa_{\bar{p}} \quad \forall \bar{p} \in P \quad (2.1)$$

- (2) Todos los profesores son asignados un número de lecciones tal que alcancen el porcentaje mínimo de su disponibilidad total máxima semanal. El control del valor de este parámetro $at \in [0, 1]$ de objetivo de disponibilidad se dejará en manos del usuario, pudiendo explorar posteriormente diversos modelos en el algoritmo metaheurístico programado. El conjunto de lecciones requeridas por todas las asignaturas cuya docencia se le asigna debe alcanzar el porcentaje relativo definido at respecto a su disponibilidad máxima semanal.

$$\sum_{(p=\bar{p}, m, g, l) \in I} wr_{m,g,l} z_{p,m,g,l} \geq at \cdot wa_{\bar{p}} \quad \forall \bar{p} \in P \quad (2.2)$$

- (3) Todos los tutores son asignados un número de lecciones en su propio grupo tutorizado tal que alcancen el porcentaje mínimo de su lecciones asignadas en el total de los grupos. El control del valor de este parámetro $gt \in [0, 1]$ de objetivo de disponibilidad se dejará en manos del usuario, pudiendo explorar posteriormente diversos modelos en el algoritmo metaheurístico programado. El conjunto de lecciones requeridas por todas las asignaturas cuya docencia se le asigna en el grupo tutorizado debe alcanzar el porcentaje relativo gt definido respecto al total de lecciones asignadas en esa semana.

$$\sum_{(t=\bar{t}, m, g=g_{\bar{p}}, l=l_{\bar{p}}) \in I} wr_{m,g,l} z_{t,m,g,l} \geq gt \cdot \left(\sum_{(t=\bar{t}, m, g, l) \in I} wr_{m,g,l} z_{t,m,g,l} \right), \quad \forall \bar{t} \in T \quad (2.3)$$

- (4) Todas las asignaturas reciben una única docencia por un profesor concreto, evitando la impartición de lecciones en una misma asignatura por varios profesores diferentes.

$$\sum_{(p, m=\bar{m}, g=\bar{g}, l=\bar{l}) \in I} z_{p,m,g,l} = 1 \quad \forall (\bar{m}, \bar{g}, \bar{l}) \in SB \quad (2.4)$$

- (5) Todos los grupos tienen la docencia de su asignatura correspondiente a la materia Tutoría asignada a su propio tutor. Sea $m_{TU} \in M$ el índice correspondiente a la asignatura de Tutoría:

$$z_{t,m_{TU},g_t,l_t} = 1 \quad \forall t \in T \quad (2.5)$$

- Variables de decisión binarias de docencia:

$$z_{p,m,g,l} \in \{0, 1\}, \quad \forall (p, m, g, l) \in I$$

La comprobación de la factibilidad del problema en función de las variables de decisión binarias definidas y las restricciones planteadas dará lugar a una solución de referencia que será empleada como estructura de docencia en la comprobación de la existencia de un horario en la segunda etapa del problema correspondiente a la organización de lecciones. Este conjunto de valores de los valores de decisión podrá ser representado como Z^0 , de tal manera que la asignación o no de una docencia $(p, m, g, l) \in I$ vendrá dada por el valor de su variable de decisión correspondiente $z_{p,m,g,l}^0 \in Z$ una vez resuelto el problema.

Etapla 2: Lecciones. Para esta segunda etapa del problema consideraremos la solución de docencia $z_{p,m,g,l} \in Z, \quad \forall (p, m, g, l) \in I$ obtenida en la primera etapa, y a partir de ahí, trataremos de encontrar un horario factible para el problema de lecciones. Iremos regla por regla, incluyendo únicamente reglas de obligado cumplimiento en el caso de las restricciones de lecciones.

- (6) Todas las asignaturas reciben el número de lecciones semanales requeridas por el profesor designado en la etapa anterior repartidas entre todos los huecos del horario. Teniendo en cuenta la solución de docencia de referencia Z^0 obtenida tras la primera etapa plantearemos la siguiente condición.

$$\sum_{(p=\bar{p}, m=\bar{m}, g=\bar{g}, l=\bar{l}, d, s) \in J} x_{p,m,g,l,d,s} = w r_{\bar{m},\bar{g},\bar{l}} z_{\bar{p},\bar{m},\bar{g},\bar{l}}^0 \quad \forall (\bar{p}, \bar{m}, \bar{g}, \bar{l}) \in I \quad (2.6)$$

- (7) Ningún profesor presenta duplicidades en su horario, asegurándonos de que en todos los huecos del horario en los que se encuentre disponible se le asigne una o ninguna lección, quedando en este último caso libre para el hueco.

$$\sum_{(p=\bar{p}, m, g, l, d=\bar{d}, s=\bar{s}) \in J} x_{p,m,g,l,d,s} \leq 1 \quad \forall \bar{p} \in P, \quad \forall (\bar{d}, \bar{s}) \in H \mid a_{\bar{p},\bar{d},\bar{s}} = 1 \quad (2.7)$$

- (8) Ningún grupo presenta duplicidades en su horario, asegurándonos de que en todos los huecos del horario se le asigne exactamente una sesión, sin contemplar la posibilidad de que se quede libre.

$$\sum_{(p, m, g=\bar{g}, l=\bar{l}, d=\bar{d}, s=\bar{s}) \in J} x_{p,m,g,l,d,s} = 1 \quad \forall (\bar{g}, \bar{l}) \in GR, \quad \forall (\bar{d}, \bar{s}) \in H \quad (2.8)$$

- (9) Todos los días se asigna a cada asignatura un número de lecciones comprendido entre el mínimo y el máximo diario definido.

- (9.1) Cada día se alcanzan las lecciones mínimas diarias definidas para cada asignatura.

$$\sum_{(p, m=\bar{m}, g=\bar{g}, l=\bar{l}, d=\bar{d}, s) \in J} x_{p,m,g,l,d,s} \geq dmin_{\bar{m},\bar{g},\bar{l}} \quad \forall (\bar{m}, \bar{g}, \bar{l}) \in SB, \quad \forall \bar{d} \in D \quad (2.9)$$

- (9.2) Cada día no se superan las lecciones máximas diarias definidas para cada asignatura.

$$\sum_{(p, m=\bar{m}, g=\bar{g}, l=\bar{l}, d=\bar{d}, s) \in J} x_{p,m,g,l,d,s} \leq dmax_{\bar{m},\bar{g},\bar{l}} \quad \forall (\bar{m}, \bar{g}, \bar{l}) \in SB, \quad \forall \bar{d} \in D \quad (2.10)$$

- Variables de decisión binarias de docencia:

$$x_{p,m,g,l,d,s} \in \{0,1\}, \quad \forall (p,m,g,l,d,s) \in J$$

Comprobaremos la factibilidad del problema de horarios para ver si es posible obtener una organización de lecciones correcta en esta segunda etapa del problema global con la información de referencia de docencia establecida en la primera etapa, Z^0 . En caso de existir una solución factible X^0 para este segundo subproblema, la asignación de una lección concreta vendrá dada por el valor asignado a su variable de decisión, $x_{p,m,g,l,d,s}^0$.

Solución global. Obtendremos diferentes soluciones de docencia en la primera etapa hasta que la utilización de una de estas como información de referencia en la segunda etapa dé lugar a un problema que resulte ser factible. En ese caso, la solución al problema global será el conjunto de las soluciones ambas etapas del problema, Z^0 y X^0 respectivamente, que han dado lugar a una solución factible en cada uno de los dos subproblemas. Esta solución marcará la organización del horario del centro obtenida.

2.3.2. Ejemplo de centro de estudio.

Pasaremos a caracterizar un caso concreto de un centro para poner en práctica el modelo teórico genérico. Según hemos podido comprobar a lo largo de la anterior subsección 2.3.1, en concreto a lo largo de la subsubsección 2.3.1, implica caracterizar los objetos matemáticos de constantes propios del centro, ya que tanto las variables como las restricciones de las subsubsecciones 2.3.1 y 2.3.1 respectivamente se definen de manera genérica en función de estas características del centro entendidas como constantes numéricas. Por lo tanto, a lo largo de esta subsección pasaremos a dar una expresión concreta de los objetos matemáticos correspondientes al centro de Educación Primaria que utilizaremos a lo largo de esta memoria para ilustrar el problema tratado.

Comenzaremos por caracterizar los índices principales del problema. Dispondremos de:

- Profesores: 22 docentes en el centro.
- Materias: 11 materias diferentes.
- Grado: 6 grados diferentes.
- Letra: un máximo de 3 letras por grado.
- Día: 5 días.
- Sesión: 6 sesiones por día.

Expresaremos los objetos que recogen los índices principales de la subsubsección 2.1.1, respectivamente, P , M , G , L , D y S . Junto al número del índice correspondiente al elemento estableceremos un alias. A su vez, definiremos los objetos subindicados.

Para organizar mejor la serie de definiciones a plantear, nos centraremos en cada uno de los objetos compuestos de la subsección 2.1.2, empleando un apartado para cada uno de estos que nos permita definir sus objetos asociados.

Huecos. Comenzaremos por definir la estructura de huecos en función de los días y sesiones que ocupen cada uno. Tendremos los días Lunes, Martes, Miércoles, Jueves y Viernes, con todas las sesiones 1ª, 2ª, 3ª, 4ª, 5ª y 6ª permitidas durante todos los días, formando por lo tanto una estructura con todas las combinaciones de días y sesiones dando lugar a un hueco válido:

Sesión / Día	Lunes	Martes	Miércoles	Jueves	Viernes
1 ^a	✓	✓	✓	✓	✓
2 ^a	✓	✓	✓	✓	✓
3 ^a	✓	✓	✓	✓	✓
4 ^a	✓	✓	✓	✓	✓
5 ^a	✓	✓	✓	✓	✓
6 ^a	✓	✓	✓	✓	✓

Una vez definidas las características de la estructura de huecos para el centro, traduzcámoslas a constantes matemáticas. En primer lugar, definiremos los dos índices fundamentales para posteriormente definir el objeto compuesto en función de estos dos.

- Días: objeto resumiendo el índice fundamental de cada día con su alias entre paréntesis.

$$D = \{1 \ (L) \ 2 \ (M) \ 3 \ (X) \ 4 \ (J) \ 5 \ (V)\}$$

- Sesión: objeto resumiendo el índice fundamental de cada sesión con su alias entre paréntesis.

$$S = \{1 \ (1^a) \ 2 \ (2^a) \ 3 \ (3^a) \ 4 \ (4^a) \ 5 \ (5^a) \ 6 \ (6^a)\}$$

- Huecos: objeto compuesto de huecos recogiendo el índice de día y sesión al que pertenecen respectivamente, y mostrando entre paréntesis el alias de cada hueco.

$$H = \left\{ \begin{array}{l} (1,1) \ (L, 1^a) \ (1,2) \ (L, 2^a) \ (1,3) \ (L, 3^a) \ (1,4) \ (L, 4^a) \ (1,5) \ (L, 5^a) \ (1,6) \ (L, 6^a) \\ (2,1) \ (M, 1^a) \ (2,2) \ (M, 2^a) \ (2,3) \ (M, 3^a) \ (2,4) \ (M, 4^a) \ (2,5) \ (M, 5^a) \ (2,6) \ (M, 6^a) \\ (3,1) \ (X, 1^a) \ (3,2) \ (X, 2^a) \ (3,3) \ (X, 3^a) \ (3,4) \ (X, 4^a) \ (3,5) \ (X, 5^a) \ (3,6) \ (X, 6^a) \\ (4,1) \ (J, 1^a) \ (4,2) \ (J, 2^a) \ (4,3) \ (J, 3^a) \ (4,4) \ (J, 4^a) \ (4,5) \ (J, 5^a) \ (4,6) \ (J, 6^a) \\ (5,1) \ (V, 1^a) \ (5,2) \ (V, 2^a) \ (5,3) \ (V, 3^a) \ (5,4) \ (V, 4^a) \ (5,5) \ (V, 5^a) \ (5,6) \ (V, 6^a) \end{array} \right\}$$

Grupos. Pasaremos a caracterizar los diversos grupos que componen la estructura de grados y letras del centro educativo propuesto. En nuestro modelo propuesto los grados entre 1º y 3º tendrán un total de dos letras *A* y *B* conformando dos grupos diferentes por grado, mientras que los grados entre 4º y 6º tendrán un total de tres letras *A*, *B* y *C* conformando tres grupos diferentes por grado. La estructura de combinaciones de grados y letras que den lugar a grupos válidos por lo tanto será la siguiente:

Grado / Letra	A	B	C
1º	✓	✓	✗
2º	✓	✓	✗
3º	✓	✓	✗
4º	✓	✓	✓
5º	✓	✓	✓
6º	✓	✓	✓

Teniendo en cuenta esto, podremos expresar los siguientes objetos matemáticos asociados a los grupos. En primer lugar, definiremos los dos índices fundamentales para posteriormente definir el objeto compuesto en función de estos dos, finalizando con la definición de los objetos subindicados.

- Grados: conjunto de índices de grados con su alias entre paréntesis.

$$G = \{1 \ (1^\circ) \ 2 \ (2^\circ) \ 3 \ (3^\circ) \ 4 \ (4^\circ) \ 5 \ (5^\circ) \ 6 \ (6^\circ)\}$$

- Letras: conjunto de índices de letras con su alias entre paréntesis.

$$L = \{(1 \ (A) \ 2 \ (B) \ 3 \ (C))\}$$

- Grupos: definición del objeto compuesto que recoge los dos índices del grupo permitido con el detalle de su alias entre paréntesis.

$$GR = \left\{ \begin{array}{cccccc} (1,1) & (1^o,A) & (1,2) & (1^o,B) & & \\ (2,1) & (2^o,A) & (2,2) & (2^o,B) & & \\ (3,1) & (3^o,A) & (3,2) & (3^o,B) & & \\ (4,1) & (4^o,A) & (4,2) & (4^o,B) & (4,3) & (4^o,C) \\ (5,1) & (5^o,A) & (5,2) & (5^o,B) & (5,3) & (5^o,C) \\ (6,1) & (6^o,A) & (6,2) & (6^o,B) & (6,3) & (6^o,C) \end{array} \right\}$$

- Tutores asociados: definición del objeto subindicado en GR recogiendo el índice del profesor que se encarga de la tutorización de cada grupo. Se incluye el alias del profesor entre paréntesis.

$$TGR = \left\{ \begin{array}{ccccc} 1 & (PR1_1) & 2 & (PR1_2) & \\ 3 & (PR1_3) & 4 & (PR1_4) & \\ 5 & (PR1_5) & 6 & (PR1_6) & \\ 7 & (PIN_1) & 8 & (PIN_2) & 9 & (PIN_3) \\ 10 & (PIN_4) & 11 & (PIN_5) & 12 & (PEF_1) \\ 13 & (PEF_2) & 14 & (PEF_3) & 15 & (PFR_1) \end{array} \right\}$$

Asignaturas. En primer lugar, comenzaremos por definir la lista de materias impartidas en el centro. Estableceremos un alias para cada una de esta para su posterior uso:

Asignatura	Alias
Lengua	<i>LE</i>
Matemáticas	<i>MA</i>
Ciencias Naturales	<i>CN</i>
Ciencias Sociales	<i>CS</i>
Inglés	<i>IN</i>
Educación Física	<i>EF</i>
Música	<i>MU</i>
Plástica	<i>PL</i>
Valores	<i>VA</i>
Tutoría	<i>TU</i>
Francés	<i>FR</i>

Tendremos que definir qué materias se encuentran definidas en qué grupos, caracterizando sus requerimientos de lecciones semanales, diarios mínimos y máximos. Serán mostrados en ese orden para cada asignatura permitida. Aquellas combinaciones de materia y grupo que no sean requeridas, sin embargo, mostrarán un símbolo **X** para distinguirlas.

Grupo / Asignatura	LE	MA	CN	CS	IN	EF	MU	PL	VA	TU	FR
(1º,A)	5,1,1	5,1,2	3,0,1	2,0,1	4,0,1	4,0,1	2,0,1	2,0,1	2,0,1	1,0,1	✗
(1º,B)	5,1,1	5,1,2	3,0,1	2,0,1	4,0,1	4,0,1	2,0,1	2,0,1	2,0,1	1,0,1	✗
(1º,A)	5,1,1	5,1,2	3,0,1	2,0,1	4,0,1	4,0,1	2,0,1	2,0,1	2,0,1	1,0,1	✗
(2º,B)	5,1,1	5,1,2	3,0,1	2,0,1	4,0,1	4,0,1	2,0,1	2,0,1	2,0,1	1,0,1	✗
(3º,A)	5,1,1	6,1,2	2,0,1	2,0,1	4,0,1	4,0,1	2,0,1	2,0,1	2,0,1	1,0,1	✗
(3º,B)	5,1,1	6,1,2	2,0,1	2,0,1	4,0,1	4,0,1	2,0,1	2,0,1	2,0,1	1,0,1	✗
(4º,A)	5,1,1	6,1,2	2,0,1	2,0,1	4,0,1	4,0,1	2,0,1	2,0,1	2,0,1	1,0,1	✗
(4º,B)	5,1,1	6,1,2	2,0,1	2,0,1	4,0,1	4,0,1	2,0,1	2,0,1	2,0,1	1,0,1	✗
(4º,C)	5,1,1	6,1,2	2,0,1	2,0,1	4,0,1	4,0,1	2,0,1	2,0,1	2,0,1	1,0,1	✗
(5º,A)	5,1,1	5,1,2	3,0,1	2,0,1	4,0,1	3,0,1	1,0,1	2,0,1	2,0,1	1,0,1	2,0,1
(5º,B)	5,1,1	5,1,2	3,0,1	2,0,1	4,0,1	3,0,1	1,0,1	2,0,1	2,0,1	1,0,1	2,0,1
(5º,C)	5,1,1	5,1,2	3,0,1	2,0,1	4,0,1	3,0,1	1,0,1	2,0,1	2,0,1	1,0,1	2,0,1
(6º,A)	5,1,1	5,1,2	2,0,1	2,0,1	4,0,1	3,0,1	2,0,1	2,0,1	2,0,1	1,0,1	2,0,1
(6º,B)	5,1,1	5,1,2	2,0,1	2,0,1	4,0,1	3,0,1	2,0,1	2,0,1	2,0,1	1,0,1	2,0,1
(6º,C)	5,1,1	5,1,2	2,0,1	2,0,1	4,0,1	3,0,1	2,0,1	2,0,1	2,0,1	1,0,1	2,0,1

Una vez caracterizada la estructura general, traduzcámosla a objetos de constantes matemáticas. Comenzaremos por definir el índice fundamental restante de materias, para posteriormente definir el objeto completo de asignaturas y caracterizar por último sus objetos subindicados.

