UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MEJORA DE ASIGNACIÓN DE SALAS EN UCAMPUS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

CRISTÓBAL MARIO SEPÚLVEDA COBO

PROFESOR GUÍA:
JOCELYN SIMMONDS WAGEMANN
PROFESOR GUÍA 2:
WILLY MAIKOWSKI CORREA

MIEMBROS DE LA COMISIÓN:
JOSÉ A. PINO URTUBIA
TOMÁS BARROS ARANCIBIA

SANTIAGO DE CHILE
2020

Resumen

La Facultad de Ciencias Física y Matemática de la Universidad de Chile necesita realizar todos los semestres una distribución de salas y cursos, en que a cada curso se le asigna un espacio físico en donde realizar sus clases. Por ejemplo, en el semestre de otoño de 2019, se distribuyeron 835 cursos en 117 salas. La supervisión de este proceso está a cargo de la Subdirección de Gestión Docente (SGD) y lo realiza a través de una plataforma desarrollada por Ucampus, centro encargado de desarrollar sistemas de apoyo a la gestión curricular en instituciones de Educación Superior. Este problema depende de una diversidad de variables, tales como la capacidad de la sala, el cupo del curso, los choques entre clases, entre otras cosas, y se espera que una buena distribución sea aquella que optimice el espacio. La solución que actualmente resuelve el problema se basa en información histórica, repitiendo las distribuciones pasadas y siendo ajustada posteriormente de forma manual por la SGD a través de la plataforma web de Ucampus. No obstante, esta misma puede mejorarse ya que, además de presentar problemas tales como asumir que una solución pasada va a ser siempre efectiva, ignora una posible optimización sobre los datos. Además, la plataforma previa no entrega retroalimentación. Por lo tanto, una mejor solución es aquella que responda a estos dos problemas principales.

Para mejorar la distribución de salas se decidió usar un acercamiento de programación matemática, específicamente programación lineal entera-mixta. Esto dado que dicho acercamiento es el más cercano al escenario planteado, al generar un modelo que ilustra el problema de decisión sobre la mejor asignación de curso y sala. De esa forma, la solución se desarrolló en Ucampus, donde se profundizó en la plataforma sumando validaciones a la distribución de salas para mejorar la información entregada al usuario. Lo anterior se realizó tomando como base la solución previa, modificando la comunicación entre módulos para contemplar la optimización y respetando la convención que existe en Ucampus y, de esa forma, la escalabilidad del sistema.

En términos de eficiencia, la solución desarrollada fue validada contrastando el nivel de ocupación promedio logrado tanto por la solución previa como la actual. Dado eso, se logró aumentar la ocupación de un 67,8 % a un 85,3 %. Esto permitirá un mejor uso del espacio y, por lo tanto, la liberación de salas que previamente se usaban de forma ineficiente. Por otra parte, en términos de interfaces, se validó en conjunto con la SGD corroborando la satisfacción de los usuarios del sistema y validando la retro-alimentación brindada.

Agradecimientos

Gracias a mis padres por ser un eterno pilar de apoyo, por entregarme mis valores y motivarme a ser una mejor persona cada día y desde el primer momento. A mi familia, por confiar plenamente en mí, y darme su soporte y seguridad. A mis tías Edmée, Gignna y Barbara por brindarme un hogar en Santiago y ser un apoyo tan grande durante estos últimos años, y toda mi vida. A Daniela por darme la motivación para seguir creciendo y ser una fuente de felicidad por tanto tiempo, también a sus hermanas Carolina y María José por ser mi segunda familia.

Gracias a Ucampus, a Willy por su ayuda constante, por darme una motivación para hacer este trabajo, y estar allí cada vez que necesitara un empujón en la dirección correcta.

Gracias a Jocelyn, profesora guía de esta memoria, por la dedicación, confianza y tiempo invertido en este trabajo. A todos mis profesores por motivar el crecimiento de mis conocimientos y permitir que llegara a este punto de mi carrera.

Gracias a mis amigos. Por toda su compañía y soporte en todos estos años de Universidad, por confiar en mí y darme su felicidad y vibrante amistad.

Finalmente, gracias a todos esas personas que de una u otra forma han ayudado a componer mi crecimiento como humano e ingeniero. A La Radio Integral por darme otro motor de vida fuera de lo académico, y ayudarme a descubrir otra pasión en la vida. Al cine y la música por dejarme existir fuera de la universidad, a Tus Amigos Nuevos y Sufjan Stevens por darme un pilar de motivación durante estos años de carrera.

Tabla de Contenido

Índice de Ilustraciones

Capítulo 1

Introducción

Durante el proceso de Inscripción Académica de la Facultad de Ciencias Físicas y Matemáticas (FCFM), al igual que el resto de las facultades de la Universidad de Chile, es necesario realizar una distribución pertinente de salas. Cada semestre se revisa el catálogo de cursos que serán dictados, y se le asigna una sala en la facultad basándose en la disponibilidad horaria del curso y la sala, y el cupo y capacidad de estos, respectivamente. En el semestre de otoño de 2019, por ejemplo, se asignaron 835 cursos en 117 salas, dictados por 623 profesores de cátedras. Estos cursos se distribuyeron en 2.119 instancias de horario semanal, o sea, bloques de horario que ocupa cada curso en sus distintas cátedras y clases auxiliares. Sin dicha distribución, un curso no tiene lugar físico donde realizar sus clases, imposibilitando su realización. Es por esto que esta etapa es sumamente importante para el correcto desarrollo de un semestre académico.

El algoritmo vigente observa la distribución de salas histórica y la repite, dando como única restricción el que una sala tenga mayor o igual capacidad al cupo de su curso asignado. Esta solución es funcional, pues distribuye todos los cursos disponibles a una sala cuya capacidad satisface la del curso, mientras que la componente histórica oculta las demás restricciones que hoy en día son parte del conocimiento de las personas encargadas. La Subdirección de Gestión Docente (SGD) es la entidad de la FCFM que supervisa la distribución de salas. Para esto, la persona encargada utiliza la plataforma web de Ucampus, en donde se encuentra la acción de distribución. Así, la implementación actual realiza la asignación y le afirma al usuario si es que ésta se realizó con éxito o no. Esta información luego debe ser verificada por la SGD, comprobando entre cursos y profesores que no haya un problema con el cálculo realizado por la plataforma. El proceso de cálculo toma unos minutos, pero la confirmación de los datos puede llevar varios días según la SGD.

El proceso de distribución de salas actualmente es llevado a cabo en un sistema realizado por el Centro de Tecnologías de la Información Ucampus, entidad encargada del desarrollo de plataformas de apoyo a la gestión para lograr la automatización de procesos. Ucampus se encarga de desarrollar e idear una serie de soluciones tecnológicas para resolver los problemas de las facultades de la Universidad de Chile, agrupadas en dos plataformas: U-Cursos y Ucampus [7]. U-cursos es una plataforma encargada de entregar herramientas de apoyo a la docencia para los cursos realizados en distintas facultades. En cambio, Ucampus es una

plataforma de apoyo a la gestión interna de las facultades en términos curriculares.

1.1. Problema y motivación

A pesar de ya existir una solución funcional en Ucampus, la cual asigna correctamente todos los cursos en el catálogo, se han observado una serie de limitaciones. Una solución en base a la información histórica solo observa el panorama a corto plazo. Los cupos de los cursos tienden a variar con el tiempo y se suelen abrir nuevas secciones en base a la cantidad de estudiantes que van ingresando cada año. La FCFM planea aumentar el cupo de estudiantes que ingresa a Plan Común en un 25 % con el transcurso de los años, lo que implicaría más masa de gente que se debe distribuir en el mismo espacio. Una distribución basada en información histórica no se adaptaría a los cambios que los cursos puedan tener en el tiempo, necesitando que cada cierta cantidad de años deba volver a replantearse para acatar el panorama del momento. Este acercamiento tiene el costo de necesitar constantemente de recursos humanos que tengan que reevaluar lo que se considere como "información histórica" para así ser definida y repetida en el tiempo.

Asimismo, esta solución produce la existencia de administradores demasiado valiosos dado su conocimiento sobre la distribución de salas histórica. Al necesitar reevaluar cómo distribuir todas las salas y cursos, se producen ciertas reglas las cuales solo son de conocimiento de las personas que las distribuyen, como por ejemplo, si cierta sala históricamente ha sido la mejor para cierto curso. Esta sobredependencia de ciertos administradores es problemática ya que toda la información que es de su conocimiento debe ser traspasada o legada en caso que fuese necesario. Un escenario así no sería ideal y podría apoyarse a través de un acercamiento distinto de la solución.

Por otra parte, se observa que la distribución de horarios es dependiente de la asignación de salas. Al momento de hacer una distribución de horarios, se debe tomar en consideración que posteriormente deberán adecuarse a lo que históricamente se ha realizado en la asignación de espacio. Por lo tanto, si la distribución de salas se automatiza para realizarse a partir de lo que se obtenga de la distribución de horarios, entonces esta última dejará de ser dependiente, permitiendo que en múltiples iteraciones ambas distribuciones vayan mejorando, en lugar de limitarse a la información histórica.

Maximizar el uso del espacio de la facultad es igualmente fundamental. Por un lado, porque desde la administración de la Universidad es importante tener un buen manejo de los recursos físicos, ya que al haber una asignación adecuada de salas al cupo de los cursos, es posible liberar salas que se están ocupando ineficientemente para su posterior uso. Además, esto es importante dado que un óptimo uso del espacio recae directamente en la satisfacción de los estudiantes y el equipo docente al momento de realizar clases.

De igual manera, es posible minimizar la distancia entre las salas de los cursos y el departamento al que pertenecen. Esto en la búsqueda de aumentar la satisfacción entre los estudiantes y docentes, por un lado ya que los estudiantes perciben un mayor sentido de pertenencia cuando sus cursos se dictan en su mismo departamento, y además porque así se

evita el tener que recorrer largas distancias entre clases. Por ejemplo, si un profesor de un curso de Geología tiene que hacer clases en el Edificio Escuela, tendrá que recorrer desde ambos extremos de la facultad para llegar hasta su sala; lo que es además complejo porque debe mover material de estudio potencialmente frágil en este trayecto. Una mejora al sistema debería tomar en consideración los edificios y departamentos para evitar la existencia de estos escenarios.

Finalmente, la validación de la solución es un proceso que puede llevar muchos días dado que la plataforma entrega muy poca retroalimentación. El sistema permite al usuario accionar la distribución de salas, pero una vez esta se completa el único mensaje que obtiene es si se realizó con éxito o no. Que el sistema permita revisar los resultados mediante la misma plataforma disminuiría considerablemente los tiempos de validación. Asimismo, para la SGD es muy importante la información que se obtenga de la distribución, cosa de poder tomarla en consideración para futuras asignaciones. Por lo mismo, es posible entregar mayor retroalimentación, como cuántos cursos y salas fueron asignados, cuáles de estos fueron asignados exitosamente y qué sala se les asignó, entre otras cosas. Igualmente, según la SGD, quienes son usuarios del sistema, la plataforma es compleja de navegar al solo presentar la acción de asignación de salas, sin la presencia de etapas intermedias que permitan evaluar los resultados previo a la asignación definitiva, u observar posibles complicaciones que puedan preverse antes de accionar la distribución.

Dadas estas razones, una mejora en el sistema implicaría un beneficio para toda la comunidad de la Facultad y abriría la posibilidad a futuras mejoras.

1.2. Objetivos

La memoria tiene como objetivo general optimizar la distribución de salas realizada en la plataforma web de Ucampus, mejorando la eficiencia de la asignación, manteniendo los requisitos fundamentales de la solución previa, y reduciendo los tiempos de validación de la distribución realizada.

Junto a esto, se propone considerar los siguientes objetivos específicos:

- Comprender qué involucra la componente histórica dentro de la solución para clarificar todos los requisitos fundamentales que deben mantenerse en una nueva solución.
- Generar un modelo matemático que refleje todas las variables involucradas en la distribución permitiendo que el modelo se perpetúe en el tiempo.
- Entregar una solución que mejore el nivel de ocupación de salas respecto a la solución previa para usar el espacio eficientemente.
- Integrar la solución a la plataforma de Ucampus respetando las convenciones del sistema.
- Entregar retroalimentación a través de la plataforma que permita reducir los tiempos de validación de la solución.

1.3. Solución propuesta

Tomando la base teórica de los problemas de optimización, se exploraron una serie de alternativas. En este análisis, la aproximación que mejor representó al escenario del problema fue la programación lineal, más específicamente, programación lineal entera-mixta. La programación lineal entera-mixta se encarga de resolver problemas de decisión en que para un conjunto de opciones, hay que escoger entre una respuesta discreta o binaria [12]. Un ejemplo sencillo es el "problema de la mochila" en que hay que decidir qué objetos llevar a una excursión, basándose en la capacidad de la mochila, el tamaño del objeto y el beneficio de este. El problema se simplifica a que por cada objeto a considerar, hay que decidir si se lleva o no se lleva [12]. En el caso del problema de asignación de salas, lo que hay que decidir es si por una combinación particular de curso y horario, una sala es óptima. Al igual que el problema de la mochila, este caso se reduce a que por cada combinación de curso, horario y sala, el algoritmo debe decidir si considerarlo o no dentro de la solución óptima, por lo que se concluye que el problema de asignación de salas es un escenario de programación lineal entera-mixta. Otras alternativas contemplaban programación dinámica, programación genética, entre otras, sin embargo, no se adecuaban a los elementos básicos del problema, dado principalmente porque el acercamiento es lineal [12] [19].

Dadas las necesidades que se establecieron sobre la solución, el objetivo del problema de decisión es maximizar el nivel de ocupación. Además, se deseó maximizar la cantidad de cursos que son asignados, y la cercanía entre la sala asignada a un curso y su departamento. Con estas tres aristas a considerar, la solución debería entregar la mejor distribución posible, insertando en una base de datos un pre-calculo de la combinación de salas y cursos que puedan ser consumidos por la plataforma de Ucampus.

A través de una interfaz web, la persona encargada puede observar este pre-cálculo, decidiendo si satisface las necesidades esperadas. De ser así, estas son insertadas en la base de datos como solución definitiva. La plataforma provee una interfaz validada por los usuarios del sistema, y que mejora elementos sobre la solución anterior, volviéndola más agradable de usar.

1.4. Metodología

Para evaluar los problemas de la solución previa, y poder determinar aquellos requisitos que tenían que cumplirse para poder llegar a una respuesta satisfactoria, se realizaron una serie de entrevistas con la SGD y la secretaría del Departamento de Ciencias de la Computación (DCC). Estas entrevistas permitieron tener una visión global sobre el desarrollo de la distribución de salas, desde la parte administrativa del cálculo de la solución a través de la plataforma, hasta los posteriores arreglos que tuvieran que hacerse de forma manual. Con estas conversaciones se identificaron los principales requisitos: maximizar la ocupación de la salas por los cursos; minimizar la distancia entre las salas asignadas a los cursos y los departamentos a los que pertenecen; agregar etapas de validación a la plataforma web; y entregar retroalimentación posterior al cálculo de la solución.

Con estos objetivos en mente, se desarrolló la solución a través de una metodología incremental en dos iteraciones. La primera iteración tenía como meta llegar a una respuesta funcional que asignara todos los cursos a una sala. Esto tenía como fin alcanzar una respuesta al menos tan buena como la anterior. La siguiente iteración tuvo como objetivo el mejorar el cálculo de la solución para que tomara en consideración los requisitos tales como: maximizar el nivel de ocupación de las salas y minimizar la distancia entre los cursos y sus departamentos.

La solución posteriormente fue validada con la SGD. Esto permitió tener la aprobación de los usuarios de la plataforma, cosa de así asegurar que cumpliera con las necesidades que ellos esperaban del proyecto. Junto a esto, se validó la eficiencia de la solución mediante distintas validaciones de datos: se calculó la cantidad de cursos distribuidos; se determinó el nivel de ocupación comparándolo con el de la solución previa; y se definió el nivel de preferencia de los cursos sobre ciertas salas dados los departamentos y edificios respectivos, a fin de compararlo con los resultados de la solución previa.

1.5. Estructura del documento

En los siguientes capítulos, esta memoria examinará el contexto del problema, su situación actual y puntos fundamentales sobre los que se deseaba trabajar (Capítulo 3); el análisis ejercido sobre el problema y diseño de la solución (Capítulo 4); la implementación de la solución, los pasos que llevaron a cubrir los elementos básicos, además de aquellos que se pulieron para encontrar su mejor versión alcanzable (Capítulo 5); la validación realizada para comprobar que se había llegado a una respuesta satisfactoria (Capítulo 6); y finalmente, las conclusiones al respecto (Capítulo 7).

Capítulo 2

Marco teórico

Como durante esta memoria se hablará sobre diversos temas que pueden no ser de conocimiento general, en este capítulo se explicarán aquellos que se consideren necesarios para la correcta comprensión del tema. En la Sección 2.1 se detallará el problema de asignación de salas. En la Sección 2.2 se profundizará en los conceptos necesarios de la optimización, su base teórica y distintas variantes. Finalmente, en la Sección 2.3 se describirán brevemente algunas de las herramientas utilizadas en este trabajo, como Gurobi, JSON y AMPL.

2.1. Asignación de Salas

La Asignación de Salas, conocida en inglés como Room Assignment, es el problema de distribuir salas a un conjunto de cursos o eventos, tomando en consideración las capacidades de las salas disponibles, los cupos de los cursos, y la disponibilidad de horario. Dado que es un problema clásico, se han planteado una serie de posibles respuestas [26], cada una basándose en una versión distintiva de éste, adecuándose a sus respectivas limitaciones. Para contextualizar el problema, más adelante se describirá el cómo diversas universidades responden a esta disyuntiva.

Tomando en consideración que el problema de Asignación de Salas tiene una gran cantidad de limitaciones (cursos, salas, horarios, cupo de los cursos, capacidad de las salas, choques entre cursos) su complejidad se clasifica como NP-Completo. Se define como NP-Completo a todo problema que es parte de NP y se considera NP-Difícil. Esto implica que el problema puede ser reducido en tiempo polimonial a un problema ya existente clasificado como NP-Completo [8]. Los problemas NP-Completos son considerados como los más difíciles de computar, es por esto que este problema se decide resolver mediante heurísticas que usan algoritmos computacionales de optimización, los cuales disminuyen la complejidad del problema encontrando la solución óptima dentro de un rango alcanzable [21]. En este caso, se considera como óptima una respuesta que, según parámetros a definir, sea mejor a aquella encontrada por la solución previa.

Para abordar el problema propuesto es necesario establecer las restricciones básicas a las

cuales éste se enfrenta:

1. La capacidad de una sala debe ser mayor o igual a la cantidad de alumnos que debe haber en un curso.
2. Un curso no puede dictarse en dos salas al mismo tiempo.
3. En una sala no pueden dictarse dos cursos al mismo tiempo.

Para el segundo punto es necesario notar que existen casos como las clases auxiliares simultáneas, en que una misma sección es separada en dos clases, por lo que para la misma "clase" se requieren dos salas. No obstante, es una excepción a la regla, y se observa que en las bases de datos institucionales, bases en donde se almacena la información del catálogo en Ucampus, no hay forma de diferenciar estas de aquellas que no realizan auxiliares simultaneas. Por lo tanto, no se consideran como parte del escenario al ser indistinguibles del resto de los cursos.

Otras restricciones se relacionan con la necesidad de que los estudiantes y docentes realicen su clases cómodamente. Estas restricciones, denominadas restricciones blandas, velan por la satisfacción y comodidad de los estudiantes y docentes [21]. Entre ellas, se puede identificar el que las salas de un curso cuenten con proyectores, el minimizar la distancia de las salas entre clases, el que las cátedras de un mismo curso se dicten en la misma sala, el maximizar el nivel de ocupación de las salas, entre otras cosas.

