UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Inscripción a turnos en cursos con cupos, considerando preferencias y superposición de horarios

Informe de Proyecto de Grado presentado por

Guillermo Waltes, Maximiliano Díaz

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Carlos Testuri
Fernando Islas

Montevideo, 14 de diciembre de 2024

Agradecimientos

En primer lugar queremos agradecer a nuestros tutores Carlos Testuri y Fernando Islas, quienes nos acompañaron a lo largo de la realización de este proyecto de grado. Sus comentarios y correcciones fueron claves en todas las etapas del proyecto.

También queremos agradecer a nuestros familiares y amigos, quienes nos apoyaron constantemente. Sin ellos esto tampoco habría sido posible.

Resumen

El presente proyecto de grado aborda la problemática de la inscripción de estudiantes a turnos de cursos con cupos en la Universidad de la República (UdelaR), considerando las preferencias de los estudiantes y la superposición de horarios. El proceso actual de inscripción, realizado a través del Sistema de Gestión Administrativa de la Enseñanza, presenta diversos inconvenientes como falta de transparencia, colapsos por saturación de la plataforma y desventajas para estudiantes con conexión lenta a internet. Como resultado, los estudiantes no siempre logran inscribirse en sus opciones preferidas. Se analizan varios artículos que modelan el problema de creación de cronogramas de cursos universitarios (UCTP, por sus siglas en inglés). Este proyecto propone una solución computacional como un problema de optimización multi-objetivo, utilizando la técnica de programación entera. La solución permite resolver el problema de asignación de estudiantes a turnos de cursos, tomando en cuenta las limitaciones de capacidad, la superposición de horarios y las preferencias estudiantiles. Se definen tres objetivos ordenados: maximizar la satisfacción de las preferencias de los estudiantes, maximizar la cantidad de estudiantes asignados a al menos un curso, y obtener una asignación equitativa entre los distintos estudiantes. Estos objetivos son jerarquizados utilizando ordenado lexicográfico y resueltos de forma secuencial. El modelo es implementado mediante el lenguaje de modelado algebraico AMPL con el *solver* Gurobi. Además, se desarrolla una aplicación de consola en *Python* que permite ejecutar el modelo utilizando distintas fuentes de datos. Se realiza un análisis de la solución utilizando seis instancias con datos reales, proporcionados por el Servicio Central de Informática de la UdelaR. Se concluye que el modelo cumple con el objetivo principal de asignar a los estudiantes a sus cursos y turnos más preferidos. Además, todos los estudiantes son asignados a por lo menos un curso, por lo que el segundo objetivo también se cumple. El objetivo de buscar una asignación equitativa se logra de forma parcial, siendo alcanzado en algunas instancias pero no en todas. Se establecen lineamientos para trabajos futuros que permitan obtener mejores resultados respecto a este último objetivo. Finalmente, el modelo produce asignaciones de calidad dentro de tiempos de ejecución razonables, mostrando su viabilidad para ser utilizado en situaciones reales dentro de la UdelaR.

Palabras clave: programación entera, optimización, multi-objetivo, optimización lexicográfica, equidad, preferencias, superposición horaria

Índice general

Capítulo 1

Introducción

Cada semestre la Universidad de la República (UdelaR) se enfrenta a diversas problemáticas en el proceso de inscripción de los estudiantes a cursos. En varias carreras los estudiantes seleccionan los horarios en que desean cursar las asignaturas. Sin embargo, puede ocurrir que en algunos de los horarios la oferta de cupos sea superada por la demanda de inscripción, lo que implica la necesidad de gestionar listas de espera. Este escenario requiere la implementación de múltiples períodos de inscripción, asegurando que las nuevas elecciones de los estudiantes no coincidan en horario con las inscripciones previamente realizadas.

Usualmente, la selección de horarios se realiza a través del Sistema de Gestión Administrativa de la Enseñanza (SGAE), y como es realizada en momentos puntuales en el tiempo, carece de información completa y resulta poco transparente. Además, al focalizar el tráfico en el sistema en un espacio de tiempo determinado, este colapsa ocasionalmente, provocando pérdida de tiempo para los estudiantes e inequidad en las oportunidades de selección. Como los estudiantes tienen muy poco tiempo para inscribirse a los cursos y horarios, usualmente no se inscriben a los cursos que prefieren sino a los que logran hacerlo. Asimismo, la elección es dependiente de la estabilidad y velocidad de la conexión a internet del estudiante, generando inequidad a estudiantes con peor acceso a internet.

Una opción para resolver este problema es afrontarlo de manera asíncrona por etapas. En una primera etapa se recaba la información sobre las preferencias de los estudiantes, donde estos eligen los cursos y turnos de cursos que desean cursar, ordenados en forma de *ranking*. Una vez que la información de todos los estudiantes es recabada, en una etapa posterior se construye una asignación de estudiantes a turnos de cursos que cumpla con las limitantes de oferta y que no asigne estudiantes a horarios superpuestos. Este escenario puede ser modelado como un problema de creación de cronograma de cursos universitarios (UCTP, por sus siglas en inglés). Este es un problema amplio que abarca diferentes casuísticas, pero en términos generales es definido por Gozali Alfian Akbar y Fujimura Shigeru (2020) como el problema de asignar eventos de enseñanza en un cierto tiempo y a un salón considerando las restricciones de las partes interesadas de la universidad, como estudiantes y profesores. Resolverlo implica

asignar los estudiantes a los turnos de curso disponibles, tomando en cuenta las preferencias de los estudiantes, los cupos disponibles, y la superposición entre horarios. Resolver este problema de forma manual es una tarea extenuante y que requiere mucho tiempo, y posiblemente no se pueda resolver de forma óptima. Desde el punto de vista computacional, este es un problema NP-Completo, lo que implica que no se conoce un algoritmo eficiente que pueda resolverlo en tiempo polinomial para todas las instancias posibles.

