Heurística basada en programación entera binaria para el problema de asignación de salones en la Universidad de los Andes

Trabajo de Tesis
presentado al
Departamento de Ingeniería Industrial

por

Elkin Castro Medina

Asesor: Andrés Medaglia Gonzalez, Ph. D.

Para optar al título de
Magíster en Ingeniería Industrial

Departamento de Ingeniería Industrial
Universidad de Los Andes
Julio de 2004

Heurística basada en programación entera binaria para el problema de asignación de salones en la Universidad de los Andes

Aprobado por:

Andrés Medaglia Gonzalez, Ph. D., Asesor

Fernando Palacios, Ph. D., Coasesor.

Gonzalo Mejía, Ph. D.

Angello Ávila, Oficina de Admisiones y Registro.

Fecha de Aprobación

A mi mamá, Myriam.

Reconocimientos

Quiero agradecer a mi asesor Andrés Medaglia, no solo por su apoyo continuo e imprescindibles aportes durante el desarrollo de este trabajo, sino además por su enorme disposición y voluntad en ayudarme a encontrar nuevas oportunidades de desarrollo personal y profesional; a Fernando Palacios, por su acompañamiento durante esta investigación; a Angello Ávila de la Oficina de Admisiones y Registro de la Universidad de los Andes, por sus contribuciones a este trabajo, su conocimiento en el tema y su disponibilidad. A Roberto Zarama, por su apoyo durante estos dos años; a German Riaño y Natalia Santamaria, por desarrollar uniandes-tesis, plantilla en LaTeX con la que escribí este documento. Al Academic Partner Program de Dash Optimization, por suministrar las licencias de Xpress-MP.

Estoy infinitamente agradecido con mi mamá Myriam Medina. Ella constantemente tuvo fe en este trabajo y en sus resultados, llevo siempre en su mente y en sus oraciones a todos los que trabajamos en este proyecto. Sin su amor, cuidados y paciencia, no hubiera sido posible su finalización. A mi hermano, quien esperaba varios días a que soltara el computador; y a Gloria Rosales por su amistad auténtica, por acompañarme y compartir conmigo en periodos muy importantes de la vida.

Finalmente, agradezco a Julio Cesar Goez y Rolando Avendaño por su amistad, por haber escrito y corrido conmigo el primer modelo de asignación de salones en el curso de optimización; y a David Rojas, por su paciencia, amistad y patrocinio, en todo lo que a Julio y a mi se nos ocurría hacer.

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Lista de Algoritmos

Introducción

El proceso de asignación de salones es un trabajo difícil y dispendioso que toda universidad debe llevar a cabo al menos dos veces al año, y consiste en encontrar el salón apropiado, en un horario especifico, para cada una de las secciones que se quieren programar. Una asignación de salones satisfactoria, debe tener en cuenta las preferencias de los departamentos y de los profesores, por salones y horarios específicos; así como también las normas académicas y administrativas; y requerimientos culturales de la comunidad universitaria.

La asignación de salones generalmente se desarrolla en un ámbito donde la demanda por salones y horarios de clase particulares, tiende a crecer; los recursos como los salones, son escasos y de capacidades limitadas; y hay objetivos que compiten por estos recursos: el propósito principal de los profesores es satisfacer sus preferencias por un horario y un salón específicos, mientras que uno de los objetivos de la administración, es la utilización eficiente de la planta física. Entonces, el problema de asignación de salones se puede definir a la luz de sus objetivos, recursos limitados y restricciones [16]

Objetivo Maximizar la satisfacción del número de secciones con salón asignado, respecto a los recursos y las restricciones.

Recursos

- Número limitado de salones disponibles para una sección particular.
- Capacidad restringida de los salones.
- Horarios de funcionamiento de los edificios.
- Estudiantes registrados en una clase particular.
- Cantidad de profesores disponibles.
- Número de secciones de cada curso.

Restricciones

- Intensidad horaria de cada sección.
- Disponibilidad efectiva de salones.
- El nivel de ocupación de los salones debe ser alto, para que una sección se programe en éste.
- Preferencias de horario y de salón de los profesores.
- En algunos modelos se evita dictar ciertas materias simultáneamente, porque existe un número importante de estudiantes inscrito en ambas (conflicto de estudiante).
- A un profesor no se le puede asignar mas de una clase simultáneamente (conflicto de profesor).
- En un salón no se puede dictar mas de una clase al tiempo (conflicto de salón).

Este problema ha interesado a la comunidad de Investigación de Operaciones y numerosos procedimientos de solución han sido propuestos. Mulvey [18] presenta el concepto de 'slot' como la dupla (salón, hora de clase) el cual es muy útil para

la visualización de las asignaciones que se hacen en este tipo de problemas. De la misma manera, define una sección como la dupla (profesor, grupo de estudiantes) y modela el problema como la asignación de un 'slot' a cada sección. Esta formulación, que se reduce a una red, se resuelve en un esquema iterativo de interacción hombre-máquina donde la intuición del programador del horario y su intervención en la construcción de éste son importantes. Dinkel et al. [10] presentan un sistema de soporte a la decisión formulando el problema como un problema de flujo en una red con capacidad y construyen una estructura de penalización en la función objetivo. El problema se resuelve con un optimizador de redes y en caso que la solución no sea factible, se incorporan las restricciones violadas y se usa un optimizador de programación entera.

De otro lado, Carter [6] resuelve cada día de la semana separadamente para reducir el tamaño del problema. Aun más, cada día se divide en tres partes y para cada parte se resuelve una serie de problemas de asignación usando relajación lagrangiana. Hertz y Robert [12] resuelven una serie de problemas de asignación bien sea con heurísticas, como búsqueda tabú, algoritmos de propósito especifico para flujo en redes, o ramificación y acotamiento, dependiendo de la estructura del problema encontrada en cada paso del esquema de solución que proponen.

Por la complejidad del problema, algunos autores han usado algoritmos de búsqueda local. Mooney et al. [17] aplican un algoritmo de búsqueda local a una formulación de programación cuadrática entera del problema de asignación de un 'slot' a cada sección que se quiere programar. También se presenta un algoritmo de preprocesamiento que identifica los conjuntos de patrones de tiempo y de salones factibles para cada sección.

Los enfoques más recientes resuelven instancias del problema con optimizadores

enteros. Dimopoulou y Miliotis [8] formulan un modelo de programación entera, y para simplificar las instancias de estos problemas se definen conjuntos de cursos y conjuntos de horas diseñados para satisfacer los requerimientos administrativos y académicos específicos. Daskalaki et al. [7] proponen un programa entero binario y resuelven una instancia para el departamento de una universidad. Asimismo, Dimopoulou y Miliotis [9] proponen un sistema distribuido, basado en una red de computadores para construir el horario. Al interior del sistema se resuelve un modelo de programación entera, que asigna cursos a franjas horarias y salones.

Capítulo I

Asignación de salones en la Universidad de los Andes

El proceso de asignación de salones lo hace la Oficina de Admisiones y Registro, y comienza desde el semestre inmediatamente anterior para el cual se está elaborando el horario. El procedimiento involucra, desde el ingreso de la información relevante de los cursos que ofrecerá cada Departamento, por parte del Coordinador Académico; el uso de una heurística de asignación de salones; hasta la negociación con los Coordinadores Académicos, de las secciones que quedaron si salón.

1.1. Proceso de asignación de salones

El proceso de asignación de salones tiene tres etapas

Etapa 1 Los Coordinadores Académicos de cada Departamento ingresan en Banner la información de cada curso que se ofrecerá el semestre siguiente. Este proceso dura dos semanas y se realiza entre la semana 10 y 12 del semestre inmediatamente anterior para el cual se elabora el horario. Además de otros datos, cada sección que se ingresa cuenta como mínimo, con un tamaño y un horario (i.e., dias, horas de inicio y de finalización de las sesiones de clase). En éste punto, el sistema Banner impide que a un profesor se le asigne más de

una clase al tiempo, es decir, el Sistema impide que se presenten conflictos de profesor.

Las secciones de materias de primer semestre, cursos magistrales, cursos de maestría y cursos con requerimientos especiales (profesores con discapacidades, problemas de acceso, requerimientos de seguridad, etc.) cuentan con un salón pre-asignado, el cual es fijado por la Oficina de Admisiones y Registro, previa solicitud de los Coordinadores Académicos. Aun más, las clases de maestría preferiblemente se deben dictar en los edificios LL o AU. Banner también impide que se presenten conflictos de salón, es decir, impide que en un salón se dicte más de una clase simultáneamente.

Etapa 2 En ésta Etapa que dura tres semanas, la Oficina de Admisiones y Registro usa una heurística de asignación de salones, que toma como datos de entrada la información de las secciones, y el inventario de salones de la Universidad.

Durante éste periodo, los datos que ingresaron los Coordinadores en la Etapa 1 pueden cambiar: se crean/cancelan nuevos cursos y secciones, las capacidades y horarios cambian, etc. Esto hace que la heurística se use repetidas veces durante ésta etapa, y aún más, varias veces durante un mismo día.

Etapa 3 Finalmente, la Oficina de Admisiones y Registro debe negociar con los Coordinadores Académicos, las secciones que no tienen salon(es) asignado(s). Este proceso de negociación dura de dos a tres semanas. Durante la negociación, se plantean entre otras alternativas, cambios de horario y cambios en la capacidad de la sección.

Al final de la negociación, las secciones para las cuales no fue posible encontrar salon(es), no se ofrecerán.

1.2. Heurística de asignación de salones

1.2.1. Archivos de entrada

La heurística de asignación de salones toma como datos de entrada dos archivos. El primero, `phyyyypp.txt`, extraído de Banner, contiene la información de cada sección [19, 20]: CRN, Departamento que dicta la sección, numero de materia, sección, créditos, nombre de la materia, cupo máximo, inscritos, cupos disponibles, hora de inicio, hora de finalización, salón pre-asignado, días en los que se dicta la clase: L para lunes, M para martes, I para miércoles, J para jueves, V para viernes, S para sábado. Se supone que los horarios de clase fijados en este archivo, no tienen conflictos de profesor, y que las secciones con salón pre-asignado, no tiene conflictos de salón. El segundo archivo, `sayyyypp.prn`, es el inventario de salones disponibles de la Universidad. Para cada salón se tiene: nombre del salón, capacidad y tipo de salón. Si el tipo de salón es 0, entonces la Oficina de Admisiones y Registro puede disponer de éste para hacer la asignación. Si este atributo es 1, el salón lo asigna algún Departamento, es decir, se dice que el salón es privado.

1.2.2. Algoritmo

La heurística [19] asignará a cada sección, de ser posible, uno o varios salones de la siguiente manera:

1. A las secciones con salón pre-asignado se les fija el salón respectivo, y se aumenta en cierto porcentaje la capacidad de todas las secciones, para tener cupos adicionales disponibles.

2. Se comienza la asignación por las secciones que se dictan más dias a la semana y tienen más cupo, teniendo en cuenta, que a cada sección se le debe asignar

el salón de menor tamaño que la aloje, y que esté sin asignar.

3. Si para alguna sección no se puede encontrar un solo salón para todas sus sesiones, se buscan más salones, hasta tener uno para cada sesión de clase.

Capítulo II

Formulación del problema de asignación de salones

El problema de programación de salones se modela como la asignación de una sección a un 'slot'. Donde un 'slot' se define como un conjunto de días de la semana, una hora de inicio y una hora de finalización de clase (patrón de tiempo) y un salón. Varios autores han utilizado el concepto de 'slot' [10, 17, 18]. Siguiendo el enfoque de Money et al. [17] se definen los siguientes conjuntos, elementos y operaciones

Conjunto de profesores

$\mathcal{I}$: Conjunto de todos los profesores (Índice: i)

Conjuntos de secciones

$\mathcal{S}$: Conjunto de todas las secciones (Índice: s)

$\overline{\mathcal{S}}$: Secciones de cursos prioritarios: cursos de primer semestre, maestría, cursos con sesiones magistrales y cursos con requerimientos especiales.

$\mathcal{S}_i$: Secciones asignadas al profesor i.

Conjuntos de salones

$\mathcal{R}$: Conjunto de todos los salones (Índice: r)

$\mathcal{R}_m$: Conjunto de salones para maestría.

Conjuntos de 'slots'

$\mathcal{L}$: Conjunto de todos los 'slots' (Índice: l)

$\mathcal{L}_s$: Conjunto de 'slots' con salones y patrones de tiempo factibles candidatos para la sección s.

$\mathcal{L}_i$: Conjunto de 'slots' factibles candidatos para el profesor i.

Conjunto de patrones de tiempo

$\mathcal{T}$: Conjunto de todos los patrones de tiempo (Índice: t)

'Slot' y patrón de tiempo

$l = (r,\ t)$: l es un 'slot', donde r es un salón y t un patrón de tiempo.

$t =$ (días de la semana, hora de inicio, hora de finalización)

Funciones sobre 'slots'

$t(l) \in \mathcal{T}$ es el patrón de tiempo del 'slot' l y $r(l) \in \mathcal{R}$ es el salón del 'slot' l.

Cruce de patrones de tiempo

$t,\ u \in \mathcal{T}$ se cruzan, $t \approx u$, si tienen al menos un día en común y una o más fracciones de sesión que se cruzan.

Además, se define α_{sl} como el coeficiente de ajuste de la sección s al 'slot' l, $\alpha_{sl} = \lambda\beta_{s,\,t(l)} + (1-\lambda)\rho_{s,\,r(l)}$. Donde $\beta_{s,\,t(l)}$ es la preferencia del profesor por dictar la sección s con el patrón de tiempo del 'slot' l, y $\rho_{s,\,r(l)}$ mide la calidad de la asignación de salón, como la densidad de ocupación del salón del 'slot' l cuando se le asigna la sección s. λ, es el peso relativo de cada coeficiente de ajuste, $0 \leq \lambda \leq 100$ y por último, $0 \leq \beta_{s,\,t(l)},\ \rho_{s,\,r(l)} \leq 1$. Para el caso particular en el que la sección s tiene salón pre-asignado, $\rho_{s,\,r(l)}$ puede ser mayor que uno en una cantidad especificada por el usuario.

Las variables de decisión se definen como $x_{sl} = 1$ si a la sección s se le asigna el 'slot' l y $x_{sl} = 0$, en caso contrario. El modelo de asignación de salones se formula así

$$\text{Maximizar} \quad \sum_{s \in \mathcal{S}} \sum_{l \in \mathcal{L}_s} \alpha_{sl} x_{sl} \tag{1}$$

Sujeto a:

$$\sum_{l \in \mathcal{L}_s} x_{sl} = 1 \qquad \forall\ s \in \overline{\mathcal{S}} \tag{2}$$

$$\sum_{l \in \mathcal{L}_s} x_{sl} \leq 1 \qquad \forall\ s \in \{\mathcal{S} \backslash \overline{\mathcal{S}}\} \tag{3}$$

$$\sum_{\substack{l \in \{l\,:\, r(l)=r(l') \\ \wedge\ t(l) \approx t(l')\}}} \sum_{s\,:\, l \in \mathcal{L}_s} x_{sl} \leq 1 \qquad \forall\ l' \in \mathcal{L} \tag{4}$$

$$\sum_{s \in \mathcal{S}_i} \sum_{\substack{l \in \{l\,:\, l \in \mathcal{L}_s\ \wedge \\ t(l) \approx t(l')\}}} x_{sl} \leq 1 \qquad \forall\ i \in \mathcal{I},\ l' \in \mathcal{L}_i \tag{5}$$

La función objetivo (1) representa el propósito principal de la Oficina de Admisiones y Registro, de maximizar el número de secciones asignadas y optimizar el uso de la planta física. El conjunto de restricciones (2) asegura que las secciones especiales se programarán. En (3) se evita duplicidad de asignación de 'slots' a cada sección. Las restricciones (4) impiden que en un salón se dicte mas de una clase simultáneamente (i.e., conflicto de salón). Finalmente, el conjunto de restricciones (5) determina que un profesor no puede dictar más de una clase simultáneamente (i.e., conflicto de profesor).

Las restricciones (4) y (5) consideran los conflictos por 'slot' de una forma agregada. Alternativamente, se pueden considerar parejas de asignaciones que producen

conflicto

$$x_{sl} + x_{s',l'} \leq 1 \qquad \forall\ l \in \mathcal{L}_s,\ l' \in \mathcal{L}_{s'},\ (s,\, l) \neq (s',\, l'), \tag{6}$$

$(s,\, l)$ y $(s',\, l')$ tienen conflicto de profesor o de salón.

La formulación (1)-(5) (llamada de aquí en adelante Formulación 1) tiene considerablemente menos restricciones que la formulación (1)-(3) y (6) (llamada de aquí en adelante Formulación 2). Se observa además, que la Formulación 2 es más fuerte que la 1, en el sentido que aproxima mejor el casco convexo [3, 11, 21]. Asimismo, es común que para problemas enteros binarios se busquen restricciones booleanas o lógicas del estilo (6) que involucran una o dos variables, y después, ó bien adicionarlas al problema ó usarlas para fijar el valor de ciertas variables [21].

La adición de las restricciones (6) se logra, bien sea resolviendo directamente la Formulación 2 ó utilizándolas en algoritmos de generación de cortes como 'Branch-and-Cut' (B&C) y 'Cut-and-Branch' (C&B) [3, 11, 21].

Capítulo III

Creación del horario

La creación del horario para la Universidad de los Andes comprende dos etapas: preprocesamiento de datos y solución de la(s) instancia(s) del preprocesamiento en Xpress-MP. En el preprocesamiento se construye el conjunto de 'slots' factibles $\mathcal{L}_s$ para cada una de las secciones, y para la segunda fase se implementó la Formulación 1 en Xpress-MP.

3.1. Preprocesamiento

El algoritmo de preprocesamiento de datos toma entre otros datos de entrada, los archivos descritos en la Sección 1.2.1. Para cada una de las secciones que se quieren programar, se construye un conjunto de salones factibles candidatos y un conjunto de patrones de tiempo candidatos. Los elementos de estos conjuntos se combinan, generando el conjunto de 'slots' factibles candidatos, $\mathcal{L}_s$, para cada una de las secciones. De esta forma, se genera una instancia reducida para la Formulación 1 y para la 2. Se implementó el algoritmo de preprocesamiento de datos basado en las ideas propuestas por Mooney et al. [17] usando SAS® [13, 14].

La selección del conjunto de patrones de tiempo candidatos, comienza considerando como primer elemento, el patrón de tiempo que originalmente se propone

para cada una de las secciones. Para los otros elementos, se consideran desplazamientos de las horas de inicio de las clases, conservando los días intactos. El usuario especifica la longitud, la cantidad y el sentido (más temprano o más tarde) de los desplazamientos. Después, se calcula el coeficiente de preferencia por un patrón de tiempo, $\beta_{s,\, t(l)}$, utilizando una función de penalización que asignará el mayor peso al patrón de tiempo preferido, y pesos menores al resto de patrones.

La construcción del conjunto de salones candidatos, inicia con la elaboración de una lista de salones factibles para cada una de las secciones, basándose en el tamaño de éstas. Al mismo tiempo se calcula $\rho_{s,\, r(l)}$, como la densidad del salón cuando éste aloja a la sección correspondiente. Si una sección tiene un salón pre-asignado, la densidad de ocupación es mayor o igual a 1, en una cantidad especificada por el usuario. El número de salones factibles candidatos por sección, es un argumento que especifica el usuario, y la selección de los salones se hace con una heurística que distribuye de manera balanceada, los salones entre las secciones respectivas.

Para construir el conjunto de salones candidatos, cada salón se inicializa con una pseudo-capacidad horaria, definida como un múltiplo de las horas semanales disponibles del salón. Después, se seleccionan los salones candidatos para cada sección, comenzando por la sección con menos salones factibles. Los salones con las mayores densidades de ocupación se escogen primero, y posteriormente se seleccionan los salones con la mayor pseudo-capacidad remanente. A medida que se va seleccionando salones, sus pseudo-capacidades se reducen en la intensidad horaria de la sección correspondiente.