2.1.1. Soluciones de otras universidades

Dado que éste es un problema que se ha dado múltiples veces y que acata directamente a los establecimientos de educación superior, cada Universidad necesita encontrar su propia solución al problema. Por esto se detallará el cómo distintas instituciones responden a esta disyuntiva. Se nota que las universidades descritas no hacen profunda transparencia de sus procesos o recursos, por lo que a pesar de ser de posible utilidad, no podrá detallarse el tamaño de sus procesos fuera del número de estudiantes.

El primer caso a revisar es el de California State University en East Bay [4], en donde cursan alrededor de 15.000 estudiantes. El proceso de asignación de salas empieza aproximadamente dos meses antes del inicio de clases y se basa en un procesamiento por lotes. Así, se toman en consideración las siguientes prioridades: movilidad o discapacidad del docente, que el cupo del curso se adecue a la capacidad de la sala, uso de tecnología del curso, horarios de los docentes, y proximidad a los departamentos institucionales. Este proceso se hace en conjunto a la distribución de horarios, por lo que para una base cualquiera de cursos, horarios y salas, se busca la mejor combinación de aquellas variables.

En el caso de Rice University [24], con aproximadamente 7.000 estudiantes, la mayor prioridad al problema se pone en la frecuencia y los patrones de uso de salas y preferencia de profesores. De esa forma, tomando la información de los cursos que los departamentos van a realizar, se asegura además que la capacidad de la sala satisfaga el cupo del curso, y que los atributos de la sala correspondan a lo deseado por éste. Además, se pone prioridad en minimizar la distancia recorrida por docentes y estudiantes.

Después, se observa el cómo Yale University [25] responde al problema, institución con más de 12.000 estudiantes. Allí, primero ciertos cursos son pre-asignados por los departamentos, y luego se realiza el resto de las distribución. Todo requerimiento especial, tal como acceso para discapacitados, o equipamiento tecnológico, debe ser tomado en consideración, siendo la arista fundamental de la asignación, además de que el espacio de la sala pueda soportar el número de asistentes del curso asignado.

Para Brown University [23], donde estudian alrededor de 9.000 personas, se indican los siguientes factores durante la distribución: inscripción actual e histórica del curso; relación de los atributos de la sala y las necesidades del docente; y disponibilidad de la sala respecto al total de potenciales cursos para cierto horario. En este caso se da flexibilidad a los requerimientos de un curso ya que se acepta que no siempre se pueden complacer dichas necesidades.

Finalmente, University of the Pacific [14], que posee más de 6.000 estudiantes, toma en consideración los siguientes factores: los cursos; el tipo de clases, divididas en primarias y secundarias, en donde las primeras se componen de las cátedras principales, mientras que las segundas son laboratorios o sesiones de discusión; las salas disponibles; el número de horas de un curso a la semana; la cantidad de estudiantes semanal que toma un curso; y la proporción de la capacidad de una sala ocupada por el número de estudiantes de un curso. De esa manera, con esa información, se notan una diversidad de factores al momento de distribuir, entre los que destacan: minimizar la cantidad de estudiantes por clases; uso de "horario prime" (de 9:00 a 15:00 horas), en donde se deben distribuir, a lo sumo, un 60 % de las clases; prioridad de horario por ciertos cursos; acuerdos previos realizados por los departamentos; necesidades de equipamiento por ciertos cursos sobre las salas; accesos adecuados; entre otras cosas.

Para poder llegar a una solución matemática es necesario formular un modelo de optimización el cual genere una respuesta óptima para el problema. En la siguiente sección se detallará la base teórica con que la que se decide abordar este escenario.

2.2. Optimización

En esta sección, primero se definirá qué es un problema de optimización y cómo se puede ocupar programación lineal para resolver esta clase de problemas. Luego, se mencionarán otras técnicas que se pueden usar para resolver problemas de optimización.

La optimización o programación matemática [6], se puede definir como una técnica que determina la mejor solución a un problema. Un problema de optimización se define mediante un modelo matemático, el cual busca ser una representación simplificada del mundo real. El modelo matemático [5] comprende tres conjuntos básicos de elementos:

- Variables de decisión: Incógnitas o decisiones que deben determinarse al resolver el modelo.
- Restricciones: Conjunto de relaciones que ciertas variables están obligadas a obedecer, y que por lo tanto el modelo debe cumplir.

- Función objetivo: Función que mide la efectividad de la solución como una función matemática de las variables de decisión. Una solución óptima será aquella que obtenga el mayor o menor resultado de la función objetivo, dependiendo del problema.

La programación matemática [12] se encarga de describir estos problemas de optimización tomando en consideración las variables de decisión y restricciones, cosa de maximizar o minimizar la función objetivo.

2.2.1. Programación lineal (PL)

La programación lineal [16] es una variante de la programación matemática que trata exclusivamente con restricciones y funciones objetivo matemáticamente lineales.

Un problema de programación lineal identifica cuatro componentes básicas:

- El conjunto de datos
- El conjunto de variables relacionadas al problema con su respectivo dominio definido
- El conjunto de restricciones lineales que limitan la solución del problema
- La función lineal que debe ser maximizada o minimizada

Un problema clásico de programación lineal es el "problema de la dieta" [16]. El problema de la dieta consiste en determinar las cantidades de distintos nutrientes que deben ingerirse para asegurar ciertas condiciones de nutrición y minimizar el coste de compra de los nutrientes. De modo más preciso supóngase que se conocen los contenidos nutritivos de ciertos alimentos, sus precios y la cantidad mínima diaria de nutrientes aconsejada. El problema consiste en determinar la cantidad de cada alimento que debe comprarse para que se satisfagan los mínimos aconsejados y se alcance un precio total mínimo. En este problema tanto la restricción (que la suma total de nutrientes sea mayor al mínimo saludable) como la función objetivo (minimizar el costo de los alimentos) son lineales.

Programación lineal entera-mixta

Si bien la programación lineal optimiza escenarios lineales con soluciones continuas, esto no siempre se adapta a problemas reales puesto que en muchas instancias las respuestas están restringidas a valores binarios o discretos. La programación entera-mixta [16] es una subdivisión de programación lineal en que la solución se decide entre valores enteros o binarios.

Un ejemplo básico de programación lineal entera-mixta es el "problema de la mochila" [16]. Considérese un excursionista que debe preparar su mochila. Asimismo, considérese que hay una serie de objetos de utilidad para el excursionista, pero que el excursionista solo puede llevar un número limitado de objetos. El problema consiste en elegir un subconjunto de objetos de tal forma que se maximice la utilidad que el excursionista obtiene, pero sin rebasar su capacidad de acarrear objetos.

En un escenario de programación lineal, la respuesta estaría en los límites de soluciones

continuas. En el "problema de la dieta", la respuesta es la cantidad de alimentos que se deben ingerir, los cuales son números enteros para cada alimento. Mientras, en el "problema de la mochila" la solución es una respuesta binaria (sí o no) por cada objeto que desea llevarse.

2.2.2. Otros tipos de optimización

Se detallarán otros enfoques de programación matemática a modo de comparación y análisis posterior. Estos se mencionarán brevemente para cubrir sus bases teóricas, y así en el Capítulo 4 determinar por qué se decantó por la programación lineal en su lugar.

El primer enfoque a describir es el de programación dinámica [20]. Este método soluciona problemas basándose en decisiones secuenciales con etapas sucesivas, en donde se busca minimizar el coste total de tales decisiones. En cada etapa se valora tanto el coste actual de tomar una decisión como el de las decisiones futuras que emergerán a consecuencia de ella.

Otro de los planteamientos posibles es la programación genética [19]. Este tipo de optimización se basa en la selección natural, tomando una población inicial, la cual, mientras no cumpla una condición de término, irá generando nuevos individuos, quienes heredarán características de sus predecesores. Con un grado de mutación, este enfoque permite entregar un amplio rango de soluciones desde su estado inicial. Así, su sentido es llegar a un nivel óptimo en base a los atributos heredados.

2.3. Herramientas

En esta sección, se entregan descripciones relevantes de las principales herramientas ocupadas en este trabajo de memoria.

2.3.1. Gurobi

Gurobi [15] es un software optimizador que resuelve un problema matemático, que toma la descripción de un problema y calcula su solución. Permite resolver problemas de programación lineal y programación cuadrática, y soporta una serie de lenguajes de programación, tales como C, C++, Java, Python, entre otros. Este software se considera un framework el cual recibe como entrada un script realizado en alguno de los lenguaje mencionados que detalle el problema matemático. Gurobi interpreta cómo puede ser abordado el problema y qué algoritmos usar, y en base a eso entrega una solución. Gurobi se acepta como un framework en lugar de una librería, ya que exige ciertas reglas en su implementación para funcionar correctamente, como seguir cierta estructura en su implementación, en lugar de simplemente implementarse y usarse a disposición del programador.

2.3.2. JSON

JSON [10] (siglas para JavaScript Object Notation, inglés para "Notación de Objetos de JavaScript") es un formato en texto estándar, que representa datos estructurados en sintaxis de objetos de JavaScript. Se utiliza para transmitir información en aplicaciones web.

2.3.3. AMPL

AMPL [2] (siglas de A Mathematical Programming Language, inglés para "Un lenguaje de programación matemática") se describe como uno del los tantos lenguajes de programación algebraico, capaz de expresar y describir en notación algebraica problemas de optimización.

Capítulo 3

Contexto

En este capítulo se detallarán las circunstancias que rodean al problema a abordar. En la Sección 3.1 se detallará el proceso que envuelve a la distribución de salas, en la Sección 3.2 se describirá la forma en que Ucampus respondía al problema, y en la Sección 3.3 se profundizará en los problemas que el proceso enfrentaba previamente.

3.1. Proceso actual de asignación de salas

En la FCFM se cursan varias carreras terminales, nueve de Ingeniería Civil y cuatro de Licenciatura. Un estudiante que ingresa a la facultad tiene que cursar cuatro semestres de ramos de Plan Común, un programa previo que involucra cursos de todas las carreras, antes de decidir por una especialidad. Estas especialidades son organizadas por distintos departamentos, mientras que Plan Común es coordinado por la Escuela de Ingeniería. Todos los departamentos disponen de recursos como cursos y profesores, mientras que la Escuela dispone de las salas y la organización de los cursos obligatorios. Así, la relación académica entre los departamentos y la Escuela es mutua entre lo que el departamento puede disponer y lo que la Escuela necesita. La Subdirección de Gestión Docente (SGD) es quien hace de principal organizadora sobre los procesos académicos de la Escuela.

Por otro lado, en la estructura administrativa de la Facultad, los departamentos organizan los cursos y profesores de sus respectivas carreras, cosa de poder dictar sus clases correctamente según las necesidades del departamento y la Facultad. Asimismo, los departamentos se organizan independientemente para luego informar a la SGD sobre qué cursos se dictaran en qué horario tentativo y con qué profesor durante el semestre. Esto es análogo a lo que se enfrenta con Plan Común, siendo organizado directamente por la SGD y organizado según los recursos disponibles por los departamentos. Así, la SGD coordina todos estos datos para luego organizar la distribución de salas basándose en la información consensuada con los departamentos.

La distribución de salas se organiza directamente en la plataforma web de Ucampus, la que a su vez está a cargo de la SGD, siendo quien ordena y gatilla dicha operación en conjunto

a la organización general del proceso. Si bien existen una serie de procesos previos a esta acción, relativos a la organización de los departamentos y la Escuela, todo queda en manos finalmente de la distribución de salas realizada por la plataforma y la posterior validación que hace la SGD sobre esta respuesta. Por lo mismo, para entender correctamente cómo es que actualmente se realiza la asignación de salas, es necesario entender tanto la solución misma que existe en la plataforma de Ucampus como los distintos procesos previos llevados administrativamente.

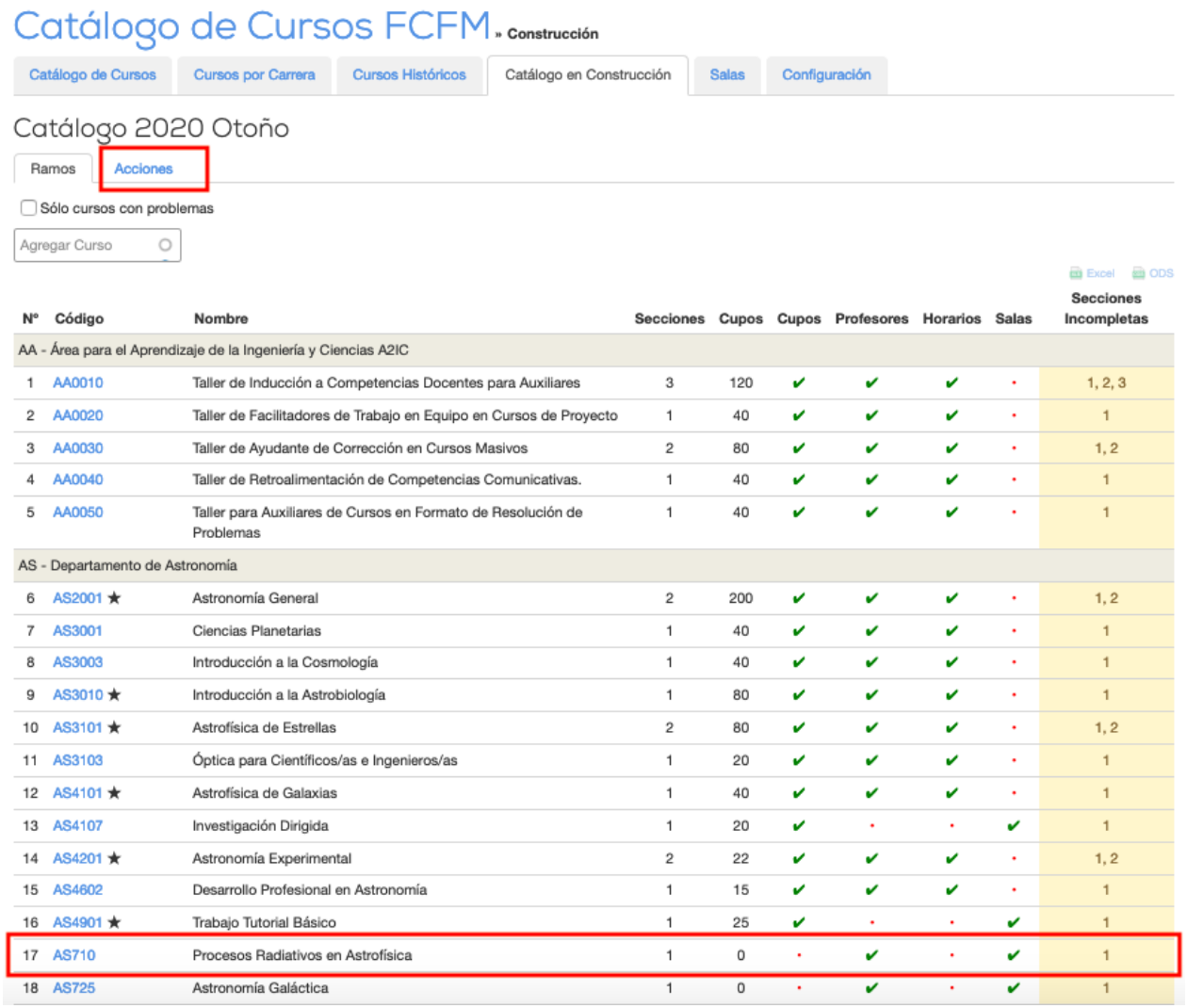
3.2. Solución en Ucampus

Previamente había una solución funcional al problema de distribución de salas en la plataforma web de Ucampus. Ésta recibe de entrada los cursos que van a dictarse durante el semestre, su respectivo cupo, las salas disponibles y los horarios determinados, datos coordinados por la SGD según los procesos previos tales como la asignación de horarios y la organización con los departamentos. Con esta información, la plataforma debe encontrar una distribución de salas para un curso y su horario específico, es decir, encontrar un lugar físico donde pueda ser dictado.

Dado que los datos de entrada son los cursos, sus cupos y los horarios asociados, la persona encargada, en este caso la SGD, debe velar para que toda esa información esté correctamente definida previamente. Para ello debía ingresar al módulo administración de cursos del "Catálogo de cursos" en Ucampus, donde se observaba el estado de cada curso en ese instante. En la Figura 3.1 se observa dicho módulo, donde se muestra una vista parcial de los cursos configurados en el sistema para un semestre. En este ejemplo, el curso marcado en una caja roja (Procesos Radioactivos en Astrofísica) tiene una sección, con cupo aún no establecido (0), profesor asignado y horario no determinado. El sistema ilustrado en la figura muestra la sala como establecida (la columna sala tiene una verificación verde) porque aún no se han precisado todos los parámetros allí ilustrados (cupo y horarios, específicamente, los cuales muestran una cruz roja en la interfaz). Una vez hecha una asignación, se puede revisar en esta misma plataforma.

En el momento en que todos los datos de entrada estén completos, o sea, cuando todos las columnas contaran con una marca de verificación en verde, puede ejecutarse la distribución de salas, la que se encuentra en la pestaña "Acciones" (marcado en una caja roja en la Figura 3.1). Al realizar la asignación el sistema recibe la información de las bases de datos de la Universidad de Chile sobre los horarios, cursos y salas, no tan solo respectivos al semestre presente sino que con información histórica. Esto debido a que esta distribución tomaba asignaciones pasadas y las repetía con un semestre de desfase. Así, se buscaba que si un curso tuvo una sala específica el semestre ante-pasado, entonces la elección de dicha sala tendría prioridad siempre y cuando el cupo del curso no excediera la capacidad.

De producirse una asignación exitosa, entonces el sistema entregaría el mensaje "Excelente! No hay cursos con problemas" como se ilustra en la Figura 3.2. Esto significa que todos los cursos fueron asignados a una sala para su horario específico. No obstante, si la distribución fallaba el usuario debía comprobar los datos que se ingresaron al sistema para revisar qué es



N°	Código	Nombre	Secciones	Cupos	Cupos	Profesores	Horarios	Salas	Secciones Incompletas
AA - Área para el Aprendizaje de la Ingeniería y Ciencias A2IC									
1	AA0010	Taller de Inducción a Competencias Docentes para Auxiliares	3	120	✔	✔	✔	·	1, 2, 3
2	AA0020	Taller de Facilitadores de Trabajo en Equipo en Cursos de Proyecto	1	40	✔	✔	✔	·	1
3	AA0030	Taller de Ayudante de Corrección en Cursos Masivos	2	80	✔	✔	✔	·	1, 2
4	AA0040	Taller de Retroalimentación de Competencias Comunicativas.	1	40	✔	✔	✔	·	1
5	AA0050	Taller para Auxiliares de Cursos en Formato de Resolución de Problemas	1	40	✔	✔	✔	·	1
AS - Departamento de Astronomía									
6	AS2001 ★	Astronomía General	2	200	✔	✔	✔	·	1, 2
7	AS3001	Ciencias Planetarias	1	40	✔	✔	✔	·	1
8	AS3003	Introducción a la Cosmología	1	40	✔	✔	✔	·	1
9	AS3010 ★	Introducción a la Astrobiología	1	80	✔	✔	✔	·	1
10	AS3101 ★	Astrofísica de Estrellas	2	80	✔	✔	✔	·	1, 2
11	AS3103	Óptica para Científicos/as e Ingenieros/as	1	20	✔	✔	✔	·	1
12	AS4101 ★	Astrofísica de Galaxias	1	40	✔	✔	✔	·	1
13	AS4107	Investigación Dirigida	1	20	✔	·	·	✔	1
14	AS4201 ★	Astronomía Experimental	2	22	✔	✔	✔	·	1, 2
15	AS4602	Desarrollo Profesional en Astronomía	1	15	✔	✔	✔	·	1
16	AS4901 ★	Trabajo Tutorial Básico	1	25	✔	·	·	✔	1
17	AS710	Procesos Radiativos en Astrofísica	1	0	·	✔	·	✔	1
18	AS725	Astronomía Galáctica	1	0	·	✔	·	✔	1

Figura 3.1: Plataforma web de administración de cursos en Ucampus, estado de la distribución de salas. Ilustra la información que se ingresa al proceso de asignación de salas, y también se utiliza para verificar los resultados de una asignación, una vez generado.

lo que podría estar generando complicaciones.