En este trabajo se alcanza el objetivo de implementar una solución computacional al problema y aplicarla a instancias de mediano y gran porte. Dicha solución permite resolver un problema base del tipo UCTP donde las asignaciones realizadas cumplen con criterios de calidad alineados al contexto de la UdelaR. En primer lugar, busca satisfacer lo mejor posible las preferencias de los estudiantes. Luego, maximiza la cantidad de estudiantes asignados a por lo menos un curso, con el fin de promover la participación estudiantil y que estos no queden desvinculados de la facultad. Por último, se busca evitar disparidades entre los estudiantes, evitando favorecer desproporcionadamente a ciertos estudiantes con respecto a otros.

Para el desarrollo de la solución se investiga de qué manera son abordados los problemas de esta índole, dado que son usuales en contextos educativos de todo el mundo. Luego, se hace uso del método de programación entera para la resolución del problema y se logra construir un modelo de tres fases el cual contempla todos los aspectos necesarios de la realidad. Dicho modelo se conforma de tres fases, jerarquizadas utilizando el método de optimización lexicográfica, el cual permite ordenar los objetivos según importancia, y es resuelto utilizando la resolución secuencial.

Con el propósito de evaluar la solución diseñada, se implementa el modelo utilizando el lenguaje de modelado algebraico AMPL. Para validar el correcto funcionamiento de este, y analizar cómo se comporta frente a diferentes escenarios, se generan casos de pruebas. Además, se desarrolla una aplicación de consola utilizando el lenguaje de programación *Python*, que permite a los usuarios resolver el problema utilizando fuentes de datos en distintos formatos.

Finalmente, se generan métricas que aportan información sobre las asignaciones realizadas. Dichas métricas dan información desde un enfoque tanto cuantitativo como cualitativo, utilizando los criterios de calidad establecidos. Una vez generadas las métricas, se analizan los resultados obtenidos utilizando datos reales provistos por el Servicio Central de Informática de la UdelaR (SeCIU). Estos datos fueron obtenidos en entornos académicos reales de la Facultad de Derecho de la UdelaR, recabados mediante encuestas a estudiantes, donde se les solicitó que señalen un orden de preferencia para los cursos y turnos que deseaban asistir.

El presente trabajo está estructurado en capítulos. En el Capítulo 2 se presenta la revisión de antecedentes con el estudio de los conceptos más relevantes, así como el estudio de otras soluciones a este problema y las distintas formas de resolverlo. En el Capítulo 3 se presenta la solución propuesta, compuesta por el modelo de programación entera y la aplicación de escritorio para ejecutarlo. Luego, en el Capítulo 4 se valida el modelo construido utilizando distintos ca-

sos de prueba, y se analizan los resultados provenientes de los datos provistos por el Servicio Central de Informática de la UdelaR (SeCIU). Por último, en el Capítulo 5 se detallan las conclusiones del trabajo junto con los lineamientos para investigaciones futuras.

Capítulo 2

Revisión de antecedentes

En este capítulo se introducen los conceptos más relevantes relacionados a los problemas similares al presentado en el Capítulo 1. En la Sección 2.1 se presenta una categorización de dichos problemas, en la Sección 2.2 se presenta un estudio de las restricciones fuertes y débiles de dichos problemas, mientras que en la Sección 2.3 se hace lo mismo con las preferencias. A continuación, en la Sección 2.4 se presenta un estudio de artículos que utilizan la programación lineal o entera para modelar el problema, y en la Sección 2.5 se hace lo mismo con el coloreo de grafos y los algoritmos genéticos. Luego, en la Sección 2.6 se hace un estudio de la programación multi-objetivo, y por último en la Sección 2.7 se presentan problemas usuales en contextos universitarios.

2.1. Categorización

Ante la necesidad o deseo de desarrollar actividades, se requiere encontrar cuándo y dónde realizarlas. Todas estas situaciones pueden ser englobadas por los problemas de despacho (*Scheduling Problem, SP*). Los SP son definidos por Wren (1996) como el arreglo de objetos en un patrón temporal o espacial de forma que se alcancen, o casi, algunos objetivos, y que se satisfagan, o casi, las restricciones sobre la forma en que pueden arreglarse los objetos. Dentro de los problemas de SP se encuentran los de programación de horarios (*Timetabling Problem, TP*), en los cuales existe la componente temporal.

Como los TP aparecen en una gran variedad de situaciones, y estas son muy diferentes entre sí, en la literatura se encuentran clasificaciones según el dominio del problema. Si se enfoca en el ámbito educativo, se tienen los problemas aplicados al bachillerato *(High School Timetabling Problems)* y a la universidad *(University Timetabling Problems)*. A su vez, dentro de los problemas aplicados a la universidad existen dos grandes problemas TP: creación de cronogramas de cursos universitarios *(University Course Timetabling Problem, UCTP)* y elaboración de cronogramas de exámenes universitarios *(University Examination Timetabling Problem, UETP)*. En la Figura 2.1 se muestra gráficamente la ca-

tegorización descrita, la cual es tomada de Babaei, Karimpour, y Hadidi (2015).