- Materias: conjunto de índices de las diferentes materias con su alias de detalle entre paréntesis.

$$M = \left\{ \begin{array}{ccccccccc} 1 & (LE) & 2 & (MA) & 3 & (CN) & 4 & (CS) & 5 & (IN) \\ 6 & (EF) & 7 & (MU) & 8 & (PL) & 9 & (VA) & 10 & (TU) & 11 & (FR) \end{array} \right\}$$

- Asignaturas: conjunto de tuplas de índices subindicado respectivamente en M y en GR . Recogeremos respectivamente los índices de materia, grado y letra para cada asignatura permitida, mostrando entre paréntesis un detalle con el alias de la asignatura. Debido al enorme número de combinaciones entre grupos y materias ($15 \times 11 - 2 \times 3 = 159$), mostraremos únicamente determinada parte del objeto de índices S , mostrando los casos ilustrativos de la materia Lengua (LE , índice 1) por estar definida para todos los grupos y de la materia Francés (FR , índice 11)

$$SB = \left\{ \begin{array}{l} (1,1,1) (LE,1^\circ,A) \quad (1,1,2) (LE,1^\circ,B) \\ (1,2,1) (LE,2^\circ,A) \quad (1,1,1) (LE,2^\circ,B) \\ (1,3,1) (LE,3^\circ,A) \quad (1,3,2) (LE,3^\circ,B) \\ (1,4,1) (LE,4^\circ,A) \quad (1,4,2) (LE,4^\circ,B) \quad (1,4,3) (LE,4^\circ,C) \\ (1,5,1) (LE,5^\circ,A) \quad (1,5,2) (LE,5^\circ,B) \quad (1,5,3) (LE,5^\circ,C) \\ (1,6,1) (LE,6^\circ,A) \quad (1,6,2) (LE,6^\circ,B) \quad (1,6,3) (LE,6^\circ,C) \\ \dots \\ (11,5,1) (FR,5^\circ,A) \quad (11,5,2) (FR,5^\circ,B) \quad (11,5,3) (FR,5^\circ,C) \\ (11,6,1) (FR,6^\circ,A) \quad (11,6,2) (FR,6^\circ,B) \quad (11,6,3) (FR,6^\circ,C) \end{array} \right\}$$

- Requerimientos semanales: describiremos el objeto subindicado en SB que recoge estos aspectos de la asignatura. De nuevo, por razones de espacio y claridad, mostraremos únicamente las

lecciones requeridas para las materias Lengua y Francés a lo largo de los diferentes grupos.

$$WR = \begin{pmatrix} 5 & 5 \\ 5 & 5 \\ 5 & 5 \\ 5 & 5 & 5 \\ 5 & 5 & 5 \\ 5 & 5 & 5 \\ \dots \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{pmatrix}$$

- Requerimientos diarios mínimos: describiremos el objeto subindicado en *SB* que recoge este aspectos de la asignatura. De nuevo, por razones de espacio y claridad, mostraremos únicamente las lecciones requeridas para las materias Lengua y Francés a lo largo de los diferentes grupos.

$$DMIN = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ \dots \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- Requerimientos diarios máximos: describiremos el objeto subindicado en *SB* que recoge este aspectos de la asignatura. De nuevo, por razones de espacio y claridad, mostraremos únicamente las lecciones requeridas para las materias Lengua y Francés a lo largo de los diferentes grupos.

$$DMAX = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ \dots \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Profesores. Comenzaremos por una descripción del total de los profesores de los que dispone el centro, estableciendo su índice, un alias para cada uno en función de su especialidad, su disponibilidad semanal máxima, su necesidad de vigilar recreos y su especialidad ⁵.

⁵Esta última definirá las asignaturas que es capaz de impartir cada uno de los profesores

Índice	Alias	Disp. semanal máx.	Vig. recreos	Especialidad
1	$PR1_1$	30	✓	$PR1$
2	$PR1_2$	30	✓	$PR1$
3	$PR1_3$	30	✓	$PR1$
4	$PR1_4$	30	✓	$PR1$
5	$PR1_5$	30	✓	$PR1$
6	$PR1_6$	30	✓	$PR1$
7	$PR1_7$	15	✓	$PR1$
8	PIN_1	30	✓	PIN
9	PIN_2	30	✓	PIN
10	PIN_3	30	✓	PIN
11	PIN_4	30	✓	PIN
12	PIN_5	30	✓	PIN
13	PIN_6	8	✗	PIN
14	PEF_1	30	✓	PEF
15	PEF_2	30	✓	PEF
16	PEF_3	30	✓	PEF
17	PEF_4	30	✓	PEF
18	PMU_1	30	✓	PMU
19	PMU_2	15	✓	PMU
20	PMU_3	15	✓	PMU
21	PFR_1	30	✓	PFR
22	PFR_2	8	✗	PFR

Además, deberemos de caracterizar la disponibilidad de los diferentes profesores a lo largo de los huecos del horarios:

- Por norma general, todos los profesores se encontrarán disponibles en todos los huecos del horario.
- $PR1_7$ se encuentra disponible todos los días entre 1ª y 3ª sesión.
- PIN_6 se encuentra disponible todos los días entre 4ª y 6ª sesión.
- PMU_2 se encuentra disponible entre Lunes y Miércoles durante todas las sesiones.
- PMU_3 se encuentra disponible entre Miércoles y Viernes durante todas las sesiones.
- PFR_2 se encuentra disponible entre Lunes y Miércoles durante todas las sesiones.

Por último, podremos resumir la lista de materias de cuya docencia puede encargarse un profesor con la siguiente tabla. Esta cualificación será la misma para todos los profesores de un mismo tipo de especialidad:

Profesor / Material	LE	MA	CN	CS	IN	EF	MU	PL	VA	TU	FR
$PR1_1$	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗
$PR1_2$	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗
$PR1_3$	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗
$PR1_4$	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗
$PR1_5$	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗
$PR1_6$	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗
$PR1_7$	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗
PIN_1	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗
PIN_2	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗
PIN_3	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗
PIN_4	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗
PIN_5	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗
PIN_6	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗
PEF_1	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	✗
PEF_2	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	✗
PEF_3	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	✗
PEF_4	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	✗
PMU_1	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗
PMU_2	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗
PMU_3	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗
PFR_1	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓
PFR_2	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓

Con esta información en mente, podremos expresar los siguientes objetos matemáticos asociados a los profesores:

- Objeto de índice fundamental de profesores P . Además del índice, expresaremos su alias entre paréntesis.

$$P = \left\{ \begin{array}{cccccc} 1 (PR1_1) & 2 (PR1_2) & 3 (PR1_3) & 4 (PR1_4) & 5 (PR1_5) & \\ 6 (PR1_6) & 7 (PR1_7) & 8 (PIN_1) & 9 (PIN_2) & 10 (PIN_3) & \\ 11 (PIN_4) & 12 (PIN_5) & 13 (PIN_6) & 14 (PEF_1) & 15 (PEF_2) & 16 (PEF_3) \\ 17 (PEF_4) & 18 (PMU_1) & 19 (PMU_2) & 20 (PMU_3) & 21 (PFR_1) & 22 (PFR_2) \end{array} \right\}$$

Una vez definido el objeto de índices, podremos definir sus objetos subindicados correspondientes.

- Objeto de disponibilidad semanal total WA' . Teniendo en cuenta la indexación en P :

$$WA' = \left\{ \begin{array}{ccccc} 30 & 30 & 30 & 30 & 30 \\ 30 & 15 & 30 & 30 & 30 \\ 30 & 30 & 8 & 30 & 30 & 30 \\ 30 & 30 & 15 & 15 & 30 & 8 \end{array} \right\} \quad (2.11)$$

- Objeto de disponibilidad semanal actualizada WA . Teniendo en cuenta la disponibilidad total de los profesores y su reserva o no de una lección para vigilancia del recreo, el objeto de constantes asociado quedará:

$$WA = \left\{ \begin{array}{ccccc} 29 & 29 & 29 & 29 & 29 \\ 29 & 14 & 29 & 29 & 29 \\ 29 & 29 & 8 & 29 & 29 & 29 \\ 29 & 29 & 14 & 14 & 29 & 8 \end{array} \right\}$$

- Objeto de calificación para enseñar materias. Teniendo en cuenta la indexación en P por filas y en M por columnas:

$$Q = \left\{ \begin{array}{c|cccccccccccc} & LE & MA & CN & CS & IN & EF & MU & PL & VA & TU & FR \\ \hline PR1_1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ PR1_2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ PR1_3 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ PR1_4 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ PR1_5 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ PR1_6 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ PR1_7 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ PIN_1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ PIN_2 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ PIN_3 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ PIN_4 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ PIN_5 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ PIN_6 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ PEF_1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ PEF_2 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ PEF_3 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ PEF_4 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ PMU_1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ PMU_2 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ PMU_3 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ PFR_1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ PFR_2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \right\}$$

- Objeto de disponibilidad semanal. Recordemos que este objeto se encuentra indexado en P y en H respectivamente, por lo que será de carácter tridimensional. Esto junto al elevado número de combinaciones de profesores y huecos ($22 \times 5 \times 6 = 660$) y a la repetición de disponibilidades entre profesores nos llevará a incluir únicamente los ejemplos representativos de la expresión de este objeto para diferentes profesores:

- Profesores que se encuentran disponibles todas las jornadas, $\forall p \in P \quad p \notin 7, 13, 19, 20, 22$:

$$A_p = \left\{ \begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array} \right\}$$

- $PR1_7$ disponible únicamente entre 1ª y 3ª hora:

$$A_7 = \left\{ \begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right\}$$

- PIN_6 disponible únicamente entre 4ª y 6ª:

$$A_{13} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- PMU_2 disponible únicamente de Lunes a Miércoles:

$$A_{19} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

- PMU_3 disponible únicamente de Miércoles a Viernes:

$$A_{20} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

- PRF_2 disponible únicamente de Lunes a Miércoles:

$$A_{22} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Quedará otro aspecto por caracterizar dentro de la información de los profesores, y es el relacionado con la tutorización de grupos. Definiremos el subconjunto de Tutores $T \subset P$ en función del conjunto de Profesores original.

- Lista de tutores: incluiremos en este conjunto el índice de cada profesor que resulte tutor. Mostraremos un detalle entre paréntesis del alias del profesor por clarificar.

$$T = \begin{pmatrix} 1 (PR1_1) & 2 (PR1_2) & 3 (PR1_3) & 4 (PR1_4) & 5 (PR1_5) \\ 6 (PR1_6) & 8 (PIN_1) & 9 (PIN_2) & 10 (PIN_3) & 11 (PIN_4) \\ 12 (PIN_5) & 14 (PEF_1) & 15 (PEF_2) & 16 (PEF_3) & 21 (PRF_1) \end{pmatrix}$$

- Grupos tutorizados: en este conjunto subindexado en T mostraremos los índices de grupo tutorizado para cada tutor. Mostraremos entre paréntesis un detalle del alias del grupo por clarificar.

$$GRT = \begin{pmatrix} (1,1) (1^\circ, A) & (1,2) (1^\circ, B) & (2,1) (2^\circ, A) & (2,2) (2^\circ, B) & (3,1) (3^\circ, A) \\ (3,2) (3^\circ, B) & (4,1) (4^\circ, A) & (4,2) (4^\circ, B) & (4,3) (4^\circ, C) & (5,1) (5^\circ, A) \\ (5,2) (5^\circ, B) & (5,3) (5^\circ, C) & (6,1) (6^\circ, A) & (6,2) (6^\circ, B) & (6,3) (6^\circ, C) \end{pmatrix}$$

Capítulo 3

Algoritmo: planteamiento, programación y resolución.

En este apartado nos encargaremos tanto del algoritmo utilizado para resolver el problema como de su puesta en marcha práctica a través de su programación. Comentar que el algoritmo será del tipo utilizado en el artículo [3] enlazado en la referencia, como continuación del anterior Trabajo de Fin de Máster. Será planteado a lo largo de la sección 3.2 y sus resultados analizados dentro de la sección 3.4 para finalizar el presente capítulo 3. Por otra parte, la escritura del código implicará reescribir todas las instrucciones a Java y a las librerías de Cplex en concreto con respecto al código utilizado en el artículo [3]. Introduciremos los principales aspectos de la programación orientada a objetos a lo largo de la sección 3.1 para posteriormente indicar como influye esto en la estructura del código a lo largo de la sección 3.3.

3.1. Principios generales de la programación orientada a objetos.

La programación orientada a objetos implica un paradigma de programación de uso extendido en la actualidad. Se basa en los siguientes principios:

- Clase: definiremos para un tipo de objeto concreto la información que este contiene así como la manera de actuar de la clase. Será el núcleo central en el que definiremos el comportamiento que rige los elementos que tienen un papel en el proyecto de Java, sentando las bases de su funcionamiento. Se articulará en torno a dos elementos fundamentales:
 - Atributo: información contenida por la clase, definiendo el tipo de campo concreto al que corresponde así como las limitaciones en su acceso. Comentaremos esto más adelante.
 - Método: funciones para operar los datos existentes, tanto existentes en librerías predefinidas para Java como algunas definidas de forma detallada por el usuario que desarrolla el proyecto concreto. A su vez, también podremos definir la capacidad de acceso del programa principal al método como en el caso de los atributos.

En este sentido para crear un objeto concreto deberemos de instanciar una clase. La creación de esta nueva instancia asigna al objeto un identificador determinado y le asocia los atributos y métodos de la citada clase a la que hemos definido que pertenece.

- Encapsulamiento: abstracción de la estructura del programa que la separa de su implantación, asegurando un acceso seguro a los datos. Definiremos todos los atributos de las clases como privados, de tal manera que únicamente podamos accederlos a través de los métodos públicos de objetos correspondientes a instancias de la misma clase. Para ello se definirán los limitadores de acceso a atributos y métodos de una clase, pudiendo darse los casos de:

- **Público:** accesible por objetos pertenecientes a todo tipo de clases. Es parte externa visible, y por lo tanto presenta el nivel de protección mínimo.
- **Protegidos:** accesible por objetos pertenecientes a su clase determinada y a la clase superior que la engloba.
- **Privado:** accesible únicamente por objetos pertenecientes a la propia clase definida. Es la parte interna más privada, y por lo tanto presenta el nivel de protección máxima.

Definiremos los atributos de la clase como privados mientras que los métodos que se encargan de serán ubicados dentro de esa misma clase y declarados como públicos, para que los atributos puedan ser accedidos desde el programa principal y no queden totalmente aislados. Al ser contenidos dentro de la misma clase que los atributos, los métodos de esta podrán acceder a los atributos pese a su carácter de privacidad. Esto proporciona un acceso controlado a la información contenida en los objetos de cada tipo de clase, asegurada la privacidad de los datos a la vez que evitamos cometer accesos no deseados que modifiquen la información de los atributos cuando únicamente queríamos consultarlos. Para asegurar este acceso correcto a lo largo de los procesos tanto de instanciación de los objetos como de modificación y consulta de la información contenida en estos definiremos tres tipos de métodos incluidos obligatoriamente en todas las clases definidas dentro del proyecto:

- **«Builders»:** funciones constructoras del objeto que se encargan de instanciar la clase concreta a la que pertenecen. Podremos instanciar los objetos de dos maneras diferentes, con sus dos constructores asociados a definir en cada clase incluida en el proyecto:
 - **Instancia vacía:** constructor sin atributos de entrada llamando únicamente a la clase superior a través del método «super()». Creará una instancia con referencia asociada pero atributos de valor nulo. Esta instancia, una vez creada a través del constructor vacío, podrá ver modificada su información posteriormente si es accedida correctamente.
 - **Instancia con información:** constructor con atributos de información externa al objeto que crea una instancia con su identificador asociado y asigna la información requerida todos o un número determinado de los atributos a través de la información de entrada del constructor. Todos o parte de los atributos tras la instanciación tendrán asignada información y no serán nulos. Igualmente, su información podrá ser modificada posteriormente si se accede adecuadamente.

Serán definidos como tipo público para poder ser accedidos desde el cuerpo principal del programa o desde otros métodos. Esto asegurará la correcta instanciación de los objetos correspondientes a esta clase.

- **«Setters»:** funciones que asignan el valor a un atributo privado concreto dentro de la clase. El valor de entrada de esta función podrá ser asignado como valor al atributo privado de la clase en cuestión por estar definido este método «Setter» dentro de la misma clase que este, permitiendo su acceso y modificación aún a través del modificador «private». Esto asegurará la correcta modificación de la información.
 - **«Getters»:** funciones que consultan y devuelven el valor del atributo privado de la clase en cuestión por las mismas razones expresadas en el anterior punto. Se asegura de devolver el valor sin modificarlo, por lo que asegurará la correcta consulta de la información.
- **Polimorfismo:** podremos definir métodos con el mismo nombre incluso dentro de la misma clase siempre y cuando los atributos sean diferentes, generando funciones diferentes con un identificador de nombre común. Esto es útil cuando pretendamos definir acciones comunes para variables diferentes, representando esa acción de manera conceptual a través de un mismo identificador. Aporta claridad al código para no depender de un gran número de nombres diferentes para identificar procesos comunes.