Teniendo en cuenta la complejidad y la cantidad de detalles del algoritmo de preprocesamiento, ésta Sección presenta el algoritmo desde una perspectiva general, para luego mostrar los detalles y las variantes de la implementación. Se inicia

entonces, presentando los archivos y argumentos de entrada al código de preprocesamiento; posteriormente se discute el algoritmo de preprocesamiento de datos; y finalmente, se muestran las dos variantes del algoritmo de preprocesamiento.

3.1.1. Argumentos y archivos de entrada

3.1.1.1. Archivo de secciones: `phyyyypp.txt`

`phyyyypp.txt` es un archivo de datos separado por tabuladores que se genera con la información de Banner. `yyyy` corresponde al año para el que se elabora la programación de cursos y `pp` el periodo para el cual se elabora el horario: primer semestre, 10; vacaciones de mitad de año, 19; y segundo semestre, 20. Por ejemplo, para las pruebas computacionales se usó el archivo del primer semestre del año 2004, `ph200410.txt`., en particular, el que se generó de Banner al comienzo de la Etapa 2 del proceso de generación del horario, descrito en el Capítulo 1. La Tabla 1 muestra las columnas para cada una de las secciones que se quieren programar [19, 20].

Campo	Tipo de dato	Descripción	Ejemplo
CRN	Numérico (cinco dígitos)	identificador de la sección en Banner	12582
Departamento	Cadena alfa-numérica (cuatro caracteres)	Departamento que ofrece la materia	IEL1
Numero materia	Numérico (tres dígitos)	Numero de la materia	201
Sección	Numérico (uno a dos dígitos)	sección de la materia	1
Créditos	Numérico (un digito)	Créditos de la materia	3
Nombre materia	Cadena alfa-numérica (menos de 35 caracteres)	Nombre de la materia	TEORIA SEÑALES
Cupo máximo	Numérico (uno a tres dígitos)	Capacidad máxima de estudiantes de la sección	80
Inscritos	Numérico (uno a tres dígitos)	Estudiantes inscritos en la sección	0
Cupos disponibles	Numérico (uno a tres dígitos)	Cupos disponibles en la sección	80
Hora inicio	Numérico (tres a cuatro dígitos)	Hora inicio clase en formato de 24 horas	1000
Hora fin	Numérico (tres a cuatro dígitos)	Hora fin clase en formato de 24 horas	1050
Salón pre-asignado	Cadena alfa-numérica (dos a cinco caracteres)	Salón pre-asignado a la sección	O202
Lunes	Un carácter	Clase el lunes	L
Martes	Un carácter	Clase el martes	
Miércoles	Un carácter	Clase el miércoles	M
Jueves	Un carácter	Clase el jueves	
Viernes	Un carácter	Clase el viernes	V
Sábado	Un carácter	Clase el sábado	

Tabla 1: Columnas del archivo de secciones.

En este archivo se garantiza que no hay conflictos de profesor, ni conflictos de salón, como se explica en la Sección 1.1, Etapa 1. Al comienzo de la Etapa 2, se

estima que en un 60 % de las secciones se conoce el profesor, entonces, de forma aleatoria se asignó profesor a ésta cantidad de secciones con datos extraídos de Banner, y que correspondieron al horario definitivo del primer semestre del 2004, de tal forma que se puedan formular las restricciones (5) del Capitulo 2. El profesor se identifica con el campo `SPRIDEN_ID`, el cual es un dato numérico de entre tres y nueve dígitos.

3.1.1.2. Archivo de salones disponibles: `sayyyypp.prn`

El inventario de salones lo genera la Oficina de Planta Física de la Universidad, está delimitado por espacios, su nombre tiene la misma sintaxis que la del archivo de secciones y consta de los campos mostrados en la Tabla 2 [19, 20].

Campo	Tipo de dato	Descripción	Ejemplo
Nombre del salón	Cadena alfa-numérica (dos a cinco caracteres)	salón en el inventario	R203
Capacidad	Numérico (dos a tres dígitos)	Capacidad del salón	30
Tipo de salón	Numérico (un dígito)	salón público (0), salón privado (1)	1

Tabla 2: Columnas del archivo de salones.

La Oficina de Admisiones y Registro puede únicamente disponer de los salones públicos, pues los privados los asignan los Departamentos a los cuales pertenecen.

3.1.1.3. Archivo de edificios disponibles para clase: `edificios.txt`

El archivo de edificios, es un archivo creado para propósitos del algoritmo de generación de horario que se propone en este documento, y no corresponde a un archivo de uso estándar de la Oficina de Admisiones y Registro. Para todos los edificios donde están los salones públicos, se tiene el horario de funcionamiento, y se especifica en cuáles de ellos se dictan clases de maestría con una `M` en la última columna. Este archivo se muestra en la Figura 1.

```
AU      700     2100     M
B       700     2100
C       700     1900
G       700     2100
LL      700     2100     M
M       700     1900
O       700     1900
PU      700     1900
R       700     1900
S       700     1900
TM      700     1900
W       700     2100
Z       700     1900
```

Figura 1: Archivo `edificios.txt`

3.1.1.4. Argumentos de entrada

El algoritmo de preprocesamiento de datos tiene los argumentos de entrada presentados en la Tabla 3.

Argumento	Tipo de dato	Descripción	Ejemplo
`pickRho`	Entero no negativo	# de salones por sección que se escogen por tener mejor ρ	2
`pickPseudo`	Entero no negativo	# de salones por sección que se escogen por pseudo-capacidad remanente	3
`pseudoCap`	Entero positivo	Coeficiente de la capacidad horaria semanal de cada salón	5
`gap_plus`	Entero no negativo	# de patrones de tiempo que inician más tarde que el patrón deseado	2
`gap_tiempo_plus`	Entero no negativo	Valor en minutos para los corrimientos tardíos	30
`gap_minus`	Entero no negativo	# de patrones de tiempo que inician más temprano que el patrón deseado	1
`gap_tiempo_minus`	Entero no negativo	Valor en minutos para los corrimientos tempranos	30
`coefAumentoSecc`	[0,100]	Porcentaje en el cual se aumentara la capacidad real de la sección	20
`coefAumentoPref`	[0,100]	Porcentaje en el cual se aumenta la preferencia por un salón pre-asignado	30
`horaInicioMag`	Cadena caracteres	Hora en formato de 24 h, después de la cual comienzan las clases de magíster	'1300'
`lambda`	[0,100]	Peso relativo de cada coeficiente de ajuste	66

Tabla 3: Argumentos del algoritmo de preprocesamiento.

La configuración de la Tabla 3, significa que una sección dada tendrá a lo sumo un conjunto de cinco salones candidatos (dos por ρ y tres por pseudo-capacidad remanente); dentro de los salones asignados por ρ está el pre-asignado, si lo tiene; cuatro patrones de tiempo (dos patrones que inician más tarde al deseado, uno inicia más temprano al deseado y el patrón de tiempo deseado); la capacidad de la sección se aumenta en un 20 %; y la densidad del salón pre-asignado, si lo tiene, es 1.3. Como

se consideran dos patrones de tiempo tardíos y `gap_tiempo_plus` = 30, uno de ellos comienza media hora más tarde y el otro, una hora más tarde; el patrón que inicia más temprano, lo haría media hora antes al deseado.

3.1.2. Algoritmo de preprocesamiento de datos

En esta sección se presenta detalladamente el algoritmo de preprocesamiento de datos (ver Algoritmo 1 página 15). Éste algoritmo tiene cuatro partes: En la primera se leen los archivos de entrada; posteriormente se generan los 'slots' candidatos para cada sección; el tercer paso es la creación de las matrices de conflictos de 'slots'; y finalmente, se escriben las instancias para Xpress-MP en archivos de texto.

El algoritmo de preprocesamiento tiene dos variantes que se explican en las Secciones 3.1.3 y 3.1.4, respectivamente. Los pasos marcados con ***** en el Algoritmo 1 son los que cambian, dependiendo de la implementación que se haga, el resto de pasos son idénticos en ambas variantes y son los que se tratan en ésta Sección. La explicación de los pasos marcados con *, se posterga hasta las Secciones 3.1.3 y 3.1.4.

3.1.2.1. Lectura de archivos

Los archivos de entrada, una vez se leen, se transforman en data sets de SAS ®. En particular, de los archivos `sayyyypp.prn` y `phyyyypp.txt` se generan los siguientes data sets

`edificios`: Lista de con los edificios disponibles para dictar clase. Ejemplo:

> `horaAbreSeg` y `horaCierraSeg` corresponden a las horas en las que el edificio abre y cierra respectivamente, en segundos.

Algoritmo 1 Algoritmo de preprocesamiento

Entrada: Archivos: `phyyyypp.txt`, `sayyyypp.prn` y `edificios.txt`. Argumentos: `pickRho`, `pickPseudo`, `pseudoCap`, `gap_plus`, `gap_tiempo_plus`, `gap_minus`, `gap_tiempo_minus`, `coefAumentoSecc`, `coefAumentoPref`, `horaInicioMag` y `lambda`

Salida: `seccEsp.dat`, lista de secciones con salón pre-asignado; `slotsCfl1.dat`, matriz de conflicto de 'slots'; `slotsCfl2.dat`, matriz de conflictos de 'slots' por parejas, (ver Ecuación (6)); `alpha.dat` matriz de coeficientes de la función objetivo, (ver Ecuación (1))

1: Lectura de archivos
2: Generación de 'slots' candidatos por sección

1. Eliminación de observaciones del data set `secciones_bruto`
2. Simplificación (compresión) del data set `secciones_bruto` y generación del data set `secciones` *
3. Generación del conjunto de salones factibles por sección *
4. Generación del conjunto de salones candidatos por sección
5. Generación del conjunto de patrones de tiempo candidatos por sección y generación de 'slots'

3: Matrices de conflicto

1. Matriz de conflicto de 'slots' *
2. Matriz de conflicto de 'slots' por parejas *

4: Escritura de la instancia para Xpress-MP

1. `seccEsp.dat`
2. `slotsCfl1.dat`
3. `slotsCfl2.dat`
4. `alpha.dat`

edificio	horaAbre	horaCierra	maestria	horaAbreSeg	horaCierraSeg
AU	700	2100	M	25200	75600
B	700	2100		25200	75600
O	700	1900		25200	68400

Tabla 4: Dos observaciones del data set `edificios`.

`edificiosmag`**:** Lista de los edificios disponibles para clases de maestría. Corresponde a las observaciones con `maestria` = `M` en el data set de la Tabla 4.

`salonespub`**:** Salones disponibles para hacer la asignación. Estos salones corresponden a las observaciones en las que el tipo de salón es cero, en el archivo `sayyyypp.prn`. Ejemplo:

salon	capacidad	horaAbreSeg	horaCierraSeg	capSemanaTiempo
AU108	20	25200	75600	302400
O100	30	25200	68400	259200

Tabla 5: Dos observaciones del data set `salonespub`.

`capSemanaTiempo` es la disponibilidad horaria semanal del salón en segundos.

`salonesmag`**:** Listado de los salones para clases de maestría y el cual tiene la misma estructura que `salonespub`.

`salonespriv`**:** Lista de salones privados que corresponde a las observaciones con tipo de salón en uno, Tabla 2.

`secciones_bruto`**:** En éste data set se lee el archivo de secciones, `phyyyypp.txt`. En la Tabla 6 se presentan dos secciones a modo de ejemplo, las cuales se despliegan en columnas por razones de espacio, y servirán como ejemplo en adelante.

nombre	TEORIA SEÑALES	OPTIMIZACION	crn	12582	12645	horainicio	1000	1100
salon	O202		depto	IEL1	IEL1	horafinal	1050	1220
lunes	1	0	materia	201	300	spriden_id		2942563
martes	0	0	seccion	1	1	salonB	1	0
miercoles	1	1	creditos	3	3	horaCeroSeg	36000	39600
jueves	0	0	capseccion	80	40	horaFinalSeg	39000	44400
viernes	1	1	inscritos	0	0	tiempoClaseSeg	3000	4800
sabado	0	0	capresidual	80	40			

Tabla 6: Dos observaciones del data set `secciones_bruto`.

En este ejemplo, Teoría de Señales se dicta lunes, miércoles y viernes en el O202 (tiene salón pre-asignado), de 10 a 10:50 AM. Su CRN es 12582, el código de materia es IEL1-201, etc. `horaCeroSeg` y `horaFinalSeg` corresponden a la hora de inicio y de finalización de la clase en segundos, mientras que `tiempoClaseSeg` es la duración de la clase en segundos. `salonB` es 1 si la sección tiene salón pre-asignado y 0 en caso contrario.

3.1.2.2. Generación de 'slots' candidatos por sección

El primer paso de esta parte del algoritmo, consiste en eliminar ciertas observaciones del data set `secciones_bruto`. En particular, se eliminan las observaciones con salón privado, pues éstos salones son asignados por los Departamentos dueños de éstos; se eliminan las observaciones que tienen un salón que no está en el inventario de salones; y por último se eliminan las observaciones que tienen una capacidad superior a la del salón público más grande. En la Tabla 7 se presentan tres observaciones que se borrarían de éste data set (se eliminaron algunas variables por cuestión de espacio). Las dos primeras observaciones se borran porque el salón TX101 es privado, y la tercera se borra porque el salón Z126 no está en el inventario de salones.

nombre	salon	crn	seccion	capseccion	horainicio	horafinal	spriden_id	salonB	idseccion
TALLER BASICO ARTES PLASTIC. 1	TX101	11764	1	20	1000	1250	39689687	1	142
TALLER BASICO ARTES PLASTIC. 1	TX101	11764	1	20	1000	1250	39689687	1	143
LABORATORIO DE QUIMICA INORGAN	Z126	10074	5	16	800	1020	79582016	1	2377

Tabla 7: Observaciones del data set `secciones_bruto` que se borrarían.

Además de las 23 variables que se muestran en la Tabla 6, se adiciona una nueva, `idseccion`, la cual numera consecutivamente cada una de las observaciones de éste data set. Esta variable identificará de manera única las secciones que entrarán en el proceso de asignación.

Después de este proceso de eliminación de observaciones, el data set `secciones_bruto` se compacta. Este proceso depende de la inclusión de o no, de la variable `spriden_id` (profesor) y de la estructura de CRNs del archivo de secciones. El proceso de compactación se explica detalladamente en la Secciones 3.1.3 y 3.1.4, para cada una de sus dos variantes, respectivamente. Luego de la compactación, el data set `secciones_bruto` se transforma en el data set `secciones`. A partir de éste punto, la variable `idseccion` identifica de forma única cada una de las observaciones del data set `secciones`, y por lo tanto, (como se verá en las Secciones 3.1.3 y 3.1.4) cada observación de este data set, corresponde a una sección para propósitos de asignación de salón. En la Tabla 8 se muestran dos observaciones del data set `secciones`, las cuales se despliegan en columnas por cuestión de espacio.

lunes	1	0	**capseccion**	80	40	**horaFinalSeg**	39000	44400
martes	0	0	**horainicio**	1000	1100	**tiempoClaseSeg**	3000	4800
miercoles	1	1	**horafinal**	1050	1220	**spriden_id**		2942563
jueves	0	0	**salonB**	1	0	**materia**	201	300
viernes	1	1	**idseccion**	1394	1401			
sabado	0	0	**horaCeroSeg**	36000	39600			

Tabla 8: Dos observaciones del data set `secciones`.

En este ejemplo la clase de Teoría de Señales con CRN 12582, se identifica ahora con el `idseccion` 1394; tiene una capacidad para 80 estudiantes; se dicta lunes, miércoles y viernes de 10 a 10:50 AM; y tiene salón pre-asignado, `salonB` $= 1$. De otro lado, la clase de Optimización cuyo CRN es 12645, tiene una capacidad para 40

estudiantes; se identifica con el `idseccion` 1401; tiene sesiones miércoles y viernes de 11:00 AM a 12:20 PM; y el profesor que la dicta se identifica con el número 2942563. En el data set `secciones` no aparece el salón de las secciones que tienen uno pre-asignado, éstos registros están en el data set `seccVsSalon`, Tabla 9 página 19.

idseccion	salon
1394	O202
1401	

Tabla 9: Dos observaciones del data set `seccVsSalon`.

Una vez se tiene el data set `secciones`, se construye el conjunto de salones factibles para cada sección, es decir para cada `idseccion`. Para la construcción de estos conjuntos se consideraron dos variantes, que se explicarán en detalle en las Secciones 3.1.3 y 3.1.4, respectivamente. Es muy importante anotar, que la omisión momentánea de la explicación de los algoritmos de compactación del data set `secciones_bruto`, y de generación del conjunto de salones factibles, no es impedimento para tratar los algoritmos que siguen. El conjunto de salones factibles de cada una de las secciones, está en el data set `salones_factibles`. Un salón se considera factible para una sección particular, si la capacidad del salón es mayor o igual al tamaño de la sección, aumentado en un porcentaje dado por el `coefAumentoSecc`. Para el ejemplo que estamos tratando, `coefAumentoSecc` $= 20\,\%$ y la sección con `idseccion` $= 1394$ tiene nueve salones factibles, mientras que la sección 1401 tiene 40 salones factibles. Si una sección tiene un salón pre-asignado, este se incluye en el conjunto de salones factibles y se le asigna una densidad de ocupación ρ, aumentada en un porcentaje dado por `coefAumentoPref`, que para este ejemplo es del $30\,\%$. En la Tabla 10 de la página 20, se presentan algunas observaciones del data set

salones_factibles para las secciones que se tomaron como ejemplo.

idseccion	spriden_id	salon	capacidad	capseccion	materia	rho
1394		O202	80	80	201	1.3
1394		B202	110	80	201	0.872727273
1394		O101	121	80	201	0.79338843
1401	2942563	AU101	48	40	300	1
1401	2942563	AU201	50	40	300	0.96
1401	2942563	B203	53	40	300	0.905660377

Tabla 10: Algunas observaciones del data set salones_factibles.

Inmediatamente se tienen los conjuntos de salones factibles, se procede a calcular los conjuntos de salones candidatos para cada sección, comenzando por la sección con menos salones factibles. El tamaño del conjunto de salones candidatos, viene dado por los parámetros pickRho y pickPseudo. En este ejemplo tienen el valor de dos, es decir, para cada sección se seleccionan primero los dos salones con la densidad de ocupación más alta, y posteriormente se seleccionan los dos salones con la mayor pseudo-capacidad remanente. El Algoritmo 2 de la página 21 presenta este procedimiento, donde se hará referencia a variables y a data sets todavía no definidos, y de los cuales se ofrecerán ejemplos.

El data set salones_factibles_smry se genera a partir del data set salones_factibles (Tabla 10) y por cada sección muestra el numero de salones factibles. Este data se ordena por la variable salones (salones factibles por sección). La Tabla 11, muestra las líneas que contienen las secciones de los ejemplos.

Después de crear el data set salones_factibles_smry, se calcula la pseudo-capacidad (pseudoCapTiempo) de cada salón publico, cantidad que se adiciona como variable en el data set salonesPub. Si un salón se abre de 7:00 AM a 9:00 PM de lunes a sábado, entonces tiene una capacidad horaria semanal de 302400 segundos; si pseudoCap=4, entonces la pseudo-capacidad semanal es de 1209600 segundos. Un

Algoritmo 2 Generación del conjunto de salones candidatos por sección

Entrada: `pickRho`, salones por sección que se seleccionan por tener la densidad de ocupación más alta; `pickPseudo`, salones por sección que se escogen por tener la pseudo-capacidad remanente más alta.

Salida: Data set `varsXpres`, el cual contiene los salones candidatos por sección.