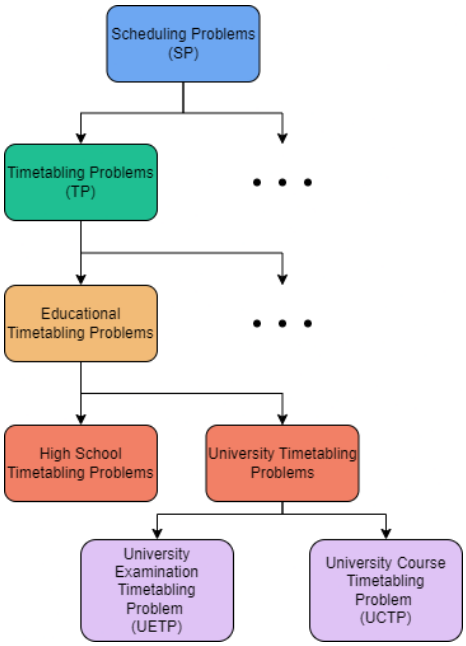

Figura 2.1: Diagrama de clasificación de los problemas de despacho educativo

Aunque la definición de la instancia del problema depende de los requisitos de cada universidad, hay elementos que suelen ser recurrentes. Babaei y cols. (2015) identifican seis elementos y los definen de la siguiente manera:

- **Evento**: una actividad programada, por ejemplo, un curso.
- **Franja horaria**: un intervalo de tiempo en donde cada evento es programado.
- **Recurso**: los recursos son utilizados por los eventos, por ejemplo, los salones y las franjas horarias.
- **Restricción**: una restricción es un condicionamiento en la programación de eventos, por ejemplo, la capacidad de salones, las franjas horarias disponibles, etcétera.
- **Personas**: incluye a los profesores y estudiantes, y cualquier otra persona interesada en la programación de horarios.
- **Conflictos**: los conflictos se dan entre los eventos. Por ejemplo programar más de un evento para un salón en la misma franja horaria.

El UCTP es definido por Gozali Alfian Akbar y Fujimura Shigeru (2020) como el problema de asignar eventos de enseñanza en un cierto tiempo y a un salón considerando las restricciones de las partes interesadas de la universidad, como los estudiantes, profesores y departamentos. Por otro lado, el problema UETP consiste en asignar exámenes a períodos de tiempo y salones. Según Burke, de Werra, y Kingston (2004), este tipo de problemas son muy similares al UCTP en su formulación general, pero suelen tener dos diferencias grandes. En el problema UETP se busca maximizar la utilización de los salones permitiendo asignar múltiples exámenes para el mismo salón al mismo tiempo, tomando en consideración que la capacidad del salón no sea sobrepasada. Por el otro lado, en el problema UCTP generalmente se tiene solo un evento por salón en un tiempo dado. La otra diferencia está en que en la programación de exámenes se busca espaciar los exámenes para que los estudiantes no tengan los exámenes seguidos, mientras que en el problema UCTP se busca que los estudiantes tengan las clases lo más cercanas posibles.

Los *Timetabling Problems*, además de ser considerados como *Scheduling Problems*, también forman parte de una categorización mayor que son los problemas de asignación. Faudzi, Abdul-Rahman, y Abd Rahman (2018) separan los problemas de asignación en problemas de armado de cronograma y problemas de distribución. Ya definidos los de armado de cronograma, se pasa a mencionar características de los denominados problemas de distribución (*Allocation problems, AP*).

Los AP son comunes en muchos escenarios de la realidad y se encuentran dentro del área de problemas de optimización combinatoria. En general, consiste en realizar una asignación óptima de recursos, y se distingue de los TP en que no consideran el tiempo como parte del problema. Este tipo de problemas aplicado al área educativa son conocidos como los problemas de distribución educativa (*Educational Allocation Problems*), y pueden ser categorizado en subcategorías según las características particulares: problema de distribución de estudiantes a proyectos *(Student Project Allocation, SPA)*, problema de distribución de nuevos estudiantes *(New Student Allocation Problem, NSAP)* y problema de distribución de espacio *(Space allocation problem, SAP)*. En la Figura 2.2 se muestra gráficamente la categorización descrita.

SPA es el problema de asignar estudiantes a proyectos tutorados por profesores. Este tipo de problemas en general involucra preferencias, ya sea de estudiantes sobre proyectos y/o de profesores sobre los estudiantes. Por otro lado, NSAP es el problema de distribuir nuevos estudiantes a las clases que les corresponda según su nivel de formación, con el fin de lograr que los estudiantes con nivel similar estén en la misma clase. Por último, SAP es el problema de asignar recursos a espacios físicos, por ejemplo, dictado de clases a salones. Este tipo de problemas involucra capacidades y/o disponibilidad de los espacios. Ejemplos de estos problemas se pueden encontrar en Anwar y Bahaj (2003), Hassim, Haniza, Shibghatullah, y Shahbodin (2014) y Gosselin y Truchon (1986) respectivamente.

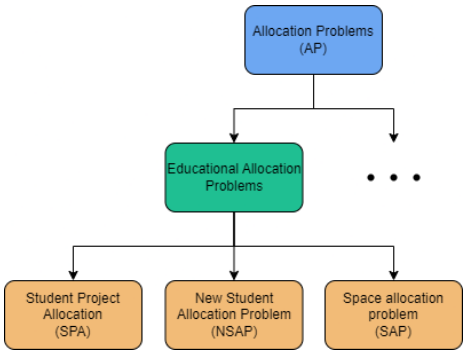

Figura 2.2: Diagrama de clasificación de los problemas de distribución educativa

2.2. Estudio de restricciones

Las restricciones pueden ser categorizadas como *restricciones fuertes* y *restricciones débiles*. Las primeras son las restricciones que una solución del problema debe cumplir, y usualmente se vinculan a normas o reglas del dominio, y las segundas son las que son preferibles que la solución cumpla, y se vinculan a objetivos del problema. En la programación matemática estas son modeladas de distinta forma, mientras que las restricciones fuertes suelen aparecer modeladas como restricciones del modelo, las débiles aparecen como parte de la función objetivo.