- **Abstracción:** en ciertas ocasiones queremos utilizar métodos que no implique necesariamente controlar el acceso, consulta y modificación de los atributos de una clase, sino que pretenderemos que cumplan otro tipo de funciones que no requieran de la instanciación de una clase para crear una instancia de objeto concreta. Para ello definiremos los métodos abstractos, que podrán ser accedidos directamente a través del identificador de la clase a la que pertenecen sin necesidad de instanciar un objeto previamente. La clase no necesita ningún cambio en el modificador de acceso a esta para definir un método abstracto dentro de esta, pudiendo incluso mezclar métodos abstractos y no abstractos dentro de una determinada clase, incluso acompañados de atributos.

Será a lo largo de la sección 3.3 cuando, entre otras cosas, explicaremos las implicaciones de este paradigma en la reescritura del código del organizador de horarios en Cplex/Java.

3.2. Planteamiento del algoritmo.

En primer lugar, comenzaremos la sección comentando el algoritmo usado en el artículo referenciado [3]. Este se basa en la obtención iterada de un número determinado de soluciones del problema de docencia, para posteriormente comprobar una a una cada una de estas soluciones en el problema de horarios para comprobar si alguna da lugar a un problema factible y por lo tanto nos permite describir la solución global al problema como las soluciones a ambas subetapas de este. Aparte de los parámetros de asignación mínima de disponibilidad y de tutorización mínima, objetivos representados respectivamente por a_t y g_t , definiremos el límite de iteraciones $nIter$ del algoritmo, marcando el número máximo de soluciones de docencia que comprobaremos en el problema de horarios antes de considerar que un determinado modelo no es factible¹. Por lo, dados los valores concretos de control de calidad a_t y g_t así como el límite máximo de soluciones a comprobar $nIter$ el esquema del anterior algoritmo descrito en pseudocódigo sería:

- Generamos el modelo del problema de docencia para los valores de parámetros de control de calidad determinados a_t y g_t así como el modelo del problema de lecciones.
- Obtenemos $nIter$ soluciones al problema de docencia generado, cada una de estas identificada como Z_i^* , $\forall i \in (1, nIter)$.
- Bucle iterativo con límite máximo de iteraciones, $\forall i \in (1, nIter)$:
 - Comprobamos la factibilidad del problema de lecciones generado a partir de la solución de referencia correspondiente Z_i^* al problema de docencia generada como información de entrada para esta segunda parte, hasta que se cumpla una de las dos siguientes condiciones:
 - En una iteración determinada $j \in (1, nIter)$ encontramos una solución X_j^* factible para el problema de lecciones. Detenemos el algoritmo considerando el problema global de organización de horarios como factible, siendo su solución el conjunto de las soluciones de ambas etapas: la solución del problema de docencia Z_j^* en la primera etapa que da lugar a un problema factible en la segunda etapa de lecciones junto a la propia solución X_j^* obtenida tras la comprobación exitosa de la factibilidad del citado problema de lecciones.
 - Alcanzamos el límite máximo de iteraciones $nIter$ sin que ninguna solución al problema de referencia de docencia genere un problema de horarios infactible, considerando a

¹Realmente, esto es una asunción por parte de la programación para agilizar la ejecución del algoritmo. Sin embargo, sin explorar todas las soluciones del árbol de docencia y emplearlas en la etapa de horarios para comprobar la factibilidad de todas ellas no podremos asegurar que el problema global sea infactible. Sin embargo, debido al carácter heurístico del algoritmo una vez comprobada la infactibilidad de un número determinado, marcando ese límite máximo por el citado parámetro de iteraciones máximas.

efectos prácticos el modelo con esa determinada exigencia en el control de calidad a través del ratio objetivo de asignación de profesores y tutores como infactible².

A partir del algoritmo previamente definido, expresaremos los cambios que llevaremos en este para adaptarlo a la resolución del modelo genérico planteado a lo largo de este Trabajo de Fin de Máster.

Como el modelo matemático del presente trabajo resulta notablemente menos pesado por la eliminación de las restricciones de sincronización de desdobles así como de lecciones de Religión y Valores incluidas en el artículo de referencia [3], podremos expresar un algoritmo más ambicioso que con la misma estrategia de partición del problema global en dos etapas explore una variedad mayor de diferentes modelos en función de nuestra intensidad de requerimientos de calidad dada por los parámetros a_t y g_t . Este algoritmo puede ser útil para obtener rápidamente soluciones que asignen unos porcentajes objetivo de lecciones respecto a la disponibilidad para los profesores y de lecciones dentro del grupo respecto al número total de lecciones asignadas para los tutores no excesivamente alto, siendo este modelo poco exigente pero fácilmente resoluble. Es de esperar que para estos problemas iniciales con bajo requerimientos sean resueltos en pocas iteraciones del algoritmo base, sin llegar a alcanzarse el límite máximo de soluciones a comprobar definido, con valor $nIter$. Esto nos proporcionará acceso rápido a soluciones del problema que podremos consultar a los pocos segundos de ejecutar el algoritmo para ir haciéndonos una idea de posibles asignaciones principalmente en el aspecto de docencia entre profesores y grupos, mientras soluciones de mayor calidad por presentar requerimientos más exigentes en sus parámetros de control a_t y g_t van siendo generadas. Será esperable que estas últimas soluciones con mayores ratios de asignación de profesores y tutores tomen más tiempo para ser halladas y que, llegado cierto punto, los requerimientos sean tan exigentes por el alto valor de los parámetros a_t y/o g_t que hagan el problema infactible, impidiendo obtener soluciones de mayor calidad a partir de determinado punto.

Por último y antes de definir el pseudocódigo del problema, comentaremos que este es un método heurístico, por lo que la comprobación de $nIter$ soluciones del problema de docencia a través del estudio de la factibilidad del problema de lecciones generado para cada pareja de valores de parámetros a_t y g_t definiendo diferentes niveles de exigencia en la calidad de la solución no asegura la condición de infactibilidad del problema global en el caso de que todas las comprobaciones sean negativas, ya que puede darse el caso de soluciones que requieran un número mayor de pasos para que la primera etapa genere una información de entrada para la segunda que haga esta última etapa factible. Uno esperaría que para valores menores de un parámetro fijado el otro, al ser los requerimientos de calidad de la sección 2.3 del tipo menor o igual, si el modelo de mayor exigencia (mayor valor de los parámetros) resulta factible el modelo de menor exigencia (menor valor de los parámetros) resulte igualmente factible. Sin embargo, como el método es heurístico y no tenemos gran control sobre el orden en que las soluciones al problema de docencia son generadas, no podremos asegurar que esa solución factible al problema de lecciones se alcance en un número menor o igual de etapas, pudiendo darse el caso de modelos que tardarían un mayor número de etapas en ser resueltos que el límite máximo definido $nIter$. De esta manera, aunque fuesen requerimientos menos exigentes que el modelo de mayor calidad que ha resultado factible, el carácter heurístico del algoritmo y en concreto el límite máximo de soluciones a comprobar $nIter$ impiden que hallemos una solución al problema global que en principio era más ligero de resolver, considerándolo como infactible según el criterio de nuestro algoritmo y pasando a una pareja nueva de valores de a_t y g_t . Como comprobaremos un gran número de modelos diferentes, esto no será una amenaza excesiva para los resultados obtenidos. Comprobaremos esto analizando el resultado de la ejecución del algoritmo programado en la sección 3.4.

Definiremos una serie de parejas de parámetros $(a_t^i, g_t^i), \forall i \in (1, nModel)$, siendo $nModel$ el número máximo de modelos a comprobar, cada uno de estos con un bucle iterativo de comprobación de $nIter$ soluciones del problema de docencia en el problema de lecciones, representando diferentes requeri-

²Como ya hemos comentado previamente, este es un método heurístico, por lo que sin comprobar la factibilidad todas las soluciones árbol del problema de docencia generado en el problema de horarios no podríamos inferir realmente la infactibilidad del problema.

mientos de calidad. Por lo tanto el esquema en pseudocódigo del algoritmo que proponemos a para este Trabajo de Fin de Máster será:

- Bucle iterativo recorriendo todas las parejas de parámetros (a_t^i, g_t^i) definidos previamente con límite máximo de modelos, $\forall j \in (1, nModel)$:
 - Generamos el modelo del problema de docencia para los valores de parámetros de control de calidad determinados a_t^i y g_t^i así como el modelo del problema de lecciones.
 - Obtenemos $nIter$ soluciones al problema de docencia generado, cada una de estas identificada como Z_j^i , $\forall j \in (1, nIter)$.
 - Bucle iterativo con límite máximo de iteraciones, $\forall j \in (1, nIter)$:
 - Comprobamos la factibilidad del problema de lecciones generado a partir de la solución de referencia correspondiente Z_j^i al problema de docencia generada como información de entrada para esta segunda parte, hasta que se cumpla una de las dos siguientes condiciones:
 - ◇ En una iteración determinada $k \in (1, nIter)$ encontramos una solución X_k^i factible para el problema de lecciones. Detenemos el algoritmo considerando el problema global de organización de horarios para la pareja de parámetros de exigencia de calidad (a_t^i, g_t^i) como factible, siendo su solución el conjunto de las soluciones de ambas etapas: la solución del problema de docencia Z_k^i en la primera etapa que da lugar a un problema factible en la segunda etapa de lecciones junto a la propia solución X_k^i obtenida tras la comprobación exitosa de la factibilidad del citado problema de lecciones.
 - ◇ Alcanzamos el límite máximo de iteraciones $nIter$ sin que ninguna solución al problema de referencia de docencia genere un problema de horarios infactible, considerando a efectos prácticos el modelo con esa determinada exigencia en el control de calidad (a_t^i, g_t^i) como infactible bajo las suposiciones de nuestro algoritmo.

Tendremos que implantar esta nueva estructura del algoritmo en el bucle principal del proyecto, contenido en el código principal contenido en la clase «Main.java» del paquete «control». Resultará bastante sencillo, ya que además de la estructura de resolución de docencia y comprobación en horarios con un límite máximo de iteraciones desarrollada previamente en el artículo [3] deberemos implantar un bucle externo que varíe los valores de ambos parámetros de control a_t y g_t para definir diferentes modelos matemáticos cuya factibilidad comprobaremos a través del algoritmo definido en el citado artículo [3]. Para cada pareja de valores de parámetros pasaremos a obtener ese límite máximo de soluciones para el problema concreto de docencia definido por sus valores de parámetros de control a_t y g_t y comprobarla la factibilidad del problema de docencia para las todas las soluciones generadas en la primera etapa hasta que encontremos una que resulte factible o se alcance el límite máximo, punto en el que consideraríamos que el modelo concreto para la pareja de valores determinada es infactible. Al definir el código de manera modular y bajo los preceptos de la POO, resultará más sencillo implantar el bucle con corrección sin preocuparnos por problemas como el llenado de la memoria de Java por mala gestión en las instanciaciones.

Por último, llevaremos a cabo un comentario acerca de nuestro criterio de infactibilidad para un modelo, extendiendo lo comentado anteriormente. Como el carácter del algoritmo es metaheurístico, considerar el problema global como infactible para determinados valores de a_t y g_t tras $nIter$ comprobaciones fallidas de factibilidad en la etapa de lecciones a partir de la información de referencia de la solución de la etapa de horarios será una asunción correcta para agilizar el algoritmo de exploración de modelos. Sin embargo, este criterio no tendrá por qué corresponder exactamente con la factibilidad real del problema global, ya que algunas de las parejas de valores de parámetros (a_t^i, g_t^i) darán lugar a problemas factibles para los que necesitaríamos comprobar un número de soluciones mayor que el límite máximo definido $nIterMax$. Sin embargo, asumir la infactibilidad en ese caso nos permitirá una

exploración mucho más veloz del espacio de soluciones del problema del centro de Educación Primaria concreto, obteniendo soluciones tempranas con requerimientos menos exigentes con gran velocidad para ir haciéndonos a la idea de estas mientras otras soluciones con más exigencia a la calidad de la docencia de profesores y tutores van siendo generadas.

3.3. Programación del algoritmo. Aplicación de POO.

En primer lugar, comentaremos los cambios referidos a las adaptaciones del código al paradigma de programación orientado a objetos, refiriéndonos a ejemplos concretos implantados en el código del proyecto de Cplex/Java adjunto a la memoria de este Trabajo de Fin de Máster.

- Se definen todos los objetos principales, tanto del modelo matemático como de generación y resolución de modelos en Cplex, en clases determinadas en función de los pretextos del paradigma POO que definiremos en los siguientes puntos de esta lista de cambios. Anteriormente índices matemáticos como los días o las sesiones eran almacenados como objetos de tipo primitivo «String». Esto no permite un acceso controlado a los datos, por lo que en la reescritura del código se incluyen definiciones de un día o sesión concreto, a partir de sus clases respectivas «Day.java» y «Session.java», así como sus objetos de array para englobar todas los días o todas las sesiones, a partir de sus clases correspondientes «DayArray.java» y «SessionArray.java».
- Definiremos todos los atributos dentro de las clases del proyecto como privados, mientras que sus métodos de construcción («builders»), modificación («setters»), consulta («getters») y definidos por el usuario como públicos, para que puedan ser accedidos desde el resto del programa y no queden aislados. Antes los atributos eran accedidos directamente tanto con el objetivo de consulta como de modificación de estos, corriendo gran riesgo de cambiar sus valores de manera indeseada frente a cualquier ligero cambio en el código. Este riesgo se agranda más todavía por la presencia de un doble bucle iterativo para el algoritmo de resolución del modelo. Con esta nueva implantación, el acceso será controlado. Ofreceremos el ejemplo de una de las clases más sencillas definidas en el proyecto, por carácter resumido y explicativo de los puntos a observar. Ofrecemos el ejemplo de la clase «Parameters.java», dentro del paquete «constants».

```
package constants;

//+++++++IMPORTING+++++++
//-----READING-----
//Setting path to input file
import java.io.File;
//Setting scanner to string read
import java.util.Scanner;

//+++++++MAIN BODY+++++++
public class Parameters {

//-----PRIVATE FIELDS-----
//Number of professors
private int nTeachers;
//Number of matters
private int nMatters;
//Number of grades
private int nGrades;
//Number of letters by grade
private int nLetters;
```



```

//Number of days
private int nDays;
//number of sessions by day
private int nSessions;

//-----BUILDERS-----
//Global
public Parameters(int nTeachers, int nMatters,
int nGrades, int nLetters, int nDays,
int nSessions) {
super();
this.nTeachers = nTeachers;
this.nMatters = nMatters;
this.nGrades = nGrades;
this.nLetters = nLetters;
this.nDays = nDays;
this.nSessions = nSessions;
}
//From aux object read
public Parameters(String[] auxread) {
super();
this.nTeachers = Integer.parseInt(auxread[0]);
this.nMatters = Integer.parseInt(auxread[1]);
this.nGrades = Integer.parseInt(auxread[2]);
this.nLetters = Integer.parseInt(auxread[3]);
this.nDays = Integer.parseInt(auxread[4]);
this.nSessions = Integer.parseInt(auxread[5]);
}
//Empty
public Parameters() {
super();
}

//-----SETTERS & GETTERS-----
public int getnTeachers() {
return nTeachers;
}
public void setnTeachers(int nTeachers) {
this.nTeachers = nTeachers;
}
public int getnMatters() {
return nMatters;
}
public void setnMatters(int nMatters) {
this.nMatters = nMatters;
}
public int getnGrades() {
return nGrades;
}
public void setnGrades(int nGrades) {
this.nGrades = nGrades;
}

```

```

    }
    public int getnLetters() {
    return nLetters;
    }
    public void setnLetters(int nLetters) {
    this.nLetters = nLetters;
    }
    public int getnDays() {
    return nDays;
    }
    public void setnDays(int nDays) {
    this.nDays = nDays;
    }
    public int getnSessions() {
    return nSessions;
    }
    public void setnSessions(int nSessions) {
    this.nSessions = nSessions;
    }

    //-----READ & DISPLAY-----
    //Method type to read the parameters of the problem from
    //text input
    public void readParameters() throws Exception {

    //Variable declaration
    //Where the file can be found
    File path;
    //To read from text file
    Scanner sc;
    //Number of parameters to read
    int nParameters = 6;
    //Auxiliar to store strings read
    String auxread1;
    //Auxiliar to split strings read
    String[] auxread2 = new String[nParameters];
    //On screen information
    System.out.println("-----Reading parameters-----
    -----\n");

    //We define the path where the file is located, including its name
    path = new File(System.getProperty("user.dir")
    + "/input/parameters.csv");

    //We create the scanner to the file and load it from the path
    sc = new Scanner(path);

    //We read the header of the file
    auxread1 = sc.nextLine();

    //We store the line read