1: Crear data set `salones_factibles_smry`. Ordenarlo por `salones` ascendentemente.
2: Crear data set de salida `varsXpress`.
3: `pseudoCapTiempo` $\leftarrow$ `capSemanaTiempo` $\cdot$ $pseudoCap$. Adicionar la pseudcapacidad de cada salón público al data set `salonesPub`.
4: $numSec \leftarrow$ Número de observaciones del data set `secciones`
5: **for** $i = 1$ to $numSec$ **do**
6: $seccionID \leftarrow$ `idseccion` de la observación i de `salones_factibles_smry`.
7: $numSal \leftarrow$ `salones` de la observación i de `salones_factibles_smry`.
8: Crear data set `salones_factibles_`$seccionID$. Ordenarlo por `rho` descendentemente y por `pseudoCapTiempo` descendentemente.
9: **if** $numSal <$ `pickRho`+`pickPseudo` **then**
10: Incluir en el conjunto de salones candidatos, los salones de `salones_factibles_`$seccionID$, tales que `pseudoCapTiempo` $\geq$ `tiempoClaseSeg`.
11: `pseudoCapTiempo` $\leftarrow$ pseudoCapTiempo - Intensidad horaria semanal de la sección i. Para cada salón asignado.
12: **else**
13: Incluir en el conjunto de salones candidatos, los primeros `pickRho` salones del data set `salones_factibles_`$seccionID$, tales que `pseudoCapTiempo` $\geq$ Intensidad horaria semanal de la sección i.
14: Ordenar `salones_factibles_`$seccionID$ por `pseudoCapTiempo` descendentemente y por `rho` descendentemente.
15: Incluir en el conjunto de salones candidatos, los primeros `pickPseudo` salones del data set `salones_factibles_`$seccionID$, tales que `pseudoCapTiempo` $\geq$ Intensidad horaria semanal de la sección i.
16: `pseudoCapTiempo` $\leftarrow$ pseudoCapTiempo - Intensidad horaria semanal de la sección i. Para cada salón asignado.
17: **end if**
18: Actualizar data set `salonesPub` con las pseudo-capacidades que se cambiaron.
19: Adicionar a `varsXpress` las observaciones de `salones_factibles_`$seccionID$ que se asignaron.
20: **end for**

idseccion	salones
1623	9
1394	9
1615	9
1565	9
1001	9
1104	9
1003	9
1365	9
1351	9
662	10
735	10

idseccion	salones
1415	40
1414	40
1413	40
1799	40
1405	40
1401	40
1809	40
1808	40
1807	40
1398	40
1795	40

Tabla 11: Algunas observaciones del data set `salones_factibles_smry`.

calculo análogo se hace para un salón que abre de 7:00 AM a 7:00 PM, el cual tendría una pseudo-capacidad de 1036800, Tabla 12.

salon	capacidad	horaAbreSeg	horaCierraSeg	capSemanaTiempo	pseudoCapTiempo
AU108	20	25200	75600	302400	1209600
O100	30	25200	68400	259200	1036800

Tabla 12: Dos observaciones del data set `salonesPub`.

Una vez se dispone de los data sets `salonesPub` y `salones_factibles_smry`, el uno modificado y el otro creado, se calcula el número de observaciones del data set `secciones`, de tal manera que ésta cantidad sea el límite superior de un contador que recorrerá cada una de sus observaciones, comenzando por la sección que tenga menos salones factibles, es decir, comenzando por la primera observación del data set `salones_factibles_smry` y recorriéndolo en orden, pues este data set está ordenado por el numero de salones factibles en forma ascendente.

La clave del ciclo del Algoritmo 2, está en recorrer secuencialmente el data set `salones_factibles_smry`, asignar a $seccionID$ el `idseccion` y a $numSal$ el

valor de `salones` (Número de salones factibles de la sección `idseccion`). Posteriormente se extraen del data set `salones_factibles` (Tabla 10), las observaciones cuya variable `idseccion` sea igual a $seccionID$, de tal forma que se construye el data set `salones_factibles_`$seccionID$. A éste data set se agregan variables del data set `salonesPub` en las que hay correspondencia de `salon`, y análogamente, se adicionan variables del data set `secciones`, pero en las que existe correspondencia de `idSeccion`. La Tabla 13 de la página 24, muestra el data set `salones_factibles_`$seccionID$, correspondiente a la sección 1394, las observaciones se han fraccionado por razones de espacio.

Los salones candidatos corresponden a las observaciones tales que, `asignacion`=1. Aún más, cada vez que se selecciona un salón esta variable toma el valor de 1, indicando que el salón respectivo fue asignado, y así evitar duplicidad en la selección de salones candidatos. En particular, los salones O202 y B202 se seleccionan por tener la más alta densidad de ocupación, mientras que los salones G104 y O102 se seleccionan por haber tenido las pseudo-capacidades remanentes más altas en su momento. Si una sección tiene un salón pre-asignado, éste siempre se incluye en el conjunto de salones candidatos, porque las primeras asignaciones se hacen por densidad de ocupación. Por ejemplo, la sección 1394 tiene como salón pre-asignado el O202 (Tabla 9), el cual se seleccionó para la lista de salones candidatos. Las variables `horaAbreSeg` y `horaCierraSeg` corresponden al horario de funcionamiento en segundos, del salón correspondiente.

Las asignaciones de salones candidatos para la sección 1401, se muestran en la Tabla 14 de la página 25, y corresponden a las líneas en las cuales `asignacion` = 1. Para la sección 1401, los salones AU101 y AU103 entran en el conjunto de salones candidatos por tener las densidades de ocupación más altas, de otro lado, los salones

idseccion	spriden_id	salon	capacidad	capseccion	materia
1394		B202	110	80	201
1394		G104	123	80	201
1394		O101	121	80	201
1394		O102	121	80	201
1394		O103	121	80	201
1394		O104	121	80	201
1394		O202	80	80	201
1394		R209	192	80	201
1394		R210	192	80	201

rho	horaAbreSeg	horaCierraSeg	pseudoCapTiempo	asignacion
0.872727273	25200	75600	595800	1
0.780487805	25200	75600	630000	1
0.79338843	25200	68400	637800	0
0.79338843	25200	68400	629400	1
0.79338843	25200	68400	630600	0
0.79338843	25200	68400	635400	0
1.3	25200	68400	984600	1
0.5	25200	68400	598200	0
0.5	25200	68400	630600	0

horainicio	horafinal	horaCeroSeg	horaFinalSeg	tiempoClaseSeg
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000

lunes	martes	miercoles	jueves	viernes	sabado
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0

Tabla 13: Data set `salones_factibles_1394`.

O301 y O303 entran en la lista de salones candidatos, por haber tenido las pseudo-capacidades remanentes más altas en el momento de la asignación.

Dentro del ciclo del Algoritmo 2, una vez se selecciona la totalidad de los salones candidatos para una sección particular, solo las observaciones del data set `salones_factibles_`$seccionID$, que tienen la variable `asignacion` en 1 se transfieren al data set `varsXpress`. La parte del data set `varsXpress` que corresponde a las secciones 1394 y 1401 se muestra en la Tabla 15 de la página 27.

idseccion	spriden_id	salon	capacidad	capseccion	materia
1401	2942563	AU101	48	40	300
1401	2942563	AU102	48	40	300
1401	2942563	AU103	48	40	300
1401	2942563	AU104	48	40	300
1401	2942563	O301	70	40	300
1401	2942563	O302	70	40	300
1401	2942563	O303	70	40	300
1401	2942563	O304	70	40	300

rho	horaAbreSeg	horaCierraSeg	pseudoCapTiempo	asignacion
1	25200	75600	412800	1
1	25200	75600	416400	0
1	25200	75600	412800	1
1	25200	75600	417000	0
0.685714286	25200	68400	466800	1
0.685714286	25200	68400	467400	0
0.685714286	25200	68400	466800	1
0.685714286	25200	68400	447600	0

horainicio	horafinal	horaCeroSeg	horaFinalSeg	tiempoClaseSeg
1100	1220	39600	44400	4800
1100	1220	39600	44400	4800
1100	1220	39600	44400	4800
1100	1220	39600	44400	4800
1100	1220	39600	44400	4800
1100	1220	39600	44400	4800
1100	1220	39600	44400	4800
1100	1220	39600	44400	4800

lunes	martes	miercoles	jueves	viernes	sabado
0	0	1	0	1	0
0	0	1	0	1	0
0	0	1	0	1	0
0	0	1	0	1	0
0	0	1	0	1	0
0	0	1	0	1	0
0	0	1	0	1	0
0	0	1	0	1	0

Tabla 14: Fracción del data set `salones_factibles_1401`.

Inmediatamente se tiene el conjunto de salones candidatos para cada una de la secciones (`varsXpress`), se procede a la generación del conjunto de patrones de tiempo candidatos por sección, y a la generación de 'slots'. Esto se logra tomando las observaciones del data set `varsXpress`, y a cada una de ellas se le asigna un conjunto de `gap_plus+gap_minus+1` patrones de tiempo candidatos. De esta manera, se combina cada una de las observaciones de `varsXpress` con los elementos del conjunto de patrones de tiempo candidatos. El Algoritmo 3 de la página 26, muestra este procedimiento.

Con el propósito de hacer más claro el Algoritmo 3, se presenta a continuación

Algoritmo 3 Generación del conjunto de patrones de tiempo candidatos por sección y generación de 'slots'

Entrada: `gap_plus` y `gap_minus`, numero de patrones de tiempo que inician más tarde y más temprano al deseado; `gap_tiempo_plus` y `gap_tiempo_minus`, longitud en minutos del desplazamiento hacia un patrón que inicia más tarde y más temprano; λ, coeficiente de ponderación de las preferencias por un patrón de tiempo y el ajuste del salón a la sección; y `horaInicioMag`, hora a partir de la cual se dictan clases de maestría.

Salida: Data set `varsXpres`, el cual contiene los 'slots' para cada sección.

1: **for all** j observación de `varsXpress` **do**
2: $gap\_tiempo\_plus \leftarrow$ `gap_tiempo_plus`$\cdot 60$
3: $gap\_tiempo\_minus \leftarrow$ `gap_tiempo_minus`$\cdot 60$
4: Leer observación j de `varsXpress`
5: $t0\_orig \leftarrow$ `horaCeroSeg`
6: $tf\_orig \leftarrow$ `horaFinalSeg`
{Generación del patrón de tiempo original.}
7: $\beta \leftarrow 1$
8: $\alpha \leftarrow \lambda\beta + (100 - \lambda)\rho$ {Calculo del coeficiente de la función objetivo}
9: Escribir estos datos en `varsXpress`
{Generación de patrones de tiempo tempranos}
10: **for** $i = 1$ to `gap_minus` **do**
11: **if** $(t0\_orig - gap\_tiempo\_minus \cdot i \geq$ `horaAbreSeg`$)$ and (`materia` $\leq$ 399 or `materia` ≥ 900) **then**
12: Calcular patrón de tiempo temprano y escribir en `varsXpress`
13: **else if** $(t0\_orig - gap\_tiempo\_minus \cdot i \geq$ `horaAbreSeg`$)$ and $(400 \leq$ `materia` $\leq 499)$ **then**
14: `horaInicioMagSeg` $\leftarrow$ `horaInicioMag` es segundos
15: **if** $(t0\_orig - gap\_tiempo\_minus \cdot i \geq$ `horaInicioMagSeg`$)$ or $(t0\_orig <$ `horaInicioMagSeg`$)$ **then**
16: Calcular patrón de tiempo temprano y escribir en `varsXpress`
17: **end if**
18: **end if**
19: **end for**
{Generación de patrones de tiempo tardíos}
20: **for** $i = 1$ to `gap_plus` **do**
21: **if** $tf\_orig - gap\_tiempo\_plus \cdot i \leq$ `horaCierraSeg` **then**
22: Calcular patrón de tiempo tardío y escribir en `varsXpress`
23: **end if**
24: **end for**
25: **end for**

idseccion	spriden_id	salon	capacidad	capseccion
1394		B202	110	80
1394		G104	123	80
1394		O102	121	80
1394		O202	80	80
1401	2942563	AU101	48	40
1401	2942563	AU103	48	40
1401	2942563	O301	70	40
1401	2942563	O303	70	40

materia	rho	horaAbreSeg	horaCierraSeg	asignacion
201	0.872727273	25200	75600	1
201	0.780487805	25200	75600	1
201	0.79338843	25200	68400	1
201	1.3	25200	68400	1
300	1	25200	75600	1
300	1	25200	75600	1
300	0.685714286	25200	68400	1
300	0.685714286	25200	68400	1

horainicio	horafinal	horaCeroSeg	horaFinalSeg	tiempoClaseSeg
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000
1000	1050	36000	39000	3000
1100	1220	39600	44400	4800
1100	1220	39600	44400	4800
1100	1220	39600	44400	4800
1100	1220	39600	44400	4800

lunes	martes	miercoles	jueves	viernes	sabado
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
0	0	1	0	1	0
0	0	1	0	1	0
0	0	1	0	1	0
0	0	1	0	1	0

Tabla 15: Fracción del data set `varsXpress` que corresponde a las secciones 1394 y 1401.

el procedimiento para calcular el patrón de tiempo temprano con respecto al patrón deseado.

1: `horaCeroSeg` $\leftarrow t0\_orig - gap\_tiempo\_minus \cdot i$

2: `horaFinalSeg` $\leftarrow tf\_orig - gap\_tiempo\_minus \cdot i$

3: $\beta \leftarrow 1 - (1 - 0{,}6)i/$`gap_minus`

4: $\alpha \leftarrow \lambda\beta + (100 - \lambda)\rho$

5: Escribir estos datos en `varsXpress`

De la misma manera, se presenta por aparte el proceso para calcular el patrón de tiempo tardío.

1: `horaCeroSeg` $\leftarrow t0\_orig + gap\_tiempo\_plus \cdot i$

2: `horaFinalSeg` $\leftarrow tf\_orig + gap\_tiempo\_plus \cdot i$

3: $\beta \leftarrow 1 - (1 - 0{,}6)i/$`gap_plus`

4: $\alpha \leftarrow \lambda\beta + (100 - \lambda)\rho$

5: Escribir estos datos en `varsXpress`

Ciertas restricciones aplican para la generación de patrones de tiempo candidatos: no se pueden generar patrones de tiempo por fuera de las horas de funcionamiento de los salones; para una sección de maestría no es posible generar un patrón de tiempo antes de la hora en que comienzan las clases de maestría (`horaInicioMag`), a no ser que la hora a la que comience la clase en el patrón de tiempo original esté antes de `horaInicioMag`.

El coeficiente de referencia por un patrón de tiempo, se calcula con la función lineal $\beta = 1-(1-0{,}6)\frac{i}{desplazamientos}$, donde *desplazamientos* es `gap_plus` ó `gap_minus`, según el caso. β es mayor o igual que 0.6, porque se asume que nadie dictaría clase con un patrón de tiempo poco preferido. En la Figura 2 de la página 29, se muestra la estructura de β cuando `gap_plus` $= 5$ y `gap_minus` $= 3$.

Del Algoritmo 3, se observa que en el data set `varsXpress` se deposita la información de 'slots' de todas las secciones. Aún más, cada observación del data set `varsXpress` corresponde a un 'slot', por lo tanto cada observación se numera consecutivamente, obteniendo así un identificador para cada 'slot' generado. La Tabla 16 de la página 30, muestra una parte de los 'slots' generados para las secciones 1394 y 1401.

Como `pickRho` $=$ `pickPseudo` $= 2$, cada sección tiene a los sumo cuatro salones

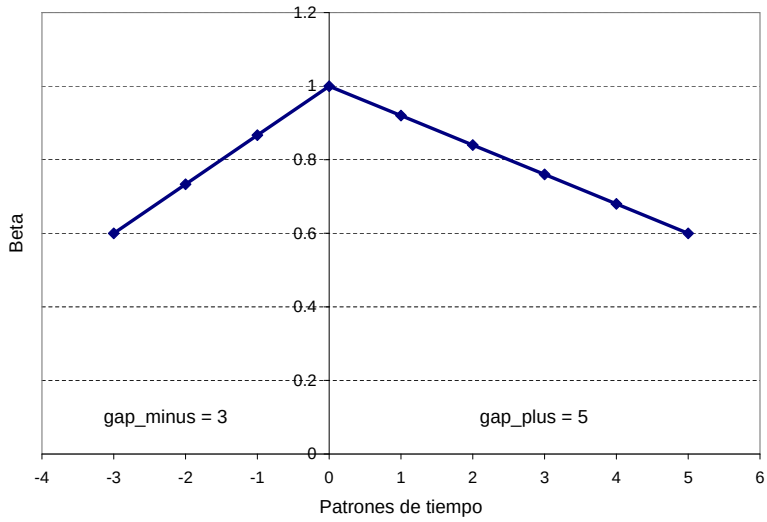

Figura 2: β vs. Patrones de tiempo

candidatos, por otro lado, como `gap_plus` = `gap_minus` = 2, entonces cada sección puede tener como máximo, cinco patrones de tiempo candidatos (el patrón de tiempo original más cuatro patrones desplazados con respecto al original). Por lo tanto, cada sección puede tener a lo sumo 20 'slots' disponibles, aunque este no siempre es el caso, porque hay secciones tan grandes que tienen menos salones factibles a los que se requieren. De otro lado, hay secciones que se dictan a horas cercanas a las de apertura o cierre de los salones, o secciones de maestría que se dictan alrededor de la hora de inicio de estas clases, lo que reduce el numero de patrones de tiempo candidatos y por ende el de 'slots'.

3.1.2.3. Matrices de conflicto

Las combinaciones de salones y patrones de tiempo que se generan, ocasionan la aparición de conflictos de salón (mas de una clase simultáneamente en un mismo salón) y conflictos de profesor (a un profesor se le asigna mas de una clase simultáneamente), es decir hay 'slots' que están en conflicto y que por lo tanto no se

idseccion	spriden_id	salon	capacidad	capseccion	materia
1394		B202	110	80	201
1394		B202	110	80	201
1394		B202	110	80	201
1394		B202	110	80	201
1401	2942563	AU101	48	40	300
1401	2942563	AU101	48	40	300
1401	2942563	AU101	48	40	300
1401	2942563	AU101	48	40	300

rho	horaAbreSeg	horaCierraSeg	asignacion	horainicio	horafinal
0.872727273	25200	75600	1	1000	1050
0.872727273	25200	75600	1	930	1020
0.872727273	25200	75600	1	900	950
0.872727273	25200	75600	1	1030	1120
1	25200	75600	1	1100	1220
1	25200	75600	1	1030	1150
1	25200	75600	1	1000	1120
1	25200	75600	1	1130	1250

horaCeroSeg	horaFinalSeg	tiempoClaseSeg	lunes	martes	miercoles
36000	39000	3000	1	0	1
34200	37200	3000	1	0	1
32400	35400	3000	1	0	1
37800	40800	3000	1	0	1
39600	44400	4800	0	0	1
37800	42600	4800	0	0	1
36000	40800	4800	0	0	1
41400	46200	4800	0	0	1

jueves	viernes	sabado	alpha	beta	slotId
0	1	0	95.67272727	1	18481
0	1	0	82.47272727	0.8	18482
0	1	0	69.27272727	0.6	18483
0	1	0	82.47272727	0.8	18484
0	1	0	100	1	18613
0	1	0	86.8	0.8	18614
0	1	0	73.6	0.6	18615
0	1	0	86.8	0.8	18616

Tabla 16: Fracción del data set `varsXpress` que corresponde a las secciones 1394 y 1401 con 'slots' calculados.

pueden asignar a la vez. Para manejar esta situación, y poder generar las restricciones (4), (5) y (6) se generan las matrices de conflicto, representadas en los data sets `slots_Cruce` para las restricciones (4) y (5) y `slotsCruceNR` para las restricciones (6). La estructura de estos dos data sets es la misma y se muestra en la Tabla 17 de la página 31. Cada fila de estos data sets representa una asignacion que no se puede hacer simultáneamente. Esto se puede representar en dos matrices booleanas (una para cada data set), donde la entrada $(i,\ j)$ corresponde a una observación

de alguno de estos data sets, la cual toma el valor de *true* indicando que los 'slots' están en conflicto. Por ejemplo, los 'slots' 18481 y 1543 nos son factibles si se asignan simultáneamente, por lo tanto la entrada (18481, 1543) de las matrices de conflicto es *true*.