Alghamdi, Alsubait, Alhakami, y Baz (2020) elaboran una revisión bibliográfica y realizan un resumen de las restricciones fuertes y débiles presentadas allí. Cabe destacar que la categorización de las restricciones como fuertes o débiles depende del contexto del problema, y lo que en un problema puede ser catalogado como fuerte, en otro puede ser catalogado como débil. Se realiza una revisión bibliográfica de problemas similares al introducido en el Capítulo 1 y a continuación se presenta la Tabla 2.1 con restricciones fuertes y débiles. En la Sección 2.4 se presenta el resumen de los problemas resueltos en los artículos (Chartier, Ellison, y Langville, 2014), (Abdallah, 2016) y (Phillips, Walker, Ehrgott, y Ryan, 2017), mientras que en la Sección 2.5.2 se encuentra el resumen de (Abdallah, 2016).

2.3. Estudio de preferencias

Las preferencias son un componente importante que puede formar parte de los problemas de asignación. Las preferencias tienen características diferentes entre sí dependiendo del problema a resolver, pero en general se pueden identificar dos entidades: la entidad que tiene preferencia, y la entidad sobre la cual se tiene la preferencia.

Si se enfoca en el ámbito educativo, las entidades que tiene preferencias habitualmente son estudiantes o profesores, mientras que las entidades sobre

Art.	Restricciones fuerte	Restricciones débiles
1	• Debe haber un vehículo por grupo • El tamaño de los grupos debe estar entre 3 y 5 personas • El estudiante debe ser asignado a lo sumo a 1 clase • El estudiante debe ser asignado a clases a las cuales tenga disponibilidad • Las maestras deben tener a lo sumo una clase cubierta	• Lograr respetar preferencia horaria de los estudiantes • Lograr respetar preferencia de amigos de los estudiantes • Lograr respetar diversidad • Lograr la mayor participación de estudiantes
2	• Cada estudiante tiene que ser asignado a un único proyecto • Un proyecto puede ser asignado a lo sumo a un estudiante • El proyecto asignado a un estudiante tiene que haber sido escogido por él	• Lograr uniformidad en la asignación de proyectos a profesores • Lograr que los estudiantes sean asignados a su primera elección
3	• No exceder la capacidad de las salas en un período de tiempo • No hay cursos recomendados para un período cuyos exámenes sean asignados al mismo día • El examen es programado para un único día y período • Hay exámenes que deben ser programados para el mismo día	• Lograr que los estudiantes tengan la menor cantidad de exámenes en el mismo período de tiempo • Lograr que los estudiantes no tengan más de un exámen por día
4	• Cada salón en cada período está asignado a lo sumo a un evento • Todos los eventos son asignados a exactamente un salón en un período • Dos eventos de la misma currícula no pueden ser movidos para el mismo período de tiempo • En el caso de un evento ocupe dos períodos, estos períodos deben ser asignados consecutivamente en el tiempo y no se debe cambiar el salón	• Se quiere un cronograma similar al original

[1] (Chartier y cols., 2014)
[2] (Anwar y Bahaj, 2003)
[3] (Abdallah, 2016)
[4] (Phillips y cols., 2017)

Tabla 2.1: Tabla de restricciones fuertes y débiles, según bibliografía

las cuáles se establecen preferencias varían entre cursos, horarios, proyectos, universidades, etcétera. También puede suceder que en un problema se tengan preferencias *two-sided*; o sea que una entidad prefiere otra determinada entidad, pero que también exista la preferencia inversa. Por ejemplo, Gartner y Kolisch (2021) describen el caso donde los estudiantes establecen una preferencia sobre universidades, y al mismo tiempo las universidades prefieren a los estudiantes con mejores notas.

En la mayoría de los problemas las preferencias se modelan como un *ranking*. Para ilustrar cómo se modelan las preferencias se presenta un ejemplo inspirado en varios problemas estudiados en la literatura, en el cual se le quiere asignar estudiantes a cursos. Sea S el conjunto de estudiantes, C el conjunto de cursos y P la matriz de preferencias entre estudiantes y curso. Para cada estudiante i y para cada curso j, existe una entrada $p_{i,j}$ en P con valores entre 0 y N, donde 0 representa que el estudiante no quiere cursar el curso, y N representa la mayor prioridad por parte del estudiante. Cabe destacar que existen problemas donde se quiere representar los rankings de forma opuesta: cuánto mayor sea el valor en $p_{i,j}$, menor es la preferencia del estudiante. Si bien este es un enfoque común para representar las preferencias hay otros casos donde $p_{i,j}$ toma valores binarios, que representa si un estudiante prefiere un curso o no. Un ejemplo en la literatura donde las preferencias son modeladas de esta manera se encuentra en Chartier y cols. (2014), cuyo contexto es resumido en la Sección 2.4, donde se quiere maximizar el número de estudiantes a los que se les asigna un curso elegido, y se modela la función objetivo:

$$\max \sum_{c \in C} \sum_{s \in S} p_{i,j} x_{i,j} \tag{2.1}$$

donde $p_{i,j}$ es binaria con valor 1 si el estudiante i prefiere al curso j y 0 en otro caso, y $x_{i,j}$ es una variable binaria de decisión que tendrá valor 1 si el estudiante i es asignado al curso j y 0 en otro caso. En este caso, no es de interés modelar si un estudiante prefiere un curso sobre otro, ya que solo se quiere que la mayor cantidad de estudiantes sean asignados a un curso elegido.