```

```

auxread1 = sc.nextLine();
//We split it between the parameters, splited by ";",
//in this order; number of professors
auxread2 = auxread1.split(";");

//We assign their values casting strings to integer
this.setnTeachers(Integer.parseInt(auxread2[0]));
this.setnMatters(Integer.parseInt(auxread2[1]));
this.setnGrades(Integer.parseInt(auxread2[2]));
this.setnLetters(Integer.parseInt(auxread2[3]));
this.setnDays(Integer.parseInt(auxread2[4]));
this.setnSessions(Integer.parseInt(auxread2[5]));

//Finally, we close the scanner
sc.close();

//On screen information
System.out.println("Parameters read\n");
}

//Method type to display the parameters of the problem
public void displayParameters() throws Exception {

//On screen information
System.out.println("Displaying parameters\n");

//Displaying information
System.out.println("Parameters:\n\n Number of teachers = " +
this.getnTeachers() +
"\n Number of matters = " + this.getnMatters() +
"\n Number of grades = " + this.getnGrades() +
"\n Number of letters = " + this.getnLetters() +
"\n Number of days = " + this.getnDays() +
"\n Number of sessions = " + this.getnSessions() + "\n\n");
}
}

```

Podemos comprobar el primer bloque de campos privados

- En este sentido a cada uno de sus clases se le asignan los métodos relacionados con esta que operan sobre sus atributos, de manera que la estructura del código resulte más natural. Un ejemplo de esto es la función «readParameters()» definida para la clase «Parameters» que acabemos de reflejar en la memoria.
- Utilización de polimorfismo en aquellas funciones que representen funciones similares. En este caso, dentro de la clase «Professor.java» al instanciar la clase de este tipo para profesores dentro del código principal del proyecto comenzaremos por leer únicamente la información de los profesores incluida en el archivo «professors.csv» de la carpeta «input» del proyecto, dejando los atributos de cualificación para enseñar materias y disponibilidad de momento nulos hasta la posterior asignación de estos a través de la lectura de los archivos de entrada «qualifications.csv» y «availabilities.csv» respectivamente. De esta forma, tendremos que definir tres constructores:
 - Vacío: crea instancia con atributos nulos y no necesita información de entrada.

```
public Professor() {
    super();
}
```

- Constructor completo a partir de todos los atributos que contiene la clase, asignando cada entrada recibida a cada uno de los atributos de la clase.

```
public Professor(String teacher, int gTeacher, double availability,
    int isTutor, String grade, String letter,
    int tGrade, int tLetter, int isAvailableRecess,
    int[] canTeachMatter, int[][] isAvailable) {
    super();
    this.teacher = teacher;
    this.gTeacher = gTeacher;
    this.availability = availability;
    this.isTutor = isTutor;
    this.grade = grade;
    this.letter = letter;
    this.tGrade = tGrade;
    this.tLetter = tLetter;
    this.isAvailableRecess = isAvailableRecess;
    this.canTeachMatter = canTeachMatter;
    this.isAvailable = isAvailable;
}
```

- Constructor parcial a través de los atributos cuya información es contenida en el archivo de entrada «professors.csv». Genera instancia de la clase con todos los atributos asignados a partir de la información de entrada del constructor salvo aquellos correspondientes a la cualificación de los profesores para enseñar materias y su disponibilidad en huecos.

```
public Professor(String teacher, int gTeacher, double availability,
    int isTutor, String grade, String letter,
    int tGrade, int tLetter, int isAvailableRecess) {
    super();
    this.teacher = teacher;
    this.gTeacher = gTeacher;
    this.availability = availability;
    this.isTutor = isTutor;
    this.grade = grade;
    this.letter = letter;
    this.tGrade = tGrade;
    this.tLetter = tLetter;
    this.isAvailableRecess = isAvailableRecess;
}
```

Observamos como hemos podido definir estos tres constructores de tal manera que cumplan funciones diferentes pero tengan un mismo nombre identificador, clarificando su función dentro del código. Esto se consigue a través del polimorfismo, a partir de la diferencia entre argumentos de los diferentes constructores, requiriendo únicamente la información que va a asignar a la instancia de la clase creada.

- Abstracción en los métodos de escritura por pantalla y a archivo de texto. Tanto la comprobación de la corrección del modelo por pantalla como la escritura de la solución del organizador de horarios creado a partir de las librerías de Cplex necesitan de entradas correspondientes a multitud de objetos. Por lo tanto, no definiremos estos métodos dentro de ninguna clase concreta, sino que

crearemos clases aparte para la escritura por pantalla y a archivo de texto denominadas respectivamente «Display.java» y «Write.java» dentro del paquete «displayWrite». Dentro de estas clases definidas de manera usual incluiremos los métodos con modificador «abstract» que permitirán su posterior acceso sin instanciar la clase. Pondremos un ejemplo con el código resumido (partes no reflejadas en la memoria identificadas con puntos suspensivos) para la clase de escritura contenida en «Write.java».

```
package displayWrite;

//Importing related classes
//For File and FileOutputStream classes
import java.io.*;

import constants.*;
import constantsArray.VarDocency;
import constantsArray.VarTimetable;

public class Write {

    public static void writeDocencySolution(double[] z, VarDocency var,
        String mode, Professor[] professor, Group[][] group,
        Subject[][][] subject, int nTeachers, int nMatters,
        int nGrades, int nLetters, double availabilityTarget,
        double groupTarget, double time,
        int aux) throws IOException {

        //VARIABLE DECLARATION
        //Path to null archive
        File path;
        //Pointer to write
        FileOutputStream fos;
        //To transform integer variables from double
        int[] auxz;

        ...

        //We close the pointer to the output file
        fos.close();
    }

    public static void writeTimetableSolution(double[] x,
        VarTimetable var, String mode, Professor[] professor,
        Group[][] group, Subject[][][] subject, Slot[][] slot,
        int nTeachers, int nMatters, int nGrades, int nLetters,
        int nDays, int nSessions, double availabilityTarget,
        double groupTarget, double time, int aux) throws IOException {

        //VARIABLE DECLARATION
        //Path to null archive
        File path;
        //Pointer to write
        FileOutputStream fos;
```

```

        //To transform integer variables from double
int[] auxx;

...

        //We close the pointer to the output file
        fos.close();
    }
}

```

Los citados modificadores «abstract» permiten acceder posteriormente a estos métodos a través del identificador de la clase sin instanciar ningún objeto concreto. Observemos cómo ocurre esto en el siguiente fragmento de código de la función «Main.java»:

```

//Writing docency to output file
Write.writeDocencySolution(problem0.getSolutionMult().getV()[i],
model0.getVarDocency(), "csv", professors.getProfessors(),
groups.getGroups(), subjects.getSubjects(),
teachers.getnTeachers(), matters.getnMatters(),
grades.getnGrades(), letters.getnLetters(),
availabilityTarget, groupTarget, time, aux);

//Writing timetable to output file
Write.writeTimetableSolution(problem.getSolution().getV(),
model.getVarTimetable(), "csv", professors.getProfessors(),
groups.getGroups(), subjects.getSubjects(), slots.getSlots(),
teachers.getnTeachers(), matters.getnMatters(),
grades.getnGrades(), letters.getnLetters(), days.getnDays(),
sessions.getnSessions(), availabilityTarget, groupTarget, time, aux);

```

- Reutilización de instancias. En fase de desarrollo del código del artículo referenciado [3] se encontraron grandes problemas a la hora de gestionar las instancias en el bucle, consumiendo la memoria reservada de la CPU por parte de Java y dando problemas de tipo «overheap» al sobrepasarse, deteniendo la ejecución del problema e impidiendo la generación asociada de una solución para el problema de organización de horarios. Esto se solucionó generando instancias desde el cuerpo principal del código y pasándolas una y otra vez a los métodos dentro del bucle iterativo como argumentos. Sin embargo, desde el punto de vista de la programación esta es una solución pobre para salir del paso, por lo que en esta reescritura del código llevada a cabo entre otros contenidos a lo largo del presente Trabajo de Fin de Máster nos hemos asegurado de una correcta utilización de las instancias bajo el paradigma de la programación orientada a objetos.

En segundo lugar, comentaremos los cambios generales en la estructura de la programación, no vinculados intrínsecamente con el paradigma de programación orientado a objetos citado anteriormente.

- Estableceremos una estructura de las clases por paquetes en base a la relación de las funciones que desempeñan estas dentro del código global. Comentaremos los paquetes y sus clases contenidas:
 - Paquete «constants». Contendrá varios tipos de clases:
 - Aquellas que contienen la definición de las constantes matemáticas de los índices principales definidos en la sección 2.3 a su nivel más fundamental, representando por ejemplo un único día, una cierta sesión, un determinado grupo o una asignatura en concreto. Será el caso de las clases «Day.java», «Grade.java», «Group.java», «Letter.java», «Matter.java», «Parameters.java», «Professor.java», «Session.java», «Slot.java», «Subject.java» y

- Aquellas que contienen la definición de objetos usados por Cplex para construir las variables del modelo, como «Var.java» o aquellas utilizadas para almacenar soluciones únicas o múltiples del problema, respectivamente «Solution.java» y «Solution-Mult.java».
- Paquete «constantsArray». Contendrá varios tipos de clases:
 - Aquellas dedicadas a almacenar los objetos compuestos creados a partir de los índices principales definidos para el problema matemático. Las clases «DayArray.java», «GradeArray.java», «GroupArray.java», «LetterArray.java», «MatterArray.java», «ProfessorArray.java», «SessionArray.java», «SlotArray.java», «SubjectArray.java» serán las representantes de este tipo.
 - Aquellas dedicadas a la creación de objetos usados por Cplex para la creación de variables del modelo, en concreto «VarDocency.java» y «VarTimetable.java».
- Paquete «control». Contiene la clase «Main.java», correspondiente a la clase principal del programa que se encarga de la gestión general de la ejecución del proyecto. Será comentado más adelante.
- Paquete «displayWrite». Contiene las clases «Display.java» y «Write.java», con métodos para comprobar por pantalla la validez de la solución generada por el organizador en el primer caso y para escribir por salida de archivo de texto las soluciones de docencia y lecciones establecidas en el segundo.
- Paquete «model». Contiene dos tipos de clases:
 - Aquellas que definen las matrices del problema entero con variables binarias de ambas subetapas del problema matemático como objetos de Cplex para posibilitar su resolución posterior, en concreto «ModelDocency.java» y «ModelTimetable.java».
 - Aquellas que definen los objetos en Cplex para crear y resolver el problema entero en variables binarias para cada iteración del bucle, siendo el caso de la clase «Problem.java».

Esto nos servirá en primer lugar para clarificar el funcionamiento del código, haciéndolo más comprensible de cara a otros desarrolladores que puedan querer modificar el código adjunto a la memoria. Además, mejora la modularidad de la implantación del modelo, siendo los elementos más representativos de sus funciones concretas, lo que facilita enormemente futuras evoluciones del código por cambios en el modelo matemático o de resolución algorítmico añadiendo nuevas clases que representen comportamientos no tenidos en cuenta anteriormente.

- El formato de entrada de los archivos que contienen la información de cualificación de docencia de materias y disponibilidad de horario de los profesores, respectivamente «qualifications.csv» y «availabilities.csv», pasará a ser plano. La información por lo tanto se encontrará ofrecida línea a línea para el nivel máximo de detalle del objeto, como ocurre con el resto de archivos de entrada de extensión «.csv» contenidos en la carpeta «input» del proyecto. En este sentido, el archivo de cualificaciones pasará a tener un registro por cada profesor y materia del modelo. Por otra parte, el archivo de disponibilidades tendrá en este nuevo proyecto un registro para cada profesor y hueco del horario, implicando este último cada día y sesión. Veamos un ejemplo de esto con el inicio del archivo de entrada de «availabilities.csv»:

```
teacherAvailabilities;dayAvailabilities...
PR1_1;L;1a;1
PR1_1;L;2a;1
PR1_1;L;3a;1
PR1_1;L;4a;1
PR1_1;L;5a;1
```

```

PR1_1;L;6a;1
PR1_1;M;1a;1
PR1_1;M;2a;1
...
```

- Programación de las nuevas restricciones del modelo en las clases «ModelDocency.java» y «ModelTimetable.java» para las etapas de docencia y lecciones respectivamente, así como implantación del algoritmo de resolución en el programa principal del proyecto.

En resumen, tendremos un código mejor estructurado, permitiendo una explicación de la ejecución del código más clara. En este sentido, explicaremos las etapas principales del código principal contenido en la clase «Main.java», describiendo como crea y se relaciona con las clases anteriormente definidas.

3.4. Resultados.

Antes de comentar propiamente los resultados de la ejecución del algoritmo para el modelo concreto de centro descrito a lo largo de este Trabajo de Fin de Máster haremos una descripción de la ejecución de este algoritmo. Desglosaremos paso a paso los procesos que lleva a cabo caracterizando las clases implicadas en este proceso y dando algunos detalles técnicos de relevancia.

- 1. Lectura de la información de entrada del algoritmo. Leeremos todos los archivos de texto en formato «.csv» que se encuentran en la carpeta «input» del proyecto. Para ello, leeremos en primer lugar la información de los objetos simples instanciados a partir de clases dentro del paquete «constants» ejecutando su método de lectura asociado, definido dentro de la propia clase. Un ejemplo de esto podrá ser la lectura de la información del archivo de texto «parameters.csv» y su almacenado dentro del objeto «parameters» de la clase «Parameters» a través del método «readParameters(...)» definido dentro de esta propia clase. En segundo lugar, leeremos la información de los objetos compuestos instanciados a partir de clases dentro del paquete «constantsArray» ejecutando su método de lectura asociado, definido dentro de la propia clase. En este caso, el ejemplo podría ser la lectura de la información del archivo de texto «subjects.csv» y su almacenado dentro del objeto «subjects» de la clase «Subjects» a través del método «readSubjects(...)» definido dentro de esta propia clase.
- 2. Para cada pareja de valores de control de calidad a_t y g_t estudiados en la ejecución del proyecto, construiremos los objetos que representan el problema de restricciones lineales en variables binarias a partir de la información de entrada leída en el paso 1, tanto para el modelo de docencia de la primera etapa del problema global como para el modelo de lecciones de la segunda. En este punto comenzaremos a instanciar objetos a partir de clases definidas en Cplex. Generaremos el modelo de docencia guardándolo en un objeto llamado «model0» de la clase «modelDocency» a través de la ejecución en este orden de las funciones «buildVarDocency», «buildnVar», «buildnRestr» y «buildCplexObjects». Por otra parte, el modelo de lecciones será guardado en un objeto llamado «model» de la clase «modelTimetable» a través de la ejecución en este orden de las funciones «buildVarDocency», «buildnVar», «buildnRestr» y «buildCplexObjects».
- 3. Resolvemos el problema global para esos valores de calidad a_t y g_t . Para ello:
 - 3.1. Crearemos los problemas de las etapas de docencia y lecciones almacenándolos en los objetos de nombre «problem0» y «problem» respectivamente, ambos de la clase «Problem» a partir de la función «startProblem(...)». De nuevo, estas instanciaciones y asignaciones de valores requerirán del uso de clases de la librería de Cplex para Java.
 - 3.2. Crearemos $nIter$ soluciones al problema de docencia a partir de la función «populateProblem(...)» definida dentro de la propia clase «Problem». Sus soluciones se almacenarán dentro de su atributo «solutionMult» de la clase «SolutionMult» para poder ser accedidas posteriormente.

- 3.3. Modificaremos el problema de lecciones contenido en el objeto «problem» a través del método «changeAb(...)» y a partir la información de referencia obtenida como solución del problema de docencia. Esta información se encuentra almacenada en el objeto «problem», y podremos acceder a ella a través de la concatenación de métodos «SolutionMult().getV()[i]», donde i representa el índice de la iteración en Java³. Una vez tenemos el problema de lecciones correcto en el objeto «problem» trataremos de encontrar una solución factible a la etapa de lecciones a través del método «solveProblem». Comprobaremos si el problema es factible, aspecto indicado por el campo «status» dentro del atributo «solution» del objeto «problem» correspondiente al problema de lecciones. Podremos consultar esta información a través de la concatenación de métodos «getSolution().getStatus()».
 - Si el problema de lecciones es factible, detendremos el bucle iterativo. Habremos encontrado una solución al problema global, terminando el algoritmo metaheurístico y finalizando el proyecto del organización de horarios con la escritura a archivo de texto de las soluciones generadas para docencia y lecciones respectivamente a partir de los métodos «writeDocencySolution(...)» y «writeTimetableSolution(...)» en la carpeta «output» dentro del directorio raíz del proyecto.
 - En caso de que el problema de lecciones no sea factible, volveremos al punto 3.3. comprobando una nueva solución del problema de docencia hasta que una de estas soluciones generadas anteriormente hagan factible al problema de lecciones, deteniendo el algoritmo al encontrar una solución factible para el problema global, o hasta que se alcance el límite máximo de iteraciones por modelo, considerando según nuestro criterio que el modelo de colegio para esos requerimientos de calidad no resulta factible.

Para más información acerca de la ejecución del algoritmo, puede consultarse el archivo de registro asociado a esta. Este archivo tiene como nombre «log.txt» y se encuentra en la carpeta «output» dentro del directorio raíz del proyecto, y contiene la salida por pantalla de la consola al ejecutar el proyecto en Java/Cplex volcada en un archivo de texto.

En el código adjunto a esta memoria, programaremos y resolveremos el algoritmo propuesto para los siguientes valores de las constantes que lo defines:

- Parámetros de requerimientos de la calidad de profesores y tutores. Comprobaremos todos los modelos correspondientes a combinaciones de parámetros ambos desde 0,5 de valor inicial y con una variación de 0,05 para cada uno de estos parámetros de manera independiente, hasta alcanzar para ambos el ratio máximo de 1,0. Esto implicará comprobar modelos desde un 50 % de requerimiento en la asignación de lecciones de profesores y tutores hasta un 100 % de asignación total. Matemáticamente, podremos reflejar estas combinaciones como (a_t^i, g_t^j) , $\forall i \in (0, 10)$, $\forall j \in (0, 10)$ | $a_t^i = 0,5 + 0,05 \cdot i$ y $g_t^j = 0,5 + 0,05 \cdot j$.
- Número de iteraciones máximas $nIter$ a comprobar por modelo de valores concretos de a_t^i y g_t^j de valor 1500. Pasadas esas 1500 iteraciones consideraremos que los requerimientos de calidad a la solución hacen al modelo global infactible como criterio de parada de nuestro algoritmo metaheurístico.