En las Secciones 3.1.3 y 3.1.4 se explica detalladamente el código en SAS® para generar los data sets que representan conflictos de 'slots'.

slotId	slotIdCruce
18481	1543
18481	1544
18481	1545
18613	117
18613	120
18613	121

Tabla 17: Algunas observaciones del data set `slots_Cruce`.

3.1.2.4. Escritura de la instancia para Xpress-MP

Una instancia de este problema consta de cuatro archivos, Tabla 18 página 32.

`seccEsp.txt` Este archivo lista las secciones que tienen salón pre-asignado, y se genera a partir del data set `secciones`, imprimiendo solamente las observaciones que tienen salón asignado.

`slotsCfl1.txt` y `slotsCfl2.txt` Estos archivos corresponden a la impresión de los data sets `slots_Cruce` y `slotsCruceNR` respectivamente, con la palabra *true* impresa al frente de cada observación, indicando que los 'slots' están en conflicto.

`alpha.txt` Este archivo lista los coeficientes de la función objetivo α_{sl}. Para generar

este archivo, se toman la variables `idseccion`, `slotId` y `alpha`, de cada una de las observaciones del data set de la Tabla 16 en la página 30.

slotsCfl1.txt

```
matrizCfl1:  [(1 1 ) true
              (1 2 ) true
              (1 3 ) true
              .
              .
              .
              (21001 35374 ) true
              (21001 35375 ) true
              (21002 1978 ) true
              (21002 1979 ) true
              .
              .
              .
              (35834 35834 ) true]
```

seccEsp.txt

```
seccEsp:  [85
           86
           88
           .
           .
           .
           1394
           1418
           1419
           1420
           .
           .
           .
           2406 ]
```

slotsCfl2.txt

```
matrizCfl2:  [(1 2 ) true
              (1 3 ) true
              (1 5353 ) true
              .
              .
              .
              (35824 35834 ) true
              (35825 35826 ) true
              (35825 35827 ) true
              (35825 35828 ) true
              .
              .
              .
              (35833 35834 ) true]
```

alpha.txt

```
alpha:   [(1 1 ) 97
          (1 2 ) 83
          (1 3 ) 70
          .
          .
          .
          (1393 18479 ) 76
          (1393 18480 ) 63
          (1394 18481 ) 96
          (1394 18482 ) 82
          .
          .
          .
          (2406 35834 ) 84 ]
```

Tabla 18: Archivos de la instancia para Xpress-MP.

3.1.3. Multiples patrones de tiempo por sección

En esta Sección se trata una de las primeras variantes del código de preprocesamiento, y es cuando se consideran multiples patrones de tiempo por cada sección a asignar. Esto puede ocasionar conflictos de profesor, y por lo tanto la variable `spriden_id` (profesor) del archivo de `secciones` se debe tener en cuenta.

Cuatro de los pasos del Algoritmo 1 en la página 15, cambian si se tiene en cuenta el profesor o no, y corresponden a las subsecciones que se explican en esta parte.

3.1.3.1. Simplificación (compresión) del data set `secciones_bruto` y generación del data set `secciones`

En el data set `secciones_bruto` es se lee el archivo `phyyyypp.txt`. Este data set se transforma en el data set `secciones_neto`, después que se eliminan ciertas observaciones, Sección 3.1.2.2. Posteriormente, `secciones_neto` se divide en tres data sets, cada uno de los cuales tiene una estructura de datos diferente. La estructura de datos al interior de éstos, permite agrupar (comprimir) ciertas observaciones, que para efectos de la asignación de un salón, constituyen una sección (`idseccion`). La Figura 3 de la página 34, muestra el diagrama de flujo del proceso de división y compresión de cada una de las partes del data set `secciones_neto`.

El código de separación, esta diseñado para generar un subconjunto de observaciones (data set intermedio) del data set inmediatamente superior, el cual es mutuamente excluyente (en cuanto a observaciones) con el data set que está a la izquierda, y al mismo nivel del que se generó. Por ejemplo, ninguna de las observaciones del `data set intermedio 1`, está en el data set `secciones_Magistrales1`; la misma relación existe entre el `data set intermedio 2` y el data set `secciones_Magistrales2`. Una vez que en cada partición se han agrupado (comprimido) las observaciones correspondientes, los resultados de las compresiones se agrupan, para así generar el data set `secciones`. En este data set cada observación corresponde a una sección a la cual se le debe asignar un salón.

En el data set `secciones_Magistrales1`, se tienen en cuenta observaciones del data set `secciones_neto` con el mismo salón pre-asignado, que conforman una clase

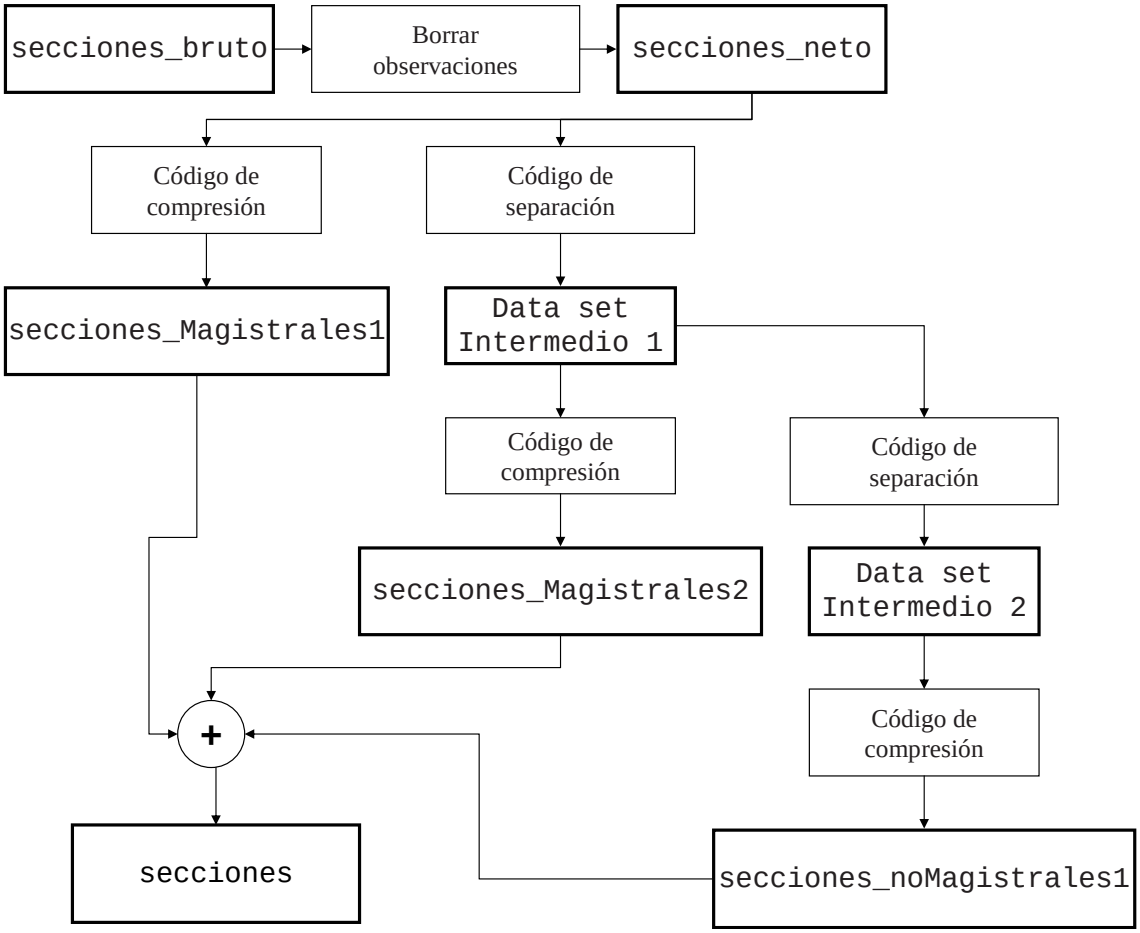

Figura 3: Diagrama de flujo del proceso de división y compactación del data set `secciones_neto` cuando se conoce el profesor.

magistral de la misma materia. La instrucción en SQL, correspondiente al código de compresión que genera el data set `secciones_Magistrales1`, se transcribe en el Apéndice A, porque explica de forma sintética el procedimiento de compactación.

En la Tabla 19 de la página 35, se muestra un ejemplo de cuatro CRNs (cuatro secciones) que se reúnen en el O104 para la clase magistral de Física III. Estas cuatro observaciones se comprimen en una sola, la cual se muestra en la misma Tabla. Esta única observación, recoge la información de las cuatro secciones, en particular las cuatro secciones reunidas suman 88 estudiantes; tienen el mismo patrón de tiempo;

y el mismo profesor. Estos hechos se trasladan a la observación, resultado de la compresión.

Clase magistral de Física III del data set secciones_neto

nombre	salon	lunes	martes	miercoles	jueves
FÍSICA III	O104	1	1	0	1
FÍSICA III	O104	1	1	0	1
FÍSICA III	O104	1	1	0	1
FÍSICA III	O104	1	1	0	1

viernes	sabado	crn	depto	materia	seccion
1	0	12398	FISI	103	21
1	0	12399	FISI	103	22
1	0	12401	FISI	103	24
1	0	12402	FISI	103	25

creditos	capseccion	inscritos	capresidual	horainicio	horafinal
3	22	0	22	1100	1150
3	22	0	22	1100	1150
3	22	0	22	1100	1150
3	22	0	22	1100	1150

spriden_id	salonB	horaCeroSeg	horaFinalSeg	tiempoClaseSeg	idseccion
41382968	1	39600	42600	3000	1144
41382968	1	39600	42600	3000	1146
41382968	1	39600	42600	3000	1150
41382968	1	39600	42600	3000	1152

Compresión de la clase magistral en el data set secciones_Magistrales1

lunes	martes	miercoles	jueves	viernes	sabado
1	1	0	1	1	0

capseccion	horainicio	horafinal	salonB	idseccion	horaCeroSeg
88	1100	1150	1	1144	39600

horaFinalSeg	tiempoClaseSeg	spriden_id	materia
42600	3000	41382968	103

Tabla 19: Compresión en el data set `secciones_Magistrales1`.

Para generar el data set `secciones_magistrales2`, se genera el data set intermedio 1: `secciones_neto_particion`. Éste data set contiene el resto de observaciones que no fueron tenidas en cuenta en el data set `secciones_magistrales2`. Una vez se dispone del data set `secciones_neto_particion`, se genera el data set

secciones_magistrales2 con un código de compactación que comprime las sesiones magistrales donde se reúnen materias diferentes. En el Apéndice A se encuentra el código en SQL que genera el data set secciones_magistrales2. En la Tabla 20 de la página 36, se muestra un ejemplo de esta situación. Estas líneas se borraron, porque el salón LL102 no está en el inventario de salones, y por lo tanto no llegaron hasta la instancia de la compresión.

nombre	salon	lunes	martes
SEMINARIO 1 MATERIA CONDENSADA	LL102	0	1
SEMINARIO 2 MATERIA CONDENSADA	LL102	0	1

viernes	sabado	crn	depto
0	0	12537	FISI
0	0	12538	FISI

creditos	capseccion	inscritos	capresidual
2	30	0	30
2	30	0	30

spriden_id	salonB	horaCeroSeg	horaFinalSeg
19299653	1	50400	57000
19299653	1	50400	57000

Tabla 20: Sesión magistral de materias diferentes que se comprimirían.

La Tabla 21 de la página 37, presenta las dos únicas observaciones del data set secciones_neto_particion que se comprimieron, y el resultado de la compresión. Las observaciones que se comprimieron son de materias totalmente diferentes (en contraste con las de la Tabla 20, que son la misma materia, pero ofrecida a alumnos de semestres diferentes) y su compresión en una sola sección para propósitos de asignación de salón es probablemente incorrecta. Aislar este tipo de situación es muy difícil, porque las observaciones que se comprimieron cumplen con todas las características de una sesión magistral. Además, sí hay casos donde materias diferente se reúnen en un mismo salón, con el mismo patrón de tiempo, y con el mismo

profesor, Tabla 20.

Observaciones del data set secciones_neto_particion a comprimir

nombre	salon	lunes	martes	miercoles	jueves
ESPECIALIZ.AUTOMATIZACION	W580	0	0	0	1
ESPECIALIZ.SISTEMAS TRANSMISIO	W580	0	0	0	1

viernes	sabado	crn	depto	materia	creditos
1	0	12027	ESPE	922	0
1	0	14201	ESPE	925	0

capseccion	inscritos	capresidual	horainicio	horafinal	salonB
20	0	20	900	1750	1
25	0	25	900	1750	1

idseccion	horaCeroSeg	horaFinalSeg	tiempoClaseSeg	spriden_id
798	32400	64200	31800	
801	32400	64200	31800	

Compresión de las observaciones de arriba en el data set secciones_Magistrales2

lunes	martes	miercoles	jueves	viernes	sabado
0	0	0	1	1	0

capseccion	horainicio	horafinal	salonB	idseccion	horaCeroSeg
45	900	1750	1	798	32400

horaFinalSeg	tiempoClaseSeg	spriden_id	materia
64200	31800		922

Tabla 21: Compresión en el data set `secciones_Magistrales2`.

La tercera partición del data set `secciones_neto`, corresponde a las observaciones que no están ni en `secciones_Magistrales1`, ni en `secciones_Magistrales2`, las cuales se guardan en el data set intermedio 2: `secciones_noMagistrales`. El código de compresión para pasar de éste data set a `secciones_noMagistrales1`, se presenta en el Apéndice A. En el data set `secciones_noMagistrales` se comprimen las clases de la misma materia que aunque se dictan a la misma hora tienen salones diferentes para cada sesión de clase; clases de la misma materia que se dictan a la misma hora, pero unas sesiones tiene salón asignado y las otras no; clases de la misma materia que tienen sesiones en días diferentes a la misma hora, además de otras sesiones que se dictan a una hora diferente a la de las demás sesiones; y finalmente las clases cuya información está en una sola linea del data set

secciones_noMagistrales. Cada uno de estos cuatro casos se presenta en la Tabla 22 de la página 38.

Observaciones del data set secciones_noMagistrales para comprimir

nombre	salon	lunes	martes	miercoles	jueves
BIOLOGIA CELULAR-TEO	G104	1	0	0	0
BIOLOGIA CELULAR-TEO	B202	0	1	1	0
INTRODUC. A ING.DE SISTEMAS	O101	0	1	0	0
INTRODUC. A ING.DE SISTEMAS		0	0	0	1
ANÁLISIS QUÍMICO	O105	0	1	1	1
ANÁLISIS QUÍMICO		0	0	1	0
ANÁLISIS QUÍMICO		0	0	0	1
INTROD.ADMINISTRACION (ADMON)		1	0	1	0

viernes	sabado	crn	depto	materia	creditos
0	0	12903	MBIO	110	3
1	0	12903	MBIO	110	3
0	0	11687	ISIS	100	3
0	0	11687	ISIS	100	3
1	0	13929	QUIM	261	3
0	0	13929	QUIM	261	3
0	0	13929	QUIM	261	3
1	0	13106	ADMI	110	3

capseccion	inscritos	capresidual	horainicio	horafinal	salonB
100	0	100	900	950	1
100	0	100	900	950	1
45	0	45	1530	1650	1
45	0	45	1530	1650	0
40	0	40	1000	1050	1
40	0	40	1600	1650	0
40	0	40	1600	1650	0
60	1	59	700	820	0

idseccion	horaCeroSeg	horaFinalSeg	tiempoClaseSeg	spriden_id
2263	32400	35400	3000	19082124
2264	32400	35400	3000	19082124
1803	55800	60600	4800	19280710
1804	55800	60600	4800	19280710
2397	36000	39000	3000	19161874
2398	57600	60600	3000	19161874
2399	57600	60600	3000	19161874
1	25200	30000	4800	

Tabla 22: Observaciones del data set secciones_noMagistrales para comprimir.

Para el primer caso (BIOLOGIA CELULAR-TEO), la clase siempre se dicta a la misma hora, pero en salones diferentes. Para propósitos de asignación, los dos patrones de tiempo originales se pueden comprimir en uno y llevar la contabilidad de los salones que habían sido pre-asignados en otro archivo. La clase de INTRODUC.

A ING.DE SISTEMAS, tiene todas sus sesiones a la misma hora, pero unas tienen salón y las otras no. Los patrones de tiempo multiples se pueden comprimir en uno solo, y llevar la contabilidad del salón pre-asignado. ANÁLISIS QUÍMICO tiene tres observaciones. Dos de estas se dictan a la misma hora, lo cual se puede comprimir en un solo patrón de tiempo. La tercera observación tiene el salón O105 asignado y no se comprime. Finalmente, la clase de INTROD.ADMINISTRACION (ADMON) no se comprime. Los resultados de la compresión de las observaciones de la Tabla 22, se muestran en la Tabla 23 de la página 39.

Observaciones comprimidas en el data set secciones_noMagistrales1

lunes	martes	miercoles	jueves	viernes	sabado
1	1	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
0	0	1	1	0	0
1	0	1	0	1	0

capseccion	horainicio	horafinal	salonB	idseccion	horaCeroSeg
100	900	950	1	2263	32400
45	1530	1650	1	1803	55800
40	1000	1050	1	2397	36000
40	1600	1650	0	2398	57600
60	700	820	0	1	25200

horaFinalSeg	tiempoClaseSeg	spriden_id	materia
35400	3000	19082124	110
60600	4800	19280710	100
39000	3000	19161874	261
60600	3000	19161874	261
30000	4800		110

Tabla 23: Observaciones del data set `secciones_noMagistrales1`, resultado de comprimir la Tabla 22.

Cuando los data sets `secciones_Magistrales1`, `secciones_Magistrales2` y

`secciones_noMagistrales1` se calculan, todas sus observaciones se agrupan, obteniéndose el data set `secciones`. Para todas las tres particiones, y las posteriores compresiones de estas, siempre se lleva contabilidad de las secciones que tiene salón pre-asignado. Estas cuentas se depositan finalmente en el data set `seccVsSalon`, donde para cada `idseccion` (del data set `secciones`) se sabe que salón tiene pre-asignado. La parte del data set `seccVsSalon` que involucra a las secciones de las Tablas 19, 21 y 23, se muestra en la Tabla 24 de la página 24.

idseccion	salon
1144	O104
798	W580
2263	B202
2263	G104
1803	
1803	O101
2397	O105
2398	
2398	
1	

Tabla 24: Algunas observaciones del data set `seccVsSalon` tratadas en los ejemplos.

3.1.3.2. Generación del conjunto de salones factibles por sección

La generación del conjunto de salones factibles por sección (`idseccion` del data set `secciones`) se hace con los siguientes criterios:

1. Un salón se considera factible para una sección particular, si la capacidad del salón es mayor o igual al tamaño de la sección, aumentado en un porcentaje dado por el `coefAumentoSecc`.

2. Si una sección tiene salones pre-asignados estos siempre estarán en el conjunto de salones factibles, y tendrán un coeficiente de densidad de ocupación mayor o igual que 1, en un porcentaje dado por `coefAumentoPref`.