Las preferencias pueden aparecer de dos maneras distintas: pueden aparecer como requisito de la solución o como un aspecto donde es deseable cumplirlas lo mejor posible. Por un lado, un caso donde es considerado como requisito es cuando se impone que la asignación sea únicamente con alguna entidad preferida. Por ejemplo, en Anwar y Bahaj (2003), donde trata el problema de SPA con preferencias de estudiantes sobre proyectos, se exige que cada estudiante sea asignado a un único proyecto y este sea de su preferencia. Para asegurar que este proyecto sea de su preferencia se incluyen las siguientes restricciones:

$$\sum_{j \in P} s_{i,j} x_{i,j} = 1 \quad \forall i \in S \tag{2.2}$$

siendo $s_{i,j}$ un parámetro binario que vale 1 si el estudiante i prefiere al proyecto j y 0 en otro caso, y $x_{i,j}$ una variable de decisión binaria que vale 1 si el estudiante i es asignado al proyecto j o 0 sino. Esta restricción, sumada a

la restricción de que cada estudiante sea asignado a un único proyecto, asegura que todo estudiante será asignado solo a un proyecto que eligió.

Por otro lado, se dice que la solución apunta a satisfacer preferencias cuando se intenta maximizar algún aspecto referente a estas. Siguiendo con el ejemplo descrito por Anwar y Bahaj (2003), en este se quiere maximizar la satisfacción de los estudiantes, y para esto se utiliza el parámetro $s_{i,j}$ como parámetro entero, donde el estudiante i selecciona su prioridad de 1 a 4 con respecto al proyecto j. Para cumplir esto tiene la siguiente función objetivo[1]:

$$\max \sum_{j \in P} \sum_{i \in C} s_{i,j} x_{i,j} \tag{2.3}$$

2.4. Modelado mediante programación entera/lineal

Los problemas de asignación pueden ser modelados mediante programación lineal o algunas de sus variantes como programación binaria, programación entera o programación entera mixta, y las diferencias entre estas caen en el dominio de las variables de decisión. Mientras que en la programación lineal todas las variables de decisión pueden tomar valores dentro de los reales, en la programación binaria solo pueden tomar los valores de 0 o 1, y en la programación entera solo pueden tomar valores enteros. Por último, en la programación entera mixta se pueden tener variables que admitan valores enteros o valores reales.

Esta forma de modelar es habitual en problemas de asignación, tanto para problemas de distribución como para problemas de cronograma. Todas estas son subclases de la programación matemática y existen métodos de resolución exacta para resolver los problemas modelados de esta manera. Además esta herramienta de modelado nos permite modelar tanto restricciones fuertes como restricciones débiles y encontrar soluciones óptimas que satisfagan ambas. En general, permite modelar aspectos como capacidades, cupos, oferta, demanda, preferencias, entre otros. En Anwar y Bahaj (2003) se menciona que la programación entera es la única técnica que resolverá apuntado a un óptimo global siempre y cuando exista solución, y esto la convierte en una buena opción siempre y cuando se logre modelar todos los aspectos del problema.

En Anwar y Bahaj (2003) se utiliza la programación entera para modelar el problema clásico de SPA, donde se tienen proyectos individuales y grupales que se les quiere asignar un grupo de estudiantes que desarrollen el proyecto, y uno o más profesores que tutoraren el proyecto. Se resuelven los problemas de proyectos individuales y proyectos grupales como modelos independientes. En el modelo para los proyectos individuales, el objetivo principal es que los profesores tengan una carga de trabajo equitativa entre sí, y tiene como objetivo secundario asignarle a los estudiantes un proyecto de su preferencia. Por otro lado, para los proyectos grupales se agrega la restricción fuerte de que los grupos tengan un número mínimo y máximo de estudiantes asignados, y se tiene

[1] La función objetivo presentada en el artículo fue simplificada con el fin de quitar elementos no relevantes a las preferencias

como objetivo asignarle a los estudiantes un proyecto de su preferencia. Ambos modelos son modelos de programación entera, y el modelo para los proyectos individuales es multi-objetivo y lo resuelven utilizando la resolución secuencial mediante la optimización lexicográfica, lo cual es definido en la Sección 2.6. Los modelos son implementados en el *software* de propósito general LINGO 6.0, y son resueltos en menos de tres segundos. Los modelos son aplicados a datos de la Universidad de Shouthampton, Reino Unido, donde para el caso de los proyectos individuales se tienen 39 estudiantes y 60 proyectos, mientras que para el caso de los proyectos grupales se tienen 68 estudiantes y 30 proyectos. Se obtienen resultados satisfactorios, por ejemplo, en el caso del modelo grupal, el 68 % de los estudiantes es asignado a su primera elección, el 22 % a su segunda elección y el 10 % a su tercera elección.