Posteriormente, la SGD revisa las salas y los cursos distribuidos. Así, la persona encargada de la validación de los resultados debe ingresar a la plataforma de administración de cursos en "Catálogo en Construcción" mencionada previamente (ver Figura 3.1). Allí, se validaría manualmente si es que la distribución es o no satisfactoria, cosa de hacer cambios posteriores en caso de emerger algún problema. Para poder revisar la sala que se le asignó a cierto curso, se debía hacer click en su nombre y así observar toda la información relativa a este.

Entendiendo el proceso y cómo es que el usuario interactuaba con la plataforma, se procederá a detallar el diseño y la arquitectura que Ucampus usa para manejar sus funcionalidades, entre ella, la distribución de salas.

Figura 3.2: Mensaje de éxito una vez se realiza el proceso de asignación

3.2.1. Arquitectura de Ucampus

Ucampus posee una arquitectura muy bien definida. Su correcta comprensión es esencial si se desea adaptar la plataforma a un nuevo funcionamiento, dado que su escalabilidad futura depende de que se respeten sus convenciones establecidas. Por esto, se procederá a detallar la arquitectura del sistema.

La plataforma de Ucampus trabaja sobre un sistema de módulos. Esta comunicación queda ilustrada en la Figura 3.3. Allí, cada módulo organiza su código según un patrón de Modelo-Vista-Controlador (MVC) [13]. Por lo tanto existen tres carpetas de trabajo, aquella encargada del modelo se llama `include`, la de la vista `template` y la del controlador `web`. Asimismo, cada módulo se comunica con la base de datos de Ucampus llamada `MUFASA`.

Para modificar algo sobre el módulo se debe trabajar sobre las convenciones del estilo MVC. No obstante si se desea agregar una funcionalidad completamente nueva entonces debe crearse un nuevo módulo con su respectivo `include`, `template` y `web`. Por lo tanto, una nueva funcionalidad debe adecuarse a esta convención para ser añadida al sistema de Ucampus. Todos los módulos se comunican con un código `Kernel` el cual sirve de intermediario entre los usuarios y los módulos.

Previamente, la solución que Ucampus proveía al problema yacía en el controlador del módulo de catálogo de cursos. Allí, un script hacía la revisión de los datos a asignar y con base en la información histórica se encargaba de distribuir las salas a los cursos, verificando que el curso no tuviera mayor cupo a la capacidad de la sala.

Sobre esta estructura es que es necesario trabajar al querer implementar una nueva distribución de salas. Se observa que toda funcionalidad extra que no sea parte estricta de esta definición, debe desarrollarse de forma externa al módulo (o sea, fuera del modelo `include`, la

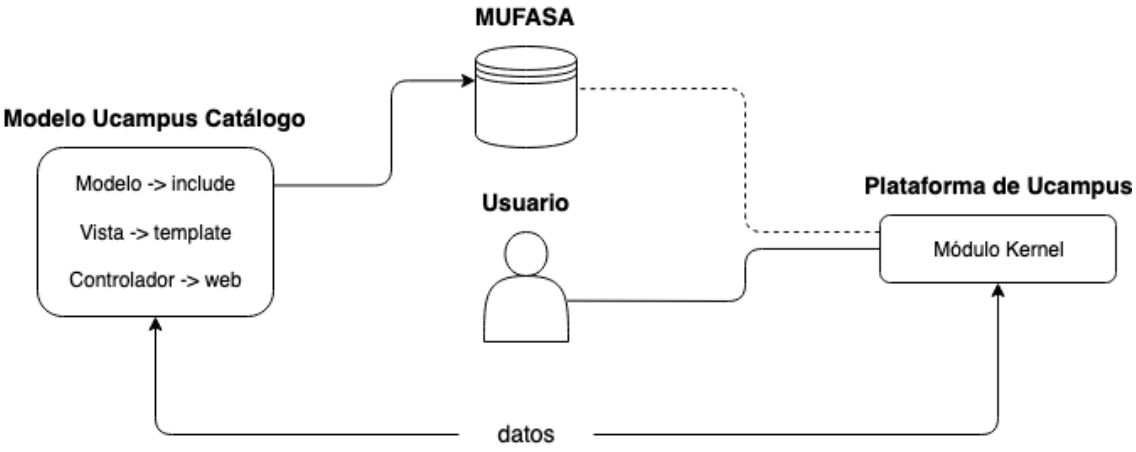


Figura 3.3: Diagrama de arquitectura de Ucampus

vista `template` y el controlador `web`), dado que alterar la estructura rompería la convención establecida por la arquitectura.

Comprendiendo la solución previa desde su desarrollo en la plataforma de Ucampus, ahora se detallará la organización de la SGD al momento de ejecutar esta acción y los problemas que la subdirección observaba en el proceso.

3.3. Problemas con el proceso y herramienta actuales

Ahora se abordará el cómo las distintas organizaciones de la Universidad respondían al problema de distribución de salas y qué problemas emergían a partir de este orden. En la Sección 3.3.1 se detallará la labor de la SGD en la asignación de salas y los problemas que han identificado al respecto. Luego en la Sección 3.3.2 se observará el lugar que los departamentos ocupan en el proceso, para entender su relevancia en la solución.

3.3.1. Subdirección de Gestión Docente

La SGD es la entidad encargada de administrar la distribución de salas. Antes de realizar la distribución, la SGD debe recibir la información relativa a todo el contexto en que se asignarán las salas. Esto implica una distribución de horarios previa la cual se organiza con cada departamento según las necesidades de éstos en conjunto con la Escuela. Una vez se realiza una distribución de horarios satisfactoria para la SGD debe asegurarse que hay un cupo determinado para cada curso, y con esos parámetros (cupo y horario) es que se puede comenzar la asignación de salas.

Una vez que se ejecuta el proceso de distribución con la solución previa, la SGD se encargaba de evaluar los resultados a través de la pestaña de administración de cursos en la plataforma web de Ucampus (ver Figura 3.1), y así dictaba si es que éstos eran satisfactorios

o no según las necesidades de la Escuela. Este proceso podía tomar varios días dado que se debía revisar caso a caso que la asignación hubiera sido correcta, y en el proceso de coordinar con los departamentos podía haber varios cambios en la distribución de salas dadas las necesidades específicas que los profesores o departamentos podían tener sobre ellas.

				CAPACIDAD		EQUIPOS		AUDIO		
Nº	EDIFICIO	Sala	Cátedra	Control	Data	Pc	Microfono	Pc	Tipo Pizarra
8	GEOLOGIA	G201	76	38	Si	Si	Si	Si	Acrílica
9	GEOLOGIA	G301	103	50	Si	Si	No	Si	Acrílica
10	GEOLOGIA	G302	88	44	Si	Si	No	Si	Acrílica
11	GEOLOGIA	G303	64	32	Si	Si	No	Si	Acrílica
12	GEOLOGIA	G304	114	57	Si	Si	Si	Si	Acrílica
13	GEOLOGIA	G305	114	57	Si	Si	Si	Si	Acrílica
14	GEOLOGIA	G306	80	40	Si	Si	No	Si	Acrílica
15	FISICA	F9	40	20	Si	Si	No	Si	Acrílica
16	FISICA	F10	152	75	Si	Si	Si (2)	Si	Tiza y Plumón(Nueva)
17	FISICA	F11	40	20	Si	Si	No	Si	Acrílica
18	FISICA	F12	56	27	Si	Si	No	Si	Cerámica Metalizada
19	FISICA	F20	108	54	Si	Si	No	Si	Tiza y Plumón(Nueva)
20	FISICA	F21	104	52	Si	Si	Si	Si	Tiza y Plumón(Nueva)
21	FISICA	F22	43	20	Si	Si	No	No	Acrílica
22	QUIMICA	Q10	148	74	Si	Si	Si(2)	Si	Tiza y Plumón(Nueva)
23	QUIMICA	Q12	59	30	Si	Si	No	Si	Acrílica
24	QUIMICA	Q13	39	20	Si	Si	No	Si	Acrílica Portátil y Cerámica Metalizada
25	QUIMICA	Q0	103	50	Si	Si	Si	Si	Tiza y Plumón(Nueva)
26	QUIMICA	QP	98	50	Si	Si	Si	Si	Tiza y Plumón(Nueva)

Figura 3.4: Vista parcial del archivo Excel con la información acerca de las salas de la Facultad

Con el fin de comprender la información con la que la SGD trabaja en la distribución de salas, desde la subdirección se entregó un archivo Excel el cual detalla toda la información de las salas sobre las que se tiene acceso (ver Figura 3.4). Entre los datos accesibles se tiene: edificio, capacidad, disponibilidad de diapositivas, tipo de pizarra, entre otras cosas. A partir de este documento se pudieron establecer las restricciones generales del problema. Por esto, se prosiguió a estudiar la información provista por la SGD, además de los datos disponibles en las diversas plataformas de la Universidad de Chile y la FCFM, en particular.

Así, se obtuvieron los siguientes datos:

1. Número de salas en la facultad: 89
2. Edificios en la facultad: 7 (Geología, Física, Química, Eléctrica, Beauchef 851, Escuela, Industrias).
3. Capacidad de estudiante por sala: Desde 12 hasta 152, con 65 en promedio y 5.693 en total al sumar todas las salas.
4. Información de vídeo y audio: Data, PC, Micrófono.
5. Tipo de pizarra: Acrílica, tiza y plumón, y cerámica metalizada.

Para entender las necesidades y limitaciones que la SGD enfrentaba durante el proceso de distribución de salas, se realizó una reunión con María José Contreras, Subdirectora de Gestión Docente, el día 15 de noviembre de 2018. Allí se dio a conocer una serie de problemas que aquejaban la situación previa, los cuales fueron aumentado con el paso del tiempo, y que se irán detallando a continuación según su importancia para la SGD.

Tal como se esbozó previamente, la solución previa se basaba en el uso de información histórica. No obstante, según la SGD está solución podía traer problemas a largo plazo dado que cada año entran más estudiantes y se abren más secciones por curso. Este método podía

Año	Ingreso
2013	776
2014	834
2015	843
2016	849
2017	962
2018	871
2019	906

Figura 3.5: Tabla con cantidad de estudiantes que ingresaron a Plan Común en la FCFM entre 2013 y 2019.

complicar la distribución, siendo especialmente complejo el caso de Plan Común, ya que se implica un enorme flujo de estudiantes que se agregan al problema. Como se observa en la Figura 3.5, entre 2013 y 2019 la matrícula aumentó en 130 personas, y se espera que en un par de años el cupo de estudiantes que ingrese a Plan Común aumente en un 25 %, lo que afectaría considerablemente el número de estudiantes que deben distribuirse en el mismo espacio. Según comentó la Subdirectora, había una gran carga desde el lado de la administración para evaluar si es factible o no soportar el crecimiento de las vacantes en el futuro en el contexto de la solución previa.

Además, para la Subdirección es muy importante que se haga un uso eficiente del espacio. En un escenario ideal, una asignación no le entrega a un curso una sala con menos espacio del que necesita, o con muchísimo más espacio del requerido. Esto lleva a que uno de los factores principales para la satisfacción de la SGD sea qué tanta diferencia habrá entre la capacidad del espacio y el cupo del curso.

Junto a esto, tal como se esbozó previamente, si un curso necesita de equipamiento específico, la sala debería contar con él. Cursos como "CC1000 Herramientas Computacionales para Ingeniería y Ciencias" y "FI1002 Sistemas Newtonianos", cuentan con laboratorios dentro de su programación, por lo que es necesario que estos se dicten en salas con la capacidad suficiente de computadores para satisfacer la demanda de estudiantes.

No obstante, según la SGD también es deseable que cursos dictados por ciertos departamentos se hagan en salas pertenecientes a dichos departamentos. Esto ya que mencionaron que, por un lado, evita el que los estudiantes y el personal académico deban moverse demasiado entre clases y, además, al tener clases dentro de su mismo departamento, se intensifica el sentido de pertenencia y comunidad, lo que conlleva a una mejora de la calidad de vida en general de los estudiantes. En el caso de Plan Común esta limitación no existe ya que dichos cursos pueden darse transversalmente en toda la facultad.

Entre otras limitantes, están los estudiantes y profesores que podrían poseer una discapacidad física. Es deseable que, de encontrarse casos así, existan las medidas necesarias para brindar comodidad a los asistentes. Tal como que, de preferencia, el curso se rinda en un primer piso, o que tenga un acceso sencillo. Asimismo, hay que tomar en consideración que un 15 % de los profesores tienen más de 65 años, lo que podría implicar que tengan necesidades semejantes, además de quizás poseer otros problemas de movilidad o desplazo.

Igualmente, se menciona que hay ciertos profesores con preferencias especiales sobre el equipamiento de las salas. Háblese de docentes que prefieren pizarras blancas o negras, proyectores o transparencias, entre otras cosas. Estas restricciones caen dentro de lo deseable, ya que resultan en casos excepcionales que bien podrían tratarse de forma directa con los profesores involucrados.

Aún cuando se maneja toda la información referente a las salas comunes, según la SGD hay mucha que se encuentra incompleta. Existen departamentos con salas propias cuyos datos no son del conocimiento de la SGD. Sin embargo esto puede descartarse, ya que para motivos de la asignación de salas, solo deben considerarse los espacios comunes.

De igual forma, la existencia de cursos no obligatorios puede inflar la percepción de uso de las salas ya que los electivos pueden ir variando y dependen tanto de la demanda estudiantil como de la disponibilidad de profesores. Aquellos que se dicten un semestre no necesariamente lo harán el siguiente, lo que lleva a que exista información que difiera entre períodos. Además, se puede producir una percepción errónea un semestre en particular sobre el espacio ocupado, ya que hay electivos que tienen pocos cupos, pero que dada una mala asignación pueden darse en espacios con mucha más capacidad. Por ejemplo, en primavera de 2018 al curso "CC5002 Desarrollo de Aplicaciones Web" se le asignó la sala S19, la cual tiene una capacidad de 80 personas para clases y 40 para evaluaciones. No obstante, el curso solo tenía 19 personas de un cupo de 40, teniendo un espacio asignado mucho mayor al verdaderamente necesario.

Con esta información sobre la situación desde la organización de la SGD se pudo tener una visión global del panorama administrativo. Aún así, es necesario tener una mirada de todos los actores involucrados por lo que se conversó con uno de los departamentos al respecto.

3.3.2. Secretaría Docente del DCC

Con el propósito de tener una percepción local respecto al proceso de distribución de salas en los departamentos, se conversó con Sandra Gaez, Secretaria Docente del DCC, sobre el proceso de organización inicial del departamento el 21 de noviembre del 2018. Dado que todos los departamentos siguen un proceso similar, se decidió entrevistar en profundidad a uno en particular para complementar lo comentado por la SGD.

Según lo conversado, antes de adentrarse específicamente en la asignación de salas, existe un proceso de distribución de cursos dentro del departamento. Al momento de determinar quién dictará qué curso, se plantea una asignación de horarios y distribución de profesores desde el departamento, la cual se envía a la SGD. Entre los docentes deciden quién está disponible y desea dictar ciertas cátedras, conversando y organizando a distintos profesores hasta tener cubierto el catálogo deseado.

Ciertos patrones tienden a repetirse. Algunas clases se dan históricamente en ciertos periodos de la semana, permitiendo que ciertas convenciones tiendan a respetarse, no generando muchas complicaciones. Por ejemplo: "CC5401-1 Ingeniería de Software II" frecuentemente se da en los horarios del martes y jueves de 10:15 a 11:45 horas. Esta información luego es confirmada por la SGD, quien valida que los horarios se encuentren disponibles.

Los mayores detalles surgen a raíz de las necesidades de algunos profesores, quienes requieren de cierto equipamiento específico al momento de dar su clase. Por ejemplo, hay profesores que al momento de realizar cátedras, necesitan que en su sala haya dos pizarras, cosa de proyectar en una las diapositivas, mientras que en la otra escribe, y esto implica la necesidad de una sala tanto con diapositivas, como con dos pizarras. No obstante, estos casos son tan específicos que rara vez generan problemas, ya que son abordados directamente desde el departamento.

Con esta información se concluyó que el peso de la organización cae directamente en la SGD, puesto que gran parte del trabajo a nivel local queda delegado a lo que finalmente la Subdirección termine distribuyendo. La organización de parte del departamento es mayoritariamente sobre asignar horarios y posteriormente pedir cambios de salas de ser necesario.

Entendiendo los límites del problema y cómo es que se aborda desde las distintas entidades de la Facultad, se tuvo un panorama del escenario enfrentado. Con esta comprensión, se procederá a analizar dicho contexto y a diseñar una solución que responda al problema.

Capítulo 4

Análisis y diseño

En este capítulo se detallará el análisis del problema realizado a partir de lo descrito en el Capítulo 3. Así, en la Sección 4.1 se describirá las consideraciones que la nueva solución debe abordar respecto al problema. Luego en la Sección 4.2 se detallará el análisis teórico realizado sobre el problema. Finalmente en la Sección 4.3 se diseñará la forma en que se abordará la solución.

4.1. Consideraciones sobre el nuevo sistema

Dada la bien definida estructura administrativa en la organización y el sector académico, se vuelve necesario respetar dicho orden. Las convenciones son funcionales y eficientes, poniendo énfasis en el lugar de la SGD al momento de realizar la distribución de salas, por lo que sería más pesado intentar alterar esta organización a respetar su estructura y trabajar sobre ella. Según la información dada en la Sección 3.3.2, se concluye que el papel que cumplen los departamentos tiene mayor peso en la asignación de horarios, y en la comunicación de sus necesidades a través de la SGD. Por lo tanto, se mueve la atención a la SGD, ya que la labor de los departamentos sale de los límites de lo contemplado por el problema planteado de distribución de salas.

La acción de distribución de salas se encuentra en la plataforma web de Ucampus. Ucampus posee una sólida arquitectura organizativa, por lo que es necesario respetar esta convención, ya que intentar cambiar la estructura rompería la posible escalabilidad futura de la solución, además de su mantención en el tiempo.

Así, es importante que una nueva solución implemente la funcionalidad que actualmente ofrece el sistema, lo que significa ser capaz de mantener la restricción de que el cupo de un curso no exceda la capacidad de una sala. La solución existente puede fallar en distribuir las salas, caso que se puede dar si es que no encuentra una respuesta que asigne todos los cursos, en cuyo caso no asigna ninguno. Por lo tanto, se otorga cierta flexibilidad sobre poder asignar absolutamente todos los cursos a una sala correspondiente.

Según comenta la SGD, hay información previa a la distribución del sistema (como salas exclusivas de los departamentos) que no está a su disposición. No obstante, tal como se esbozó en la Sección 3.3.1, esto no debería generar problemas dado que solo se desean asignar salas en base a los espacios comunes. Dicho eso, debe tomarse en consideración la existencia de estos espacios, cosa de excluirlos en la distribución y así no otorgarle a un curso una sala exclusiva a un departamento que no es parte del espacio común.

Entrando en la solución previa, se observa un gran problema dada su visión a corto plazo. Como dicha solución basa su respuesta en información histórica, este panorama puede cambiar radicalmente con el paso de los años, volviendo inútil la información que se estableció inicialmente. Dado que el cupo de ingreso de la FCFM ha aumentado en el transcurso del tiempo, y se espera que siga creciendo, necesariamente se abrirán más secciones y los cursos aumentarán su cupo, no siguiendo lo que históricamente se pensó como una distribución de salas óptima. Esto motiva a que periódicamente haya que reevaluar la información histórica, generando un gasto de recursos humanos y dejando mucha información en personas clave quienes tienen el peso de entender cuales son las asignaciones óptimas. Automatizar la solución para que evalúe el panorama cada semestre quitaría este peso, ya que por cada período se buscaría la mejor distribución en base a las características que presente en ese momento, en lugar de asumir que lo que históricamente se consideró mejor vaya a ser válido por largas instancias de tiempo.