3. Para las clases de maestría siempre se busca salón en los edificios destinados para las clases de maestría. Si la capacidad aumentada de una sección de maestría, es mayor que la capacidad del salón más grande destinado para estas clases, se buscan salones en los demás edificios del campus, de tal forma que la alojen.

Los conjuntos de salones factibles para cada una de las secciones del data set `secciones` se crean en el data set `salones_factibles`, Tabla 10, página 20.

3.1.3.3. Matrices de conflicto

Una vez se tiene el data set `varsXpress` con los 'slots' de cada sección calculados, Tabla 16 página 30, se proceden a crear dos data sets que representan las asignaciones de 'slots' que no son factibles, porque están en conflicto. Un 'slot' dado tiene conflicto con otro, si tienen el mismo salón y sus patrones de tiempo se cruzan (conflicto de salón). También se presenta el conflicto de profesor, cuando dos 'slots' pertenecen a secciones que tienen el mismo profesor y sus patrones de tiempo se cruzan.

En esta parte del código de preprocesamiento, se construyen dos matrices de conflicto: `slots_Cruce` y `slotsCruceNR`. En el data set `slots_Cruce`, por cada uno de los 'slots' que se generaron, se muestran los 'slots' con los que este tiene conflicto (de profesor ó de salón), de tal forma que las restricciones (4) y (5) se puedan generar. En la Tabla 25 de la página 42 se muestra la estructura de este data set. Asimismo, el código en SQL que genera este data set se presenta en el Apéndice B, porque de manera compacta define el procedimiento para hallar 'slots' en conflicto.

El data set `slotsCruceNR` se genera a partir del data set `slots_Cruce`, tomando únicamente los pares de 'slots' en conflicto para generar las restricciones (6). El data set `slots_Cruce` se presenta en la Tabla 25 y el código en SQL para se creación

slots_Cruce

slotId	slotIdCruce
1	1
1	2
1	3
1	5353
1	5354
.	.
.	.
.	.
1	31997
1	31998
1	35632
2	1
2	2
2	3
2	5353
2	5354
.	.
.	.
.	.
2	31999
2	35631
2	35632
.	.
.	.
.	.

slotsCruceNR

slotId	slotIdCruce
1	2
1	3
1	5353
1	5354
1	5355
.	.
.	.
1	31997
1	31998
1	35632
2	3
2	5353
2	5354
2	5355
2	11091
.	.
.	.
2	31999
2	35631
2	35632
.	.
.	.
.	.
.	.
.	.

Tabla 25: Data set de 'slots' en conflicto

está en el Apéndice B.

De la Tabla 25 se observa la diferencia estructural entre estos data sets. En el data set `slots_Cruce` se verifica el hecho que todo 'slot' está en conflicto con si mismo, y que las líneas `(1, 2)` y `(2, 1)` aunque representan el mismo par conflictivo, desde el punto de vista de las restricciones (4) y (5) son diferentes: en el primer caso el 'slot' 2 pertenece al conjunto de 'slots' conflictivos del 1, mientras que en el segundo, el 'slot' 1 es miembro del conjunto de slots conflictivos con el 2. Finalmente, se observa que desde el punto de vista de las restricciones (6), el par conflictivo `(1, 2)` es lo mismo que el `(2, 1)`, por lo tanto solo se considera un solo par en el data set

`slotsCruceNR`, y no se considera el conflicto de un 'slot' con si mismo.

3.1.4. Un único patrón de tiempo por sección

En esta Sección se considera la segunda variante del código de preprocesamiento, y es cuando se considera un único patrón de tiempo por cada sección a asignar. Si este es el caso, se puede suponer que los patrones de tiempo de las secciones del archivo `phyyyypp.txt` no tienen conflictos de profesor.

En las instancias donde solo se considera un patrón de tiempo por sección, se introducen algunos cambios en el procedimiento con el que se generan:

1. Las secciones que tienen salón pre-asignado se fijan. Es decir, estas secciones ya quedan asignadas al salón pre-asignado que tengan.

2. Se asignan los salones de las secciones con salón pre-asignado a las secciones sin salón, siempre y cuando no se presente conflicto de salón.

3. Se introduce un algoritmo de remoción de conflictos de salón para secciones con salón pre-asignado. En la Tabla 21 de la página 37, se presenta el caso de dos secciones con el mismo salón pre-asignado, que tienen conflicto de salón.

4. Se asume que el archivo `phyyyypp.txt` no tiene conflictos de profesor, por lo tanto no es necesario considerar el profesor, pues no se generaran múltiples patrones de tiempo.

5. Se asume que las secciones de maestría, de primer semestre, magistrales y secciones con requerimientos especiales, tienen salón pre-asignado.

3.1.4.1. Simplificación (compresión) del data set `secciones_bruto` y generación del data set `secciones`

El proceso de compresión de observaciones, Figura 4 página 45, inicia tomando el data set `secciones_neto` que es el resultado de eliminar ciertas observaciones del data set `secciones_bruto`, de la misma manera que en la Sección 3.1.3.1. Después, el data set `secciones_neto` se divide en dos partes mutuamente excluyentes: una parte tiene las observaciones con salón pre-asignado y la otra tiene las observaciones sin salón pre-asignado. Dentro de las observaciones con salón pre-asignado, se tiene dos grupos. En el primero, se comprimen observaciones del data set `secciones_neto` con el mismo salón pre-asignado, que conforman una clase magistral de la misma materia (`secciones_pre_asignado1`). El segundo grupo, corresponde a observaciones del data set `secciones_neto` con el mismo salón pre-asignado, que conforman una clase magistral de materias diferentes (`secciones_pre_asignado2`). A las observaciones del data set `secciones_neto` que no tienen salón pre-asignado no se les hace ningún tratamiento (`secciones_sin_salon`). Finalmente, los data sets `secciones_pre_asignado1` y `secciones_pre_asignado2` se agrupan, y posteriormente se remueven las secciones que causen conflictos de salón, para formar el data set `secciones_pre_asignado`. Este data set y el data set `secciones_sin_salon` se agrupan en el data set `secciones`. El código en SQL que divide `secciones_neto` y comprime sus particiones, se presenta en el Apéndice C.

El algoritmo que remueve las secciones que ocasionan conflictos de salón, toma los salones donde hay conflictos y solamente deja la sección que tenga la mayor capacidad. De esta manera se garantiza que las secciones más grandes tendrán salón.

En la Tabla 19 de la página 35, se ilustra como unas observaciones del data set `secciones_neto` quedarían comprimidas en el data set `secciones_pre_asignado1`.

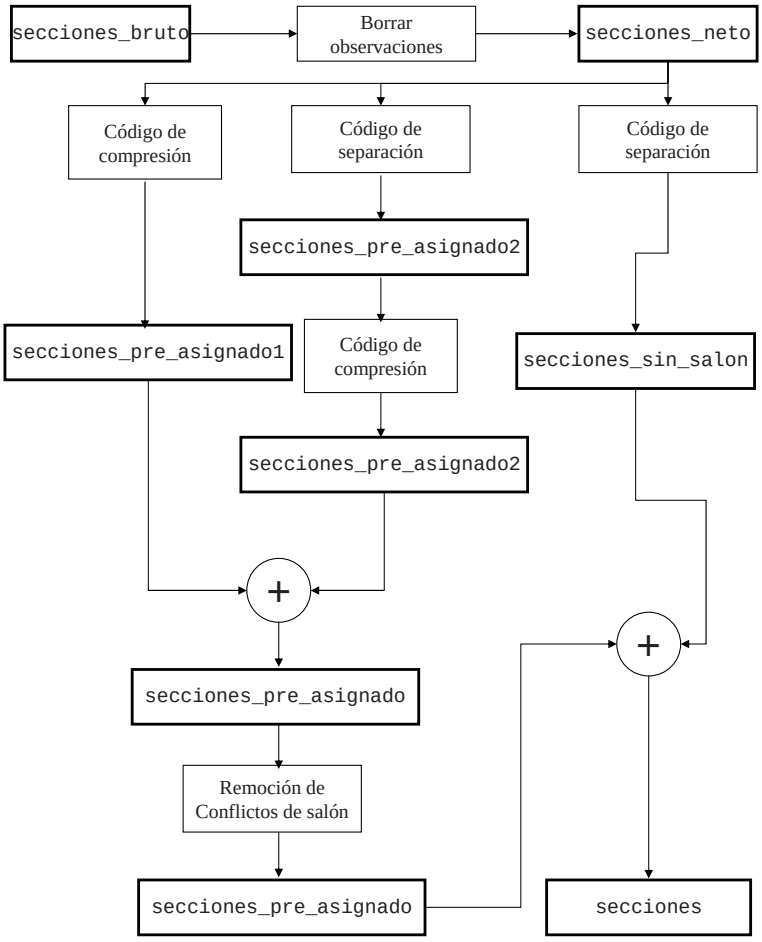

Figura 4: Diagrama de flujo del proceso de división y compactación del data set `secciones_neto` cuando se genera un solo patrón de tiempo.

De la misma manera, las observaciones del data set `secciones_neto` de la Tabla 21 en la página 37, se compactan como se muestra allí, pero esta vez en el data set `secciones_pre_asignado2`.

Para ilustrar las diferencias en los algoritmos de compresión hasta ahora presentados, se consideran las observaciones del data set `secciones_neto` de la Tabla 26 en la página 46, y las observaciones resultado de la compresión, cuando se tiene en cuenta el profesor en la compactación y cuando no. Se advierte que cada observación en el data set `secciones_neto` la dicta un profesor diferente. Además, ambas

tienen el mismo patrón de tiempo y comparten el mismo salón. El hecho que cada sección tenga un profesor diferente, hace que cuando se considera el profesor en la compactación, estas observaciones no se compriman. De otro lado, si no se considera el profesor, el algoritmo comprime las dos observaciones en una, pues se trata de dos secciones de materias diferentes que comparten el mismo salón.

Observaciones del data set secciones_neto a comprimir

nombre	salon	lunes	martes	miercoles	jueves
PRACT.DOCENTE O ADMTVA 1	AU210	1	0	0	0
PRACT.DOCENTE O ADMTVA 2	AU210	1	0	0	0

viernes	sabado	crn	depto	materia	seccion
0	0	12714	EDUC	400	1
0	0	12716	EDUC	400	1

creditos	capseccion	inscritos	capresidual	horainicio	horafinal
6	10	0	10	1700	1950
6	10	0	10	1700	1950

spriden_id	salonB	horaCeroSeg	horaFinalSeg	tiempoClaseSeg	idseccion
41631058	1	61200	71400	10200	726
51872284	1	61200	71400	10200	727

Observaciones en al data set secciones_noMagistrales1

lunes	martes	miercoles	jueves	viernes	sabado
1	0	0	0	0	0
1	0	0	0	0	0

capseccion	horainicio	horafinal	salonB	idseccion	horaCeroSeg
10	1700	1950	1	726	61200
10	1700	1950	1	727	61200

horaFinalSeg	tiempoClaseSeg	spriden_id	materia
71400	10200	41631058	400
71400	10200	51872284	400

Observación en el data set secciones_pre_asignado2

lunes	martes	miercoles	jueves	viernes	sabado
1	0	0	0	0	0

capseccion	horainicio	horafinal	salonB	idseccion	horaCeroSeg
20	1700	1950	1	726	61200

horaFinalSeg	tiempoClaseSeg	spriden_id	materia
71400	10200	51872284	400

Tabla 26: Diferencia en la manera de compactación cuando se consideran multiples patrones de tiempo (se tiene en cuenta el profesor) y cuando se considera un único patrón de tiempo para cada sección.

3.1.4.2. Generación del conjunto de salones factibles por sección

La generación del conjunto de salones factibles por sección, sigue los siguientes criterios:

1. Un salón se considera factible para una sección particular, si la capacidad del salón es mayor o igual al tamaño de la sección, aumentado en un porcentaje dado por el `coefAumentoSecc`.

2. Las secciones que tienen salón pre-asignado se fijan. Es decir, estas secciones ya quedan asignadas al salón pre-asignado que tengan.

3. Los salones de las secciones con salón pre-asignado se asignan a las secciones sin salón, siempre y cuando no se presente conflicto de salón.

4. Se asume que las secciones de maestría, de primer semestre, magistrales y secciones con requerimientos especiales, tienen salón pre-asignado. Por lo tanto, no se considera el caso especial de buscar salones en ciertos edificios para las clases de maestría, pues se supone que ya lo tienen asignado.

3.1.4.3. Matrices de conflicto

El cálculo de las matrices de conflicto de 'slots' tiene el mismo algoritmo que se explica en la Sección 3.1.3.3 de la página 41, solo que en este caso no se consideran los conflictos de profesor, únicamente se consideran los conflictos de salón. Este hecho se ve reflejado en el código de SQL para la generación de las matrices de conflictos de 'slots', que se presenta en el Apéndice D.

3.1.5. Generación de instancias diarias

Las instancias que se han discutido hasta el momento consideran que todas las sesiones de clase de una sección dada, se dictan en el mismo salón. Este no necesariamente debe ser el caso, pues se puede considerar un modelo de horario, donde las sesiones se dicten en salones diferentes. Esto se logra si se considera la generación de seis instancias diarias (una instancia para cada día de clase) en lugar de una instancia semanal, como se venía haciendo. El Algoritmo 4 de la página 49, muestra las modificaciones que se deben implementar en el Algoritmo 1, para generar instancias diarias.

3.2. Modelo matemático en Xpress-MP

3.2.1. Formulación 1

El modelo matemático (1)-(5), se implementó en Xpress-MP. El código se muestra en el Apéndice E.

Varias características de esta implementación se explican en lo que sigue. Se define el procedimiento `forward procedure printsol`, para imprimir ciertos datos cada vez que se encuentra una solución entera, `setcallback(XPRS_CB_UIS,"printsol")` [2]. Uno de los argumentos de este "callback" es `XPRS_CB_UIS`, lo que hace que en cada solución entera se ejecuten las instrucciones del procedimiento `printsol`. En particular, en el archivo `printsol.dat` se imprimen: `XPRS_BESTBOUND`, la mejor cota determinada hasta el momento; `XPRS_MIPOBJVAL`, el valor de la función objetivo de la mejor solución entera encontrada hasta el momento; y el gap relativo, `100*abs(XPRS_MIPOBJVAL-XPRS_BESTBOUND)/XPRS_BESTBOUND`. La lista completa de atributos de un problema, y su definición se encuentran en [5]. Por último, se imprime la solución entera que se encontró.

Algoritmo 4 Algoritmo de preprocesamiento para la generación de instancias diarias

Entrada: Archivos: `phyyyypp.txt`, `sayyyypp.prn` y `edificios.txt`. Argumentos: `pickRho`, `pickPseudo`, `pseudoCap`, `gap_plus`, `gap_tiempo_plus`, `gap_minus`, `gap_tiempo_minus`, `coefAumentoSecc`, `coefAumentoPref`, `horaInicioMag` y `lambda`

Salida: `seccEsp_`d`.dat`, `slotsCfl1_`d`.dat`, `slotsCfl2_`d`.dat`, `alpha_`d`.dat`; para cada día $d = 1,\ 2.\ ...,\ 6$

1: Lectura de archivos
2: Eliminación de observaciones del data set `secciones_bruto`
3: Simplificación (compresión) del data set `secciones_bruto` y generación del data set `secciones`
4: **for** $d = 1$ to 6 **do**
5: Filtrar las observaciones del día d del data set `secciones`
6: Generación del conjunto de salones factibles de las secciones del día d
7: Generación del conjunto de salones candidatos de las secciones del día d
8: Generación del conjunto de patrones de tiempo candidatos de las secciones del día d y generación de 'slots'
9: Matriz de conflicto de 'slots' del día d
10: Matriz de conflicto de 'slots' por parejas del día d
11: Escritura de `seccEsp_`d`.dat`
12: Escritura de `slotsCfl1_`d`.dat`
13: Escritura de `slotsCfl2_`d`.dat`
14: Escritura de `alpha_`d`.dat`
15: **end for**

La matriz de conflictos `matrizCfl1`, se declara de tipo `boolean` para ahorrar memoria. Además, todos los elementos del bloque `declarations` son dinámicos, pues en el momento de su declaración no se conoce su tamaño. El tamaño de los elementos que allí se declaran, solo se define hasta la ejecución de los bloques `initializations`. Posteriormente, con la instrucción `create(x(s,l))`, solo se crean las variables de decision para las cuales `alpha` está definido. Las instrucciones `finalize(secciones)`, `finalize(slots)` y `finalize(seccEsp)`, convierten los respectivos conjuntos dinámicos, en conjuntos constantes. De esta manera, los arreglos que estos conjuntos indexan, se vuelven estáticos. Siempre que sea posible, es mejor usar arreglos estáticos que dinámicos, porque Mosel maneja más eficientemente los estáticos [4].

Las restricciones (4) y (5) se escriben como un solo conjunto de restricciones, donde en cada restricción se consideran los 'slots' que tienen conflicto, bien sea de profesor ó de salón, con el 'slot' que se fija. Esto se puede hacer, porque en la matriz de conflictos `matrizCfl1` por cada 'slot', se consideran los 'slots' con los que hay conflictos (de profesor o de salón).

La instrucción `setparam("XPRS_MIPRELSTOP", 0.05)` determina el gap relativo de parada en 5 %. Esta instrucción es muy importante en modelos donde no se puede alcanzar un gap relativo de 0 %, en un tiempo computacional razonable, pero si se puede alcanzar un gap del 5 % en un tiempo computacional prudente.

Finalmente, en el archivo `solucion.dat` se imprime las cantidades `XPRS_BESTBOUND` y `XPRS_MIPOBJVAL`; el tiempo de corrida en segundos, `gettime-starttime`; y la solución optima.

3.2.2. Formulación 2

Con el propósito de escribir la formulación (1)-(3) y (6) sin incluir las restricciones (6) simultáneamente, se propone la implementación de un algoritmo de 'Branch-and-Cut' [3, 11, 21]. Incluir explícitamente las restricciones (6) incrementa considerablemente el tamaño del problema, y aunque las restricciones (4) y (5) son suficientes para definirlo, esta formulación no es muy fuerte [3, 21], ocasionando que la solución de la relajación lineal, no arroje una muy buena aproximación de la solución entera que se quiere obtener. Para ilustrar la diferencia numérica entre las restricciones (4 y 5), y las restricciones (6), considérese una instancia con un único patrón de tiempo por sección, y `pickRho = pickPseudo = 2`, para las secciones que no tiene salón pre-asignado, es decir, cuatro salones candidatos por sección sin salón pre-asignado y las secciones con salón pre-asignado se fijan en éste. Las restricciones (4 y 5) suman 2419 restricciones, mientras que las restricciones (6), 8933. Ahora se considera una instancia semanal con `pickRho = pickPseudo = 2`, y `gap_plus = gap_minus = 2`, es decir, cuatro salones candidatos y cinco patrones de tiempo por sección. Las restricciones (4 y 5) ascienden a 35,834, mientras que las restricciones (6) suman 994,716.

Para la Formulación 2, Apéndice F, se discute un algoritmo de 'Branch-and-Cut' donde las restricciones (4) y (5) definen el problema, y se usan las restricciones (6) en un algoritmo de 'Branch-and-Cut', que ejecuta el Algoritmo 5 en cada nodo del árbol de 'Branch-and-Bound' [11].