Para comparar el modelo con otras soluciones existentes, los autores toman como referencia un programa imperativo AssignProj el cual fue desarrollado por Teo y Ho (1998). Para realizar la comparación se ejecuta el modelo utilizando el *data set* provisto en Teo y Ho (1998), y se comparan las asignaciones y el tiempo de ejecución. En cuanto al tiempo de ejecución, el modelo es altamente superior con una duración de 6,6 minutos para llegar a una solución factible, mientras que AssignProj no logra una solución factible en menos de 24 horas. Con respecto a la calidad de la solución, esta es altamente superior. En primer lugar, el modelo llega a una solución factible, logrando asignar todos los estudiantes, en cambio AssignProj deja 45 estudiantes sin asignar. Además el modelo también logra que la mayoría de los estudiantes estén asignados a sus primeras opciones.

Otros autores que utilizan esta técnica de modelado son Chartier y cols. (2014), que presentan un problema donde se consideran las preferencias horarias pero no se busca crear un cronograma. El problema a resolver es de distribución, y consiste en asignar estudiantes voluntarios al dictado clases escolares, donde el horario de estas ya está predefinido. El objetivo principal es maximizar la cantidad de estudiantes asignados, pero también se busca optimizar otros objetivos. Por ejemplo, al momento de asignar se busca maximizar dos preferencias distintas del estudiante: su preferencia horaria y su preferencia de compañero a la hora de dictar la clase juntos. Además, se busca que haya diversidad racial y por género en los grupos.

Para resolver este problema, los autores plantean un modelo de programación entera mixta. Como se tiene más de un objetivo a cumplir, este modelo es un modelo multi-objetivo y lo resuelven utilizando la resolución secuencial mediante la optimización lexicográfica. En este se desarrollan cuatro fases en las que se busca maximizar un objetivo distinto en cada una, donde estos son resueltos en orden de importancia. Esto último es visto con más detalle en la Sección 2.6.

Para evaluar el modelo se utilizan dos *data sets*, uno chico (46 estudiantes, 19 clases y 19 maestras) y uno mediano (99 estudiantes, 150 clases y 26 maestras), y se realiza una comparación de cada fase con un modelo escalarizado (se realiza una combinación lineal entre las funciones objetivos y los pesos), y también con el punto de utopía (cada fase es corrida sin tomar en consideración los valores obtenidos en las fases anteriores, y por lo tanto el valor más alto que una fase puede alcanzar). En términos generales el modelo por fases logra mejores

valores objetivos en comparación con el modelo escalarizado, logrando valores muy cercanos al punto de utopía. Esto no se cumple para la última fase la cual busca la diversidad entre los grupos, ya que el modelo escalarizado da mejores resultados en términos de diversidad a expensas de los otros objetivos. Todas estas pruebas fueron realizadas utilizando el *solver* Xpress-MP.

Otro ejemplo que surge en la literatura es Gartner y Kolisch (2021), donde consideran el problema de asignar estudiantes de intercambio que se nominaron a universidades internacionales. En este problema se busca maximizar la cantidad de estudiantes asignados, y se consideran preferencias por parte de estudiantes, que eligen a sus universidades predilectas, y de las universidades, que buscan a los más capacitados. Los autores modelan el problema utilizando programación entera mixta, y en particular como el problema de la mochila múltiple, donde se busca maximizar la preferencia de los estudiantes y las aptitudes de los estudiantes.

Para resolver el problema se utiliza la API del *solver* IBM ILOG CPLEX en Java. En este caso, la universidad cuenta con una heurística que ha sido usada para resolver el problema en años previos, y esta es utilizada como línea base para la evaluación del modelo. Los autores con el fin de poder comparar las soluciones con el modelo propuesto definen algunos conceptos de "justicia", el cual establece entre otras cosas que estudiantes con mejores calificaciones tienen prioridad frente a estudiantes con calificaciones más bajas. Para evaluar el modelo se dispone de nueve casos de prueba donde se busca evaluar distintos objetivos, entre los que se encuentran el número de nominaciones y el número relativo de nominaciones a la primera, segunda y tercera universidad elegida. En todos se obtienen buenos resultados, con un porcentaje de asignación de estudiantes a universidades de entre un 61 % y un 85 %. Además, en todos los casos se resolvió en menos de un segundo.

2.5. Otras formas de resolución del problema

Existen otras formas de modelar los problemas estudiados. En particular, se estudia el modelado mediante el coloreo de grafos en la Sección 2.5.1, y el modelado mediante algoritmos genéticos en la Sección 2.5.2.

2.5.1. Modelado mediante coloreo de grafos

Uno de los métodos que se puede utilizar para resolver el problema de *timetabling*, y en particular los problemas de UCTP y UETP, consiste en modelarlo como un problema de coloreo de grafos. Este problema es extensamente estudiado en la teoría de grafos, y entre los algoritmos de resolución se encuentran *Saturation algorithm*, *Recursive Largest First algorithm*, *Simulated Annealing algorithm* y *Greedy algorithm* (Ganguli y Roy, 2017).

Ganguli y Roy (2017) presentan una manera de resolver el problema de UCTP coloreando un grafo de conflictos, en el cual se crea un vértice por cada curso, y las aristas entre los vértices representan si los cursos están en conflicto.

Los conflictos pueden ser causados por varias razones, por ejemplo, la asignación del mismo profesor a distintos cursos, la presencia de estudiantes inscriptos a ambas clases, o la necesidad de utilizar el mismo salón para el dictado de ambos cursos. La resolución del problema consiste en buscar un coloreo de vértices para el grafo construido, donde cada color representa un período de tiempo, aunque también se podría resolver el problema de coloreo de aristas, ya que según los autores son problemas equivalentes. Como el algoritmo asigna colores distintos a nodos adyacentes, se garantiza que cursos en conflicto se programen en horarios distintos.