Además, hay una estrecha relación entre el proceso de asignación de horarios y distribución de salas. Esto significa que una posible respuesta del primero estaría limitado por el segundo, ya que al momento de organizar los horarios es necesario tomar en consideración el cómo ciertos cursos se han dado históricamente en ciertas salas, restringiendo la asignación de horarios al proceso posterior. Automatizar la respuesta de la distribución de salas, cosa de dejar de depender de la componente histórica, independizaría ambos procesos, dando más flexibilidad y generando, no solo una mejor asignación de salas, sino que también una de horarios.

De igual forma, una vez el sistema encuentra una solución, el proceso de validación puede ser muy lento, pudiendo llevar días. Esto ocurre porque el sistema entrega nula retroalimentación una vez que se ejecuta la asignación, y la única forma de revisar las salas asignadas a cada curso es a través de la plataforma de administración de cursos, la que solo permite ver esta información al revisar un curso a la vez. Si la nueva plataforma web de distribución de salas entregara información sobre dicha asignación al momento en que se encuentra una respuesta y en un formato más sencillo de revisar, acortaría considerablemente los tiempos de validación al poder evaluar toda la información al instante en que se encuentra una solución, siendo además consultable posteriormente en la misma plataforma de distribución de salas.

Asimismo, para la Facultad es importante que se haga un buen uso del espacio físico. Por esto, el sistema debería mejorar el nivel de ocupación de las salas, de manera que éstas se adecuen lo mejor posible a la cantidad de estudiantes que asisten a un curso. Cumpliendo con los requisitos ya efectuados por el sistema (que el cupo no exceda la capacidad de la sala), la forma de medir el éxito del sistema a construir, en términos de qué tan buena es la asignación en sí misma, sería comparando el nivel de ocupación, en donde si este aumenta en promedio

se podrá concluir que la respuesta del sistema es mejor a lo que existía previamente. De igual forma, se compararán los mínimos, máximos y desviación estándar de ambas soluciones para evaluar si es que la nueva solución mejora la respuesta en comparación a la previa.

Del mismo modo, como detalló la SGD, la satisfacción de los estudiantes y docentes aumenta cuando sus clases se realizan en el mismo edificio de su Departamento, dado el sentido de pertenencia desarrollado. Esto excluye Plan Común ya que sus clases se diversifican entre distintos departamentos. Además, existen carreras donde los espacios predilectos no son en el mismo edificio del departamento. Por ejemplo, Ingeniería Civil Mecánica e Ingeniería Civil en Computación tienen sus departamentos en Beauchef 851, pero no usan las salas específicas del departamento para hacer clases, sino que las comunes de Beauchef 851. Una buena solución debería tomar en consideración esta cercanía para mejorar la comodidad del estudiantado y equipo docente.

Se espera de igual forma que ciertos cursos que necesiten de equipamiento especial para realizar sus clases, tales como diapositivas o computadores, sean distribuidos en salas con dicho material. No obstante, las bases de datos de la Universidad no ilustran qué salas cuentan con equipamiento o qué cursos lo necesitan, por lo que no sería posible hacer esta asignación de forma automatizada en este momento, dejando en manos de cambios posteriores a la asignación en caso que haya cursos que necesiten del equipamiento. Dentro de los archivos entregados por la SGD (ver Figura 3.4) hay información sobre qué equipamiento posee qué sala, pero agregar esto queda fuera del alcance de la memoria ya que aún sumando esta información a las bases de datos de la Universidad, faltaría conocer qué cursos necesitan qué equipos.

Igualmente, si bien se identificó como objetivo el querer otorgarle a un curso una sala basada en las preferencias del docente encargado, no se ve como posible dicho objetivo dadas las características del contexto. Para poder automatizar un sistema que se adecúe a dichas especificaciones, las bases de datos tendrían que almacenar información sobre qué tipo de sala prefiere qué profesor, datos que aún no se encuentran disponibles.

De la misma forma, es importante que en caso de encontrarse estudiantes o profesores con problemas de movilidad el sistema sea capaz de asignar una sala tomando esto en consideración. No obstante, las bases de datos no tienen esta información, por lo que no sería posible distribuir en base a las necesidades de los estudiantes o los profesores. En la misma línea, si un profesor tiene una preferencia sobre un tipo de sala, no sería factible coordinar esta información de forma automática ya que es parte del juicio del docente y no un parámetro en la información. Por lo mismo, estos puntos serían parte de una etapa de ajuste manual a la distribución de salas durante el proceso de validación de la respuesta.

Además, al analizar el cómo otras universidades, tales como California State, Rice, Yale, Brown y Pacific, responden al problema, se encontraron varias consideraciones en común, no obstante, la prioridad que se le da a cada una varía considerablemente entre distintas instituciones. Por ejemplo, todas vislumbran el que la capacidad de la sala debe ser mayor al cupo del curso asignado, pero mientras California State, Rice y Brown buscan disminuir la brecha entre capacidad y cupo, Pacific privilegia el que haya la menor cantidad posible de estudiantes por sala. Asimismo, se presentan casos como California State y Pacific, que realizan sus asignaciones de sala en simultaneo a las de horarios, caso que no se adecúa al contexto

de este problema. Además, Yale establece como única prioridad durante la distribución los requerimientos especiales tales como equipamiento, movilidad o preferencia de los docentes. Varios de estos casos mantienen como parte de la solución un sistema de preferencias en que saben qué es lo que un curso, docente o estudiante desea de su sala. Se evidencia que, dado que esta información aún no existe dentro de las bases de datos de Ucampus y, por lo tanto, las limitaciones de este problema, no puede hacerse énfasis en este aspecto. De todas formas, se enfatiza el que cada institución de alguna forma da prioridad a lo que los docentes y estudiantes desean según la visión del establecimiento, que en el caso de la FCFM y los límites de la memoria se traduciría a la cercanía entre edificios y departamentos.

Finalmente, se evidencia un problema en el uso del espacio derivado a que ciertos cursos no utilicen completamente su cupo disponible. Sin embargo, este problema deriva directamente de la eficiencia del uso del cupo, saliendo de los límites del alcance de la memoria, al no ser posible establecer cómo es que un cupo será ocupado durante el semestre.

Ahora se procederá a detallar el análisis teórico del problema de optimización a resolver. Para esto se considerará preliminarmente al problema como uno de programación matemática, cosa de simplificar el análisis teórico. Posteriormente, se revisará si es que el enfoque de programación matemática se adecúa o no al escenario descrito.

4.2. Análisis teórico

Entendiendo el problema según el contexto generado, se procederá a analizar las bases teóricas que lo definen con el fin de poder avanzar a diseñar una solución que se adecúe al escenario presentado. Esto implica crear un modelo matemático que incluya los conjuntos de datos junto a sus parámetros; las restricciones del problema y los objetivos buscados.

Para el problema de distribución de salas es posible identificar tres conjuntos de datos: *Cursos*, *Salas* y *Horarios*. Estos conjuntos representan los cursos que se desean dictar durante el semestre correspondiente, el espacio físico en que deben ser cursados, y el periodo de la semana en que se realizarán, respectivamente. Se denotará como *HorariosC* a los horarios asignados a un curso en particular.

Cada uno de estos conjuntos dispone de una serie de parámetros. Un curso tiene un *cupo* máximo de estudiantes que pueden tomar dicho curso y una sala posee una *capacidad* máxima de asistentes soportados por el espacio físico.

Estos elementos son insumos para el problema. Sin embargo, para poder detallar las otras aristas del modelo (restricción y objetivos) es necesario pensar en cuál es el resultado esperado. Dado que se desea asignar una sala a una combinación dada de *curso-horario*, se pensó en una variable binaria *asig* que determine si una tupla *curso-horario-sala* es parte o no de la solución. Esta variable de decisión está descrita por la función ilustrada en la expresión 4.1.

$$asig(s, c, h) = \begin{cases} 1 & \text{si sala } s \text{ se asigna al curso } c \text{ con horario } h \\ 0 & \text{en caso contrario} \end{cases} \tag{4.1}$$

Otra forma de abordar este problema hubiera sido mediante una función $asig(c, h)$ que, para un curso c y un horario h entregara como resultado la *sala* asignada a dicha combinación de *curso-horario*. No obstante, esta formulación hubiera complicado la definición de las restricciones y función objetivo, dada la complicación de intentar validar una respuesta cuando el valor de retorno representa a la *sala*. Mientras, la función elegida ilustrada en la expresión 4.1 es relativamente sencilla de validar al poder comparar y expresar los resultados mediante los valores binarios entregados.

4.2.1. Restricciones

Teniendo los conjuntos y parámetros, es necesario reconocer las limitaciones a las que se enfrenta el problema. Dado que la solución solo puede existir dentro de ciertas restricciones, las cuales determinan el rango de la respuesta, se identificaran las limitaciones dado el contexto.

La primera restricción que se identificó es la de Sala Única. Esta delimita que una *sala* solo puede tener asignado a lo más un *curso* para un *horario* determinado. Su justificación recae en la imposibilidad de que más de un *curso* pueda dictarse en una misma *sala* al mismo tiempo. Además hay que notar que una *sala* no necesariamente debe tener un *curso* en un *horario*. Esta restricción se define por la desigualdad en la expresión 4.2.

$$\forall s \in Salas, \forall h \in Horarios, \sum_{c \in Cursos} asig(s, c, h) \leq 1 \tag{4.2}$$

Luego se da la restricción de Curso Único en que se detalla que para cada *curso* debe haber una y solo una *sala* asignada necesariamente en sus respectivos *horarios* distribuidos. Esto primero porque se pretende que cada *curso* tengan al menos una *sala* asignada en cada *horario*, y además porque un *curso* no se puede dictar en dos *salas* al mismo tiempo. Existen casos excepcionales, como clases auxiliares, donde se divide un *curso* y se dicta en más *salas* al mismo tiempo. No obstante, se consideran casos especiales, los cuales deben revisarse particularmente, dado que en las bases de datos de la Universidad de Chile no se cuenta con información referente a qué clases realizan esta división. Esto queda definido con la ecuación 4.3.

$$\forall c \in Cursos, \forall hc \in HorariosC, \sum_{s \in Salas} asig(s, c, hc) = 1 \tag{4.3}$$

Asimismo, se detalla la restricción de Capacidad, la cual limita el que para cada asignación de *curso* y *sala*, el *cupo* no puede exceder la *capacidad* de la *sala*. Esto dado que físicamente se presume que una *sala* no puede soportar más estudiantes a lo que su *capacidad* define. Dicha restricción se expresa en la desigualdad de la expresión 4.4, donde se detalla que la diferencia entre *capacidad* y *cupo*, para toda combinación de *sala*, *curso* y *horario*, debe ser mayor o igual a 0.

$$\forall s \in Salas, \forall c \in Cursos, \forall h \in Horarios, \big(capacidad(s) - cupo(c)\big) \times asig(s, c, h) \geq 0 \quad (4.4)$$

Dadas las restricciones mencionadas es que se limita el rango en que puede encontrarse una solución, la cual busque dar con la mejor combinación de *sala*, *curso* y *horario* dadas todas las posibles asignaciones. Para esto ahora se describirán los objetivos que se busca completar con el desarrollo de esta solución.

4.2.2. Función objetivo

En consideración a lo detallado en el Capítulo 3, existen una serie de objetivos que se espera que la solución teórica alcance en función de mejorar la respuesta en comparación a la solución previa. Según su prioridad, estas son: aumentar el nivel de ocupación; y mejorar la cercanía entre departamentos y clases, tomando en consideración lo que se considera un edificio cercano y lejano para cada departamento.

Se identifica como objetivo principal el maximizar la ocupación de las *salas*, por lo que en base a esto se modela todo el problema, cuya alta prioridad se debe a la urgencia que fue informada por la SGD en la Sección 3.3.1. Esto dado que es el pilar sobre lo que se considera una buena asignación, ya que se espera que las *salas* se adecúen a las características del *curso*, y se describe al *cupo* y la *capacidad* como la propiedad principal del *curso* y la *sala*, respectivamente.

Por otro lado, según lo comentado por la SGD en la Sección 3.3.1, la satisfacción de los docentes y estudiantes se ve directamente influida por la cercanía que presenta el edificio de la *sala* con el departamento del *curso* asignado. Por lo mismo, se piensa dentro de los objetivos del modelo el poder tomar en consideración esta preferencia para aumentar la comodidad de los involucrados.

Con esta descripción del problema, se conducirá a describir el acercamiento con que se pudo plantear una posible implementación de la solución.

4.3. Diseño

Habiendo analizado el contexto del problema, se empezó a diseñar una forma de resolverlo tomando en consideración las técnicas a disposición y escogiendo aquella que responda de mejor manera al escenario. Por ello, en la Sección 4.3.1 se detallará la decisión de abordar la solución con programación lineal; luego, en la Sección 4.3.2 se confirmará el modelo descrito anteriormente, y se describirán los cambios que sean necesarios; y finalmente en la Sección 4.3.3 se diseñará la interfaz gráfica del sistema junto a la interacción con el usuario.

4.3.1. Programación lineal

Teniendo las bases teóricas descritas, es necesario decidir con qué método o técnica se abordará el problema para encontrar una solución. Preliminarmente se decidió usar programación matemática para detallar el análisis teórico, por lo que ahora se discutirá si es que es el método más adecuado para afrontar el problema.

Dado que en primera instancia este es un problema de optimización, se vuelve clara la decisión de usar la programación matemática como técnica para modelar el problema y encontrar su mejor respuesta. Existe una diversidad de tecnologías las cuales acercan el problema a lenguajes conocidos y, dado el contexto en que se desenvuelve Ucampus, permitiría implementar exitosamente la solución dentro su plataforma web, aprovechando su sistema de módulos, respetando las convenciones y escalabilidad del sistema.

Vislumbrando la programación matemática como forma de abordar el problema, es necesario definir qué tipo de programación será el empleado para describir el contexto. Existen múltiples tipos de optimización tales como programación lineal, cuadrática, dinámica, genética, entre otras.

Dada la descripción que se realizó previamente del problema, es directo que las restricciones asociadas (las expresiones 4.2, 4.3, y 4.4) son lineales. Además, dados los objetivos detallados, lo que se desea es minimizar la diferencia entre *cupo* y *capacidad*, y maximizar la preferencia de ciertos *cursos* por sobre ciertos edificios dados los departamentos a los que pertenecen. Detallando estos problemas como funciones objetivo según las características de la programación matemática, se vuelve claro que dichas funciones objetivo son lineales al ser compuestas por expresiones que lo son.

Asimismo, el escenario también se detalla como problema de programación lineal entera-mixta, en que la variable de *asignación* se describe como discreta o binaria. En este caso, la variable de asignación representaría si es que una combinación de *sala*, *curso* y *horario* pertenece a la solución óptima o no. Dada dicha representación, es directo que la respuesta sería binaria, ya que para cada variable solo existen dos respuestas, si pertenece a la solución o no. Por otro lado, otros tipos de programación matemática no-lineales no se adecuan al contexto del problema dado que la definición de su restricciones y función objetivo son de carácter lineal. Así, es sencillo concluir que la programación lineal entera-mixta sería aquella más adecuada para modelar el escenario presentado por el problema.

4.3.2. Modelo

Entendiendo el problema como uno de programación matemática de carácter lineal entera-mixta, se considera aquello detallado en el análisis y se confirma su lugar en el modelo que se busca implementar. Para esto se identificarán qué elementos deberán conservarse y cambiarse desde el modelo teórico al práctico.

Los conjuntos fundamentales de *Cursos*, *Salas* y *Horarios* se mantienen desde la teoría ya que son los datos de entrada para el problema. Asimismo, se mantienen los parámetros

descritos de *cupo* para *Cursos* y *capacidad* para *Salas*.

Las limitantes del problema se describen según lo detallado en las restricciones de Sala Única, Curso Único y Capacidad (ver las expresiones 4.2, 4.3, y 4.4, respectivamente). Con respecto a Sala Única, esta se refiere a que por cada *sala* en un *horario* puede haber máximo un curso. Aún así, este *horario* no tiene que ser necesariamente el mismo. Por ejemplo, si durante un mismo día un *curso* tiene una clase de 10:00 a 12:00 horas, y otro tiene clases de 10:15 a 11:45 horas, entonces se producirá un *choque*, por lo que no podrían tener la misma *sala*. Este concepto se define en la expresión 4.5.

$$\forall c_1, c_2 \in Cursos, \forall h_1, h_2 \in Horarios$$
$$choque(c_1, h_1, c_2, h_2) = \begin{cases} 1 & \text{si curso } c_1 \text{ en horario } h_1 \text{ choca con curso } c_2 \text{ en horario } h_2 \\ 0 & \text{en caso contrario} \end{cases} \tag{4.5}$$

Una visión superficial del problema respondería a la restricción evitando que dos cursos con un mismo *horario* puedan acceder a la misma *sala*. No obstante, en presencia de *choques* el problema se complejiza, ya que dos *cursos* podrían intersectarse incluso aunque no posean exactamente el mismo horario. Para poder elaborar en la restricción de Sala Única con presencia de *choques* se pensó en realizar un pre-cálculo de las colisiones. Para esto se tomarán todas las combinaciones de *cursos* y *horarios* a disposición, y se revisará qué *curso-horario* choca con qué otro *curso-horario*. De esta forma se puede tener la información de fácil acceso para posteriormente usarse al momento de desarrollar la restricción de Sala Única. Esta función queda definida en la expresión 4.5.

Ahora bien, considerar los *choques* dentro del problema limita bastante el rango de soluciones, por lo que deberá flexibilizarse el modelo para asegurar que el sistema encuentre una solución. Debido a esto, la restricción de Curso Único se modificará a partir de lo que se pensó teóricamente. La definición original de la restricción (ver expresión 4.3) consideraba que un *curso* debía tener una sola *sala* asignada sí o sí. No obstante, dada la necesidad de flexibilizar esta limitante, se reformulará la expresión correspondiente para aceptar a lo más una *sala* distribuida a cada *curso*. Esto queda ilustrado en la desigualdad mostrada en la expresión 4.6. De esa forma, podrá darse que ciertos *cursos* no sean distribuidos. Sin embargo, según se detallará en la función objetivo, se buscará el éxito de la solución permitiendo que el distribuir la mayor cantidad de *cursos* posibles sea parte de las prioridades del objetivo.

$$\forall c \in Cursos, \forall hc \in HorariosC, \sum_{s \in Salas} asig(s, c, hc) \leq 1 \tag{4.6}$$

Tal como se esbozó previamente, el objetivo principal del modelo se había planteado originalmente como la maximización del nivel de ocupación. Esto implica intentar disminuir la diferencia entre la *capacidad* de las *salas* y el *cupo* de los *cursos* asignados a ellas. Por lo mismo, se plantea la primera parte de la función objetivo como detalla la expresión 4.7. De esa forma, se pretende minimizar la diferencia mencionada entre todas las combinaciones de curso-horario y sala asignada.

	Beauchef 851	Beauchef 851 Norte	Edificio Escuela	Edificio Física	Edificio Química	Edificio Eléctrica	Edificio Geología
Astronomía	0	0	50	100	75	75	50
Biotecnología	0	0	50	75	100	75	50
Ciencia de los Materiales	0	0	50	75	100	75	50
Civil	0	0	50	75	75	75	75
Computación	100	100	25	0	0	0	0
Eléctrica	0	0	50	75	75	100	75
Física	0	0	50	100	75	75	50
Geofísica	0	0	25	50	50	75	100
Geología	0	0	25	50	50	75	100
Industrial	100	100	25	0	0	0	0
Matemáticas	100	100	25	0	0	0	0
Mecánica	100	100	25	0	0	0	0
Minas	0	0	25	50	50	75	100
Plan Común	50	50	50	50	50	50	50
Química	0	0	50	75	100	75	50

Figura 4.1: Tabla de distribución de preferencia de cursos

$$\min\Bigg(\sum_{s\in Salas}\sum_{c\in Cursos}\sum_{h\in Horarios}\Big(\big(capacidad(s)-cupo(c)\big)\times asig(s,c,h)\Big)\Bigg) \tag{4.7}$$

Por otra parte, en la flexibilidad de la restricción de Curso Único (ver expresión 4.6), ahora parte del objetivo del modelo es asignar la mayor cantidad de *cursos* posibles. Por esto, se detalla dentro de la función objetivo la maximización de cursos asignados, lo que queda descrito en la expresión 4.8. Así, aún cuando el modelo permita que ciertos cursos no sean distribuidos, intentará que la mayor cantidad posible lo sean.

$$\max\Big(\sum_{s\in Salas}\sum_{c\in Cursos}\sum_{h\in Horarios} asig(s,c,h)\Big) \tag{4.8}$$

Luego, se toma en consideración el deseo por maximizar la cercanía entre los *cursos* y sus departamentos. Esto implica que la sala que se le asigne a un *curso*, de preferencia, sea del departamento al que pertenece, o no quede muy lejos físicamente de éste.