La salida de este algoritmo genera un gran número de ficheros de solución tanto para docencia establecida como para las lecciones organizadas, generando un archivo por pareja de valores de parámetros que marcan unos requerimientos de calidad determinados. Como estos archivos son planos y contienen la información fila a fila, registro a registro, su análisis directo no dará gran cantidad de información. Para observar una solución completa generada por el proyecto en Cplex/Java podremos utilizar el libro adjunto de Tableau. En el estado en el que se adjunta junto con esta memoria, el libro de Tableau muestra una solución para el valor $nIter = 1500$ y los valores concretos $(a_t, g_t) = (0,75, 0,7)$. Por lo

³Recordemos que en Java el primer índice corresponde al 0

tanto, a lo largo de esta sección 3.4 no analizaremos las soluciones generadas una a una, sino que nos encargaremos de caracterizar el resultado de cada una de estas iteraciones del algoritmo metaheurístico en función de los diferentes requerimientos de calidad establecidos.

- En caso de que se haya encontrado una solución factible al problema global antes de alcanzar el límite máximo de $nIter$ iteraciones para ese modelo concreto de valores a_t y g_t , incluiremos el número de iteraciones que le ha costado al algoritmo converger y encontrar una solución adecuada al problema global así como el tiempo que le ha costado encontrarla.
- En caso de que no se haya encontrado una solución factible al problema global alcanzado el límite máximo de $nIter$ iteraciones para ese modelo concreto de valores a_t y g_t , mostraremos un símbolo **X**.

a_t / g_t	0,5	0,55	0,6	0,65	0,7	$\geq 0,75$
0,5	369 i, 62,529 s	772 i, 106,212 s	17 i, 5,308 s	3 i, 3,703 s	2 i, 2,9559 s	X
0,55	717 i, 100,754 s	1266 i, 171,433 s	6 i, 4,169 s	743 i, 101,199 s	X	X
0,6	X	711 i, 98,671 s	9 i, 3,811 s	82 i, 13,432 s	51 i, 9,870 s	X
0,65	X	31 i, 7,632 s	5 i, 3,579 s	1047 i, 143,223 s	38 i, 7,445 s	X
0,7	X	X	X	201 i, 29,530 s	1 i, 2,883 s	X
0,75	150 i, 23,079 sec	2 i, 3,486 s	X	X	6 i, 3,952 s	X
$\geq 0,8$	X	X	X	X	X	X

Observamos como se producen los resultados «extraños» pero esperados descritos anteriormente de considerar modelos con menor intensidad de requerimientos infactibles pasado el límite de iteraciones y sin embargo encontrar modelos con requerimientos más intensos que resultan factibles dentro del número máximo de iteraciones definidas. Esto es esperable debido al carácter metaheurístico del algoritmo. Notable resulta el caso de la solución hallada para el problema con valores de parámetros $a_t = 0,75$ y $g_t = 0,7$, tras únicamente 6 iteraciones del algoritmo base y un tiempo asociado de resolución de 3,952s. Sin embargo, el anterior modelo factible que encontramos para ese mismo valor de parámetro de $a_t = 0,75$ es el correspondiente a $g_t = 0,55$, sin encontrar soluciones factibles para los modelos ($a_t = 0,75, g_t = 0,6$) ni ($a_t = 0,75, g_t = 0,65$) antes de alcanzar el número máximo de 1500 comprobaciones. Tal y como hemos comentado, esto es consecuencia del carácter heurístico del algoritmo y de la asunción de un límite de comprobaciones en el bucle base del algoritmo antes de considerar que unos determinados requerimientos de calidad en la solución hacen al problema infactible.

Capítulo 4

Visualización de Horarios: Tableau

Analizaremos a lo largo de este capítulo la herramienta de análisis visual de horarios desarrollada a partir del software de Tableau. Esta visualización pretende proporcionar al usuario del organizador de horarios una herramienta para poder analizar las diferentes soluciones obtenidas a partir del organizador de horarios que hemos tratado a lo largo del capítulo 3.

Cada vez que una solución es generada en Cplex/Java no resulta sencillo visualizarla directamente desde la salida de texto que se obtiene como salida de este proyecto. Esta solución en forma de archivo de texto obtenida tras la organización del horario se encuentra agregada al mínimo nivel, reflejando cada una de las asignaciones que tienen lugar entre profesores, asignaturas y huecos. Esto implica un número de registros lo suficientemente grande para que no sea posible vincularlos entre sí a simple vista y reconstruir el horario completo sin un análisis más profundo. Por lo tanto, esto haría del proceso manual de elección de la solución deseada entre todas las generadas un proceso tedioso que puede echar atrás a muchos usuarios a la hora de usar la herramienta. La automatización de la creación de un horario en formato convencional, reflejando la asignación de cada profesor/grupo (en función de la visualización deseada) en cada hueco por días y sesiones puede aliviar esta carga de tarea, pudiendo visualizarse en pantalla a través de Tableau la solución de organización de horarios obtenida al minuto de ser generada por el proyecto en Cplex/Java.

Además de las visualizaciones estándares de horarios de profesores y grupos, así como de los correspondientes a tutores y grupos tutorizados, incluiremos visualizaciones que se dediquen a analizar la solución obtenida para cada uno de los grupos y profesores, caracterizando las lecciones asignadas por diversas categorías. Por último, también incluiremos una serie de visualizaciones de carácter más auxiliar que se encarguen de comprobar la corrección de las docencias y lecciones asignadas en función de aspectos como la capacitación de los profesores para enseñar materias o su disponibilidad semanal en huecos según lo expuesto en la sección 2.1, así como que se cumplen las reglas requeridas como la no duplicación de huecos por grupo y profesor según lo expuesto en la sección 2.2.

En general, dividiremos este capítulo en cuatro secciones. En primer lugar, analizaremos los aspectos concretos de los horarios organizados que queremos analizar a lo largo del libro de Tableau. Seguiremos por analizar la estructura de los archivos de entrada de Tableau, obtenidos como salida tras la ejecución del proyecto de organización de horarios en Cplex/Java. Describiremos los campos que componen los diversos ficheros y cómo uniremos las diversas fuentes de datos entre sí para generar un conjunto de registros preparado para el análisis que queremos llevar a cabo. Tras ello, pasaremos a explicar consideraciones generales del modelo a tener en cuenta al desarrollar cualquier visualización dentro de este libro de Tableau. Por último, llevaremos a cabo un análisis pestaña por pestaña del libro de Tableau para comentar qué se pretende mostrar en las visualizaciones presentes en cada una de ellas, cómo construirlas y, por último, comentar la navegación y utilización general del libro.

4.1. Introducción a Tableau y objetivos generales.

Comencemos por definir una serie de conceptos fundamentales que serán de utilidad a la hora de plantear la construcción genérica de cualquier libro de Tableau.

- **Dashboard:** objeto que engloba todas las visualizaciones así como las fuentes de datos a las que estos se conectan. Existen de dos tipos de libros a este respecto:
 - **Empaquetados:** la fuente de datos se encuentra incluida en el propio libro de Tableau como una extracción. En líneas generales, esta extracción es un archivo en formato «.hyper» que contiene los datos de la fuente de datos utilizados en el Dashboard con una estructura que facilita enormemente las consultas. Resulta recomendable cuando buscamos crear libros con fuentes de datos muy pesadas que puedan tardar mucho tiempo en reaccionar al interactuar con el usuario, aliviando sensiblemente el tiempo de consulta a través de la creación de una extracción a partir de la fuente de datos. Para disponer de datos actualizados, el procedimiento más empleado es refrescar estos extractos periódicamente, por lo que puede ser una buena manera de disponer de datos recientes sin sacrificar la velocidad de consulta y carga del Dashboard.
 - **No empaquetados:** la conexión a la fuente de datos se encuentra en vivo. Resulta recomendable cuando buscamos crear libros con fuentes de datos que requieran actualización constante por cambiar los datos de su fuente. Esto implica un mayor tiempo de consulta y por lo tanto una respuesta algo más lenta del Dashboard, a cambio de asegurar la actualización de los datos. Sin embargo, para fuentes de datos de tamaño reducido cuya consulta resulte ágil, como las que tratamos a lo largo de este 4, esto no supondrá problema.

La fuente que utilizaremos a lo largo del trabajo es de pequeño tamaño, por lo que la velocidad de consulta no supondrá un problema. Pero por encima de todo, no es solo que necesitemos datos actualizados del Dashboard tras cada ejecución, sino que para ello deberemos reemplazar físicamente la fuente de datos tras cada ejecución¹. Por lo tanto, queda claro que lo más adecuado en nuestro caso es construir un libro no empaquetado con conexión en vivo.

- **Pestañas:** cada una de las diferentes páginas de las que se compone el Dashboard. Generalmente, las visualizaciones que tratan de transmitir una información relacionada se agrupan en diferentes pestañas dentro de un Dashboard, de tal manera que cada una de estas pestañas analice un aspecto determinado de la fuente de datos. A su vez, la navegación entre pestañas puede ayudar a reforzar la completitud de la información que proporciona el Dashboard. Profundizaremos en esto al definir los objetivos de la visualización a desarrollar en la siguiente sección 4.2, pero de momento podremos adelantar que estableceremos botones de navegación en el Dashboard integrados con las visualizaciones.
- **Hoja:** visualización concreta, elemento base de representación de datos. Resulta el eje de cualquier Dashboard. Desarrollaremos diferentes visualizaciones en función de los requerimientos expresados en la siguiente sección 4.2. En líneas generales, incluiremos en nuestro libro tablas, diagramas de barras y valores numéricos directos.
- **Selectores:** ofreceremos selectores al usuario para poder interactuar con el Dashboard y controlar la información que refleja. Existen dos tipos diferentes: filtros y parámetros. De cara al usuario, su funcionamiento no tiene por qué presentar diferencias; sin embargo, su funcionamiento en el software de Tableau es completamente diferente. Describamos en líneas generales la función de cada uno:

¹Esto implica únicamente eliminar el archivo «timetable.csv» encontrado en la carpeta en la que se aloja el libro de Tableau que servía de fuente de datos tras la anterior ejecución del organizador de horarios y pegar el nuevo archivo «timetable.csv» obtenido tras la siguiente ejecución del proyecto en Cplex/Java

- **Filtros:** restringen los datos incluidos en ciertas visualizaciones. En nuestro caso, estableceremos filtros para los profesores y los grupos cuando pretendamos representar visualmente el horario de estos o cuando pretendamos analizarlo. Estableceremos los filtros a nivel de pestaña, afectando a todas las hojas incluidas en ellas. Como hemos comentado anteriormente, para dar una experiencia conjunta de exposición de datos, estableceremos un vínculo desde las pestañas referentes a visualizaciones de horarios, tanto de profesores como de grupos, a sus respectivas pestañas de detalle. En ese sentido, el filtro será común tanto a la pestaña de horarios como a la de detalle correspondiente en ambos casos, para que así al filtrar por un profesor o grupo en la primera hoja general y acceder posteriormente a su segunda hoja detallada se mantenga la selección de docente o grupo de alumnos. De la misma manera, cuando volvamos de la pestaña de detalles a la pestaña general a través del vínculo establecido, también podremos mantener la selección de esta primera.
 - **Parámetros:** elemento alternativo de selección por parte del usuario que no filtra los registros sobre la base de datos, sino que ofrece una entrada al libro de Tableau para que puedan programarse las acciones a ser tenidas en cuenta de manera más detallada que un filtro directo, pudiendo superar ciertas limitaciones que causaría este filtrado en la base de datos. Será útil cuando tratemos de relacionar horarios entre profesores y grupos tutorizados. Debido a la estructura de la base de datos, todos los registros contendrán no solo la información del profesor docente de la asignatura y del grupo que la recibe sino también la del tutor del grupo y la del grupo tutorizado por parte del profesor en caso de que este resulte ser un tutor. Por lo tanto, filtrar por un valor concreto en el campo de profesor incluirá únicamente aquellas lecciones en las que sea profesor, ya que actuará sobre todos los registros del campo. De esta manera, asignaturas sobre su grupo tutorizado en las que no sea docente quedarían excluidas con el filtrado directo. Para filtrar desde un campo de origen profesor a un campo de destino tutor deberemos utilizar una programación más detallada a través de un parámetro y un campo calculado dependiente del valor del parámetro y del campo de destino tutor. Será este campo calculado el que nos sirva para filtrar correctamente. Esto funciona de manera análoga para los grupos y las lecciones de sus tutores.
- **Dimensiones y métricas:** las columnas de la base de datos generaran los diferentes campos, con sus valores definidos por los diferentes registros de esta misma base de datos. Estos campos se dividirán en dos tipos:
- **Dimensiones:** se encargan de representar valores cualitativos. Por ello, serán de gran utilidad a la hora de representar, por ejemplo, el profesor o la materia de las lecciones contenidas en cada registro de la fuente de datos. Interpretaremos todos los campos de la base de datos como dimensiones a través de Tableau. Ilustraremos las dimensiones correspondientes a los campos de la fuente de datos en la subsección 4.2.
 - **Métricas:** miden valores numéricos y requieren de una agregación como, por ejemplo, una suma o un promedio. En nuestro caso, las métricas implicarán recuento de número de lecciones y similares, por lo que obtendremos la mayoría de métricas a través de conteos y conteos definidos en campos calculados de Tableau. Los números contenidos en la base de datos serán interpretados como dimensiones discretas, de tal manera que obtendremos sus métricas como campos calculados fijando determinado campo de agregación y el tipo de agregación a llevar a cabo. Ilustraremos ambos casos de obtención de métricas a partir de campos calculados en la subsección 4.3.
- **Campos calculados:** dimensiones o métricas que no proceden directamente de la fuente de datos, sino que son creados a través del software de Tableau a partir de determinadas dimensiones y métricas de la base de datos o a partir de otros campos calculados. Nos servirán para varios propósitos, como son:

- Calcular recuentos de lecciones para profesores y grupos, caracterizando por ciertos aspectos como la asignación entre tutores y grupos tutorizados, la sesión o el día en que se asignan.
- Obtener los valores de requerimientos para asignaturas y de disponibilidad para profesores. Como esta información se repetirá para cada una de las lecciones contenida en cada registro de la fuente de datos, deberemos de realizar un cálculo con un nivel de detalle sobre el campo apropiado y aplicar una agregación correcta para que se escoja el valor correctamente una única vez a través del campo calculado.
- Nivel de detalle: control detallado del nivel de agregación de un campo en Tableau. En nuestro caso, utilizaremos el método «FIXED», correspondiente a fijar una dimensión y agregar (sumando, promediando...) todos los registros correspondientes para cada uno de los diferentes valores que presente esa dimensión. En ese sentido, se parece a las estructuras del tipo «GROUP BY» empleadas en lenguajes de bases relacionales como el SQL. Nos servirá para fijar correctamente los recuentos en campos calculados de Tableau.
- Acciones: efectos sobre el Dashboard en función de la interacción del usuario pero a través de la interacción con otros elemento del Dashboard en lugar de ofrecer un selector como en el caso de los filtros o de los parámetros. En nuestro caso utilizaremos principalmente la acción de resaltado, que al poner el ratón sobre una unidad de una visualización concreta, como puede ser una celda de una tabla o una barra en un gráfico de filas o columnas, ilumina en otras visualizaciones y/o en la suya propia otras unidades que contengan los mismos valores de la dimensión de la celda seleccionada por el usuario. Además, aprovecharemos las acciones de cambio de pestaña para navegar entre estas a través de clicks sobre iconos integrados en el Dashboard con ese fin.

4.2. Archivos de entrada.

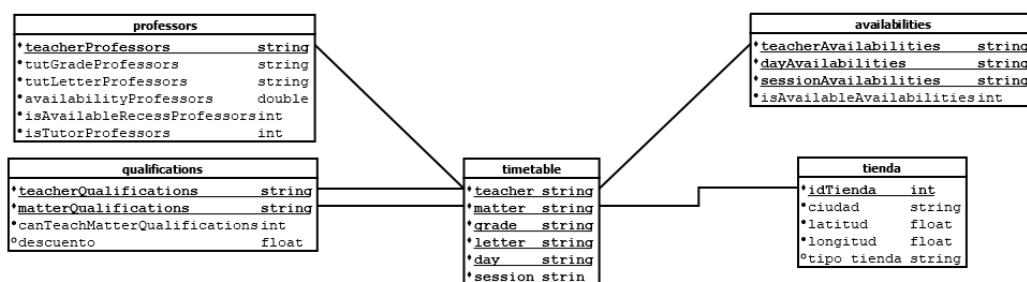
Hablaremos en esta sección de los archivos de texto a utilizar como fuentes de datos para Tableau, describiendo sus campos y cómo relacionar estas fuentes para crear la base de registros deseada. Todos las fuentes serán archivos en formato CSV, en los que los campos se encontrarán separados entre sí por el carácter punto y coma (;) y los registros por un salto de línea ({ }n). Pasaremos a analizar uno a uno los archivos utilizados, caracterizando su origen y la información que contiene.