En el programa del Apéndice F se define la función `cb_node` para adicionar cortes en cada nodo del árbol de 'Branch-and-Bound', `setcallback(XPRS_CB_CM, " cb_node")`. Antes de invocar nuestra propia estrategia de cortes, se deshabilita la generación de cortes del solver de Xpress-MP, `setparam("XPRS_CUTSTRATEGY",`

Algoritmo 5 Algoritmo de 'Branch-and-Cut'

1: Resolver la relajación lineal
2: Leer los valores de las variables de decision
3: Identificar las restricciones (6) violadas
4: Adicionar las restricciones violadas al problema
5: Resolver otra vez la relajación lineal

`0)`; `XPRS_PRESOLVE` se apaga; y con la instrucción `setparam("XPRS_EXTRAROWS", 5000)`, se reserva espacio para 5000 cortes. La función `cb_node`, únicamente adiciona cortes en los primeros tres niveles del árbol, `depth<4`; posteriormente se leen los valores de las variables de decisión y después se identifican las restricciones (6) que se violaron. Las restricciones que se violaron se guardan en un arreglo en forma normalizada (todos los términos variables y constantes al lado izquierdo de la desigualdad, `1 - x(r,k) - x(s,l)`) [11]. Mientras que en otro vector se almacena el tipo de relación de desigualdad (`CT_GEQ` para $\geq$). Una vez se tienen los vectores de restricciones violadas, estas se adicionan con el procedimiento `addcuts`, `addcuts(cutid, type, cut)`, donde `cutid` es un vector de identificadores definidos por el usuario; `type` es el vector de tipos de relación de desigualdad; y `cut` es el vector de términos de los cortes.

El prototipo de la función `cb_node` depende del tipo de callback que se use. En este caso se usa `XPRS_CB_CM`, que admite valores booleanos únicamente [2]. Entonces, en cada nodo del árbol de 'Branch-and-Bound', la función `cb_node` se llamaría repetidamente, mientras retorne `true`. El valor de retorno de `false`, hace que solo se llame una vez por nodo [3].

3.3. Métodos de solución

Hasta ahora se han discutido dos tipos de instancias para el problema de asignación de salones:

Multiples patrones de tiempo por sección. En estas instancias se considera un conjunto de patrones de tiempo por cada sección a asignar, además del patrón de tiempo que viene en el archivo `phyyyypp.txt`. Estos desplazamientos de patrones de tiempo pueden ocasionar conflictos de profesor, por lo tanto la variable `spriden_id`, se tiene en cuenta en el preprocesamiento.

Un único patrón de tiempo por sección. Cuando este es el caso, se supone que en el archivo de secciones `phyyyypp.txt`, no hay conflictos de profesor, y que las secciones con salón pre-asignado, no tienen conflictos de salón.

Para cada uno de los dos casos anteriores, se pueden generar tanto instancias semanales, como instancias diarias. Estos casos se tratarán en las secciones que siguen.

3.3.1. Solución de una instancia semanal

Para la solución de una instancia semanal, se decide si ésta tendrá multiples patrones de tiempo, ó un único patrón de tiempo por sección. Posteriormente se escoge el código de preprocesamiento adecuado para cada caso, y se genera la instancia correspondiente, la cual se resuelve en Xpress-MP. Si la instancia tiene multiples patrones de tiempo por sección, se usa el programa del Apéndice E, de otro lado, si la instancia tiene un único patrón de tiempo, no se incluyen las restricciones (2), porque estas asignaciones se efectúan en el preprocesamiento. La Figura 5 de la página 54, muestra el diagrama de flujo de este proceso.

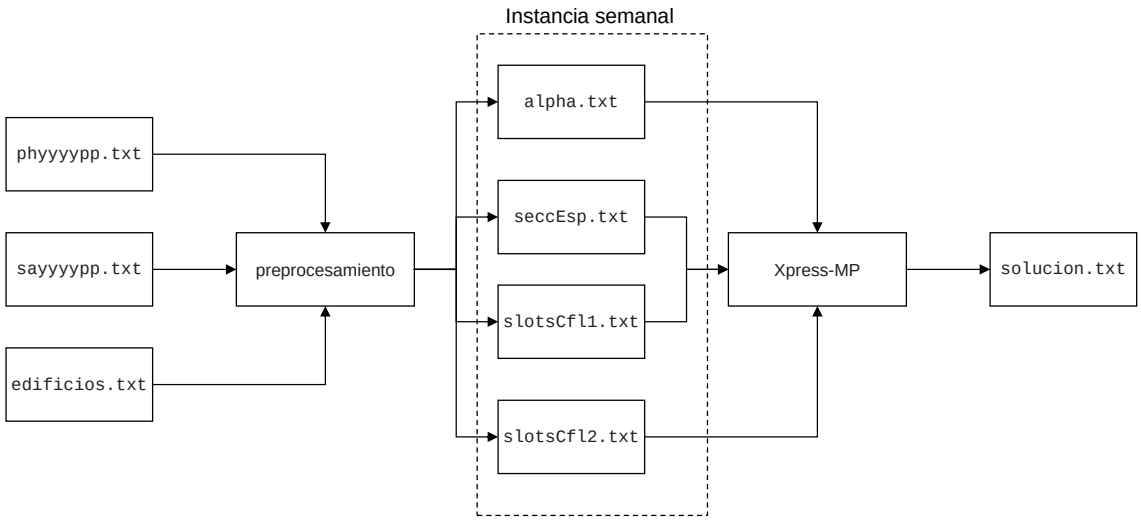

Figura 5: Método de solución de instancias semanales.

3.3.2. Solución de instancias diarias

Para la generación de instancias diarias se sigue el mismo proceso que en la Sección anterior. Naturalmente, los archivos `alpha.txt, seccEsp.txt, slotsCfl1.dat` y `slotsCfl2.dat`, se generan para cada día que hay clase. Además, se incluye un código de posprocesamiento, en el que se toman las seis soluciones (una para cada día de clase) y se calculan las secciones que efectivamente fueron asignadas en el horario que se requiere, pues estas son las que se tienen en cuenta para la elaboración del horario final. El código en SAS para el posprocesamiento de instancias diarias con un único patrón de tiempo, se presenta en al Apéndice G. La Figura 6 de la página 55, muestra el diagrama de flujo para la solución de instancias diarias.

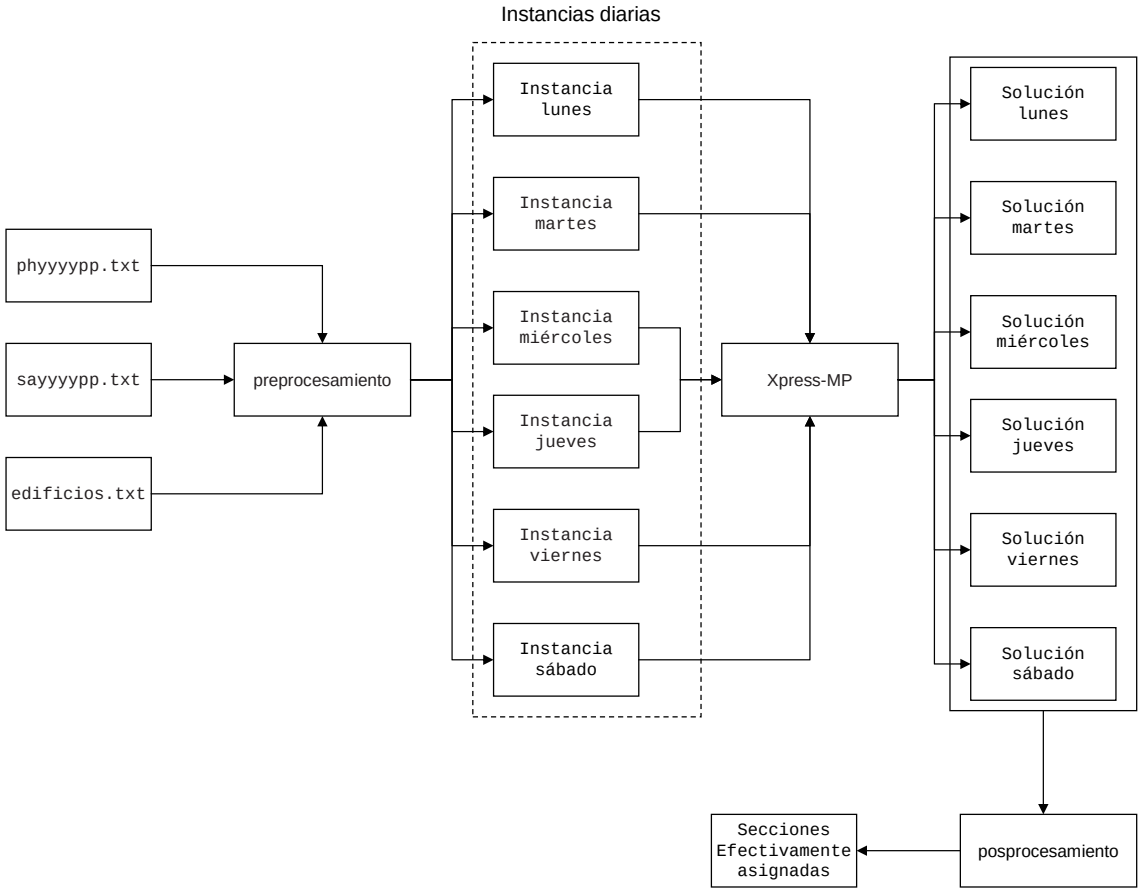

Figura 6: Método de solución de instancias diarias.

Capítulo IV

Resultados computacionales

Las pruebas de aplicación se hicieron con el archivo de secciones `ph200410.txt`, que corresponde al primer semestre del año 2004. Este archivo se generó de Banner al comienzo de la Etapa 2 de construcción del horario, la cual se describe en el Capítulo 1. Este archivo tiene 2406 líneas, que especifican exactamente el horario de cada una de las secciones que se quieren programar, además de algunas secciones, para las cuales se especifica el salón.

El inventario de salones consta de 192 elementos, de los cuales, 127 son públicos, los salones restantes, son administrados por los Departamentos de la Universidad, y por lo tanto, la Oficina de Admisiones y Registro no puede disponer de ellos. Las pruebas computacionales se hicieron con los salones públicos únicamente. Finalmente, se usó el archivo de edificios de la Figura 1 en la página 13.

Para las pruebas computacionales se resolvieron 24 instancias. Las características de cada instancia se muestran en la Tabla 27.

Para todas las instancias se consideró un único patrón de tiempo por sección; el porcentaje en el cual se aumenta el tamaño de la sección, `coefAumentoSecc`, se fijó en 0; λ, el peso relativo de los coeficientes de ajuste, se dejó en 0; y la 1:00 PM, es la hora de inicio de las clases de maestría. Con estos parámetros, se tienen 1957

	Instancias semales											
pickRho	1	2	3	4	5	12						
pickPseudo	1	2	3	4	5	12						
pseudoCap	10					2	4	6	8	10	12	24
Duración preprocesamiento (min)	26	26	26	26	26	18	18	18	18	18	18	18
Alternativas (num. 'slots')	2,870	5,539	7,094	8,307	8,587	1,548	3,515	5,045	6,513	7,876	9,670	16,828
Restricciones por 'slot'	2,870	5,539	7,094	8,307	8,587	1,548	3,515	5,045	6,513	7,876	9,670	16,828
Restricciones por pares de 'slots'	19,125	60,036	88,304	114,474	117,443	4,316	19,905	43,055	73,809	113,758	162,680	503,045
Identificador instancia	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12

	Instancias diarias											
pickRho	2	4	6	8	10	12						
pickPseudo	2	4	6	8	10	12						
pseudoCap	24					2	4	6	8	10	12	24
Duración preprocesamiento (min)	55	55	55	55	55	55	55	55	55	55	55	55
Alternativas (num. 'slots' prom.)	2,405	4,808	7,209	9,573	11,848	12,807	14,033	14,033	8,896	10,979	12,807	14,033
Restricciones por 'slot'	2,405	4,808	7,209	9,573	11,848	12,807	14,033	14,033	8,896	10,979	12,807	14,033
Restricciones por pares de 'slots'	13,365	36,822	71,827	118,137	173,778	204,786	237,814	243,745	112,909	160,671	204,778	237,831
Identificador instancia	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12

Tabla 27: Características de las instancias.

secciones para asignar, de las cuales 521 tienen salón pre-asignado. El algoritmo de remoción de conflictos de salón de secciones con salón pre-asignado, remueve el salón de cinco secciones de estas, entonces, se pre-asignan 516 secciones.

Se tienen 12 instancias semanales. Cinco de estas instancias tienen el coeficiente de pseudo-capacidad fijo en 10, y el número de salones candidatos se incrementa de dos en dos hasta 10. Para las siete instancias semanales restantes, se deja fijo el número de salones candidatos en 24, mientras se incrementa el coeficiente de pseudo-capacidad de dos en dos hasta 12, y después se salta a 24.

En cinco de las 12 instancias diarias, se fija el coeficiente de pseudo-capacidad en 24. Es importante anotar, que en la generación de instancias diarias, este coeficiente multiplica la capacidad horaria diaria de cada salón, a diferencia de las instancias semanales, donde se multiplica la capacidad horaria semanal del salón respectivo. Para las instancias con coeficiente de pseudo-capacidad fijo, los salones candidatos se incrementan de cuatro en cuatro hasta 20. Para las siete instancias diarias restantes, los salones candidatos se fijan en 24, y la pseudo-capacidad se aumenta en pasos de dos unidades hasta 12, y después se pasa a 24.

Las instancias se generaron en un computador con procesador Pentium IV de 2 GHz, y 256 MB de RAM. Para cada instancia se muestra el tiempo en minutos que tardo en generarse. Se observa que la generación de instancias diarias, dura el menos el doble que la de las instancias semanales, porque en un día se pueden generar tantas variables, como en una instancia semanal, proceso que se repite seis veces. Para las instancias semanales se muestra el número de 'slots' (alternativas) que se generaron. En las instancias diarias se muestra el promedio de 'slots' de lunes a viernes. Cabe anotar que el número de 'slots', es el número de variables del problema. La línea de restricciones por 'slot', muestra la cantidad de restricciones (4) y (5) del problema respectivo. En las instancias diarias, esta cantidad corresponde al promedio de lunes a viernes de restricciones de este estilo. Es muy importante decir, que la cantidad de restricciones (4) y (5), coincide con el número de 'slots', porque todo 'slot' se cruza al menos con si mismo. La fila de restricciones por pares de 'slots', muestra la cantidad de restricciones (6) que se generan en la instancia respectiva. Al igual que antes, esta cantidad corresponde al promedio de lunes a viernes, en el cuadro de instancias diarias.

La Figura 7, muestra el número de restricciones por 'slot', y el número de restricciones de pares de 'slots' contra el número de salones candidatos ó coeficiente de pseudo-capacidad, para las instancias de la Tabla 27.

Los conflictos por pares de 'slots', naturalmente son mayores en todos los casos que las restricciones por 'slot'. Para dos instancias, las restricciones por pares crecen exponencialmente. En las instancias s6 a s12, a medida que se aumenta el coeficiente de pseudo-capacidad, más salones se podrán asignar por tener pseudo-capacidades remanentes altas, lo que redunda en más conflictos. En las instancias d1 a d5, se observa que a medida que el número de salones candidatos aumenta, los conflictos

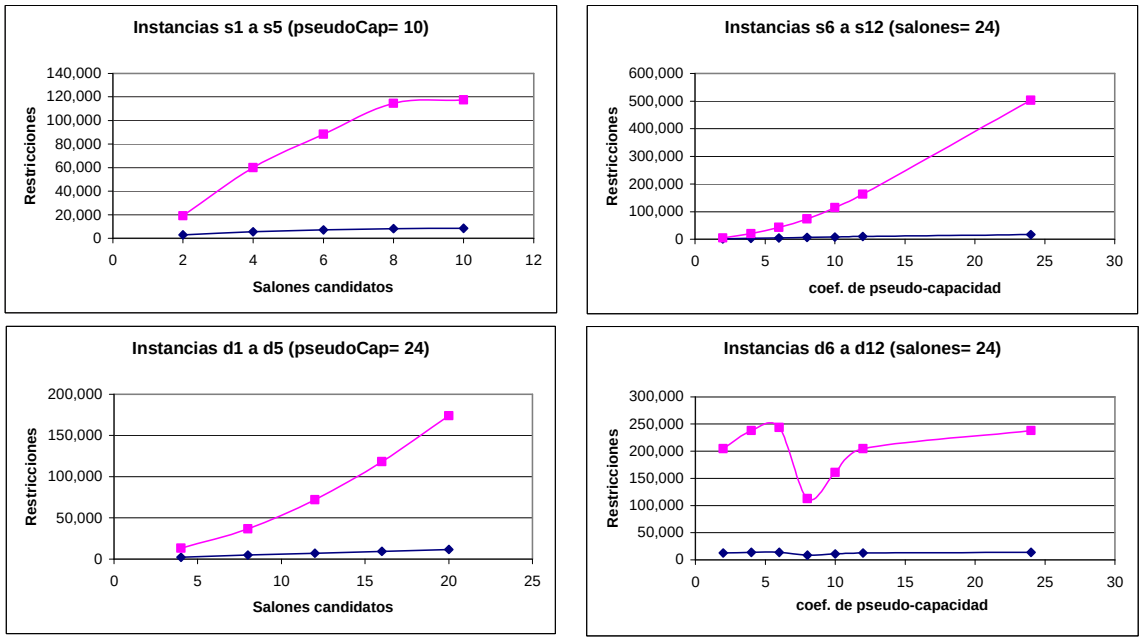

Figura 7: Número de restricciones vs. Salones ó coef. de pseudo-capacidad.

lo hacen (de esperarse), pero se demoran más en estabilizarse, como si lo hacen en las instancias s1 a s5. Esto es así, porque las asignaciones de salón en d1-d5 se hacen diariamente, entonces es más fácil encontrar conjuntos cada vez más grandes de salones candidatos por sección (si el tamaño efectivo del conjunto de salones candidatos es grande, hay más conflictos). De otro lado, como en s1-s5 las asignaciones de salón se hacen semanalmente, se llega rápidamente al conjunto máximo posible de salones candidatos, y la cantidad de restricciones se estabiliza. De la gráfica de restricciones para la instancia d6-d12, se observa que el número de restricciones por pares, es probablemente periódico, con periodo 10. El origen de esta potencial periodicidad, todavía no es claro.

Las instancias de la Tabla 27 se resolvieron con Xpress-MP versión 2004A, usando el programa matemático (1), (3) y (4). Las restricciones (2) se tuvieron en cuenta en el preprocesamiento, y como se considera un solo patrón de tiempo para cada

sección, las restricciones (5) no se incluyen, porque se asume que el archivo de secciones, `ph200410.txt`, no tiene conflictos de profesor. Todas las instancias se resolvieron en una máquina con procesador Pentium IV de 2 GHz, y 256 MB de RAM. Los resultados que se obtuvieron para cada una de las instancias generadas se muestran en la Tabla 28.

	Instancias semales											
	s1	**s2**	**s3**	**s4**	**s5**	**s6**	**s7**	**s8**	**s9**	**s10**	**s11**	**s12**
Numero de restricciones	4,305	6,974	8,448	9,655	9,934	1,894	4,223	6,073	7,750	9,223	11,091	18,263
Numero de variables	2,870	5,539	7,094	8,307	8,587	1,548	3,515	5,045	6,513	7,876	9,670	16,828
Gap	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.90%	0.37%	0.23%	1.84%
T. de ejecucíon en el optimizador (minutos)	1	2	3	4	3	1	1	1	2	3	26	29
Secciones sin salon pre-asignado que efectivamente se asignaron	293	262	262	254	250	174	223	220	225	210	216	220
% de asignación total	41.34%	39.75%	39.75%	39.35%	39.14%	35.26%	37.76%	37.61%	37.86%	37.10%	37.40%	37.61%
% de asig. de secc. Con salon preasignado	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%

	Instancias diarias											
	d1	**d2**	**d3**	**d4**	**d5**	**d6**	**d7**	**d8**	**d9**	**d10**	**d11**	**d12**
Numero de restricciones	3,007	5,410	7,811	10,174	12,449	13,408	14,635	14,635	9,498	11,581	13,399	14,635
Numero de variables	2,405	4,808	7,209	9,573	11,848	12,807	14,033	14,033	8,896	10,979	12,807	14,033
Gap	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
T. de ejecucíon en el optimizador (minutos)	5	5	5	8	13	12	13	17	8	10	13	14
Secciones sin salon pre-asignado que potencialmente se pueden asignar	937	1,192	1,196	1,155	1,121	1,105	1,119	1,120	1,064	1,089	1,096	1,105
Secciones sin salon pre-asignado que efectivamente se asignaron	551	901	919	887	866	811	856	843	755	798	815	858
% de secciones potenciales que se asignaron	58.80%	75.59%	76.84%	76.80%	77.25%	73.39%	76.50%	75.27%	70.96%	73.28%	74.36%	77.65%
% de asignación total	54.52%	72.41%	73.33%	71.69%	70.62%	67.81%	70.11%	69.44%	64.95%	67.14%	68.01%	70.21%
% de asig. de secc. Con salon preasignado	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%	99.04%

Tabla 28: Resultados.