Los autores resuelven dos problemas de cronograma de cursos, ambos de dimensiones pequeñas, y obtienen soluciones que utilizan un número mínimo de colores para colorear el grafo, lo que se traduce en un número mínimo de períodos para realizar el cronograma. A partir de esta resolución se obtiene la asignación de cursos a períodos, y por lo tanto crear el cronograma es directo. Los autores logran embeber las restricciones fuertes de sus problemas en el grafo de tal forma que dichas restricciones sean tomadas en cuenta al momento de crear la coloración. Sin embargo, no todas las restricciones pueden ser modeladas utilizando un grafo, y los autores dejan las restricciones débiles fuera del modelo. Debido a que en términos generales para el problema de coloreo hay más de una solución que utiliza el número mínimo de colores, los autores las evalúan de acuerdo a los criterios establecidos en las restricciones débiles, quedándose con la mejor de ellas.

Aunque Ganguli y Roy (2017) pudieron resolver los problemas planteados, cabe destacar que este método tiene grandes limitaciones. En primer lugar, como se mencionó anteriormente, no todas las restricciones pueden ser modeladas en un grafo, y se tiene que hacer evaluaciones externas al problema de coloreo para obtener la mejor solución. Lewis (2008) afirma que la conversión a coloreo de grafos es usualmente hecha cuando se consideran restricciones relativamente simples; cuando se tienen otras restricciones que ocurren en situaciones reales, el modelo de coloreo de grafos no es suficiente. Por otro lado, el problema de coloreo de grafos es NP-difícil, por lo que no se le conoce resolución eficiente (Ganguli y Roy, 2017).

A pesar de las limitaciones presentadas, Lewis (2008) sostiene que casi todos los problemas de *timetabling* tienen el problema de grafo subyacente de alguna forma u otra en su definición, y varios algoritmos de *timetabling* utilizan información heurística extraída de este problema como motivación.

2.5.2. Modelado mediante algoritmos genéticos

Los algoritmos genéticos son una metaheurística aplicada en muchos problemas de optimización multi-objetivo. Según Costabel (2005), los algoritmos genéticos son una técnica robusta y pueden manejar exitosamente un amplio rango de problemas, incluso algunos que son difíciles de resolver por otros métodos. En términos generales estos algoritmos no garantizan una solución óptima al problema, pero son generalmente buenos encontrando soluciones en corto tiempo. Cabe destacar que esto depende de la complejidad del problema y del

modelado del mismo. El principal campo de aplicación de estos algoritmos es donde no existan técnicas de resolución especializadas, ya que, cuando las hay, estas superan a los algoritmos genéticos en velocidad y precisión (Costabel, 2005).

Abdallah (2016) aborda un problema de construcción de cronogramas para exámenes en una universidad de Egipto. En este caso, este problema es extremadamente complejo debido a la cantidad de estudiantes, profesores y materias que integran la universidad. Este encaja dentro de la categorización UETP donde lo que se busca es poder calendarizar un conjunto de eventos (exámenes) en un numero finito de franjas temporales satisfaciendo un conjunto de restricciones, entre las que se encuentra la capacidad de las salas.

En este artículo, el autor desarrolla dos soluciones utilizando enfoques distintos: métodos exactos, elaborando un modelo de programación entera cuadrático, y un enfoque heurístico implementando un algoritmo genético. El modelo cuadrático es utilizado para minimizar el número de conflictos, que es la cantidad de estudiantes que tienen exámenes al mismo horario; pero esta solución no minimiza el número de choques, que es la cantidad de estudiantes que tienen más de un examen el mismo día. Para esto implementa un algoritmo genético que abarca ambos objetivos: minimizar conflictos y choques. Los modelos son evaluados con datos de una escuela de 1.285 estudiantes y 6.500 cursadas. El modelo cuadrático da una solución con pocos conflictos pero muchos choques, mientras que el algoritmo genético genera una solución con pocos conflictos y pocos choques.

2.6. Optimización multi-objetivo

En el contexto de la optimización, puede ocurrir que se requiera tomar en consideración más de un objetivo, los cuales pueden ser modelados mediante funciones nombradas función objetivo. En este contexto se definen los problemas de optimización multi-objetivo, que son los problemas donde se optimiza más de una función objetivo. En términos matemáticos este tipo de problemas son formulados de la siguiente manera (Miettinen, 1998):

$$\min_{x \in X} \{f_1(x), f_2(x), ..., f_k(x)\} \tag{2.4}$$

Donde X es el conjunto factible y $k \geq 2$ es la cantidad de funciones objetivo.

Estos problemas deben ser abordados con herramientas y enfoques distintos a los problemas de único objetivo, ya que podrían presentar conflictos entre los objetivos. Estos conflictos se dan cuando no existe una solución que optimice a todas las funciones objetivo simultáneamente. El caso más simple de resolver es cuando no hay conflictos entre las funciones objetivos. Como este caso es poco probable que ocurra, es necesario utilizar técnicas para lidiar con los conflictos. En el caso no trivial, se define una solución como Pareto-óptima si cualquier mejora en alguno de los objetivos empeora otro objetivo distinto (Miettinen, 1998). En otras palabras, una solución es Pareto-óptima si no existe otra solución que es mejor o igual en todos los objetivos, y tiene un objetivo estrictamente

mejor (Miettinen, 1998). Es importante disponer de información adicional ya que pueden existir muchas soluciones Pareto-óptimas (potencialmente infinitas) que son consideradas igual de buenas complejizando la elección de la mejor solución para el problema particular.