- Archivo de referencia: «timetable.csv». Contiene todas las lecciones asignadas, libres y no disponibles si las analizamos por parte de los profesores. Como los grupos no tienen la posibilidad de encontrarse libres o no disponibles, excluyendo esos registros no asignados o sin disponibilidad obtendremos todas las lecciones asignadas para los grupos. Nos servirá como archivo base de la fuente de datos, entorno al que realizaremos la unión en estrella para generar los registros completos de lecciones. Será obtenido a partir de la salida del proyecto de Cplex/Java y será reemplazado por otro archivo de la misma estructura generado tras cada ejecución de este organizador de horarios. Cada una de las salidas obtenidas en el algoritmo dará lugar a un posible fichero a utilizar como fuente para el mismo modelo de centro escolar de Educación Primaria resuelto.
- Archivos auxiliares: contienen la información auxiliar de tutores, grupos y asignaturas del modelo de centro escolar estudiado. Son los mismos archivos empleados como parte de la entrada del proyecto de Cplex/Java, concretamente de la carpeta «input». Por lo tanto, serán comunes a todas las ejecuciones del algoritmo para un mismo modelo de centro de Educación Primaria estudiado, por lo que no tendrán por qué ser reemplazados en el estudio de cada solución obtenida por el organizador de horarios, pudiendo mantenerlos. Estos serán:
 - «professors.csv». Contiene la información del alias de los profesores, su carácter de tutores, su grupo tutorizado en caso afirmativo, su disponibilidad máxima semanal de lecciones y su

reserva de lecciones para vigilar recreo.

- «qualifications.csv». Contiene la información del alias de los profesores y las materias junto con la capacidad de estos primeros de impartir estas segundas.
- «availabilities.csv». Contiene la información del alias de los profesores y de los huecos (días y sesiones) junto con la disponibilidad de estos primeros durante estos segundos.
- «groups.csv». Contiene la información de los alias de las combinaciones grupos (grados y letras), su carácter permitido o no y su tutor en caso de que el grupo se encuentre definido. La unión tipo «Left Join» que realizaremos, comentada posteriormente en esta misma sección, asegurará que no se creen registros con grupos no permitidos.
- «subjects.csv». Contiene la información de los alias de las combinaciones de asignaturas (materias, grados y letras), su carácter requerido o no y sus requerimientos de lecciones tanto semanales como diarios mínimos y máximos en caso de que la asignatura se encuentre definida. La citada unión «Left Join» que realizaremos asegurará que no se creen registros con asignaturas no permitidas.

Una vez definidos los archivos de texto que actuarán como fuente de datos, pasaremos a definir sus campos y las uniones establecidas entre estos para lograr la estructura de registros deseada. Realizaremos un sencillo modelo en estrella en torno al archivo de lecciones «timetable.csv» unido a los archivos «professors.csv», «qualifications.csv», «availabilities.csv», «groups.csv» y «subjects.csv», cuyo esquema relacional de uniones será el siguiente:



Todas las uniones reflejadas desde los archivos auxiliares al archivo central de referencia de lecciones asignadas serán del tipo «Left Join». Este tipo de unión genera un registro para cada uno de los registros originales de referencia, tratando de unirlos con un registro del archivo auxiliar que comparta el valor de los campos de unión. De esta forma:

- Evitamos la creación de registros para grupos, asignaturas y tutores nulos contenidos en los archivos auxiliares que romperían la estructura deseada.
- Obtenemos para las lecciones asignadas su información de los archivos auxiliares, caracterizando aspectos como los requerimientos de la asignatura, el grupo tutorizado por el profesor docente de la lección o el tutor del grupo que recibe la lección.
- Obtenemos un registro con los campos del registro original para las lecciones libres o no disponibles para profesores, ya que al no encontrar registros en los archivos auxiliares se mantendrá el contenido del archivo de referencia dejando estos campos de los archivos auxiliares en valor nulo. De esta manera, no romperemos la estructura buscada en los registros de lecciones.

4.3. Consideraciones generales.

En esta sección trataremos acerca de aspectos comunes a todas las visualizaciones, de control de los registros en la fuente de datos y de obtención de métricas, principalmente. También hablaremos del

control de agregación en los recuentos de lecciones por profesores, grupos y demás aspectos del horario y la creación de campos auxiliares.

- Lecciones desde el punto de vista de los profesores y de los grupos. Como ya hemos comentado anteriormente, el archivo de referencia central «timetable.csv» contiene un registro para cada lección asignada entre profesor y grupo en un hueco concreto para impartir una materia determinada pero también para cada hueco en el que un profesor se encuentra libre. En estos registros tanto la asignatura como el grado y la letra tendrá valor «FREE». Controlaremos este aspecto a través de un campo calculado de Tableau llamado «Subject (Teachers)» a partir de los campos originales de la base de datos que controlan este aspecto: «Matter», «Grade» y «Letter». Su fórmula será:

```
IF([Matter] = "FREE" AND [Grade] = "FREE" AND [Letter] = "FREE")
THEN
    "FREE"
ELSEIF([Matter] = "N/A" AND [Grade] = "N/A" AND [Letter] = "N/A")
THEN
    "N/A"
ELSE
    "(" + [Matter] + ", " + [Grade] + ", " + [Letter] + ")"
END
```

De esta forma, este campo nos servirá para:

- Representar la asignación en un hueco determinado para un profesor, mostrando la asignatura (materia y grupo) correspondiente en el caso de que esté asignada la lección, «FREE» cuando el profesor no se encuentre asignado y «N/A» cuando el profesor no se encuentre disponible.
- Además y debido a lo expuesto en el anterior punto, este campo nos servirá para quedarnos únicamente con las lecciones asignadas. En primer lugar, a través de esto podremos representar los horarios organizados para los grupos, que no contemplan la posibilidad de encontrarse libres o no asignados. En segundo, nos permitirá hacer el análisis detallado de los grupos y los profesores midiendo el número de lecciones asignadas como un recuento de número de registros sobre esta base de datos filtrada por el campo.

En ese mismo sentido, una vez filtradas únicamente las lecciones asignadas, tendremos que representar la asignación de cada grupo en cada hueco. Esta vendrá dada por un profesor y una materia, dadas por los campos «Teacher» y «Matter» respectivamente. Para representarla, crearemos el siguiente campo calculado:

```
"(" + [Teacher] + ", " + [Matter] + ")"
```

- Cálculos de nivel de detalle con «FIXED». Emplearemos dos tipos principales:
 - Recuento de lecciones: fijando algún tipo de estructura como profesores o grupos, contamos el número de registros en la base de datos correspondientes a cada uno de esos profesores o grupos una vez filtrada la fuente de datos en la vista a través del campo «Subject (Teachers)» eliminando las lecciones libres o no disponibles y quedándonos solamente con las asignadas. Pondremos el ejemplo del calculo de lecciones asignadas para cada profesor. Fijaremos el campo «Teacher» y haremos un conteo de registros. La fórmula de su campo calculado de nombre «Assigned lessons (teacher)» será:

```
SUM({FIXED [Teacher]: SUM([Number of Records])})
```


Otro ejemplo será el campo «Assigned lessons (group)», que tras filtrar como en el caso anterior a través del campo calculado «Subject (Teachers)» eliminando lecciones libres o no disponibles hará un recuento de las lecciones asignadas sobre el grupo a través de la siguiente fórmula:

```
SUM({FIXED [Group Groups]: SUM([Number of Records]))})
```

- Obtención de constantes repetidas por la unión de fuentes de datos. Las características de los ficheros auxiliares como los requerimientos de lecciones de las asignaturas o la disponibilidad máxima semanal de lecciones de los profesores se repiten en cada registro, mientras que en nuestras comprobaciones queremos tenerlos una única vez en un nivel de agregación correcto. Para ilustrarlo, pondremos el ejemplo del cálculo de la disponibilidad semanal máxima de un profesor. Esa información se encuentra repetida en el campo «Availability Professors», por lo que fijaremos cada profesor con el campo «Teacher» y obtendremos el máximo (en realidad daría igual obtener el mínimo, debido a que los registros son todos repetidos e iguales) del campo de disponibilidad semanal aprovechando para comprobar si el profesor reserva una lección para vigilancia de recreo. Esto se hará a través del campo calculado de nombre «Weekly Availability (teacher)» y fórmula:

```
SUM({FIXED [Teacher]: MAX(
    IF([Is Available Recess Professors] = 1)
    THEN
        [Availability Professors] - 1
    ELSE
        [Availability Professors]
    END
)})
```

- Semáforos y banderas: como el modelo de organización de horarios requiere ciertos aspectos como la asignación de los profesores un porcentaje relativo mínimo de su máximo de disponibilidad semanal o la no duplicidad de lecciones para grupos o profesores en ningún hueco, crearemos una estructura de campos calculados de métricas y dimensiones para estudiar estos aspectos. En relación con los dos apartados anteriores, ilustraremos cómo se vigila que ningún profesor supera su disponibilidad semanal máxima con sus lecciones asignadas. Crearemos un campo calculado de nombre «R1 - Check Max Lessons Teacher» que fije cada profesor a través del campo «Teacher» y compruebe si las lecciones asignadas dadas por el campo calculado «Assigned lessons (teachers)» superan o no la disponibilidad máxima semanal para cada tutor dada por el campo calculado «Weekly Availability Teachers».
- Campos auxiliares de valor nulo, con fórmula:




```
NULL
```

Se utilizarán para poder crear hojas con iconos que sirvan para la navegación entre pestañas al aplicar acciones de cambio de pestaña al hacer click sobre ellos.


4.4. Visualizaciones.

Describamos una a una las visualizaciones que pretendemos llevar a cabo en el Dashboard a desarrollar con Tableau. En cada uno de los siguientes apartados incluiremos la descripción general de la vista, cómo obtenerla en Tableau y capturas en imagen de cómo se ve el resultado final. Iremos pestaña a pestaña, describiendo las hojas incluidas en esta así como la interacción con esta pestaña por parte del usuario y la navegación entre pestañas.

- 1. Primera pestaña: vistazo general a los horarios de grupos y profesores.


Timetable Overview




Teacher

PMU_2 

	L	M	X	J	V
1º	FREE	(MU, 6º, B)	(MU, 1º, A)	N/A	N/A
2º	(MU, 4º, A)	(MU, 4º, A)	FREE	N/A	N/A
3º	(MU, 6º, B)	FREE	(MU, 5º, A)	N/A	N/A
4º	(MU, 1º, A)	(MU, 5º, C)	(PL, 5º, A)	N/A	N/A
5º	(MU, 5º, A)	FREE	FREE	N/A	N/A
6º	(MU, 5º, C)	FREE	FREE	N/A	N/A

Assigned
FREE
N/A

Group

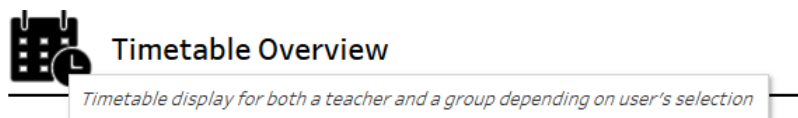
(3º, B) 

	L	M	X	J	V
1º	(PIN_3, IN)	(PR1_6, CN)	(PEF_2, EF)	(PIN_3, VA)	(PIN_3, IN)
2º	(PR1_6, CN)	(PMU_1, MU)	(PIN_3, IN)	(PR1_6, MA)	(PR1_6, TU)
3º	(PMU_1, MU)	(PEF_2, EF)	(PIN_3, VA)	(PR1_6, PL)	(PR1_6, CS)
4º	(PR1_6, CS)	(PIN_3, IN)	(PR1_6, PL)	(PR1_6, LE)	(PEF_2, EF)
5º	(PR1_6, LE)	(PR1_6, MA)	(PR1_6, LE)	(PR1_6, MA)	(PR1_6, LE)
6º	(PR1_6, MA)	(PR1_6, LE)	(PR1_6, MA)	(PEF_2, EF)	(PR1_6, MA)

CN
CS
EF
IN
LE
MA
MU
PL
TU
VA

Incluiremos las siguientes hojas:

- Icono con descripción emergente explicando la función de la pestaña como descripción emergente al poner el cursor sobre esta.



La construiremos llevando el campo auxiliar «null» a la tarjeta de «Formas» y editando la descripción emergente. Esta será el procedimiento general para crear hojas auxiliares con títulos e iconos con descripciones emergentes explicativas.

- Icono con link a pestaña 2 de horarios de tutores al hacer click sobre él. Contiene descripción emergente de su función al poner el ratón encima de este.



La construiremos de manera análoga a las hojas con descripciones emergentes. Posteriormente, estableceremos una acción del tipo «Ir a la pestaña» con origen en esta misma hoja, destino en la hoja de horarios de tutores y activada al hacer click.

- Icono con link a pestaña 3 de horarios de grupos al hacer click sobre él. Contiene descripción emergente de su función al poner el ratón encima de este².



- Título del apartado de horarios profesores con descripción explicativa.

Teacher

- Horario de los profesores por día y sesión. El selector de filtro escogido nos permite visualizar el horario de un único profesor al mismo tiempo en la hoja asociada.

	PMU_2					
	L	M	X	J	V	
1ª	FREE	(MU, 6º, B)	(MU, 1º, A)	N/A	N/A	Assigned
2ª	(MU, 4º, A)	(MU, 4º, A)	FREE	N/A	N/A	FREE
3ª	(MU, 6º, B)	FREE	(MU, 5º, A)	N/A	N/A	N/A
4ª	(MU, 1º, A)	(MU, 5º, C)	(PL, 5º, A)	N/A	N/A	
5ª	(MU, 5º, A)	FREE	FREE	N/A	N/A	
6ª	(MU, 5º, C)	FREE	FREE	N/A	N/A	

Para cada hueco de la semana organizada, se muestra la asignación del profesor, pudiendo darse los casos de:

- Tutores:
 - ◇ Tener un hueco con una lección asignada dentro del propio grupo tutorizado, mostrando tanto la materia como el grado y la letra del grupo que reciben la asignatura. Mostraremos este tipo de huecos en color gris claro.
 - ◇ Tener un hueco con una lección asignada fuera del propio grupo tutorizado, mostrando tanto la materia como el grado y la letra del grupo que reciben la asignatura. Mostraremos este tipo de huecos en color gris de intensidad media.
 - ◇ Tener un hueco libre, mostrando el texto «FREE» para indicarlo. Mostraremos este tipo de huecos en color gris oscuro.
- No tutores:
 - ◇ Tener un hueco con una lección asignada con una asignatura concreta, mostrando la tanto la materia como el grado y la letra del grupo que reciben la asignatura. Mostraremos este tipo de huecos en color gris claro.
 - ◇ Tener un hueco libre sin ninguna lección asignada, mostrando el texto «FREE» para indicarlo. Mostraremos este tipo de huecos en color gris de intensidad media.
 - ◇ Tener un hueco no disponible, mostrando el texto «N/A» para indicarlo. Mostraremos este tipo de huecos en color gris oscuro.

Construiremos la hoja llevando el campo «Days» al estante de «Columns» y el campo «Sessions» al estante de «Filas», mientras que por otra parte pondremos el campo «Subject (Teachers)» en la tarjeta de «Texto». Por último, el campo «Teachers» se colocará sobre la

²Una vez ilustradas varias descripciones emergentes, incluiremos las capturas de los iconos y títulos sin sus descripciones emergentes por no excedernos en espacio con las imágenes, limitándonos a indicar la presencia de la descripción emergente en la hoja determinada.

tarjeta de «Filtros».

Será de relevancia que este filtro será común a todas las hojas dentro de la pestaña 4 de detalles de profesores, para que al navegar entre estas se conserve la selección de profesor cuya información se enseña.

Al poner el ratón sobre un hueco de un tipo determinado, pudiendo ser uno entre asignado, libre o no disponible, resaltaremos todas los huecos del mismo tipo dentro del horario.

- Icono con link de navegación a la pestaña 4 de detalles de profesores.



Al tener filtro común con la hoja anterior de horarios de profesores, al hacer click sobre el icono de navegación accederemos a la página de detalles mostrando la información del profesor seleccionado previamente. Una vez en esa página podremos de nuevo cambiar el profesor cuya información se enseña por pantalla, y al volver a la pestaña 1 original mostraremos el horario del nuevo profesor seleccionado. Ampliaremos esto en la explicación de la pestaña 4.

- Título del apartado de horarios grupos con descripción emergente explicativa.

Group

- Horario de los grupos por día y sesión. El selector de filtro escogido nos permite visualizar el horario de un único grupo al mismo tiempo en la hoja asociada.

	(18, A)				
	L	M	X	J	V
1ª	(PR1_1, CN)	(PR1_1, MA)	(PMU_2, MU)	(PIN_4, IN)	(PIN_4, IN)
2ª	(PEF_4, EF)	(PR1_1, TU)	(PIN_4, IN)	(PEF_4, EF)	(PR1_1, VA)
3ª	(PR1_6, CS)	(PEF_4, EF)	(PR1_1, LE)	(PR1_1, PL)	(PR1_1, LE)
4ª	(PMU_2, MU)	(PIN_4, IN)	(PR1_1, VA)	(PR1_1, LE)	(PEF_4, EF)
5ª	(PR1_1, MA)	(PR1_1, CN)	(PR1_1, MA)	(PR1_1, MA)	(PR1_1, PL)
6ª	(PR1_1, LE)	(PR1_1, LE)	(PR1_1, CN)	(PR1_6, CS)	(PR1_1, MA)

Para cada hueco de la semana organizada, se muestra la asignación del grupo. Siempre tendrá asignada una asignatura, no contemplando para los grupos la posibilidad de huecos libres o no disponibles. Mostraremos la materia a la que corresponde la asignatura y el profesor que la enseña la lección concreta al grupo en cada hueco de la semana. Distinguiremos cada hueco con un color en función del tipo de materia a la que corresponda.

Construiremos la hoja llevando el campo «Days» al estante de «Columnas» y el campo «Sessions» al estante de «Filas», mientras que por otra parte pondremos el campo «Subject (Matters)» en la tarjeta de «Texto». Por último, el campo «Group Groups», que representa el grupo concreto al que corresponde la lección del registro de la fuente de datos, será llevado al estante de filtros.