Para plantear este objetivo fue necesario agregar un nuevo parámetro *preferencia*, el cual expresa la predilección de un *curso* por sobre una *sala* según el departamento del *curso* y el edificio de la *sala*. Para esto se agregó al modelo una tabla de preferencias (ver Figura 4.1) la cual expresa dicha prioridad. Las filas representan a los departamentos y las columnas a los edificios del campus. Si un edificio es de primera prioridad para un departamento, entonces obtiene un puntaje de 100, y si es de última prioridad entonces obtiene 0. Cabe notar que este último puntaje no implica que el curso no pueda ser asignado a dicho edificio, sino que tendrá mínima preferencia respecto al resto de los edificios.

Hay edificios los cuales se comparten entre varios departamentos. Los cursos de Astronomía y Física se dan en el Edificio Física; los de Ingeniería Civil Química e Ingeniería Civil en Biotecnología se dan en el Edificio Química; y Geofísica, Geología y Minas se dan en el Edificio Geología, por lo tanto, todos obtienen un puntaje de 100 sobre dichos edificios. Caso

particular es el de los departamentos de Ingeniería Civil en Computación, Ingeniería Civil Industrial, Ingeniería Civil Matemática e Ingeniería Civil Mecánica, ya que sus edificios son de uso exclusivo del departamentos, por lo que sus *cursos* no se dictan en sus *salas*, sino más bien en los espacios comunes disponibles. Por lo mismo, el edificio con mayor prioridad para dichos departamentos son Beauchef 851 y Beauchef 851 Norte.

Desde ese punto, el resto de los edificios tienen un puntaje asignado en base a qué tan lejos se encuentra el edificio en comparación a aquel de máxima prioridad. Por ejemplo, Geología tiene prioridad secundaria sobre el Edificio Eléctrica, dado que es el más cercano al Edificio Geología, por lo que obtiene un puntaje de 75 al respecto, mientras que el Edificio Beauchef 851 tiene mínima preferencia ya que se encuentra al otro extremo de la Facultad. Dos casos especiales son Ingeniería Civil y Plan Común, el primero porque su edificio no cuenta con espacio común para el curso de clases, por lo que deben disponer de los edificios cercanos, sin tener máxima prioridad por ninguno, ya que todos se encuentra a cierta distancia del Edificio Civil. Por otra parte, Plan Común no cuenta con ningún edificio específico, por lo que se le da un puntaje equitativo de 50 sobre todos los edificios, lo que expresa una preferencia neutra por cualquiera que se le sea asignado.

Esta tabla no pudo ser validada con la SGD debido a que fue lo último en ser agregado dentro del problema. De todas formas, los valores que toma para cada combinación de *curso* y *sala* pueden ser modificados según las necesidades de la SGD, por lo que será parte del trabajo futuro contemplar los valores que mejor expresen las preferencias de los estudiantes y docentes.

Con la tabla de preferencias definida, se agregó esta nueva información dentro del modelo como parte del parámetro *preferencia*, que entrega el puntaje de prioridad que un *curso* posee sobre cierta *sala*. Así, se define como parte de la función objetivo la maximización de esta preferencia por todas las ternas de *sala-curso-horario* asignadas. Esto queda formalizado en la expresión 4.9.

$$\max\Big(\sum_{s\in Salas}\sum_{c\in Cursos}\sum_{h\in Horarios}\big(preferencia(s,c)\times asig(s,c,h)\big)\Big) \tag{4.9}$$

Con el modelo definido, se pudo describir aquello detallado teóricamente en forma práctica para su posterior implementación. De esa forma, se puede proceder a precisar el diseño de la interfaz según lo analizado previamente.

4.3.3. Interfaz gráfica

Para completar el diseño de la plataforma se tuvo que pensar en la interacción que el usuario tiene con la misma. Por esto, se describirá el cómo se abordó la capa de usuario según el análisis ejercido en la Sección 4.1.

Dado que actualmente la distribución de salas existe en una pestaña la cual permite ejecutar la acción de asignación, se ha detallado que esta operación es demasiado directa

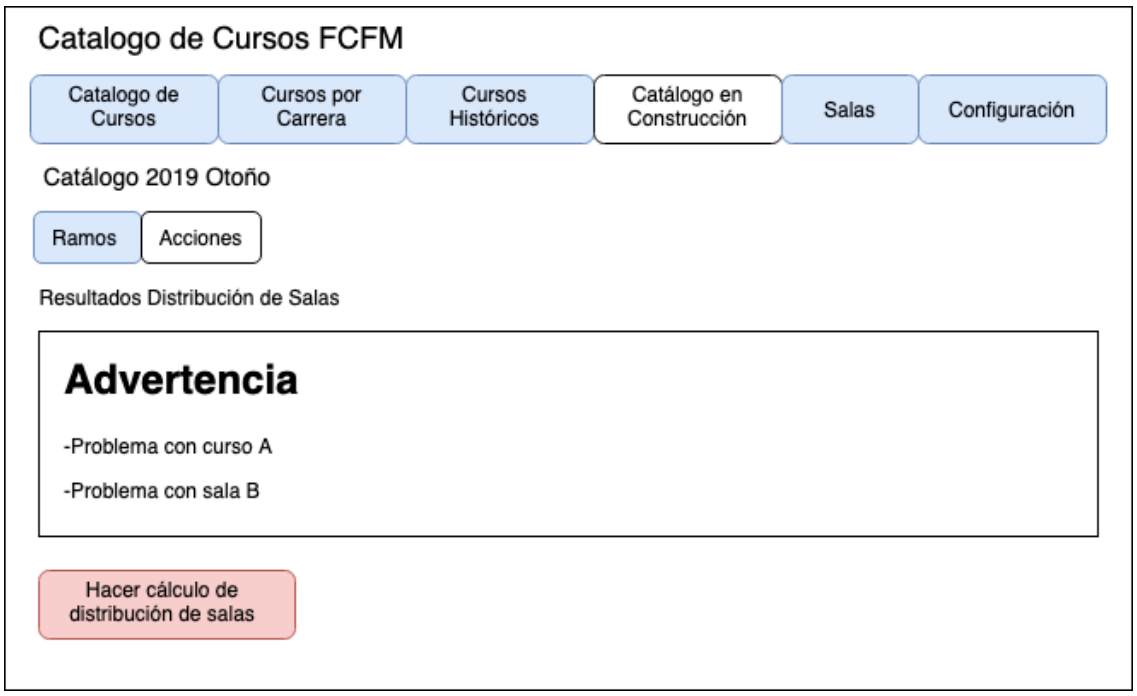


Figura 4.2: Prototipo de la interfaz de distribución de salas previo al cálculo de la solución

y definitiva, ya que en lugar de entrar a una ventana específica, ejecuta la distribución de forma inmediata. Por lo mismo, la plataforma debería tener una pestaña específica para la distribución, en que el usuario deba tener distintas etapas de confirmación antes de realizar la asignación de salas final.

Por esto se piensa en una pestaña previa a la distribución, la cual contenga una acción de asignación de salas preliminar, que luego el usuario puede revisar y, si es que está conforme con la solución, ejecute la acción de inserción final. Este diseño por etapas permite que el usuario tenga la asignación controlada y pueda organizar la información antes de tener una respuesta final sobre la distribución de salas.

Asimismo, dado que se ha detallado que la asignación es flexible a no distribuir necesariamente todos los cursos, antes de realizar dicha acción el usuario puede revisar información previa de advertencia, como si es que un horario cuenta con demasiados cursos asignados, la cual le permita prevenir un posible fallo en la respuesta. Así, por ejemplo, si el horario de 12:00 a 13:30 horas del lunes y viernes tuviera demasiados cursos, se podría prevenir el que producto de esto no todos sean asignados, permitiendo redistribuir esos cursos en mejores horarios y así evitar un problema en la asignación de salas. Esto queda ilustrado en la Figura 4.2, donde hay un texto de advertencia para el usuario previo a la posibilidad de accionar el cálculo de distribución de salas.

Igualmente, tal como se ha mencionado, un gran problema que enfrenta la SGD son los largos periodos de validación. Por lo mismo, la plataforma debería ilustrar todos los cursos que fueron asignados, en qué salas, y en qué horarios, junto a las características disponibles (como el cupo del curso, la capacidad de la sala, entre otras cosas), para que el usuario pueda juzgar correctamente la distribución y determinar si es o no satisfactoria de una manera

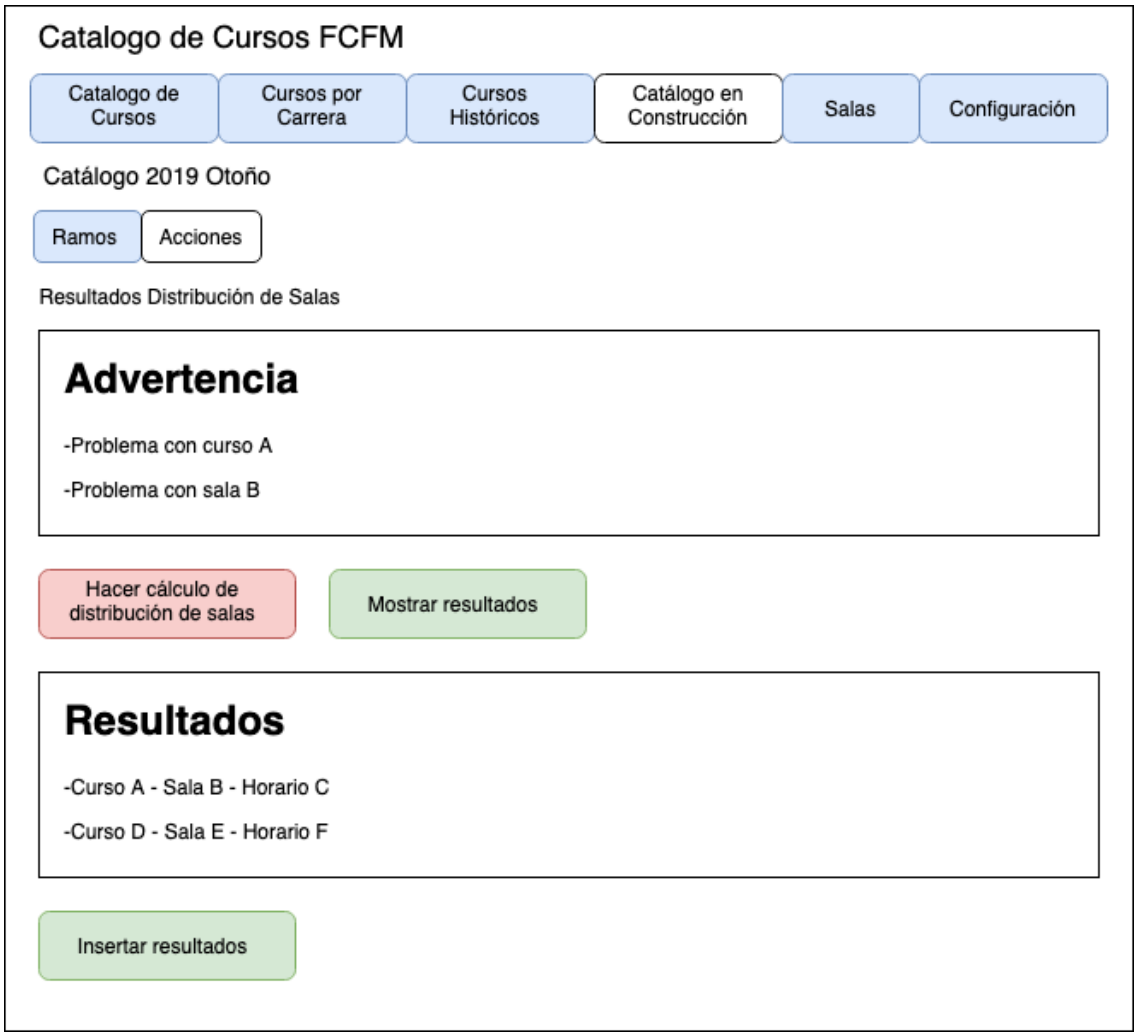


Figura 4.3: Prototipo de la interfaz de distribución de salas posterior al cálculo de la solución

global. Esto reduciría considerablemente los tiempos de validación al permitir a la persona encargada el tener acceso directo a la distribución de salas completa. Dicha información queda ilustrada en la Figura 4.3, donde se muestra la entrega de información luego de accionada la distribución de salas. Allí, se puede consultar la asignación de datos y, en caso de ser satisfactoria, se puede insertar definitivamente como respuesta a la distribución.

De esa forma queda detallada la interfaz del sistema y, con ello, el diseño general de la solución. Con esta descripción ahora se ilustrará la arquitectura diseñada para el sistema y posteriormente el cómo se implementaría este diseño en la plataforma.

4.3.4. Arquitectura

Tal como se detalló en la Sección 3.2.1, la arquitectura de Ucampus sigue el patrón de Modelo-Vista-Controlador. Por lo mismo, cada módulo desarrollado en este sistema responde a dicho estilo. De esa forma, se toma en consideración la solución previa de la distribución de salas como base.

Dado que el diseño escogido es a través de programación lineal, se debe encontrar un software que permita la modelación y la resolución del problema. En Ucampus se han desarrollado previamente modelos de optimización utilizando AMPL y solvers matemáticos. Aún así, este último ya no es opción dado el costo involucrado y la necesidad de que los solvers funcionen en un entorno de servidor web. Por otra parte, se había comenzado a evaluar el software Gurobi, el cual había tenido buena experiencia en su uso en Ucampus, además de ya tener una licencia disponible para su uso. Éste, en la experiencia de su implementación de Ucampus, mantiene la implementación del modelo de forma sencilla, cosa de ser fácil de desarrollar, y soporta múltiples lenguajes de programación. Debido a esto se decidió usarlo para abordar el diseño de la solución.

Aún así, se debía elegir un lenguaje sobre el que desarrollar el modelo en base al software Gurobi, esto debido a que ninguno de ellos es estándar dentro del desarrollo de Ucampus. Así, se eligió Python, dada la familiaridad y conocimiento del memorista sobre dicho lenguaje. No obstante, dada esta falta de estándar mencionada en Ucampus, se tuvo que idear la comunicación que debía existir entre el desarrollo de la resolución del modelo y el módulo de distribución de salas.

La Figura 4.4 ilustra la comunicación entre el modelo de optimización y el módulo de distribución de salas. Allí, los pasos 0 y 1 describen cómo el módulo de optimización recibe los datos de *salas*, *cursos* y *horarios* desde la base de datos institucional MUFASA. Esta información es procesada por el modelo y optimizada, lo que entrega un archivo con la distribución de salas. Sin embargo, tal como se mencionó previamente, estos datos deben guardarse temporalmente para luego ser validados e insertados definitivamente. Por lo mismo, se crea una tabla donde almacenar la asignación de tabla temporal `HORAS_CURSOS_TEMP`. El paso 2 ilustra la inserción del resultado en dicha tabla temporal. Luego, el paso 3 describe que el módulo de distribución de salas toma la información de la tabla temporal y, una vez validada por el usuario, el paso 4 ilustra que esta es insertada en la tabla definitiva de distribución salas `HORAS_CURSOS`.

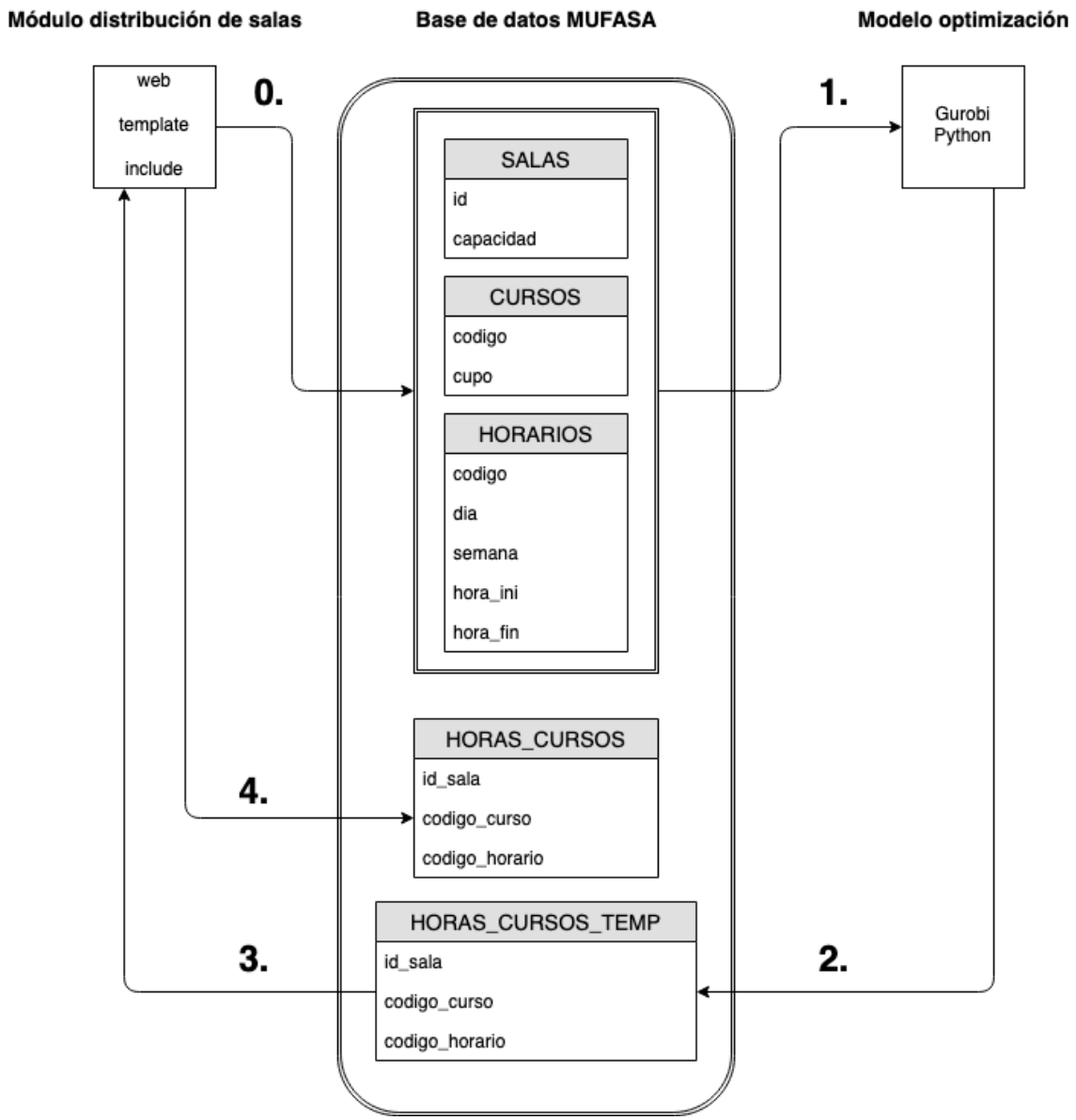


Figura 4.4: Diagrama comunicación entre el modelo de optimización Gurobi-Python y el módulo de distribución de salas modelo-vista-controlador.

Habiendo definido la estructura con la que se planea abordar la solución a desarrollar, se avanzará a describir el cómo este diseño fue implementado en el sistema.