Para las instancias semanales se muestra el número de variables y de restricciones de cada problema; en las instancias diarias estas cantidades corresponden al promedio de lunes a viernes. El gap en ambos casos, corresponde al gap relativo, y es la diferencia en valor absoluto, entre el valor de la función objetivo de la mejor solución entera encontrada hasta el momento, y la mejor cota determinada hasta el momento; dividido entre la mejor cota encontrada hasta el momento, como se define en la Sección 3.2.1. El tiempo de ejecución en el optimizador, es el tiempo que tarda Xpress-MP en resolver la instancia respectiva, para las instancias diarias, es el tiempo que se tarda en resolver los seis subproblemas. En las instancias semanales se muestran las secciones sin salón pre-asignado que se asignaron. Dentro de este conjunto, es probable que se haya asignado alguna de las cinco instancias a las cuales se les removió el salón, dado que ocasionaban conflicto. Para las instancias

diarias, las secciones sin salón pre-asignado que potencialmente se pueden asignar, corresponden a las secciones que se les encontró salón en al menos una sesión de clase. Durante el posprocesamiento se calculan las secciones que efectivamente se asignaron, es decir, las secciones a las cuales se les encontró salón para todas las sesiones de clase. El porcentaje de secciones potenciales que se asignaron, es la fracción de secciones sin salón pre-asignado que potencialmente se pueden asignar, a las que efectivamente se les encontró un salón para cada sesión de clase. Finalmente, se muestra la fracción de las 1,957 secciones a programar, que se asignaron, y la fracción de las 521 secciones con salón pre-asignado, que se programaron en este salón, las cuales ascienden a 516.

Las instancias s1 y d3 arrojaron los porcentajes de asignación total más altos, 41.3 % y 73.3 %, para instancias semanales y diarias, respectivamente. Contrario a lo que se esperaría intuitivamente, si se mantiene el coeficiente de pseudo-capacidad fijo, y se aumenta el número de salones candidatos por sección, el porcentaje total de asignación disminuye en las instancias s1-s5. Para las instancias d1-d5 el porcentaje total de asignación aumenta hasta cierto punto donde alcanza su máximo nivel, y después disminuye, Figura 8. En esta Figura se muestra el porcentaje total de asignación, y el número de secciones sin salón pre-asignado que se programaron (este dato está normalizado con respecto al máximo valor de la linea 'secciones sin salón pre-asignado que efectivamente se asignaron', del grupo de instancias respectivo) para todos los casos. Adicionalmente, para las instancias d1-d5, y d6-d12, se muestra el porcentaje de secciones potenciales que efectivamente se asignó. Para las instancias diarias, el porcentaje total de asignación en ningún caso es inferior al 50 %, en particular, para las instancias d6-d12 este porcentaje nunca es inferior al 64 %. En las instancias semanales el porcentaje total de asignación está alrededor del 38 %.

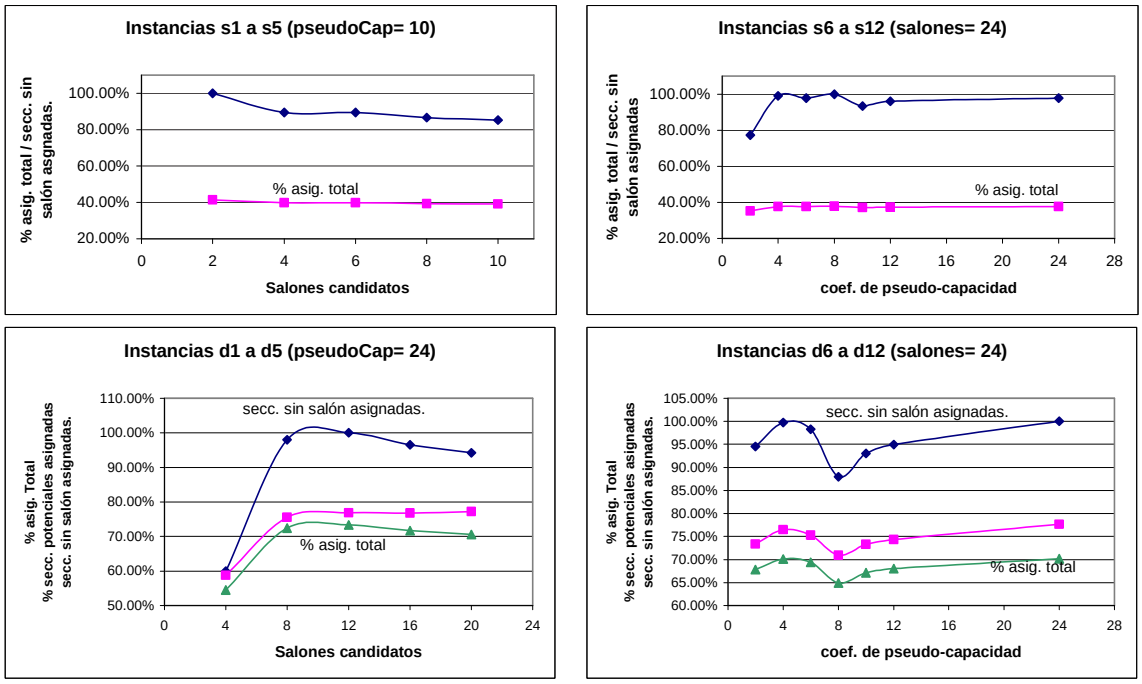

Figura 8: Resultados.

Las gráficas para las instancias d6-d12, exhiben el mismo patrón que en la Figura 7, y de la misma manera, es probable que sean periódicas, con periodo 10.

En la literatura que se consultó, solo el problema tratado por Carter [6], es comparable al de los Andes. Carter trabajó una instancia de 1400 secciones y 125 salones. La Tabla 29 muestra las estadísticas que comparan algunos problemas de asignación de salones estudiados. Las casillas que corresponden a las cantidades que no están disponibles, ó que no se pueden deducir del artículo, aparecen vacías. En la Tabla 29, la cantidad secciones / salones, corresponde al número de secciones por salón, y da una idea de que tan difícil de resolver es el problema. La instancia de los Andes tiene la segunda densidad mayor, después de la de Mooney [17], pero la instancia de los Andes es cuatro veces más grande que ésta, en cuanto a secciones y salones. El tiempo de solución, corresponde al tiempo que empleó el método de solución usado

para resolver la instancia, no en generarla. Daskalaki et al. [7], resuelven el problema para el Departamento de una universidad. Ellos resuelven el programa matemático de 19,295 variables y 17,159 restricciones con CPLEX 5.1 en 95 minutos; comparado con la instancia de 16,828 variables y 18,263 restricciones de los Andes, que se resuelve en 29 minutos. La heurística de asignación de salones que usa la Oficina de Admisiones y Registro, y que se describe en la Sección 1.2, tiene un porcentaje de asignación del 70 %, y arroja la solución en un minuto.

Autor(es)	E. Castro et al.	Herística Ofc. De Admisiones.	J. M. Mulvey	M. W. Carter	J. J. Dinkel et al.
Numero de secciones	1957	1957	--	1400	[287, 312]
Numero de salones	127	127	14	125	22
secciones / salones	15.41	15.41	--	11.20	[13.04, 14.18]
Numero de restricciones	[1894, 18263]	--	--	90	[311, 338]
Numero de variables	[1548, 16828]	--	--	125	[2009, 2658]
% de asignación	[35.26%, 73.33%]	70%	--	--	--
tiempo de solución (min.)	[1, 29]	1	--	15	--
Ámbitos de la instancia	Universidad de los Andes	Universidad de los Andes	Escuela de negocios de UCLA	Universidad de Waterloo	College of Buss. Admin. Texas A&M.

Autor(es)	E. L. Mooney et al.	A. Hertz et al.	M. Dimopoulou et al. (2001)	S. Daskalaki et al.	M. Dimopoulou et al. (2004)
Numero de secciones	[67, 492]	339	--	[33, 119]	--
Numero de salones	[29, 31]	--	--	[9, 18]	--
secciones / salones	[2.3, 15.87]	--	--	[3.67, 6.61]	--
Numero de restricciones	--	--	500	[7543, 17159]	150
Numero de variables	--	--	100	[4100, 19295]	450
% de asignación	--	--	--	--	--
tiempo de solución (min.)	--	600	--	[2.5, 95]	--
Ámbitos de la instancia	--	Colegio en Suiza.	Universidad de economía y de negocios de Atenas	Depto. Ing. Eléctrica Univ. De Patras	Universidad de economía y de negocios de Atenas

Tabla 29: Comparación de varios problemas.

Capítulo V

Conclusiones e investigación futura

El algoritmo de preprocesamiento es una parte muy importante del modelo de solución que se propone, y demostró ser sumamente útil, porque disminuye el espacio de búsqueda y permite lograr soluciones satisfactorias en un tiempo computacional razonable. En este código se implementa un novedoso mecanismo de generación de patrones de tiempo a partir de un patrón de tiempo propuesto a priori, lo que implica que se debe considerar el horario de funcionamiento de los edificios y la hora en la que se inician las clases de maestría, de tal forma que no se generen patrones tiempo fuera de estos horarios. Además, se introduce una función de penalización que depende de la longitud del desplazamiento que se considere (ver Figura 2 en la página 29). En este sentido, se podría explorar el escenario donde β dependa de la hora del día. Asimismo, se implementan mecanismos para manifestar preferencia por edificios, como es el caso de los edificios para clases de maestría. Las pre-asignaciones de salón, ecuaciones (2), se manejan desde el preprocesamiento, porque se disminuye el tamaño del problema, y además porque se observó que si se incluyen en la formulación de Xpress-MP, el modelo resulta ser infactible, posiblemente por problemas de estabilidad numérica, pues las instancias que se le pasaban, tenían pre-asignaciones

factibles. Todos los argumentos del preprocesamiento los especifica el usuario, permitiéndole así, que manifieste sus necesidades. La relación entre los argumentos del preprocesamiento es compleja, e impacta directamente en la calidad de la solución obtenida (Figuras 7 y 8, y Tabla 28). Queda por explorar la sintonización adecuada de estos parámetros. Intuitivamente se esperaba que a mayor número de salones candidatos, el porcentaje de asignación aumentara, en la práctica ocurrió lo contrario. Entonces, se plantea la posibilidad de generar dos instancias, tales que una contenga a la otra, en el sentido que los salones candidatos de las secciones de una instancia, son subconjuntos de los conjuntos de salones candidatos de la otra. Aunque en el preprocesamiento se trata de distribuir los salones dentro de las secciones balanceadamente, se observa que para algunos salones la distribución no es justa, pues tienen muchas secciones asignadas con respecto a otros salones del mismo tamaño. Aun así, el algoritmo de preprocesamiento es satisfactorio para la generación de instancias de calidad, y puede servir por sí mismo, como base para el desarrollo de una heurística de asignación de salones. Se introducen los diagramas de flujo de manipulación de archivos (Figuras 3 y 4, páginas 34 y 45) los cuales permiten escribir y visualizar de forma sintética las transformación y generación de archivos.

Dos aspectos importantes quedan abiertos al desarrollo. Uno, es la sistematización del proceso, y el otro, la explotación de las funcionalidades estadísticas de SAS®. En cuanto al primer aspecto, el método de solución que se propone, implica un preprocesamiento en SAS®, seguido de una optimización en Xpress-MP, y el posprocesamiento final, lo que involucra la manipulación dinámica de centenares de archivos (con tamaños del orden de los decenas de MB) en un PC. Este proceso se puede hacer transparente al usuario, si las aplicaciones se instalan en un servidor, al

cual la Oficina de Admisiones y Registro tenga acceso, y pueda enviar los parámetros y los archivos de entrada. Con respecto al segundo punto, consideramos que el preprocesamiento es susceptible a mejoras, basadas en las estadísticas de los perfiles de asignación de secciones a salones, franjas horarias preferidas, y uso de franjas horarias poco deseadas.

Se muestra que es posible descomponer el problema en problemas de asignación diarios, y así resolver instancias de problemas de programación entera binaria de gran escala con optimizadores comerciales. Para instancias cuyo tiempo de corrida es considerable, se pueden escribir algoritmos de generación de cortes, de tal manera que se aproxime la solución óptima en un tiempo razonable, utilizando como cortes las restricciones (6). Esto sería particularmente útil, cuando se consideran instancias con multiples patrones de tiempo, pues en estas instancias, no solamente el numero de restricciones aumenta, sino que la estructura de estas es más complicada.

En el tratamiento de un problema de asignación de salones, es preferible alcanzar una solución buena (según lo criterios administrativos y académicos) y factible en un tiempo razonable, a lograr la solución óptima después de un tiempo computacional considerable. Aun así, es importante aclarar que minimizar el porcentaje de secciones no asignadas es de especial interés. En este sentido, se puede considerar una segunda pasada, donde se tenga en cuenta las asignaciones previas y se hagan asignaciones en los intervalos libres de los salones.

Apéndice A

Instrucciones en SQL que generan los data sets comprimidos cuando se conoce el profesor

Código de compactación para la generación del data set `secciones_magistrales1`

```
proc sql noprint;
 create table secciones_magistrales1 as
 select distinct lunes, martes, miercoles, jueves, viernes, sabado,
 sum(capseccion) as capseccion, horainicio, horafinal,
 max(salonB) as salonB, min(idseccion) as idseccion, horaCeroSeg,
 horaFinalSeg, tiempoClaseSeg, max(spriden_id) as spriden_id,
 min(materia) as materia
 from datos.secciones_neto
 group by depto, nombre, spriden_id, salon, horaCeroSeg, horaFinalSeg,
 lunes, martes, miercoles, jueves, viernes, sabado
 having min(lunes)= max(lunes) and min(martes)= max(martes)and
 min(miercoles)= max(miercoles) and min(jueves)= max(jueves) and
 min(viernes)= max(viernes) and min(sabado)= max(sabado) and
 max(salon) ^= "" and min(salon) ^= "" and count(nombre)>=2
```

```
 order by idseccion;
quit;
```

Código de compactación para la generación del data set `secciones_magistrales2`

```
proc sql noprint;
 create table secciones_magistrales2 as
 select distinct lunes, martes, miercoles, jueves, viernes, sabado,
 sum(capseccion) as capseccion, horainicio, horafinal,
 max(salonB) as salonB, min(idseccion) as idseccion, horaCeroSeg,
 horaFinalSeg, tiempoClaseSeg, max(spriden_id) as spriden_id,
 min(materia) as materia
 from secciones_neto_particion
 group by depto, spriden_id, salon, horaCeroSeg, horaFinalSeg,
 lunes, martes, miercoles, jueves, viernes, sabado
 having min(lunes)= max(lunes) and min(martes)= max(martes)and
 min(miercoles)= max(miercoles) and min(jueves)= max(jueves) and
 min(viernes)= max(viernes) and min(sabado)= max(sabado) and
 max(salon) ^= "" and min(salon) ^= "" and count(nombre)>=2
 order by idseccion;
quit;
```

Código de compactación para la generación del data set `secciones_noMagistrales1`

```
proc sql noprint;
 create table secciones_noMagistrales1 as
```

```
 select distinct max(lunes) as lunes, max(martes) as martes,
 max(miercoles) as miercoles, max(jueves) as jueves,
 max(viernes) as viernes, max(sabado) as sabado, capseccion,
 horainicio, horafinal, max(salonB) as salonB,
 min(idseccion) as idseccion, horaCeroSeg, horaFinalSeg, tiempoClaseSeg,
 spriden_id, min(materia) as materia
 from secciones_noMagistrales
 group by crn, horaCeroseg, horaFinalSeg
 having (min(horaCeroSeg) = max(horaCeroSeg) and
 min(horaFinalSeg) = max(horaFinalSeg))
 order by idseccion;
quit;
```

Apéndice B

Instrucciones en SQL que generan los data sets de 'slots' en conflicto cuando se conoce el profesor

Código de generación del data set `slots_Cruce`

```
proc sql noprint;
 create table datos.slots_Cruce as
 select vx1.slotId, vx2.slotId as slotIdCruce
 from datos.varsXpress vx1, datos.varsXpress vx2
 where (vx1.salon = vx2.salon or
 (vx1.spriden_id ^= . and vx2.spriden_id ^= . and
 vx1.spriden_id = vx2.spriden_id))
 and (((vx1.lunes= 1 and vx2.lunes=1) or
 (vx1.martes= 1 and vx2.martes= 1) or
 (vx1.miercoles= 1 and vx2.miercoles= 1) or
 (vx1.jueves= 1 and vx2.jueves= 1) or
 (vx1.viernes= 1 and vx2.viernes= 1) or
 (vx1.sabado= 1 and vx2.sabado= 1))
 and (vx2.horaCeroSeg <= vx1.horaCeroSeg < vx2.horaFinalSeg
```

```
 or vx1.horaCeroSeg <= vx2.horaCeroSeg < vx1.horaFinalSeg))
 order by vx1.slotId, vx2.slotId;
quit;
```

Código de generación del data set `slotsCruceNR`

```
data datos.slotsCruceNR;
 set datos.slots_Cruce;
 if slotId < slotIdCruce;
run;
```

Apéndice C

Instrucciones en SQL que generan los data sets comprimidos cuando se genera un único patrón de tiempo

Código de compactación para la generación del data set `secciones_pre_asignado1`

```
proc sql noprint;
 create table secciones_pre_asignado1 as
 select distinct lunes, martes, miercoles, jueves, viernes, sabado,
 sum(capseccion) as capseccion, horainicio, horafinal,
 max(salonB) as salonB, min(idseccion) as idseccion, horaCeroSeg,
 horaFinalSeg, tiempoClaseSeg, max(spriden_id) as spriden_id,
 min(materia) as materia
 from datos.secciones_neto
 group by depto, nombre, salon, horaCeroSeg, horaFinalSeg, lunes,
 martes, miercoles, jueves, viernes, sabado
 having min(lunes)= max(lunes) and min(martes)= max(martes)and
 min(miercoles)= max(miercoles) and min(jueves)= max(jueves) and
```

```
 min(viernes)= max(viernes) and min(sabado)= max(sabado) and
 max(salon) ^= "" and min(salon) ^= "" and count(nombre)>=2
 order by idseccion;
quit;
```