Será de relevancia que este filtro será común a todas las hojas dentro de la pestaña 5 de detalles de grupos, para que al navegar entre estas se conserve la selección de grupos cuya información se enseña.

Al poner el ratón sobre un hueco de una materia determinada resaltaremos todas los huecos del mismo tipo dentro del horario.

- Icono con link de navegación a la pestaña 5 de detalles de grupos.



Al tener filtro común con la hoja anterior de horarios de grupos, al hacer click sobre el icono de navegación accederemos a la página de detalles mostrando la información del grupo seleccionado previamente. Una vez en esa página podremos de nuevo cambiar el grupo cuya información se enseña por pantalla, y al volver a la pestaña 2 original mostraremos el horario del nuevo grupo seleccionado. Ampliaremos esto en la explicación de la pestaña 5³.

■ 2. Segunda pestaña: vistazo general al horario de tutores. Incluye:

- Icono con descripción emergente explicativa de la función de la pestaña.



- Icono de navegación para volver a la pestaña 1.



- Título de tutores con descripción emergente.

Tutor

- Horarios de tutores.

PR1_6						
	L	M	X	J	V	
1ª	(PL, 6º, B)	(CN, 3º, B)	FREE	(PL, 6º, B)	FREE	FREE
2ª	(CN, 3º, B)	FREE	FREE	(MA, 3º, B)	(TU, 3º, B)	Inside Tutorised Group
3ª	(CS, 1º, A)	FREE	FREE	(PL, 3º, B)	(CS, 3º, B)	Outside Tutorised Group
4ª	(CS, 3º, B)	FREE	(PL, 3º, B)	(LE, 3º, B)	FREE	
5ª	(LE, 3º, B)	(MA, 3º, B)	(LE, 3º, B)	(MA, 3º, B)	(LE, 3º, B)	
6ª	(MA, 3º, B)	(LE, 3º, B)	(MA, 3º, B)	(CS, 1º, A)	(MA, 3º, B)	

Distinguiremos entre tres tipos diferentes de lecciones:

- Asignación del tutor en su grupo tutorizado. Será representado con el tono de gris más claro.
- Asignación del tutor fuera de su grupo tutorizado. Será representado con un tono de gris de intensidad medio.
- Lección libre. Será representado con el tono de gris más oscuro.

³Una vez mostrados los detalles técnicos de todas las hojas de una pestaña en lo relativo a los conceptos de Tableau, para las siguientes pestañas nuestro análisis se centrará más en su función, funcionamiento e información enseñada más que en detalles técnicos de funcionamiento. De todas formas, comentaremos las peculiaridades fundamentales de programación en Tableau. Para mayor detalle, puede consultarse el libro de Tableau adjunto a la memoria correspondiente a la visualización de horarios

Este será el caso especial de filtrado desde el campo «Tutors Professors» hasta los valores concretos del campo «Teacher». Como Tableau no deja llevar a cabo esta operación de manera directa, actuaremos a través de la creación de un parámetro de nombre «Tutor», de tipo «String» y con posibles valores correspondientes a los alias de todos los tutores. Una vez hecho esto, crearemos el campo calculado «Tutor Teacher Parameter Control (1)». Querremos quedarnos con aquellos registros cuyo profesor docente coincida con la selección de tutor llevada a cabo por parámetro, por lo que este campo calculado es un booleano de fórmula:

[Tutor] = [Teacher]

Al filtrar por este campo calculado y quedarnos con los valores «Verdadero», podremos quedarnos únicamente con el tutor seleccionado. En el caso del filtrado del horario de tutores una vez seleccionado el tutor, esto podría haberse hecho directamente con un filtro; sin embargo, en el caso del filtrado de un grupo tutorizado una vez seleccionado el tutor, esto deberá ser llevado a cabo de manera obligatoria a través de un parámetro.

A excepción del filtrado comentado, la hoja de horarios de tutores se construirá de manera análoga a la hoja de filtrado de profesores dentro de la pestaña 1.

- Título de grupos con descripción emergente y detalle del grupo tutorizado por el tutor seleccionado.

Group

- Horario del grupo tutorizado por el tutor seleccionado previamente.

(3º, B)

	L	M	X	J	V	
1ª	(PIN_3, IN)	(PR1_6, CN)	(PEF_2, EF)	(PIN_3, VA)	(PIN_3, IN)	Inside Tutorised Group
2ª	(PR1_6, CN)	(PMU_1, MU)	(PIN_3, IN)	(PR1_6, MA)	(PR1_6, TU)	Outside Tutorised Group
3ª	(PMU_1, MU)	(PEF_2, EF)	(PIN_3, VA)	(PR1_6, PL)	(PR1_6, CS)	
4ª	(PR1_6, CS)	(PIN_3, IN)	(PR1_6, PL)	(PR1_6, LE)	(PEF_2, EF)	
5ª	(PR1_6, LE)	(PR1_6, MA)	(PR1_6, LE)	(PR1_6, MA)	(PR1_6, LE)	
6ª	(PR1_6, MA)	(PR1_6, LE)	(PR1_6, MA)	(PEF_2, EF)	(PR1_6, MA)	

- Asignación del grupo con su tutor correspondiente, análogo a la asignación del tutor dentro de su grupo tutorizado. Será representado con el tono de gris más claro.
- Asignación del grupo con un profesor diferente a su tutor, análogo a la asignación del tutor fuera de su grupo tutorizado. Será representado con un tono de gris de intensidad medio.
- La opción de hueco libre para los grupos no se contempla en el modelo.

Comentaremos ahora la parte correspondiente al filtrado del horario del grupo tutorizado en función de la selección del tutor. Querremos quedarnos con aquellos registros cuyo grupo que recibe la lección coincida con el grupo tutorizado por el tutor seleccionado por parámetro, por lo que este campo calculado es un booleano de fórmula:

[Tutor Groups] = [Tutor]

De nuevo, filtraremos para quedarnos únicamente con los registros con valor «Verdadero». Este es la parte que no podría restringirse correctamente a través de un filtro estándar en Tableau, obligándonos a adoptar la estrategia del uso del parámetro.

- Por último, estableceremos un sistema de resaltado entre la hoja de tutores y la hoja de grupos tutorizados. Al poner el ratón sobre un tipo de lección entre las 3 definidas resaltará

en ambas tablas todas las lecciones correspondientes a este tipo, pudiendo ver conjuntamente las asignaciones que cumplan la correspondencia entre tutor y grupo o no desde el punto de vista tanto de estos primeros tutores como de estos segundos grupos. Por último, aclararemos que al poner el ratón sobre una lección libre en un profesor únicamente resaltaremos las lecciones de ese tipo en el horario de los profesores, ya que para los grupos no consideramos huecos libres en este modelo.

- 3. Tercera pestaña: vistazo general al horario de grupos. Incluye:
 - Icono con descripción emergente explicativa de la función de la pestaña.



- Icono de navegación para volver a la pestaña 1.



- Título de grupos con descripción emergente.

Group

- Horarios de grupos.

(6º, A)							
	L	M	X	J	V		
1ª	(PEF_2, PL)	(PEF_2, EF)	(PR1_1, LE)	(PEF_2, MA)	(PMU_3, MU)	<div> <div>Inside Tutorised Group</div> <div>Outside Tutorised Group</div> </div>	
2ª	(PEF_2, EF)	(PEF_2, CN)	(PEF_2, CN)	(PEF_2, TU)	(PEF_2, MA)		
3ª	(PR1_1, LE)	(PR1_1, LE)	(PEF_2, VA)	(PEF_2, CS)	(PEF_2, PL)		
4ª	(PIN_6, IN)	(PFR_2, FR)	(PEF_2, EF)	(PIN_6, IN)	(PR1_1, LE)		
5ª	(PEF_2, CS)	(PIN_6, IN)	(PFR_2, FR)	(PMU_3, MU)	(PEF_2, VA)		
6ª	(PEF_2, MA)	(PEF_2, MA)	(PEF_2, MA)	(PR1_1, LE)	(PIN_6, IN)		

Distinguiremos los mismos casos de lecciones que en la anterior pestaña 2.

De nuevo, este será el caso especial de filtrado desde el campo «Group Groups» hasta los valores concretos del campo «Tut Group Professors». Por las razones anteriormente expuestas, actuaremos a través de la creación de un parámetro de nombre «Group», de tipo «String» y con posibles valores correspondientes a los alias de todos los grupos. Una vez hecho esto, crearemos el campo calculado «Tutor Group Parameter Control (2)». Querremos quedarnos con aquellos registros cuyo grupo reciba la lección coincida con la selección de tutor llevada a cabo por parámetro, por lo que este campo calculado es un booleano de fórmula:

[Group] = [Group Groups]

Al filtrar por este campo calculado y quedarnos con los valores «Verdadero», podremos quedarnos únicamente con el tutor seleccionado. En el caso del filtrado del horario de grupos una vez seleccionado el valor deseado en el selector, esto podría haberse hecho directamente con un filtro; sin embargo, en el caso del filtrado de un tutor correspondiente una vez seleccionado el grupo, esto deberá ser llevado a cabo obligatoriamente a través de un parámetro.

A excepción del filtrado comentado, la hoja de horarios de tutores se construirá de manera análoga a la hoja de filtrado de profesores dentro de la pestaña 1.

- Título de tutores con descripción emergente y detalle del tutor que tutoriza el grupo seleccionado.

Tutor

- Horario del tutor del grupo seleccionado previamente.

PEF_2

	L	M	X	J	V
1ª	(PL, 6º, A)	(EF, 6º, A)	(EF, 3º, B)	(MA, 6º, A)	FREE
2ª	(EF, 6º, A)	(CN, 6º, A)	(CN, 6º, A)	(TU, 6º, A)	(MA, 6º, A)
3ª	FREE	(EF, 3º, B)	(VA, 6º, A)	(CS, 6º, A)	(PL, 6º, A)
4ª	FREE	FREE	(EF, 6º, A)	FREE	(EF, 3º, B)
5ª	(CS, 6º, A)	(EF, 5º, B)	(EF, 5º, B)	(EF, 5º, B)	(VA, 6º, A)
6ª	(MA, 6º, A)	(MA, 6º, A)	(MA, 6º, A)	(EF, 3º, B)	FREE

■ FREE
 ■ Inside Tutorised Group
 ■ Outside Tutorised Group

Distinguiremos los mismos casos de lecciones que en la anterior pestaña 2.

Comentaremos ahora la parte correspondiente al filtrado del horario del tutor correspondiente en función de la selección del grupo seleccionado. Querremos quedarnos con aquellos registros cuyo profesor que imparte la lección coincida con el tutor del grupo seleccionado por parámetro, por lo que este campo calculado es un booleano de fórmula:

`[Tut Group Professors] = [Group]`

De nuevo, filtraremos para quedarnos únicamente con los registros con valor «Verdadero». Este es la parte que no podría restringirse correctamente a través de un filtro estándar en Tableau, obligándonos a adoptar la estrategia del uso del parámetro.

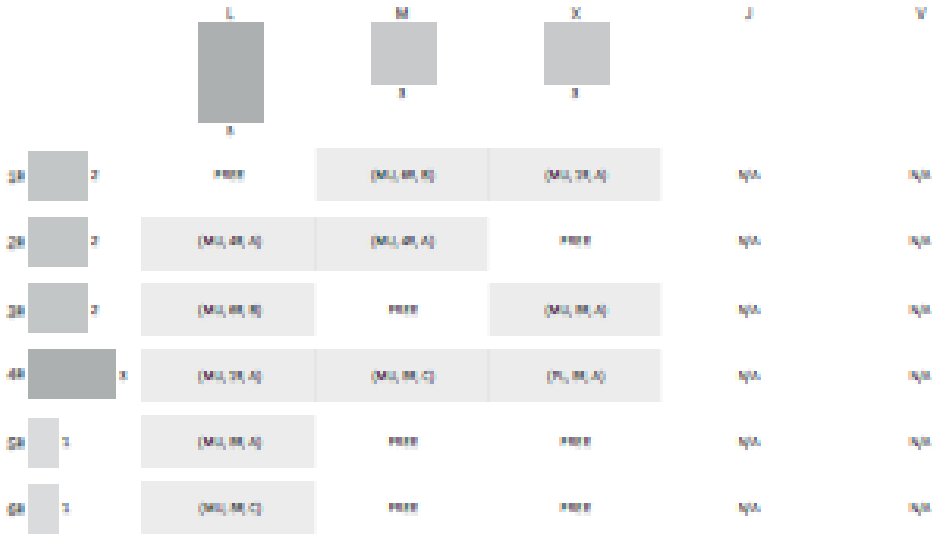
- Por último, estableceremos un sistema de resaltado de lecciones entre el horario del grupo y de su tutor correspondiente análogo al llevado a cabo en la pestaña 3.
- 4. Cuarta pestaña: análisis en detalle del horario de un profesor. Esta pestaña presentará enlaces de navegación de ida y vuelta entre la propia pestaña 4 y la pestaña 1 de vistazo general de horarios. Para que el cambio de pestañas al hacer click en los iconos de navegación tenga más sentido, el profesor filtrado en todas las hojas de esta pestaña 4 será el mismo utilizado en el horario de profesores de la pestaña 1, tal y como comentamos anteriormente. Haremos esto estableciendo un filtro en común que afecte a todas las hojas.
- En esta memoria ofreceremos la información para otro profesor diferente al mostrado en la hoja correspondiente de la pestaña 1 por mostrar mayor variedad de datos, pero al navegar entre ellas se conservaría el profesor original *PMU*₂. En concreto, escogeremos un tutor para mostrar todos los KPIs en ambas filas.



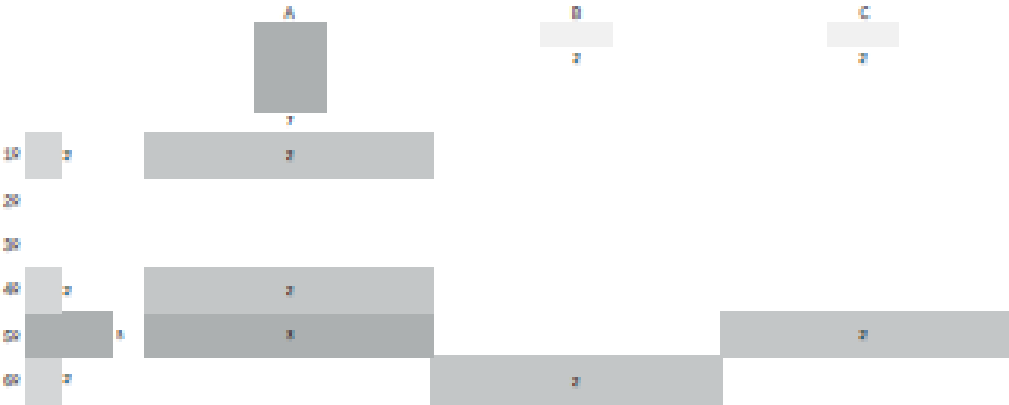
General Information

Weekly available	Assigned lessons	Availability Ratio
54	15	78.87%

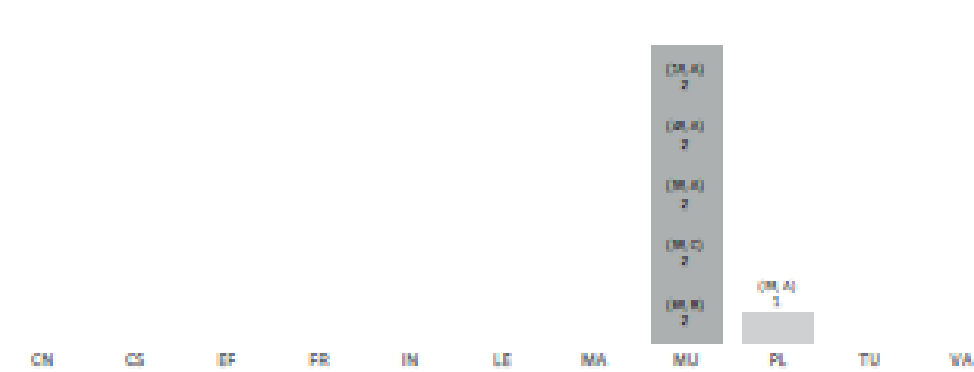
Timetable Analysis



Group Analysis



Matter Analysis



Las hojas incluidas serán:

- Icono con descripción emergente de la función de la pestaña, filtro selector del profesor mostrado en esta pestaña 4 de detalle e icono con link que al hacer click devuelve a la pestaña 1 de vistazo general de horarios. Como anteriormente ya hemos ilustrado gran variedad de descripciones emergentes e iconos, a lo largo de estas pestañas 4 y 5 únicamente nos centraremos en describir las visualizaciones contenidas.
- 6 KPIs acerca de la asignación de lecciones.