Capítulo 5

Implementación

En este capítulo se describirá la implementación de la solución según lo diseñado en el Capítulo 4. En la Sección 5.1 se detallará la entrada y salida de información del módulo además de la definición de las variables principales; en las Secciones 5.2 y 5.3 se describirán las restricciones principales y el desarrollo de choques, respectivamente; y en la Sección 5.4 se detallará la función objetivo. Además, en la Sección 5.5 se explicará el proceso de pre-asignación de cursos en que se tuvo que remodelar la solución respecto a lo que se había planificado inicialmente. Luego, en la Sección 5.6 se profundizará en la reformulación del uso de memoria que emergió debido a los problemas del modelo inicial. Finalmente, en la Sección 5.7 se detallará el desarrollo de la interfaz y cómo se implementó la interacción con el usuario.

5.1. Entrega, definición y salidas de datos

Tomando lo diseñado como solución al problema inicial en la Sección 4.3.2, el primer desafío en su implementación fue la entrega de datos hacia el modelo. Según lo ilustrado en la Sección 4.3.4, se debe entregar la información al modelo a través de la base de datos de `MUFASA`. No obstante, dado que el modelo se implementó en Gurobi, no fue posible ejercer esta comunicación directamente con la base de datos. Por lo mismo, se pensó en hacer la entrega de datos a través de archivos JSON, los cuales son interpretados por el modelo.

En el Listado 5.1 se describe el cómo el módulo de distribución de salas le entrega los datos al modelo de optimización. Entre las líneas 3 y 13 se ilustra con pseudo-código el almacenamiento de información, a través de consultas de SQL a la base de datos de `MUFASA`, la cual es guardada en variables específicas para cada dato. Luego, si es que existen cursos que distribuir (línea 15), entonces se procede a accionar la asignación de salas. En caso contrario se avisa sobre un error para ser ilustrado al usuario a través de la interfaz (líneas 32 a 34). Para ejecutar la optimización de distribución de salas, el módulo primero define un archivo en donde guarda los datos consultados (`salas`, `cursos`, `horarios`, `horarios_cursos`, entre otras variables que se detallarán en las siguientes secciones). Esto se almacena en una ruta que luego el modelo puede interpretar. Este proceso está ilustrado por las líneas 16 a 18.

```php
<?php
funcion solucion( $semestre ) {
    // Consulta SQL que trae la información de las salas
    $capacidad = obtener_informacion_salas();

    // Consulta en SQL que trae la información de los cursos
    $cupo = obtener_informacion_cursos();

    // Consulta en SQL que trae la información de los horarios
    $horarios = obtener_informacion_horarios();

    // Consulta en SQL que trae la información de la asignación de horarios y cursos
    $horas_cursos = obtener_informacion_horas_cursos();

    if( $cupo ) {
        $ruta_archivos = InfoNucleo::getParametro( SISTEMA, 'ruta_archivos' ).'/gurobi/';
        $file = $ruta_archivos.'gurobi_data_salas.json';
        file_put_contents( $file, UTIL::json_encode( $data ) );

        $url = InfoNucleo::getParametro( SISTEMA, 'ruta_web' );
        $url .= '/batch/catalogo/salas_gurobi';
        $lines = UTIL::wget( $url, [], 420 );
        if( ! $lines ) KERNEL::error( 'Error al procesar el archivo' );

        $json = UTIL::json_decode( $lines );
        if( !json ) KERNEL::error( 'Error en los datos del optimizador o no se ha encontrado una
         ↪  solución óptima' );

        insertar_en_tabla_temporal( $json );

        $error = "";

    } else {
        $error = "Error: Aun no se han establecido los cursos";
    }

    return $error;
}
?>
```

Listado 5.1: Entrega de datos al modelo.

Luego, se realiza un proceso `batch`, que ejecuta al modelo de optimización mediante el script `salas_gurobi` (líneas 20 a 22). Una vez finalizada la optimización, la respuesta es almacenada en un archivo JSON, el cual es interpretado por el módulo (línea 25). Posteriormente el script se encarga de la inserción de esta información en una tabla temporal según lo detallado en la Sección 4.3.4.

Luego se genera una variable de decisión `asignación` (ver el Listado 5.2). Allí se recorren todos los datos de `salas`, `cursos` y `horarios`, y se asigna una variable binaria en Gurobi a la combinación de esos tres atributos. Esta será la variable a optimizar. Por lo tanto, cuando empiece el cálculo de la solución óptima, aquellas combinaciones que se consideren parte de ella tendrán un valor mayor a 0, mientras que aquellas que no, tendrán un valor de 0. Por ejemplo, si la `sala` de id 1 es asignada al `curso` de código 2 en el `horario` de código 3, entonces `asignacion[1,2,3]` será igual a 1. Mientras que si la `sala` de id 4 no es asignada al `curso` de código 5 en el `horario` 6, entonces `asignacion[4,5,6]` será 0.

```
asignacion = {}
for sala_id in salas:
    for curso_codigo in cursos:
        for horario_codigo in horarios:
            asignacion[sala_id, curso_codigo, horario_codigo] = m.addVar( vtype = GRB.BINARY )

m.update()
```

Listado 5.2: Definición de la variable de decisión `asignacion` en el modelo.

```
dif = {}
for sala_id in salas:
    for curso_codigo in cursos:
        dif[sala_id, curso_codigo] = salas[sala_id]['capacidad'] - cursos[curso_codigo]['cupo']
```

Listado 5.3: Definición de variable `dif` que representa la diferencia entre la capacidad de una sala y el cupo de un curso.

En primera instancia, se determinó que sería conveniente tener fácil acceso a la diferencia entre la capacidad de una sala y el cupo de un curso. Por esto, se generó una variable `dif` que guarde dicha información. El Listado 5.3 describe el proceso de definición de la variable, en que se parte con un diccionario `dif` (línea 1) y luego se itera sobre las `salas` y `cursos` (líneas 2 y 3) para finalmente asignarle la diferencia entre la `capacidad` y el `cupo` a la llave de la id de la `sala` y el código del `curso` en la variable `dif` definida (línea 4). Esto permitiría tener fácil acceso a estos valores, dando pie a que solo se necesite el id y código de una `sala` y `curso` para obtener la diferencia entre ambos. Así, se evita el tener que sacar estos datos desde las variables `salas` y `cursos` originales, aligerando el código.

Posteriormente, se definió una función objetivo, cuyo propósito es establecer aquella función que se desea maximizar o minimizar a través de la optimización. Ésta será detallada en la Sección 5.4. Luego, se ejecuta la optimización del modelo de datos a través del método `m.optimize()`. Hay que notar que Gurobi solo entrega la solución óptima si es que esta existe. De no existir, no genera ninguna optimización. Por esto, la función objetivo debió flexibilizarse cosa de que entregara una respuesta interpretable, ya que si no muestra ninguna solución, será imposible para el usuario saber qué es lo que falló en el cálculo.

```
out = {}
if m.status == GRB.Status.OPTIMAL:
    for sala_id in salas:
        for curso_codigo in horarios_cursos:
            for horario_codigo in horarios_cursos[curso_codigo]:
                if asignacion[sala_id, curso_codigo, horario_codigo].x > 0.01:
                    if sala_id not in out:
                        out[sala_id] = []
                    curso_horario = {
                        'sala': salas[sala_id],
                        'curso': cursos[curso_codigo],
                        'horario': horarios[horario_codigo]
                    }
                    out[sala_id].append(curso_codigo)

print json.dumps(out)
```

Listado 5.4: Código que almacena la solución en un archivo JSON.

Finalmente, se obtiene el resultado de la optimización iterando sobre los valores de la variable `asignacion`. Si es que estos son mayores a 0.01, entonces se considera que dicha distribución de `sala`, `curso` y `horario` es parte de la solución óptima. Este proceso queda ilsutrado en el Listado 5.4. Allí, se forma un diccionario `out` (línea 1) y, si es que el modelo logró optimizar los datos (línea 2), se itera por sobre todas las `salas`, `cursos` y `horarios` (líneas 3 a 5). Luego, se consulta en la variable `asignacion` si es que la combinación de `sala-curso-horario` es parte de la solución óptima (línea 6). En caso positivo, se toma la id de la `sala` como llave de `out` (línea 8) y la información de la distribución (líneas 10 a 12) es agregada a la variable como parte de la solución (línea 14). Al final, se imprime la información del archivo para que esta sea retornada al módulo de distribución de salas en la plataforma de Ucampus.

5.2. Desarrollo de restricciones principales

Una vez establecida la definición de datos, para formalizar la optimización se empezaron a sumar las restricciones. Su objetivo es limitar las posibilidades sobre las que el modelo puede moverse, ajustándose a las diversas necesidades que se requieran al respecto.

```
1  # Restriccion Sala Unica
2  for sala_id in salas:
3      for horario_codigo in horarios:
4          m.addConstr( sum(asignacion[sala_id, curso_codigo, horario_codigo] for curso_codigo in
           ↪   cursos ) <= 1 )
5
6  # Restriccion Curso Unico
7  for curso_codigo in horarios_cursos:
8      for horario_codigo in horarios_cursos[curso_codigo]:
9          m.addConstr( sum(asignacion[sala_id, curso_codigo, horario_codigo] for sala_id in salas )
           ↪   <= 1 )
10
11 # Restriccion Capacidad
12 for sala_id in salas:
13     for curso_codigo in horarios_cursos:
14         for horario_codigo in horarios_Cursos[curso_codigo]:
15             m.addConstr( dif[sala_id, curso_codigo] * asignacion[sala_id, curso_codigo,
               ↪   horario_codigo] >= 0 )
```

Listado 5.5: Definición de restricciones del modelo – Sala Única, Clase Única y Capacidad.

Se tienen tres restricciones: Sala Única, Curso Único y Capacidad, las cuales fueron detalladas en la Sección 4.2.1. Su implementación está dada por el Listado 5.5. Las líneas 2 a 4 detallan Sala Única, en que para cada iteración de `sala` y `horario` se determina que puede haber, a lo más, un `curso` asignado. Por otra parte, Curso Único se describe en las líneas 7 a 9, en que para cada iteración de `curso` y `horario` debe haber, a lo más, una `sala` asignada. Se observa que, originalmente, la restricción dictaba que en la línea 9 esta comparación fuera una igualdad (o sea, que la sumatoria fuera sí o sí igual a 1). No obstante, dada la flexibilidad descrita en la Sección 4.3.2, se determina que es posible para la restricción que ciertos cursos no sean distribuidos. Finalmente, la restricción de Capacidad es detallada entre las líneas 12 a 15, donde se describe que la diferencia entre `cupo` y `capacidad`, definidos en la variable `dif`, para todos los cursos asignados, sea mayor o igual a 0.

5.3. Desarrollo de choques

Posteriormente, según lo mencionado en la Sección 4.3.2, se define el `choque` entre dos cursos como el evento en que el horario de alguno de los dos se interpone con el otro. Ésta es una profundización de la restricción de Sala Única, por lo que su implementación implica el tener que reformular lo descrito en las líneas 2 a 4 del Listado 5.5. Este cambio está dado a que, por la forma en que los horarios son identificados en la base de datos, no es factible concluir que dos cursos no chocan entre ellos solo por tener identificaciones de `horarios` distintas. Por ejemplo, el horario que tiene como módulo los jueves de 10:15 hrs a 11:45 hrs, es considerado un horario distinto a aquel que tiene como módulos los jueves de 10:00 hrs a 12:00 hrs. Sin embargo, es evidente que ambos chocan, a pesar de tener identificaciones distintas.

```
1  <?php
2  // PRE-CALCULAR CHOQUES
3  $choques = [];
4  $choques_mejorados = [];
5  foreach( $horas_cursos as $c1 => $hs ) {
6      foreach( $hs as $h1 => $v ) {
7          $horas1 = $horarios[$h1];
8          foreach( $horas_cursos as $c2 => $hs2 ) {
9              if( $c1 >= $c2 ) continue;
10
11             foreach( $hs2 as $h2 => $v2 ) {
12                 $horas2 = $horarios[$h2];
13
14                 if( $v['semana'] && $v2['semana'] && $v['semana'] != $v2['semana'] ) {
15                     continue;
16                 } if( $horas1['dia'] != $horas2['dia'] ){
17                 continue;
18                 } if( $horas1['ini'] >= $horas2['fin'] \
19                 || $horas2['ini'] >= $horas1['fin'] ) {
20                     continue;
21                 }
22
23                 if( $h1 == $h2 ) {
24                     if( ! $choques_mejorados[$h1] ) $i++;
25                     $choques_mejorados[$h1][$c1] = $c1;
26                     $choques_mejorados[$h1][$c2] = $c2;
27                 } else {
28                     $choques[$c1][$h1][$c2][$h2] = 1;
29                     $i++;
30                 }
31             }
32         }
33     }
34 }
35 ?>
```

Listado 5.6: Precálculo de choques

Para esto primero se realizó un pre-cálculo de choques en el módulo de distribución de salas. Según la arquitectura diseñada en la Sección 4.3.4, el módulo le entrega esta información al modelo de optimización junto al resto de los datos de `cursos`, `salas` y `horarios`.

Esta acción de pre-calculo queda detallada en el Listado 5.6. Allí, se revisan todas las posibles combinaciones de `curso-horario` (líneas 3 a 8) y se comparan entre ellas para evaluar si chocan. Este proceso parte descartando todos los posibles escenarios en que evidentemente

no hay colisiones. Primero se revisa si es que las `semanas` son distintas (líneas 14 y 15); los `días` son diferentes (líneas 16 y 17); o si el `horario de inicio` de un `curso` es mayor al `horario de finalización` del otro, y viceversa (líneas 18 a 21).

Una vez descartados estos escenarios, es evidente la conclusión que los dos `cursos` no colisionan respecto a sus `horarios`. Ahora, dados problemas de memoria los cuales se profundizarán más adelante, fue necesario separar los choques en dos tipos. Por una parte, los `choques mejorados`, los cuales existen cuando el código de horario es exactamente el mismo, y los `choques` normales, que existen cuando el choque ocurre entre dos `horarios` con identificación distinta. El primer caso está descrito por las líneas 23 a 26, en que, si es que el código de `horario` es el mismo, entonces se almacena en la variable `choques_mejorados`, para cierta combinación de `curso-horario`, todos los otros `cursos` que chocan en ese mismo `horario`. El segundo escenario está detallado por las líneas 27 a 30. Allí, si los identificadores de `horario` no son los mismos, entonces se almacenará en la variable `choques` todas las combinaciones de `curso-horario` que chocan para un caso particular de `curso-horario`.

El propósito de esta diferenciación es tener que ocupar menos memoria al almacenar estas combinaciones. Esto dado que si ambos `cursos` comparten el mismo `horario`, entonces no es necesario almacenar dos veces la variables `horario`, ya que es la misma para ambas combinaciones. Asimismo, se nota en la línea 9 que solo se almacenan la mitad de los casos de combinaciones posibles. Esto debido a que al almacenar dicha mitad, ya se cuenta con la otra. Por ejemplo, si se sabe que el `curso` "MA 1001-2 Introducción al Cálculo" choca con "MA 1002-1 Introducción al Álgebra" en el `horario` del lunes de 12:00 a 13:00 horas, entonces no es necesario saber que "MA 1002-1 Introducción al Álgebra" choca con "MA 1001-2 Introducción al Cálculo" en ese mismo horario. Esta diferenciación permite ahorrar el uso de memoria, evitando el almacenamiento de información repetida y reduciendo sustancialmente el espacio.

```
1  # Restriccion Horario Unico 1
2  for h in choques_mej:
3      for sala_id in capacidad:
4          m.addConstr( sum( asignacion[sala_id, c, h] for c in choques_mej[h] ) <= 1 )
5
6  # Restriccion Horario Unico 2
7  for c_1 in choques:
8      for h_1 in choques[c_1]:
9          for c_2 in choques[c_1][h_1]:
10             for h_2 in choques[c_1][h_1][c_2]:
11                 for sala_id in capacidad:
12                     m.addConstr( asignacion[sala_id, c_1, h_1 + asginacion[sala_id, c_2, h_2] <=
                        ↪  1 )
```

Listado 5.7: Restricción de choques

Luego de definir el pre-cálculo, se pudo empezar a trabajar en el modelo de optimización en sí. Para esto se agregaron las restricciones de Horario Único 1 y Horario Único 2 (Listado 5.7), las cuales remplazaron la restricción de Sala Única descrita en el Listado 5.5, pues en la existencia de las dos anteriores, la tercera es redundante. Esto dado que al evitar que existan choques, no es necesario revisar que una sala sea única, pues por defecto, al no existir choques, una misma sala no puede ocuparse por dos cursos en un mismo horario.

Estas nuevas restricciones hacen uso de las variables de `choques` y `choques_mej` antes

definida. La restricción de Horario Único 1 conserva lo que previamente se contemplaba en la restricción de Sala Única. Una sala no puede ser asignada a dos cursos en exactamente el mismo horario. En la restricción de Horario Único 2, por otra parte, se revisan todas las combinaciones de cursos y horarios que colisionan, y el modelo verifica que no se asigne la misma sala a ambos. De esa forma, se evita que el espacio físico de una sala sea compartido por dos cursos para un periodo de tiempo definido.

5.4. Desarrollo de función objetivo

Para desarrollar la función objetivo, lo primero que se tomó en consideración es el objetivo principal de la plataforma (según lo descrito en la Sección 4.3.2), el cual es disminuir la diferencia entre el `cupo` de los `cursos` y la `capacidad` de sus `salas` respectivas. Por esto, la base de la función objetivo fue la minimización de dicha diferencia, si es que ambos fueron asignados el uno al otro.

El primer acercamiento a la función objetivo, previo a las flexibilizaciones mencionadas anteriormente, llevó a que la optimización no encontrara una solución al problema. Por lo mismo, se volvió fundamental el realizar las flexibilizaciones mencionadas (como el que ya no se distribuyan todos los cursos necesariamente) y, asimismo, considerar dentro de la función objetivo la necesidad de asignar la mayor cantidad de cursos posibles.

```
m.setObjective( (
    # Maximizar asignacion
    -1 * sum (
        asignacion[sala_id, curso_codigo, horario_codigo]
            for curso_codigo in horarios_cursos
                for horario_codigo in horarios_cursos[curso_codigo]
                    for sala_id in capacidad
                        if ( sala_id, curso_codigo, horario_codigo) in asignacion
    )
    # Minimizar diferencia de espacio
    + sum(
        ( int( capacidad[sala_id] ) - int( cupo[curso_codigo] ) ) / 60.0
        * asignacion[sala_id, curso_codigo, horario_codigo]
        for curso_codigo in horarios_cursos
            for horario_codigo in horarios_cursos[curso_codigo]
                for sala_id in capacidad
                    if ( sala_id, curso_codigo, horario_codigo ) in asignacion
    )
    # Maximizar preferencia de cursos
    + -1 * sum(
        preferencia[curso_codigo, sala_id] / 50.0
        * asignacion[sala_id, curso_codigo, horario_codigo]
            for curso_codigo in horarios_cursos
                for horario_codigo in horarios_cursos[curso_codigo]
                    for sala_id in capacidad
                        if ( sala_id, curso_codigo, horario_codigo ) in asignacion
                        and ( curso_codigo, sala_id ) in preferencia
    ) ), GRB.MINIMIZE )
```

Listado 5.8: Función objetivo con flexibilidad de asignación

Por esto, una vez eliminada la restricción de Curso Único se pudo avanzar a considerar como los dos objetivos principales de la función a optimizar el maximizar la asignación y

minimizar la diferencia de espacio. Ambos fueron detallados en la función objetivo según el Listado 5.8. Para esto se consideró lo diseñado teóricamente en la Sección 4.3.2 según la Ecuación 4.8, en que se busca asignar la mayor cantidad de cursos posibles (líneas 3 a 9). Dado que la función objetivo busca minimizar, la sumatoria detallada es multiplicada por -1, para que así sea maximizada.