Código de separacion y de compactación para la generación del data set `secciones_pre_asignado2`

```
proc sql noprint;
 create table secciones_pre_asignado2 as
 select distinct crn, nombre, depto, salon, lunes, martes, miercoles,
 jueves, viernes, sabado, sum(capseccion) as capseccion, horainicio,
 horafinal, max(salonB) as salonB, min(idseccion) as idseccion,
 horaCeroSeg, horaFinalSeg, tiempoClaseSeg, max(spriden_id) as spriden_id,
 min(materia) as materia
 from datos.secciones_neto
 group by depto, nombre, salon, horaCeroSeg, horaFinalSeg, lunes,
 martes, miercoles, jueves, viernes, sabado
 having min(lunes)= max(lunes) and min(martes)= max(martes)and
 min(miercoles)= max(miercoles) and min(jueves)= max(jueves) and
 min(viernes)= max(viernes) and min(sabado)= max(sabado) and
 max(salon) ^= "" and min(salon) ^= "" and count(nombre)=1
 order by idseccion;
quit;


proc sql noprint;
```

```
 create table secciones_pre_asignado2 as
 select distinct lunes, martes, miercoles, jueves, viernes, sabado,
 sum(capseccion) as capseccion, horainicio, horafinal,
 max(salonB) as salonB, min(idseccion) as idseccion, horaCeroSeg,
 horaFinalSeg, tiempoClaseSeg, max(spriden_id) as spriden_id,
 min(materia) as materia
 from secciones_pre_asignado2
 group by depto, salon, horaCeroSeg, horaFinalSeg, lunes, martes,
 miercoles, jueves, viernes, sabado
 having min(lunes)= max(lunes) and min(martes)= max(martes)and
 min(miercoles)= max(miercoles) and min(jueves)= max(jueves) and
 min(viernes)= max(viernes) and min(sabado)= max(sabado)
 order by idseccion;
quit;
```

Código de separacion para la generación del data set `secciones_sin_salon`

```
data secciones_sin_salon (drop= nombre salon crn depto seccion
creditos inscritos capresidual);
 set datos.secciones_neto;
 if salon = "";
run;
```

Apéndice D

Instrucciones en SQL que generan los data sets de 'slots' en conflicto cuando no se conoce el profesor

Código de generación del data set `slots_Cruce`

```
proc sql noprint;
 create table datos.slots_Cruce as
 select vx1.slotId, vx1.salonB, vx2.slotId as slotIdCruce
 from datos.varsXpress vx1, datos.varsXpress vx2
 where (vx1.salon = vx2.salon) and (((vx1.lunes= 1 and vx2.lunes=1)
 or (vx1.martes= 1 and vx2.martes= 1) or
 (vx1.miercoles= 1 and vx2.miercoles= 1) or
 (vx1.jueves= 1 and vx2.jueves= 1) or (vx1.viernes= 1 and vx2.viernes= 1)
 or (vx1.sabado= 1 and vx2.sabado= 1))
 and (vx2.horaCeroSeg <= vx1.horaCeroSeg < vx2.horaFinalSeg or
 vx1.horaCeroSeg <= vx2.horaCeroSeg < vx1.horaFinalSeg))
 order by vx1.slotId, vx2.slotId;
quit;
```

Código de generación del data set `slotsCruceNR`

```
data datos.slotsCruceNR;
 set datos.slots_Cruce;
 if slotId < slotIdCruce;
run;
```

Apéndice E

Formulación 1

```
model salones
 uses "mmxprs", "mmsystem"

 parameters
  alphaFile = '.\alpha.txt'
  cflFile1 = '.\slotsCfl1.txt'
  seccEspFile = '.\seccEsp.txt'
 end-parameters

 !imprime en un archivo cada solucion entera que se encuentre
 forward procedure printsol
 setcallback(XPRS_CB_UIS,"printsol")

 declarations
  secciones, slots, seccEsp: set of integer
  alpha: array(secciones,slots) of real
  matrizCfl1: array(slots,slots) of boolean
```

```
 x: array(secciones,slots) of mpvar
end-declarations

initializations from alphaFile
 alpha
end-initializations

initializations from cflFile1
 matrizCfl1
end-initializations

initializations from seccEspFile
 seccEsp
end-initializations

forall (s in secciones, l in slots | exists(alpha(s,l))) do
 create(x(s,l))
 x(s,l) is_binary
end-do

finalize(secciones)
finalize(slots)
finalize(seccEsp)

obj:= sum(s in secciones)sum(l in slots | exists(alpha(s,l)))
```

```
                                                    alpha(s,l)*x(s,l)

!R1: LAS SECCIONES CON SALON PRE-ASIGNADO DEBEN SER ASIGNADAS
forall(s in seccEsp) preAsig(s):= sum(l in slots | exists(alpha(s,l)))
                                                    x(s, l) = 1

!R2: RESTRICCIONES QUE IMPIDEN DUPLICIDAD DE SLOT EN LA ASIGNACION
forall(s in secciones) noDup(s):=
sum(l in slots | exists(alpha(s,l))) x(s,l) <= 1

!R3: RESTRICCIONES QUE EVITAN CONFLICTOS DE SALON Y DE PROFESOR.
forall(k in slots) cfl(k):= sum(l in slots | exists(matrizCfl1(k,l)))
                  sum(s in secciones | exists(alpha(s,l))) x(s,l) <= 1

starttime:= gettime
setparam("XPRS_MIPRELSTOP", 0.05)  ! gap relativo de parada en 5%

maximize(obj)

fopen("solucion.dat",F_OUTPUT)
writeln(getparam("XPRS_BESTBOUND")," ",getparam("XPRS_MIPOBJVAL")," ",
                                                    gettime-starttime)
forall(s in secciones, l in slots | exists(alpha(s,l))
and getsol(x(s,l))>getparam("XPRS_FEASTOL"))
                                    writeln(s," ",l," ",getsol(x(s,l)))
```

```
 fclose(F_OUTPUT)



 ! *** IMPRESION DE SOLUCIONES ENTERAS ***
 procedure printsol
  setparam("XPRS_SOLUTIONFILE",0)
  fopen("printsol.dat",F_OUTPUT+F_APPEND)
  writeln(getparam("XPRS_BESTBOUND")," ",getparam("XPRS_MIPOBJVAL")," ",
  100*abs(getparam("XPRS_MIPOBJVAL")-getparam("XPRS_BESTBOUND"))
                                                  /getparam("XPRS_BESTBOUND"))
  forall(s in secciones, l in slots | exists(alpha(s,l))
  and getsol(x(s,l))>getparam("XPRS_FEASTOL"))
                                          writeln(s," ",l," ",getsol(x(s,l)))
  fclose(F_OUTPUT)
  setparam("XPRS_SOLUTIONFILE",1)
 end-procedure

end-model
```

Apéndice F

Formulación 2

```
model salones
 uses "mmxprs", "mmsystem"

 parameters
  alphaFile = '.\alpha.dat'
  cflFile1 = '.\slotsCfl1.dat'
  cflFile2 = '.\slotsCfl2.dat'  !conflictos por parejas
  seccEspFile = '.\seccEsp.dat'
 end-parameters

 ! se crea un archivo vacio que se llenara con datos en cada nodo
 ! de B&B
 fopen("cb_node.dat",F_OUTPUT)
 fclose(F_OUTPUT)

 forward function cb_node:boolean !adiciona cortes en cada nodo de B&B
```

```
!imprime en un archivo cada solucion entera que se encuentre
forward procedure printsol
setcallback(XPRS_CB_UIS,"printsol")

declarations
 secciones, slots, seccEsp: set of integer
 alpha: array(secciones,slots) of real
 matrizCfl1: array(slots,slots) of boolean
 matrizCfl2: array(slots,slots) of boolean
 x: array(secciones,slots) of mpvar

 feastol: real                          !zero tolerance
 solx: array(secciones,slots) of real   !sol. values for variables 'x'
end-declarations

initializations from alphaFile
 alpha
end-initializations

initializations from cflFile1
 matrizCfl1
end-initializations

initializations from cflFile2
 matrizCfl2
```

```
end-initializations

initializations from seccEspFile
 seccEsp
end-initializations

forall (s in secciones, l in slots | exists(alpha(s,l))) do
 create(x(s,l))
 x(s,l) is_binary
end-do

finalize(secciones)
finalize(slots)
finalize(seccEsp)

obj:= sum(s in secciones-seccEsp)sum(l in slots | exists(alpha(s,l)))
                                                    alpha(s,l)*x(s,l)

!LAS SECCIONES CON SALON PRE-ASIGNADO DEBEN SER ASIGNADAS
forall(s in seccEsp) preAsig(s):= sum(l in slots | exists(alpha(s,l)))
                                                          x(s, l) = 1

!RESTRICCIONES QUE IMPIDEN DUPLICIDAD DE SLOT EN LA ASIGNACION
forall(s in secciones-seccEsp) noDup(s):=
sum(l in slots | exists(alpha(s,l))) x(s,l) <= 1
```

```
!RESTRICCIONES QUE EVITAN CONFLICTOS DE SALON Y DE PROFESOR.
forall(k in slots) cfl(k):= sum(l in slots | exists(matrizCfl1(k,l)))
                  sum(s in secciones | exists(alpha(s,l))) x(s,l) <= 1

starttime:= gettime
setparam("XPRS_CUTSTRATEGY", 0)! deshabilitar generacion automatica de cortes
setparam("XPRS_PRESOLVE", 0)       ! Switch presolve off
setparam("XPRS_EXTRAROWS", 5000)   ! 5000 filas extra en la matriz
setparam("XPRS_MIPRELSTOP", 0.05)  ! gap relativo de parada en 5%
feastol:= getparam("XPRS_FEASTOL") ! tolerancia cero
feastol:= feastol * 10
setcallback(XPRS_CB_CM, "cb_node") ! callback para generacion de cortes
maximize(obj)

fopen("solucion.dat",F_OUTPUT)
forall(s in secciones, l in slots | exists(alpha(s,l)) and
getsol(x(s,l))>getparam("XPRS_FEASTOL")) writeln(s," ",l," ",getsol(x(s,l)))
fclose(F_OUTPUT)

!***       ***
! *** GENERACION DE CORTES EN LOS NODOS DEL ARBOL ***
!***       ***
function cb_node:boolean
 declarations
```

```
 ncut: integer
 cut: array(range) of linctr
 cutid: array(range) of integer
 type: array(range) of integer
end-declarations

depth:=getparam("XPRS_NODEDEPTH")
ncut:= 0

if(depth<4) then
 ! jalar el vector de solucion
 setparam("XPRS_SOLUTIONFILE",0)
 forall (s in secciones, l in slots | exists(alpha(s,l)))
  solx(s,l):= getsol(x(s,l))
 setparam("XPRS_SOLUTIONFILE",1)

 ! Search for violated constraints
 forall (k in slots, l in slots, r in secciones, s in secciones |
 exists(matrizCfl2(k,l)) and exists(alpha(r,k)) and exists(alpha(s,l))) do
  if (solx(r,k)+solx(s,l)-1 > feastol) then
   cut(ncut):= 1 - x(r,k) - x(s,l)
   cutid(ncut):= 1
   type(ncut):= CT_GEQ
   ncut+=1
  end-if
```

```
  end-do

  ! Add cuts to the problem

  if(ncut>0) then
   addcuts(cutid, type, cut)
  end-if
 end-if

 fopen("cb_node.dat",F_OUTPUT+F_APPEND)
 !node depth cuts_added BESTBOUND LPOBJ MIPOBJ REL_GAP
 writeln(getparam("XPRS_NODES")," ",depth," ",ncut," ",
 getparam("XPRS_BESTBOUND")," ",getparam("XPRS_LPOBJVAL")," ",
 getparam("XPRS_MIPOBJVAL")," ",
 abs(getparam("XPRS_MIPOBJVAL")-getparam("XPRS_BESTBOUND"))
                                                  /getparam("XPRS_BESTBOUND"))
 fclose(F_OUTPUT)

 returned:= false
end-function

!***       ***
!*** IMPRESION DE SOLUCIONES ENTERAS ***
!***       ***
procedure printsol
```

```
  setparam("XPRS_SOLUTIONFILE",0)
  fopen("printsol.dat",F_OUTPUT)
  writeln(getparam("XPRS_lpobjval"))
  forall(s in secciones, l in slots | exists(alpha(s,l)) and
  getsol(x(s,l))>getparam("XPRS_FEASTOL")) writeln(s," ",l," ",getsol(x(s,l)))
  fclose(F_OUTPUT)
  setparam("XPRS_SOLUTIONFILE",1)
 end-procedure

end-model
```

Apéndice G

Posprocesamiento

```
libname datos "."; filename stats "stats.txt"; \%macro calidad;

/*lectura de los archivos de solucion que arroja Xpress-MP*/ \%do
d=1 \%to 6;
 \%let x= solucion_\%unquote(&d.).dat;
 filename solucion "&x.";
 data solucion_\%unquote(&d.) (drop= valor);
  infile solucion;
  input idseccion slot valor;
  if _N_ > 1;
 run;
\%end;

\%let numSecc=; /*num. total de secciones a asignar*/ \%let
numSeccEsp=; /*num. de secc. especiales a asignar*/ \%let
numSeccEspA=;/*num. total de secciones especiales que
efectivamente se asignaron*/ /*lectura de los archivos de solucion
```

```
que arroja Xpress-MP*/
 data stats;
  infile stats;
  input totalSecc totalSeccEsp totalSeccEspAsig;
  call symput('numSecc',totalSecc);
  call symput('numSeccEsp',totalSeccEsp);
  call symput('numSeccEspA',totalSeccEspAsig);
 run;
\%put numSecc= &numSecc.; \%put numSeccEsp= &numSeccEsp.; \%put
numSeccEspA= &numSeccEspA.;

/*Al frente de cada idseccion se pone el numero de dias de clase a
la semana*/ data seccSumDias (drop= lunes martes miercoles jueves
viernes sabado capSeccion horaInicio horaFinal horaCeroSeg
horaFinalSeg tiempoClaseSeg spriden_id materia);
 set datos.secciones;
 sumDias= sum(lunes,martes,miercoles,jueves,viernes,sabado);
run;

/*A partir de los archivos de solucion que se leyeron, se estima
cuantos dias
  de clase se programaron para cada seccion*/
data solSeccSumDias (drop= slot);
 /*en los archivos de solucion solo salen las soluciones de las
   secciones que ->NO<- tienen salon pre-asignado */
```

```
 set solucion_1 solucion_2 solucion_3 solucion_4 solucion_5 solucion_6;
 by idseccion;
run;

/*num. obs. de solSeccSumDias es el numero de secciones sin salon
pre-
  asignado que potencialmente se asignaron*/
proc sql noprint;
 create table solSeccSumDias as
 select idseccion, count(idseccion) as sumDias
 from solSeccSumDias
 group by idseccion
 order by idseccion;
quit;

\%let numSeccPot=; /*num. total de secciones sin salon
pre-asignado
  que potencialmente se pueden asignar*/
proc sql noprint;
 select count(idseccion) into: numSeccPot
 from solSeccSumDias;
quit; \%put numSeccPot= &numSeccPot.;

/*num. obs. de resumenAsignacion es el numero de secciones sin
salon
```

```
  pre-asignado que efectivamente se asignaron*/
proc sql noprint;
 create table resumenAsignacion as
 select x.idseccion, salonB
 /*en el data set seccSumDias se contabilizan las secciones con salon pre-
   asignado que efectivamente se asignaron*/
 from seccSumDias x, solSeccSumDias y
 where x.idseccion= y.idseccion and x.sumDias= y.sumDias
 order by idseccion;
quit;

\%let numSeccAsig=; /*num. total de secciones sin salon
pre-asignado que efectivamente se
  asignaron*/
proc sql noprint;
 select count(idseccion) into :numSeccAsig
 from resumenAsignacion;
quit; \%put numSeccAsig= &numSeccAsig.;

/* promedio de variables de las instnacias de lunes a viernes*/

\%let totalVars=; \%do d=1 \%to 5;
 proc sql noprint;
  select count(salonB) into :numVars
  from datos.varsXpress_\%unquote(&d.)
```

```
  group by salonB
  having salonB= 0;
 quit;
 \%put numVars= &numVars.;
 \%let totalVars= \%eval(&totalVars.+&numVars.);
\%end; \%let varProm = \%sysevalf(&totalVars./5);

data _NULL_;
 file "reporte.csv";
 put "Total de secciones a asignar, &numSecc.";
 put "Total de secciones con salon pre-asignado, &numSeccEsp.";
 put;
 put;
 put;
 put "Restricciones promedio (Lunes a Viernes), &VarProm.";
 put;
 put;
 put "Secciones sin salon pre-asignado que potencialmente se pueden
                                              asignar, &numSeccPot.";
 put "Secciones sin salon pre-asignado que efectivamente se asignaron,
                                                    &numSeccAsig.";
 put "Secciones con salon pre-asignado que se asignaron, &numSeccEspA.";
 put "\% total de asignacion,
              \%sysevalf(100*(&numSeccAsig.+&numSeccEspA.)/&numSecc.)";
 put "\% de asignacion de secciones con salon pre-asignado,
```

```
                                \%sysevalf(100*&numSeccEspA./(&numSeccEsp))";
run;

\%mend calidad; \%calidad;
```

Referencias

[1] A. S. Asratian and D. de Werra, *A generalized class-teacher model for some timetabling problems*, European Journal of Operational Research **143** (2002), 531–542.

[2] Dash Associates, *Xpress-mosel language reference manual*, Dash Associates, 2002.

[3] ———, *Xpress-Mosel User Guide*, Dash Associates, 2002.

[4] ———, *Xpress-MP Essentials*, Dash Associates, segunda ed., Febrero 2002.

[5] ———, *Xpress-Optimizer Reference Manual*, Dash Associates, 2002.

[6] M. W. Carter, *A lagrangian relaxation approach to the classroom assignment problem*, Information systems and operational research (INFOR) **27** (1989), no. 2, 230–246.

[7] S. Daskalaki, T. Birbas, and E. Houstos, *An integer programming formulation for a case study in a university timetabling*, European Journal of Operational Research **153** (2004), 117–135.

[8] M. Dimopoulou and P. Miliotis, *Implementation of a university course and examination timetabling system*, European Journal of Operational Research **130** (2001), no. 1, 202–213.

[9] ———, *An automated university course timetabling system developed in a distributed environment: A case study*, European Journal of Operational Research **153** (2004), 136–147.

[10] J. J. Dinkel, J. Mote, and M. A. Venkataramanan, *An efficient decision support system for academic course scheduling*, Operations Research **37** (1989), no. 6, 853–864.

[11] S. Heipcke, *Embedding Optimization Algorithms*, White paper, Dash Optimization, Blisworth House, Blisworth, Northants NN7 3BX (UK), Noviembre 2002.

[12] A. Hertz and V. Robert, *Constructing a course schedule by solving a series of assignment type problems*, European Journal of Operational Research **108** (1998), 585–603.

[13] SAS Institute Inc., *SAS® Guide to the SQL Procedure: Usage and Reference, Versión 6*, SAS Institute Inc., Cary, NC, USA., Primera ed., 1989.

[14] _____, *SAS® Macro Lenguage: Reference*, SAS Institute Inc., Cary, NC, USA., Primera ed., 1997.

[15] L. Lamport, *LaTeXA document preparation system. User's guide and reference manual*, 14 ed., Tools and Techniques for Computer Typesetting, Addison-Wesley, 2002.

[16] J. Liebowitz, V. Krishnamurth, and I. Rodens, *Classroom scheduling: An application and tools*, Journal of Computer Information Systems (1998), 6–10.

[17] E. L. Mooney, R. L. Rardin, and W. J. Parmenter, *Large-scale classroom scheduling*, IIE transactions **28** (1996), no. 5, 369–378.

[18] J. M. Mulvey, *A classroom/time assignment model*, European Journal of Operational Research **9** (1982), 64–70.

[19] F. Salcedo, *Asignación de salones*, Dirección de Tecnologías de Información (DTI), Universidad de los Andes, Cr1 18 A-10. Bogota, Colombia, 2000.

[20] _____, *Subsistema pre-horarios salones*, Dirección de Tecnologías de Información (DTI), Universidad de los Andes, Cr1 18 A-10. Bogota, Colombia, Octubre 2000.

[21] L. A. Wolsey, *Integer Programming*, Wiley-Interscience series in discrete mathematics, John Wiley & Sons, Inc., 1998.