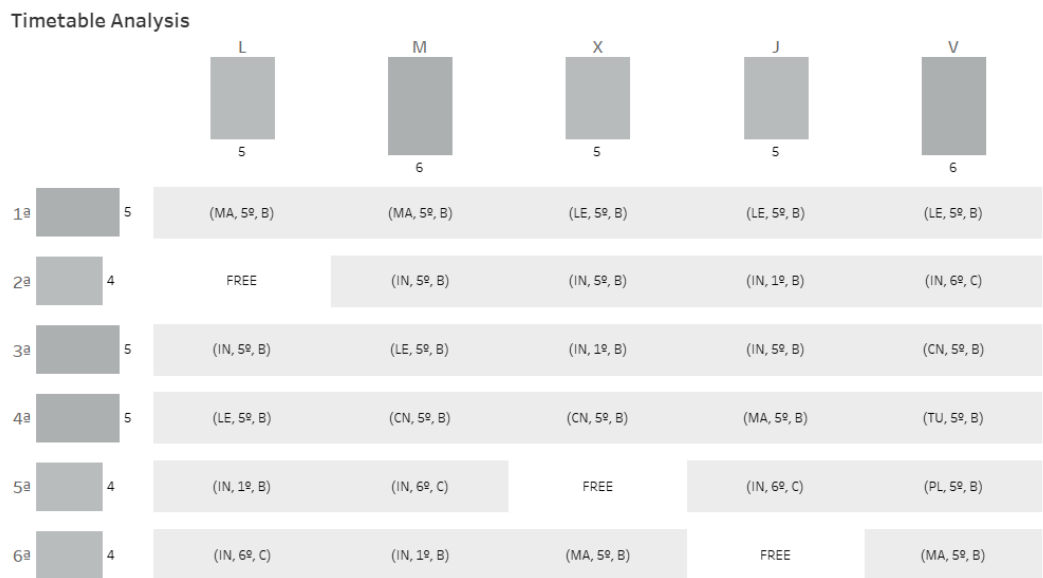
General Information

Weekly available	Assigned lessons	Availability Ratio
29	27	93,10%
Tutorised Group	Lessons for Tutorised Group	Group Ratio
(5º, B)	19	70,37%

Mostrarán, respectivamente:

- Primera fila: información común a todos los profesores. 3 KPIs incluyendo por este orden:
 - ◇ Disponibilidad máxima semanal del profesor.
 - ◇ Número de lecciones asignadas por semana. En color rojo en caso de que supere la disponibilidad máxima, en color verde en caso contrario.
 - ◇ Porcentaje de lecciones asignadas sobre el máximo disponible semanalmente. En verde en caso de que alcance el objetivo mínimo de asignación respecto a la disponibilidad máxima del profesor, en rojo en caso contrario.
 - Segunda fila: información exclusivamente para tutores. Se incluye una hoja auxiliar en un contenedor flotante fuera de los límites del Dashboard junto a las hojas de los KPIs, de tal manera que la hoja se encuentre filtrada por el carácter de tutor del profesor escogido en el selector. Incluiremos aquellos profesores que no son tutores filtrando por el campo «Is Tutor Professors» y quedándonos con el valor 0. De esta forma, cuando el profesor sea un tutor, la hoja no mostrará datos, colapsando y trayendo dentro de los límites del Dashboard las hojas de los KPIs de tutores, haciéndolas visibles. En caso de que el profesor no sea un tutor, la hoja mostrará datos y ocupará espacio, desplazando los KPIs de tutores que no tendrían sentido para el no tutor seleccionado fuera de los límites del Dashboard, evitando así que sean visibles.
- 3 KPIs mostrando, respectivamente:
- ◇ Nombre del grupo tutorizado.
 - ◇ Número de lecciones asignadas al tutor dentro del propio grupo tutorizado. En verde en caso de que el porcentaje de lecciones asignadas dentro del propio grupo tutorizado sobre el total de lecciones asignadas entre todos los grupos alcance el porcentaje mínimo de tutorización dentro del propio grupo.
 - ◇ Porcentaje de lecciones asignadas dentro del propio grupo tutorizado sobre el total de lecciones asignadas entre todos los grupos alcance el porcentaje mínimo de tutorización dentro del propio grupo. En verde en caso de que alcance el porcentaje mínimo de tutorización dentro del propio grupo.

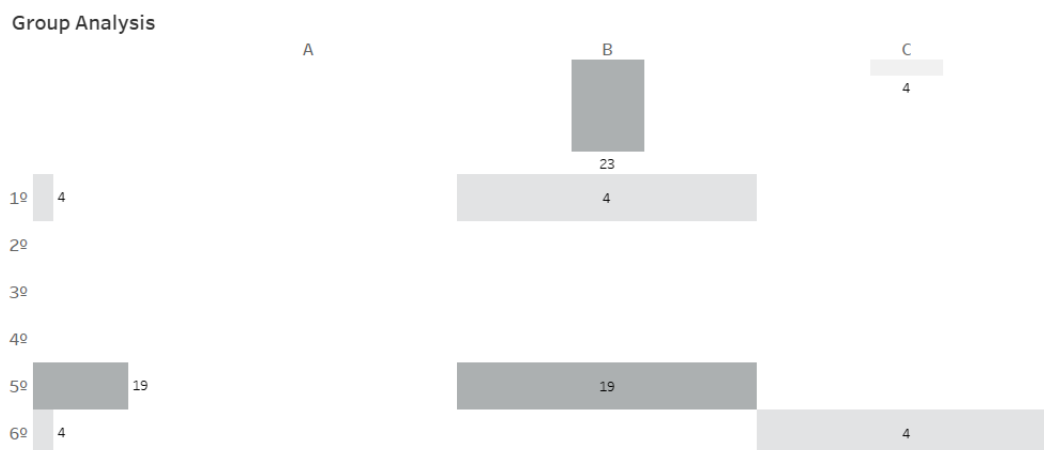
- Análisis de la asignación de la docencia del profesor por huecos, detallando por día y sesión.



En la parte central aparece el horario del profesor, mostrando las asignaturas asignadas en color gris claro y las libres en color blanco. Mostraremos por texto en cada caso la materia y la combinación de grado y letra del grupo. En la parte superior y en la parte izquierda se encontrarán dos agregaciones respectivamente por día y sesión del número de lecciones asignadas a través del campo calculado correspondiente. De esta manera, cada columna del gráfico de arriba mostrará el número total de lecciones asignadas cada día a cada profesor, con la longitud de la barra y la intensidad de su color proporcionales a este, mientras que cada fila del gráfico de la izquierda hará lo propio con la cantidad de lecciones asignadas a cada profesor a lo largo de cada una de las sesiones del horario. De esta manera, podremos observar la distribución de las lecciones asignadas a los profesores a lo largo de los huecos en que se compone el horario, detallando por día y sesión.

Al poner el ratón sobre el gráfico central de horario por día y sesión, resaltaremos la columna de día y la fila de sesión en ambos gráficos de detalles correspondientes. Por otra parte, al poner el ratón sobre un gráfico de detalle, resaltaremos sobre el gráfico central todas las lecciones en huecos contenidos en determinados días o sesiones, en función de si es a través de las columnas del gráfico superior o de las filas del gráfico de la izquierda.

- Análisis de la asignación del horario del profesor por grupos, detallando por grado y letra.

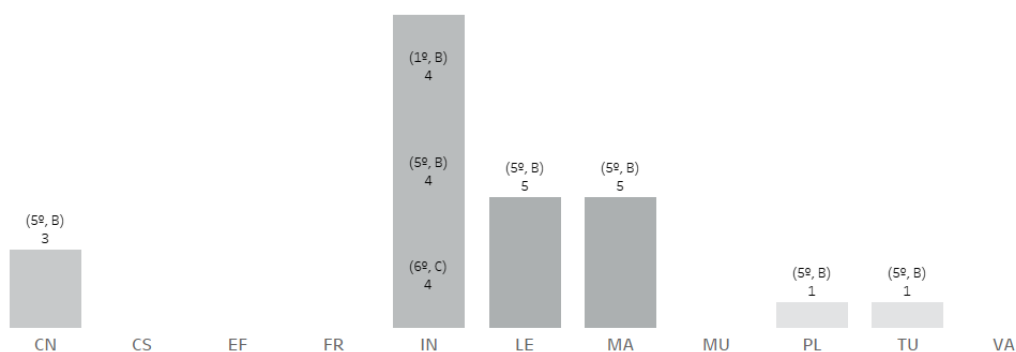


En la parte central aparece la división por grupo de las lecciones asignadas al profesor como tabla, mostrando en color proporcional a este número de sesiones asignadas así como el propio número por texto. En la parte superior y en la parte izquierda se encontrarán dos agregaciones respectivamente por letra y grado del número de lecciones asignadas a través del campo calculado correspondiente. De esta manera, cada columna del gráfico de arriba mostrará el número total de lecciones asignadas en cada letra para cada profesor, con la longitud de la barra y la intensidad de su color proporcionales a este, mientras que cada fila del gráfico de la izquierda hará lo propio con la cantidad de lecciones asignadas a cada profesor a en un determinado grado. De esta manera, podremos observar la distribución de las lecciones asignadas a los profesores a lo largo de los grupos en que se compone la estructura de organización del alumnado del centro, con detalle en los grados y letras en los que estas se distribuyen.

Al poner el ratón sobre el gráfico central de distribución por grupo, resaltaremos la columna de letra y la fila de grado en ambos gráficos de detalles correspondientes. Por otra parte, al poner el ratón sobre un gráfico de detalle, resaltaremos sobre el gráfico central las lecciones en grupos correspondientes a determinadas letras o grados, en función de si es a través de las columnas del gráfico superior o de las filas del gráfico de la izquierda.

- Distribución por materia de las sesiones asignadas al profesor con detalle en el grupo de la asignatura concreta.

Matter Analysis



Cada columna mostrará una barra con una longitud proporcional al número de lecciones asignadas por materia. Por otra parte, cada barra correspondiente a una materia concreta tendrá un recuadro para cada grupo sobre el que establezca docencia para una asignatura de ese tipo de materia. De esta forma, el recuadro de cada asignatura mostrará una intensidad de color y una longitud proporcional al número de lecciones asignadas para el profesor en la asignatura establecida en ese grupo para el tipo de materia concreta de la columna citada, así como un detalle con la materia y combinación de grado y letra del grupo junto al número de sesiones asignadas por texto.

Al poner el ratón sobre un recuadro correspondiente a la asignatura sobre un grupo concreto, se resaltarán el resto de asignaturas de diferentes materias pero dentro del mismo grupo para este profesor dentro del gráfico.

- 5. Quinta pestaña: análisis en detalle del horario de un grupo. Esta pestaña presentará enlaces de navegación de ida y vuelta entre la propia pestaña 5 y la pestaña 1 de vistazo general de horarios. Para que el cambio de pestañas al hacer click en los iconos de navegación tenga más sentido, el grupo filtrado en todas las hojas de esta pestaña 5 será el mismo utilizado en el horario de grupos de la pestaña 1, tal y como comentamos anteriormente. Haremos esto estableciendo un filtro en común que afecte a todas las hojas.

En esta memoria ofreceremos la información para otro grupo diferente al mostrado en la hoja

correspondiente de la pestaña 1 por mostrar mayor variedad de datos, pero al navegar entre ellas se conservaría el profesor original PMU_2 . En concreto, escogeremos un tutor para mostrar todos los KPIs en ambas filas. Además, técnicamente es muy análogo a lo descrito en el anterior punto, así que nos limitaremos a describir las visualizaciones que forman parte de este y su función.

- Icono con descripción emergente, icono de navegación...
- Fila de KPIs por texto.

General Information

Assigned lessons	Lessons with Tutor	Lessons with Tutor	Tutor
30	18	60,00%	PR1_1

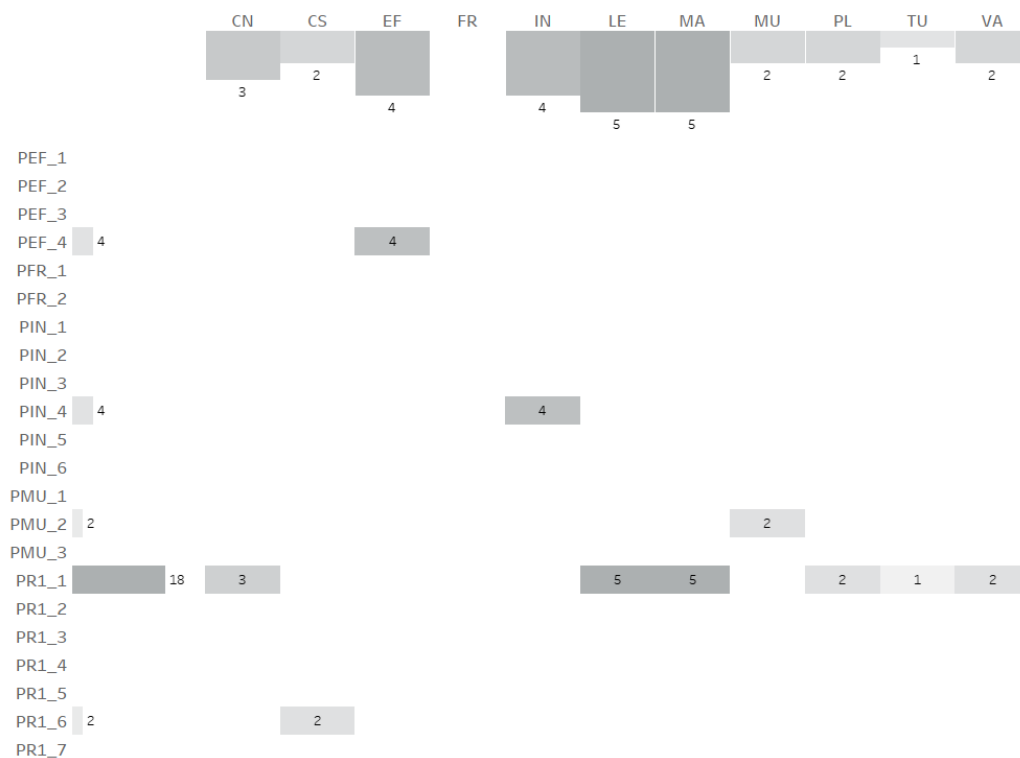
4 KPIs representando, respectivamente:

- Número de lecciones asignadas. Debe de ser 30 en todos los grupos.
 - Número de lecciones asignadas con el propio tutor del grupo.
 - Porcentaje de lecciones asignadas con el propio tutor del grupo sobre el número de lecciones asignadas totales.
 - Nombre del tutor del grupo seleccionado.
- Horario del grupo, resaltando las lecciones del mismo tipo: asignadas con el tutor de la asignatura o con otro diferente.

Timetable Analysis

	L	M	X	J	F	V
6ª	(PR1_1, LE)	(PR1_1, LE)	(PR1_1, CN)	(PR1_6, CS)		(PR1_1, MA)
5ª	(PR1_1, MA)	(PR1_1, CN)	(PR1_1, MA)	(PR1_1, MA)		(PR1_1, PL)
4ª	(PMU_2, MU)	(PIN_4, IN)	(PR1_1, VA)	(PR1_1, LE)		(PEF_4, EF)
3ª	(PR1_6, CS)	(PEF_4, EF)	(PR1_1, LE)	(PR1_1, PL)		(PR1_1, LE)
2ª	(PEF_4, EF)	(PR1_1, TU)	(PIN_4, IN)	(PEF_4, EF)		(PR1_1, VA)
1ª	(PR1_1, CN)	(PR1_1, MA)	(PMU_2, MU)	(PIN_4, IN)		(PIN_4, IN)

- Análisis de la asignación de lecciones por materia y profesor, con detalle en ambos aspectos.



En la zona central, tabla para cada materia impartida por cada profesor, anotando el número de lecciones asignadas en el grupo cuya información estamos visualizando. En cada fila del gráfico de detalle de la izquierda se recogerán el número de lecciones asignadas con cada profesor, mientras que en cada columna del gráfico de arriba se recogerán el número de lecciones asignadas con cada materia.

Al poner el ratón sobre una celda correspondiente a una materia y profesor del gráfico central, resaltaremos la columna o fila correspondiente del gráfico de detalle. Por otra parte, cuando pongamos el ratón sobre un gráfico de detalle de materias o profesor resaltaremos toda la fila o toda la columna respectivamente sobre el gráfico central.

- 6-11: pestañas de la sexta a la undécima, comprobaciones de la corrección de las variables y las restricciones del modelo. Son más bien una herramienta utilizada en el desarrollo del código en Cplex/Java y la comprobación de la corrección de sus resultados, pero se adjuntan en el libro de Tableau por su utilidad. Sirven para comprobar si las docencias y lecciones asignadas por parte del programa cumplen:
 - Restricciones de docencia y lecciones en la pestaña 6 «Model Checking», comprobando que se cumplan las condiciones de capacitación de los profesores para establecer docencia sobre una materia determinada, de disponibilidad de estos mismos en un hueco para asignar una lección, de existencia del grupo y de requerimiento de la asignatura. Se mostrarán aquellas condiciones que se cumplen con ✓ y aquellos en los que no con ✗.
 - Requerimientos de reglas del modelo, en las pestañas 7-11. Vigilaremos los requerimientos tanto de docencia como de lecciones con el mismo código de símbolos que en el caso anterior.

Bibliografía.

- 1 ÁRTON P. DORNELES, OLINTO C.B. DE ARAÚJO, LUCIANA S. BURIOL, *A fix-and-optimize heuristic for the high school timetabling problem*, Computers & Operations Research, 52, 29-38, 2014.
- 2 LANDIR SAVINIEC, MARISTELA O. SANTOS, ALYSSON M. COSTA, *Parallel local search algorithms for high school timetabling problems*, European Journal of Operational Research, 265, 81-98, 2018.
- 3 JUAN PEIRÓ, *Construcción de horarios escolares. Aproximaciones mediante programas de programación lineal entera.*, Facultad de Ciencias, Universidad de Zaragoza, Zaragoza.
- 4 ASHWIN URDHWARESHE, *Object-Oriented Programming and its Concepts*, International Journal of Innovation and Scientific Research, Vol. 26 No. 1, 2016.
- 5 *IBM ILOG CPLEX Optimization Studio: Getting Started with CPLEX*, IBM Corporation, versión 12, edición 6, 2013.
- 6 *IBM Knowledge Center: Overview (CPLEX Java API Reference Manual)*. Obtenido de https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.6.2/ilog.odms.cplex.help/refjavacplex/html/overview-summary.html.
- 7 *Ayuda de Tableau Desktop y de la creación web*. Obtenido de <https://help.tableau.com/current/pro/desktop/es-es/default.html>.