Hay que evidenciar que dado este acercamiento es que puede darse el escenario en que algunos cursos no se asignen. Por ejemplo, si hubiera muchas limitantes, la solución podría incluir que unos 10 cursos no se asignen mediante la optimización, los cuales deberían distribuirse manualmente. No obstante, en la siguiente sección se detallará una decisión sobre la plataforma la cual minimizó estos casos.

Una vez implementada esta flexibilidad sobre la función objetivo, la respuesta del sistema mejoró considerablemente llegando a una solución interpretable. En las primeras pruebas, realizadas con información del semestre de primavera 2019, se dejaron no asignados apenas 30 cursos en contraste a los más de 1.000 que sí lo fueron, junto a sus respectivos `horarios` y `salas`. Este número incluso mejoraría después de la pre-asignación de `cursos` descrita en la siguiente sección, en que se logró que la distribución asignara todos los `cursos`, no dejando ninguno fuera de la asignación.

Por otra parte, las líneas 11 a 18 del Listado 5.8 detallan la parte de la función objetivo que minimiza la diferencia de espacio. Esto toma lo diseñado la Sección 4.3.2 y la expresión 4.7, donde se toma la diferencia del `cupo` de los `cursos` y la `capacidad` de las `salas` asignadas, y se busca minimizar ese resultado. Dado que el valor de estos dos segmentos es sustancialmente distinto, es necesario normalizar los resultados para que sean equiparables.

Asimismo, cada parte de la función objetivo tuvo que ser normalizada para tener un peso neutro en comparación a la otra. El segmento de maximización de asignación se dejó intacto dado que cada asignación tiene un valor unitario. Por otra parte, en la parte de minimización de diferencia de espacio, dicha diferencia fue dividida por 60, debido a que este es el promedio de `capacidad` de las `salas`. Finalmente, en el segmento de minimización de preferencia de cursos la `preferencia` fue normalizada por 50 dado que este es el valor neutro de dicho parámetro.

5.5. Pre-asignación de cursos

Con la primera solución de base, se notó que había una serie de `cursos` cuya asignación dentro de la distribución parecía redundante. Por ejemplo, un `curso` como "Natación" siempre se da en la "Piscina", por lo que no parece necesario tener que incluirlo dentro de la distribución de salas, puesto que cualquier otra `sala` no tiene sentido como distribución para dicho `curso`. Además, se tienen `cursos` con distribuciones históricas, como "Ajedrez", el cual siempre se ha cursado en la "Sala de Juegos". Esto llevó a considerar que hay una serie de cursos que no es necesario contemplar en la distribución, puesto que ya tienen una sala por defecto. Esto no se examinó en el diseño ya que durante las distintas indagaciones sobre el contexto del problema, tanto al leer al respecto, como al conversar con la SGD, no se había

enfatizado en esto. Recién se pudo tomar en consideración al acercarse a la implementación y hacer las primeras pruebas, dando cuenta de que el sistema estaba distribuyendo dichos cursos y notando en que no deberían ser parte específica del problema.

Lamentablemente, indagando en la información disponible en la base de datos, no hay datos que permitan inferir que entre un `curso` y una `sala` haya una relación histórica. La solución actual de Ucampus responde a este problema usando la información histórica directamente. No obstante, en el escenario desarrollado por la plataforma, no es información suficiente el saber qué `sala` tuvo cierto `curso` el semestre ante-pasado, puesto que lo que se quiere conocer es un patrón de salas más que una instancia en el tiempo.

El primer acercamiento a resolver este problema fue observar el como las `salas` se asignaban a `cursos` específicos que no tenían un valor de `cupo` en la base de datos. Inicialmente se pensó en usar este dato para generar una tabla que pre-asignara todas las combinaciones de `cursos` y `salas` que sí o sí debían juntarse. Sin embargo, esto llevó a problemas como: `salas` que están en mantención y que tienen `cupo` 0 para evitar que sean asignadas; cursos que son distribuidos momentáneamente en salas especiales, pero que no necesariamente se dictan allí todos los semestres; `ramos` con ciertas secciones que se distribuyen en `salas` con `cupo` 0, pero que no representan como debe trabajarse con el `ramo` en conjunto; entre otras cosas.

Es por esto, que se decidió conversar la situación con María José Contreras, Subdirectora de Gestión Docente, para evaluar una solución al problema. Desde allí, se determinó que todos los cursos que necesiten ser asignados sí o sí con ciertas salas, son los primeros en distribuirse. Esto ya que desde los departamentos se dialogan estos casos especiales, no haciéndolos parte de la distribución oficial. Asimismo, esta respuesta contempla los casos en que profesores poseen necesidades o preferencias específicas, o en que ciertos alumnos necesitan salas particulares por alguna discapacidad física (por ejemplo, necesitar acceso de silla de ruedas). Todos estos casos quedaron contemplados dentro de una asignación previa a la distribución oficial que otorga la plataforma, por lo que se podrían considerar como resueltos.

De esa forma, en la búsqueda de los datos desde la base de datos se definió una variable `asignados`, la cual almacena las `salas` distribuidas para todas las combinaciones de `curso-horario` que ya han sido asignadas. Según la arquitectura detallada en la Sección 4.3.4, ahora en la comunicación de datos desde el módulo de distribución de salas, se entrega la información de `asignados`, junto al resto de los datos de `salas`, `cursos`, `horarios` y `choques`.

Es importante notar que esto debió realizarse así para que el modelo siguiera considerando que hay ciertas salas que ya están ocupadas. Si en lugar de esta solución se hubiera abordado una respuesta en que simplemente se traen todos los cursos desde la base de datos menos los asignados, entonces el modelo podría querer asignar salas en horarios en que ya estaban distribuidos. En su lugar, al abordar el escenario de la forma mencionada, se asegura que se contemplen los cursos ya asignados, sin hacerlos parte de la distribución.

```
1  # Cursos ya asignados
2  for curso_codigo in asignados:
3      for horario_codigo in asignados[curso_codigo]:
4          sala_id = asignados[curso_codigo][horario_codigo]['sala_id']
5          if not (sala_id, curso_codigo, horario_codigo) in asignacion:
6              asignacion[sala_id, curso_codigo, horario_codigo] = m.addVar( vtype = GRB.BINARY )
7          asignacion[sala_id, curso_codigo, horario_codigo].ub = 1
8          asignacion[sala_id, curso_codigo, horario_codigo].lb = 1
9
10 m.update()
```

Listado 5.9: Restricción a cursos ya asignados

Así, luego de realizar la primera asignación en el modelo, y antes de formular las restricciones, se desarrolló una restricción que limita los cursos que ya han sido asignados, la cual queda detallada en el Listado 5.9. De esa forma, se consideraron todas las combinaciones de `curso-horario` y su `sala` asignada (líneas 2 a 4), fijando dichas distribuciones en la variable `asignacion`, cosa de imposibilitar que dichas `salas`, `cursos` u `horarios` sean distribuidos en cualquier otra combinación.

5.6. Reformulación de uso de memoria

El acercamiento a la solución resultó ser exitoso para pequeñas muestras de información. Por ejemplo, distribuir 20 cursos en 50 salas. No obstante, una vez estas muestras fueron creciendo empezaron a surgir problemas con la memoria del ambiente de desarrollo de Ucampus. Alrededor de los 100 cursos el módulo consumió un segmento importante de la memoria disponible en la plataforma, paralizando los procesos paralelos y no llegando a una solución concreta. Se estima que la solución tomaba más del 25 % de la memoria RAM del ambiente de desarrollo, lo cual contempla alrededor de 2 GB. Para más de 1 GB de memoria ocupada el ambiente se paraliza por lo que se volvió prioritario el tener que disminuir dicho consumo, para evitar paralizar todos los procesos ejecutados en el ambiente (entre los que se encuentra la distribución de salas).

Es por esto que fue necesario adentrarse en el código para vislumbrar qué partes del módulo de optimización estaba consumiendo la memoria. En este punto de la solución aún no se tenía certeza sobre su eficacia debido a que la paralización de la plataforma no permitía llegar a una solución concreta para datos reales. Es por esto que fue fundamental responder a los problemas de memoria para, no solo optimizar la respuesta de la solución, sino que llegar a entregar una en primer lugar.

La primera reformulación que se realizó fue localizar todos los lugares en el código en que se estuviera utilizando memoria de forma innecesaria. En primera instancia, se notó que la definición de una variable para almacenar la diferencia entre la capacidad de una sala y el cupo de un curso (Listado 5.3) no ofrecía mayor utilidad, más que hacer más robusto al código. Sin embargo, en conceptos de espacio, ocupaba un segmento de información innecesario, por lo que se eliminó su presencia en pos de simplemente usar los datos sobre los que se tiene acceso, para generar la diferencia simplemente restándole la capacidad de una sala al cupo de un curso.

Además, se observó un gran problema de memoria en la manera en que se había formulado el pre-cálculo de `choques`. Tal como se mencionó previamente, una solución original simplemente revisaba los `choques` entre todos los `cursos` sin hacer una diferenciación entre aquellos que chocaran en el mismo `horario` (dos `cursos` en el `horario` de 10:15 a 11:45) y aquellos que lo hacían entre `horarios` distintos (un `curso` en el `horario` de 10:00 a 12:00 horas y otro en el de 10:15 a 11:45 horas). Esto llevaba a repeticiones de información que fácilmente podían ser almacenadas de forma continua. Por ejemplo, no es necesario saber que "MA1102-1 Álgebra Lineal" de 8:30 a 10:00 horas, choca con "MA1101-2 Cálculo Diferencial e Integral" de 8:30 a 10:00 horas. Basta con saber que ambos cursos chocan de 8:30 a 10:00 horas.

Junto a esto, la búsqueda de `choques` duplicaba la información. Inicialmente, se revisaba para cada `curso` y `horario`, todos los `cursos` y `horarios` que chocaran con este. La resolución de este problema fue detallada en la Sección 5.3.

Estos cambios disminuyeron considerablemente el uso de la memoria a alrededor de un 4 % de RAM (320 MB aproximadamente), permitiendo que el código llegara a los límites aceptables en que se pudo procesar el modelo de optimización y entregar una respuesta interpretable. Así, al menos se pudo asegurar que el proceso llegara a término con la optimización sin consumir grandes cantidades de memoria disponible, y se pudo trabajar con todos los datos necesarios.

5.7. Interfaz gráfica

Al empezar a trabajar con la interfaz, primero se notó la falta de confirmación y retroalimentación por sobre el primer cálculo. En la plataforma original, acceder a la distribución de salas implica automáticamente realizar una distribución. Además, dicha distribución se agrega automáticamente a la base de datos, no existiendo una capa media que permita advertir sobre lo que se va a insertar.

Por esto, se definieron dos capas medias de la plataforma previas a la inserción definitiva. Primero una interfaz previa al cálculo inicial, a la que se accede a través de la plataforma de distribución de salas. Esto da cierta seguridad al no tener que hacer el cálculo inmediatamente al ingresar en la plataforma.

Esta interfaz ilustra datos sobre la cantidad de cursos que serán dictados durante el semestre, y la cantidad de salas que hay disponibles (salas con cupo mayor a 0 y que no son de uso exclusivo de los departamentos, o sea que se pueden asignar libremente). Asimismo, muestra horarios potencialmente problemáticos. Considerando que en este punto, los cursos ya se encuentran asignados a sus horarios, se realiza un cálculo de cuantos horarios hay en cada curso, para ilustrar en la tabla aquellos que bordeen peligrósamente la cantidad de salas disponibles. Esto permite dar advertencia al usuario sobre posible choques, permitiendo generar cambios en la distribución de cursos y horarios, para generar una mejor asignación de salas (ver Figura 5.1). Al final de la interfaz, hay un botón de "Hacer el cálculo". Al ejecutarlo, la plataforma empieza a calcular la solución óptima. Ésta se ejecuta en la capa

Figura 5.1: Interfaz de acción de distribución de salas previo al cálculo de la optimización

lógica según el modelo ilustrado en la Sección 4.3.4.

Es necesario notar que esto contempla que ciertos cursos se dictan en salas exclusivas, como se detalló en el capítulo pasado. Por lo tanto, esta advertencia no es necesariamente invalidante para el cálculo de datos. Más bien sirve de advertencia en caso de que al realizar el cálculo se encuentren muchos cursos sin asignar.

Una vez se llega a una solución, esta se ilustra en la interfaz dando detalle del nombre del ramo, la sección, el horario, el nombre de la sala y, a modo de retroalimentación, la capacidad de la sala asignada y el cupo del curso (Figura 5.2). Esta información se almacena en una tabla temporal de pre-cálculo en la base de datos. Por lo tanto, toda solución aún no es definitiva. Esto permite que la persona encargada pueda rehacer el cálculo variando los datos según estime conveniente, a fin de generar la mejor solución posible.

Asimismo, los datos se dividen entre Cursos No Asignados y Cursos Asignados. Esta primera tabla genera retroalimentación sobre qué tan exitosa fue la solución, ilustrando cuantos cursos fueron dejados fuera, y dando la posibilidad para que la persona encargada pueda asignarlos manualmente sin interferir en el cálculo óptimo encontrado.

Finalmente, si se aprueba el cálculo realizado, al final hay un botón de "Insertar en la base de datos". El cual, al ejecutarse, mueve los datos de la tabla temporal de pre-cálculo a la base de datos de cursos y salas. De esa forma, se considera como verdaderamente asignada la distribución.

Dia	Horario	Cantidad de Cursos	Problema
Lunes	10:15 - 11:45	89	Muchos cursos en este horario
Lunes	12:00 - 13:30	96	Muchos cursos en este horario
Martes	10:15 - 11:45	102	Muchos cursos en este horario
Martes	12:00 - 13:30	96	Muchos cursos en este horario
Martes	14:30 - 16:00	82	Muchos cursos en este horario
Miercoles	10:15 - 11:45	78	Muchos cursos en este horario
Jueves	10:15 - 11:45	107	Muchos cursos en este horario
Jueves	12:00 - 13:30	104	Muchos cursos en este horario
Viernes	10:15 - 11:45	86	Muchos cursos en este horario
Viernes	12:00 - 13:30	78	Muchos cursos en este horario

Hacer el calculo | Ocultar resultados

Horarios Asignados:

Excel ODS

Codigo	Curso	Seccion	Tipo de Clase	Dia	Horario	Sala	Cupo Curso	Capacidad Sala	Diferencia	Requiere Equipo
EP5206	Academic Writing 2	1	Cátedra	2	15:00-15:45	E215	1	32	31	No
EP5206	Academic Writing 2	1	Cátedra	4	15:00-15:45	F11	1	40	39	No
DR100A	Acondicionamiento Físico Básico	1	Cátedra	2	10:00-10:45	B210	25	26	1	No
DR100A	Acondicionamiento Físico Básico	1	Cátedra	4	10:00-10:45	B210	25	26	1	No
DR100A	Acondicionamiento Físico Básico	2	Cátedra	2	11:00-11:45	Q21	25	34	9	No
DR100A	Acondicionamiento Físico Básico	2	Cátedra	4	11:00-11:45	Q23	25	40	15	No
CI5115	Agua en Minería	1	Cátedra	5	12:00-12:45	B109	19	24	5	No
DR230A	Aikido I	1	Cátedra	1	12:00-12:45	B109	20	24	4	No
DR230A	Aikido I	1	Cátedra	3	12:00-12:45	S25	20	55	35	No
DR230B	Aikido II	1	Cátedra	1	12:00-12:45	B206	10	26	16	No
DR230B	Aikido II	1	Cátedra	3	12:00-12:45	B214	10	22	12	No
MA1102	Álgebra Lineal	1	Cátedra	2	10:00-10:45	F20	95	96	1	No
MA1102	Álgebra Lineal	1	Cátedra	4	10:00-10:45	F21	95	96	1	No
MA1102	Álgebra Lineal	1	Auxiliar	5	14:00-14:45	QO	95	95	0	No

Figura 5.2: Interfaz de acción de distribución de salas posterior al cálculo de la optimización

Capítulo 6

Validación

En este capítulo se detallará la validación de la solución desarrollada en esta memoria. Para esto se tomarán en consideración cuatro aristas de la plataforma a probar: distribución total de cursos; nivel de ocupación de salas; cercanía de departamentos y edificios en la distribución de cursos; e interacción con el usuario e interfaz del sistema. Para cada uno de estos puntos se detallará la metodología o protocolo a seguir, se describirán los datos recolectados, y se hará un análisis sobre los resultados.

6.1. Distribución total de cursos

La primera prueba fue la de distribución total de cursos. Ésta consistió en validar que se asignara la mayor cantidad de cursos posibles en un semestre. Su importancia recae en que, debido a la flexibilidad dada al modelo, el resultado puede excluir ciertos cursos de la distribución total.

En este escenario, los datos a revisar fueron la cantidad total de cursos distribuidos para el semestre de primavera 2019, y la información de entrada fue: `cursos`, `salas`, `horarios` y asignación de `cursos-horarios`. Para motivos de prueba, no se consideraron los cursos pre-asignados, por lo que el sistema intentó distribuir todos los `cursos` sin distinción. No obstante, hay que tener en cuenta que los `cursos` pre-asignados representan alrededor de 300 casos de entre un total de 1.999, o sea, aproximadamente un 15 % de los datos. Por lo mismo, se puede considerar que, en el peor de los casos, una distribución debería dejar fuera, a lo más, 300 cursos, ya que representan los potencialmente pre-asignados.

Al obtener los resultados, en primera instancia la solución previa de Ucampus no dejó `cursos` sin distribuir. Sin embargo, si esta no encuentra una respuesta adecuada, no entrega una asignación de salas, y tampoco indica cuáles fueron las asignaciones problemáticas que entorpecieron a la heurística. Por otra parte, la solución desarrollada en esta memoria asignó 1.841 `cursos`, dejando 155 sin asignar. O sea, el sistema logró distribuir exitosamente el 92 % de los `cursos` disponibles para el semestre de primavera 2019.

Al analizar los resultados es importante destacar que, según lo descrito en la Sección 5.5, un escenario real tomaría en consideración la existencia de los `cursos` pre-asignados. Por lo mismo, se puede concluir que el resultado es favorable ya que los 155 casos no asignados representan poco más de la mitad del total de 300 `cursos` que no estarían distribuidos previamente, un 15 % del total según lo mencionado antes. Asimismo, al ilustrar al usuario la cantidad de `cursos` que no son distribuidos, la plataforma entrega una ventaja respecto a la solución previa, debido a que, sabiendo qué `cursos` no fueron asignados, se pueden modificar los datos para volver a computar la asignación. De esa forma, la distribución puede ejecutarse las veces que se deseen hasta obtener una solución satisfactoria, caso en que finalmente se puede insertar en las bases de datos como una respuesta definitiva.

6.2. Nivel de ocupación de salas

Luego, se validó el nivel de ocupación de los `cursos` distribuidos, comparando el resultado entre la solución previa y la desarrollada en esta memoria. Para esto también se usaron datos de `cursos` y `salas` del semestre de primavera 2019. La maximización del nivel de ocupación fue uno de los principales puntos tomados en cuenta al modelar la solución, dada la prioridad mencionada por la SGD en la Sección 3.3.1. Así, la validación consistió en tomar los `cursos` en conjunto a sus `salas` y comparar la diferencia promedio entre el `cupo` y la `capacidad`, respectivamente, entre los resultados obtenidos por la solución previa y la desarrollada. En el caso de la solución previa, la cantidad de `cursos` y `salas` de entrada al modelo fueron 569 y 85, respectivamente. Para los resultados de la solución implementada, estos datos de entrada fueron 775 y 83. Esta diferencia se debe a que se dio como condición que el `cupo` del `curso` fuera menor a la `capacidad` de la `sala`, ya que de otra forma al realizar las consultas en las bases de datos se consideraban casos como, por ejemplo, un `curso` con `cupo` 50 asignado a una `sala` con `capacidad` 0 o 1, lo que no refleja el verdadero nivel de ocupación. Es por esto que la asignación generada por la solución previa tiene 569 `cursos` de entrada, mientras que el sistema implementado recibe 775.

Después, se procedió a comparar el nivel de ocupación promedio obtenido por ambas soluciones. Dado que los resultados se almacenan en la base de datos de la Universidad, se realizaron consultas de SQL sobre dicha base de datos para evaluar el nivel de ocupación promedio de ambas respuesta. En este caso, la solución desarrollada fue almacenada en la tabla de los resultados preliminares (`HORAS_SALAS_PRE`), mientras que la solución previa se mantuvo en la tabla de resultados definitivos (`HORAS_SALAS`). Por lo mismo, se observa que las consultas para obtener el nivel de ocupación de ambos resultados son análogas, variando solo en los nombres de las tablas y sus atributos. Por ello, éstas se describirán según lo ilustrado en el Listado 6.1. En la línea 2 de la consulta se cruzan las tablas `HORAS_SALAS`, `SALAS` y `CURSOS`, dado que para realizar la validación se necesita la información de la asignación de `salas`, las `salas` y los `cursos`, respectivamente. Para que las tablas sean consistentes en su información, en las líneas 3 y 4 se asegura que el código del curso de `HORAS_SALAS` (`HOR_S_CURS_CODIGO`) sea igual al código del curso de `CURSOS` (`CURS_CURSO`); y se confirma que la id de la sala de `HORAS_SALAS` (`HOR_S_SAL_ID`) sea igual al de `SALAS` (`SAL_ID`). Luego, en la línea 5 se revisa que la información del semestre en que se está dictando el curso (`CURS_C_SEM`) sea del semestre primavera 2019 (20192). Después, en la línea 6 se consideran solo los casos de cursos que se

```
1  SELECT AVG(SAL_CAPACIDAD - CURS_CUPO)
2  FROM HORAS_SALAS, SALAS, CURSOS
3  WHERE HOR_S_CURS_CODIGO = CURS_CODIGO
4      AND HOR_S_SAL_ID = SAL_ID
5      AND CURS_C_SEM = 20192
6      AND HOR_S_H_C_SEMANA = 0
7      AND SAL_CAPACIDAD >= CURS_CUPO
```

Listado 6.1: Consulta para medir el nivel de ocupación de salas

dicten transversalmente durante el semestre, para así no tomar en cuenta escenarios como los controles y exámenes, que existen como asignaciones, pero al salir de los límites del proyecto, no deben considerarse. Posteriormente, en la línea 7 se verifica que la `capacidad` de las `salas` sea mayor al `cupo` de los `cursos`. La respuesta de esta consulta queda ilustrada en la línea 1, y resulta en el promedio de la diferencia entre la `capacidad` de las `salas` (`SAL_CAPACIDAD`) y el `cupo` de los `cursos` (`CURS_CUPO`).

Al ejecutar ambas consultas, la solución previa entregó un 67,8 % de nivel de ocupación. O sea, que para cada `curso` asignado a una `sala`, el `cupo` del primero ocupó un 67,8 % (40,7 de 60) del espacio disponible. Mientras, la solución desarrollada elevó este resultado promedio a un 85,3 % (51,2 de 60). Asimismo, al evaluar los mínimos en ambas asignaciones la diferencia mínima entre `capacidad` y `cupo` fue nula. Por otra parte, al comparar los máximos la solución previa entregó una diferencia de 89, mientras que la diferencia máxima en la solución desarrollada fue 53, lo que representa una mejora en ocupación. Finalmente, al comparar la desviación estándar de ambas soluciones, la respuesta previa dio un valor de 17,8 de variación, mientras que la desviación de la respuesta desarrollada fue de 10,2. Esto evidencia la mejoría de la nueva solución al entregar mejores resultados según las herramientas estadísticas a disposición.

De esa forma, se evidencia un claro aumento en el nivel de ocupación entregado por la solución desarrollada en comparación a la previa y, por lo tanto, un más eficiente uso del espacio respecto a lo que se tenía previamente. Además, mejorando la eficiencia de los recursos físicos en un 17,5 %, se tendrá una liberación de espacio el cual puede ser ocupado posteriormente, lo que se enlaza con la validación previa de distribución total de `cursos`.

6.3. Cercanía entre departamentos y edificios

La tercera validación fue de cercanía entre los departamentos de los `cursos` y los edificios de sus `salas` asignadas. Su importancia recae en cómo la SGD percibe la satisfacción de los docentes y estudiantes, según lo detallado en la Sección 3.3.1. Para esto se consideró la tabla de preferencias descrita en la Sección 4.3.2 y en la Figura 4.1, y se midió el promedio de preferencia asignado a cada `curso` por su `sala` distribuida. Así, si dos `cursos` de Ingeniería Civil Industrial son asignados en el Edificio Escuela y Beauchef 851, expresarán una preferencia de 50 y 100, respectivamente, y un promedio de 75 entre ambas.

A fin de obtener el promedio de preferencia asignado para cada asignación de `salas`, se realizaron dos consultas, según lo almacenado en las bases de datos. Al igual que la consulta

```
1  SELECT AVG(PREF_VALOR)
2  FROM HORAS_SALAS, PREFERENCIAS, SALAS, CURSOS, RAMOS
3  WHERE HOR_S_CURS_CODIGO = CURS_CODIGO
4      AND HOR_S_SAL_ID = SAL_ID
5      AND CURS_C_RAMO = RAMO_C_RAMO
6      AND PREF_INST_CODIGO = RAMO_INST_CODIGO
7      AND PREF_EDIFICIO = SAL_EDIFICIO
8      AND CURS_C_SEM = 20192
9      AND HOR_S_P_H_C_SEMANA = 0
```

Listado 6.2: Consulta para medir la preferencia de los cursos

anterior, aquella ilustrada en el Listado 6.2 es un ejemplo análogo para ambas consultas realizadas, en que solo varía el nombre de la tabla `HORAS_SALAS` y los atributos asociados. En este caso, en la línea 2 se cruzan las tablas `HORAS_SALAS`, `PREFERENCIAS`, `SALAS`, `CURSOS`, `RAMOS`, para así obtener la información de la asignación de `salas`, la preferencia de departamentos sobre edificios, las `salas`, los `cursos` y los `ramos`. Es necesario notar que la diferencia entre estas últimas dos tablas es que un curso es la instancia en el tiempo de un `ramo`, por lo mismo, mientras el `curso` almacena información tal como el semestre en que fue dictado, el `ramo` hace el enlace con el departamento al que el `curso` pertenece. Luego, entre las líneas 3 y 5 se verifica la información entre tablas para asegurar la consistencia de la información. Después, se obtiene el valor de preferencia que cierto departamento (`PREF_INST_CODIGO`) tiene sobre cierto edificio (`PREF_EDIFICIO`). Posteriormente, en la línea 8 y 9 se revisa que los datos sean del semestre primavera 2019, y que estos sean de cursos que se dictan durante todo el semestre.

Al medir el resultado para la solución previa, la `preferencia` calculada se expresó en un puntaje promedio de 63. Mientras, la nueva solución tuvo un resultado promedio de 40.

De esa manera, se implica un descenso en este nivel respecto a la nueva solución desarrollada, por lo que se podría considerar que la respuesta es menos buena en contraste a lo que se obtenía previamente. Esta declinación puede deberse a dos motivos principales: por un lado la tabla de preferencias descrita en la Figura 4.1 y la Sección 4.3.2, la cual podría redefinirse para expresar mejor las necesidades de la Facultad. Por otra parte está el peso asignado a cada segmento de la función objetivo implementada en la Sección 5.4. Al momento de desarrollar la función objetivo se normalizaron las tres partes (maximización de distribución total, minimización de diferencia y maximización de preferencia), por lo que una forma de combatir este puntaje sería dándole más peso al segmento relativo a la `preferencia`. No obstante, darle más importancia a dicha parte de la función objetivo afectaría los dos resultados expresados previamente (distribución total de cursos y maximización de nivel de ocupación), por lo que la forma de encontrar el equilibrio que mejor represente los intereses de la Facultad sería iterando repetidas veces hasta encontrar los pesos adecuados para asignar a cada parte de la función objetivo.

6.4. Interfaz gráfica

Finalmente, se validó la interfaz de la plataforma web de distribución de salas. Esta validación es fundamental, ya que la aprobación del usuario es primordial para el correcto enfoque

de la plataforma. Independiente de los resultados, si la interfaz no hubiera sido aprobada, el sistema no podría haberse considerado funcional. Para esto, fue necesario evaluar en conjunto a los usuarios del sistema el que la navegación e interacción fuera lo esperado. En este caso, se validó con María José Contreras, subdirectora de la SGD quien, al representar al usuario que navegaría por el sistema, permitiría determinar si los cambios fueron válidos, y considerados una mejora respecto a lo existente previamente.

En esa arista, la Subdirectora de la SGD aprobó las modificaciones, considerándolo una mejora al sistema, siendo más cómoda de navegar, y generando mayor retroalimentación sobre la información de la plataforma, lo que se espera vuelva más eficiente los tiempos de validación de los resultados en la práctica. Ahora, en lugar de tener que revisar manualmente cada curso en búsqueda por validar cada asignación específica distribuida por el sistema, el usuario puede revisar la asignación completa en conjunto, lo que se espera afecte considerablemente los tiempos de validación.

Así fue como se concluyó que la plataforma y solución desarrollada cumplían los requerimiento que se habían contemplado inicialmente. Obteniendo no solo una mejoría a partir de datos concretos, como la asignación total y el nivel de ocupación, sino que además siendo validada por la principal administradora del sistema.

Capítulo 7

Conclusión

Recapitulando, se desarrolló una plataforma de optimización que buscaba mejorar el uso del espacio físico de la facultad, respecto a la distribución de salas y cursos, junto a la comunicación entre la plataforma y el usuario. Esto llevó a mejorar la interfaz actual, cosa de entregar mejor retroalimentación, y reformular la solución propuesta, buscando dar con mejores resultados.

Usando programación lineal se estableció una base teórica que fue implementada a través del solver Gurobi con lenguaje Python. La correcta comunicación entre la información de la base de datos, la capa lógica y los datos entregados al usuario, permitieron mejorar la comodidad del administrador y trabajar con una solución optimizada, además de agregar etapas de pre-cálculo previo a la inserción final.

En general se cumplieron los objetivos planteados inicialmente. Se arribó a una solución optimizada que entrega mejores resultados generales a los de la solución actual, según las validaciones realizadas. Asimismo, la plataforma fue pulida para comodidad del usuario y se desarrolló un modelo de optimización que entrega una respuesta concreta al problema inicial.

En un principio se deseó agregar otras variables en la distribución, como el uso de equipos. No obstante, las tablas de la base de datos no contaban con la información suficiente para poder generar una verdadera relación entre requerimiento físico de cierto curso y capacidad de la sala de brindar equipamiento.

Asimismo, consideraciones como la preferencia de los profesores sobre ciertas salas, o que existan alumnos con restricciones físicas que necesiten salas específicas, son aristas que no pudieron contemplarse al no ser datos duros sobre los que trabajar. Finalmente dichas variables pudieron acomodarse al contemplar que ciertos cursos se asignan previo a la distribución de la solución. De esta forma, a pesar de no ser contemplados directamente en el rango de la solución, no se dejaron fuera completamente.

Los resultados fueron generalmente satisfactorios. Por una parte, porque según las pruebas realizadas, las respuestas entregadas fueron mayoritariamente mejores a las que había previamente, expresando una brecha menor entre la capacidad de las salas y el cupo de sus cursos

asignados. Mientras que además, el sistema fue validado con usuarios reales, quienes se vieron conformes con la nueva plataforma, aprobando el trabajo hecho y valorando la actualización del sistema. Dado que la única validación que entregó peores resultados fue la relativa a las preferencias de los cursos sobre ciertos edificios dados los departamento a los que pertenecen, es evidente que la plataforma podría mejorarse aún más, no obstante, es materia sobre la que se puede seguir profundizando en futuras iteraciones de la solución. Según los requisitos que se plantearon inicialmente, la solución puede considerarse como satisfactoria.

Esta solución es importante ya que genera un impacto directo en el día a día de los estudiantes y equipo docente de la facultad. Una mejor distribución es de relevancia directa en el correcto desarrollo de las clases. Igualmente, para la facultad es importante hacer un buen uso de su área física, no sobre o infra-dimensionado el espacio ocupado, y permitiendo tener más salas disponibles para uso futuro. En ese mismo sentido, ahora que la solución entrega mejor retroalimentación sobre la solución del sistema, se espera que los tiempos de validación se acorten considerablemente, lo que quitará peso a los administradores sobre el proceso. Asimismo, esta respuesta automatizada permitirá mejorar retroactivamente la asignación de horarios, al ya no depender de las restricciones posteriores dadas por la variante histórica de la distribución de salas.

Como parte del trabajo futuro se contempla el equilibrar el peso que se le da a las variables de distribución total, maximización de nivel de ocupación y preferencia de edificios en el modelo de optimización, para así representar correctamente las necesidades de los estudiantes, los docentes y la SGD. De igual forma, dado que no se pudo validar la tabla de preferencias con la SGD dentro del marco de la memoria, cae la necesidad de generar una que represente lo que se percibe como necesario según los agentes participantes en el desarrollo de los cursos. Es posible seguir puliendo la solución final con futuras iteraciones, pudiendo a largo plazo encontrar una combinación de peso y preferencias que exprese correctamente las necesidades esperadas. Además, con el trabajo constante y ligado a usuarios, podrían emerger nuevas necesidades de la interfaz, más aún considerando una aplicación real en que sea más evidente qué requerimientos están faltando sobre la plataforma. Asimismo, se considera la posibilidad de agregar a las bases de datos de la Universidad información sobre qué sala posee qué equipo, y qué curso necesita de este, a fin de poder sumarlo al modelo de optimización. Igualmente, podría agregarse a la información del sistema el que ciertas clases de una misma sección se dividen en dos y, por lo mismo, se realizan en dos salas, cosa de contemplarlas como parte de la distribución. Además, puede plantearse el incluir las preferencias de ciertos profesores al sistema, tomando quizás una variable histórica para así prevenir estos escenario de forma automatizada.

El desarrollo de la solución implicó un profundo aprendizaje sobre métodos de optimización. Al tener que aprender sobre esta temática, se hizo lectura sobre distintos tipos de programación matemática, adentrándose más en los conocimientos de esta área. Junto a ello, se ahondó en la comunicación entre capas, al tener que desarrollar una plataforma en que distintas partes se comunicaran. Ello llevó a trabajar constantemente con interfaces, lógica y consultas sobre bases de datos. Así, se profundizó el conocimiento sobre los lenguajes Python, PHP, HTML y SQL, esencialmente.

De igual forma, se aprendió sobre modelos de optimización a través de solvers tales como

Gurobi. Esto vino de la mano con el conocimiento adquirido sobre programación matemática, especialmente programación lineal, sobre la que se sostuvo toda la solución. Aunque de todas formas se instruyó bastante sobre otros tipos de programación con el afán de llegar a la más adecuada para este desarrollo particular.

Finalmente se concluye que el trabajo desarrollado fue satisfactorio en base a las expectativas que se habían planteado inicialmente. Se cumplieron los objetivos iniciales, y se profundizó el conocimiento sobre las áreas contempladas. La plataforma no solo entrega valor útil a la facultad, sino que además mejora la comunicación entre el sistema y el usuario, y vuelve más eficiente el proceso administrativo.

Bibliografía

[1] Kien Ming Ng Aldy Gunawan and Kim Leng Poh. Solving the Teacher Assignment-Course Scheduling Problem by a Hybrid Algorithm. *World Academy of Science, Engineering and Technology*, (33):1–7, 2007.

[2] AMPL. What is AMPL?, fecha de acceso: enero 2020. `https://ampl.com/faqs/what-is-ampl/`.

[3] José María Ferrer Julián Barquín Pedro Linares Andrés Ramos, Pedro Sánchez. *Modelos matemáticos de optimización*. PhD thesis, Universidad Pontificia Comillas ICAI-ICADE, septiembre 2010.

[4] California State University East Bay. Classroom Assignments, fecha de acceso: marzo 2020. `https://www.csueastbay.edu/us/academic-scheduling/assignments.html`.

[5] Alfonso Fernández Bes. *Optimización matemática en procesos industriales*. PhD thesis, Universidad Politécnica de Madrid, febrero 2015.

[6] Hax Bradley and Magnanti. *Mathematical Programming: An Overview*. PhD thesis, Massachusetts Institute of Technology, 1977.

[7] Centro de Tecnologías de la Información Ucampus. Acerca de U-Campus, fecha de acceso: enero 2020. `http://ingenieria.uchile.cl/investigacion/centros-y-programas/133629/centro-de-tecnologias-de-la-informacion-ucampus`.

[8] Jeff Erickson. NP-Hard Problems. *Algorithms, Etc.*, Algorithms Notes(30), diciembre 2018.

[9] M. C. S. Boeres G. S. Bello, M. C. Rangel. *An approach for the Class/Teacher Timetabling Problem using Graph Coloring*. PhD thesis, 2008.

[10] JSON. Introducing JSON, fecha de acceso: enero 2020. `https://www.json.org/json-en.html`.

[11] Ioannis Karatzas. and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, Berlin, 2nd edition, 2000.

[12] Catherine Lewis. *Linear Programming: Theory and Applications*. PhD thesis, 2008.

[13] Juan Pavón Mestras. *Estructura de las Aplicaciones Orientadas a Objetos: El patrón Modelo-Vista-Controlador (MVC)*. PhD thesis, Universidad Complutense Madrid, 2008-09.

[14] University of the Pacific. General Assignment Classroom Scheduling Policy, fecha de acceso: marzo 2020. https://www.pacific.edu/about-pacific/administration-offices/office-of-the-registrar/general-assignment-classroom-scheduling-policy.html.

[15] Gurobi Optimizacion. About Gurobi, fecha de acceso: enero 2020. https://www.gurobi.com/es/company/about-gurobi/.

[16] Natalia Alguacil Enrique Castillo Antonio J. Conejo Pablo Pedregal, Ricardo García. *Formulación y resolución de modelos de programación matemática en ingeniería y ciencias*. 2002.

[17] Martin A. Philbert. Class & Classroom Scheduling Policy. *University of Michigan*, 2018. https://www.provost.umich.edu/space/instruct/ClassClassroomSchedulingPolicy.pdf.

[18] Philip Protter. *Stochastic Integration and Differential Equations*. Springer, 1990.

[19] José Jesús Martínez Páez. *Conceptos básicos de programación genética*. PhD thesis, Universidad Nacional de Colombia, 2001.

[20] Andrés Ramos. *Programación dinámica*. PhD thesis, Universidad Pontificia Comillas ICAI-ICADE, noviembre 2015.

[21] Eliana Mirledy Toro Ocampo Ramón Alfonso Gallego Rendón, John Fredy Franco Baquero. *Problema de asignación óptima de salones resuelto con Búsqueda Tabú*. PhD thesis, Universidad Tecnológica de Pereira, Vereda La Julia, Risaralda Colombia, 2008.

[22] Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*. Number 293 in Grundlehren der mathematischen Wissenschaften. Springer, Berlin [u.a.], 3. ed edition, 1999.

[23] Brown University. Classroom Assignments & Location, fecha de acceso: marzo 2020. https://www.brown.edu/about/administration/registrar/classroom-assignments-locations.

[24] Rice University. Classroom Assignment Process, fecha de acceso: marzo 2020. https://registrar.rice.edu/facstaff/assign_process/.

[25] Yale University. Assignment of Classrooms, fecha de acceso: marzo 2020. http://catalog.yale.edu/handbook-instructors-undergraduates-yale-college/classrooms-class-lists/assignment/.

[26] Harsh vardhan Dwivedi. *Analysis and Implementation of Room Assignment Problem and Cannon's Algorithm on General Purpose Programmable Graphical Processing Units*

with CUDA. PhD thesis, Clemson University, 2011.

[27] S. Kirkpatrick; C. D. Gelatt; M. P. Vecchi. Optimization by Simulated Annealing. *Science, New Series*, 220(4598):671–680., 1983.