Para ejemplificar los conflictos entre funciones objetivo y las soluciones Pareto-óptimas, se muestra un ejemplo de una posible solución al problema presentado en Anwar y Bahaj (2003). En dicho problema, los profesores presentan proyectos para los cuales ellos podrían tutorar, y los estudiantes expresan su preferencia de proyectos. Además, se tienen dos objetivos a optimizar simultáneamente: uniformizar la cantidad de proyectos tutorados por los profesores, para que no haya ningún profesor que tenga una carga de trabajo significativamente mayor al resto, y asignar los estudiantes a los proyectos de su preferencia.

En un caso hipotético se podría tener que un profesor presente proyectos que sean muy preferidos por los estudiantes. Si se tiene una solución Pareto-óptima de este problema y, por ejemplo, se quisiera modificar dicha solución para mejorar la preferencia de los estudiantes, pasaría que el profesor que presentó los proyectos preferidos tendría que tutorar más proyectos, perdiendo uniformidad. En este caso, como forma de resolver estos conflictos, se prioriza la uniformidad de cantidad de proyectos tutorados frente a la preferencia de los estudiantes.

Para resolver los problemas de optimización multi-objetivo se tienen varios métodos, entre los que se encuentran la escalarización y la resolución secuencial mediante la optimización lexicográfica.

La escalarización consiste en convertir el problema multi-objetivo en un problema con una única función objetivo. Dicha función objetivo es una combinación lineal de las funciones objetivo, y los escalares que multiplican a las funciones se denominan pesos (Miettinen, 1998). Dado el problema (2.4), su escalarización consiste en escoger pesos $w_1, ..., w_k$ y resolver el problema de un único objetivo $\min_{x \in X} \sum_{i=1}^{k} w_i f_i(x)$. Esto disminuye considerablemente la complejidad del problema, ya que se pueden utilizar las técnicas conocidas para problemas de un único objetivo.

En casos donde se tiene un conocimiento a priori sobre la relevancia relativa de las funciones objetivo, se puede tomar en consideración esta información para decidir cuáles soluciones del problema son preferibles. Chartier y cols.(2014) y Anwar y Bahaj (2003) utilizan optimización multi-objetivo para resolver el problema de creación de cronograma, y ambos utilizan la optimización lexicográfica como método de selección entre las soluciones. Para utilizar la optimización lexicográfica se debe crear un orden total entre las funciones objetivo, ordenándolas por prioridad. Cabe destacar que utilizar la optimización lexicográfica implica que una pequeña mejora en una función prioritaria produzca una pérdida sustancial en otra función menos prioritaria (Zykina, 2004).

Para resolver el problema multi-objetivo ordenado lexicográficamente se tienen varios algoritmos, entre los cuales se encuentra la resolución secuencial. Esta consiste en realizar k fases (iteraciones), donde en cada una se optimiza una función objetivo distinta, siguiendo el orden definido. Se comienza resolviendo el problema de único objetivo de minimización f_1, con el conjunto de restricciones iniciales. Tomando el óptimo obtenido en la Fase 1, Z_1, se procede a la Fase

2, donde se resuelve el problema de único objetivo minimizando f_2. Como no se quiere empeorar el valor de f_1, se agrega la restricción $f_1(x) \leq Z_1$. Luego, se pasa a la Fase 3, donde se minimiza f_3 sujeto a las restricciones iniciales, $f_1(x) \leq Z_1$ y $f_2(x) \leq Z_2$, donde Z_2 es el óptimo obtenido en la fase anterior. Se continúa iterativamente hasta llegar a la fase final k, donde se tienen las restricciones iniciales y se mantienen los óptimos obtenidos en todas las fases anteriores. Si en alguna de las fases se obtiene una única solución, se culmina la ejecución, ya que en las fases siguientes no se podrá encontrar un mejor óptimo.

Chartier y cols. (2014) proponen un modelo con cuatro fases, donde en cada una se quiere optimizar un objetivo diferente. Como se utiliza optimización lexicográfica, las fases están ordenadas según las preferencias determinadas por la universidad. Debido a que algunas fases son computacionalmente complejas, los autores proponen los siguientes refinamientos para mejorar su complejidad computacional:

- Relajar las variables enteras o booleanas en variables reales.
- Relajar los bloques de disponibilidad horaria en bloques más grandes.
- Acotar superiormente o inferiormente la función objetivo de una fase en una fase posterior.
- En caso de que exista un parámetro con una dimensión muy alta y muchos valores nulos, definir el parámetro únicamente para los valores no nulos.
- Considerar si es fundamental encontrar el óptimo absoluto en la fase, o si con una aproximación es suficiente.

Por otro lado, Chartier y cols. (2014) presentan dos maneras para poder evaluar los modelos multi-objetivo con fases. La primera es comparar el modelo de fases con el modelo escalarizado, y utilizan los pesos elegidos mediante el proceso de jerarquía analítico de Saaty. La segunda manera es comparar cada fase con el punto de utopía, que es obtenido al correr cada fase independientemente, sin considerar las fases anteriores. Este punto nos da el objetivo ideal de cada fase, y por lo tanto es un buen punto de comparación para saber si el modelo cumple con los objetivos planteados.

2.7. Problemas usuales en contextos universitarios

En esta sección se explora dos problemas que surgen con frecuencia en el contexto universitario: el problema de seccionado y el problema de perturbación mínimal.

2.7.1. Problema de seccionado

En algunas situaciones, los cursos deben ser divididos en distintas secciones, que son copias del curso donde cada una tiene un período de tiempo, salón y

profesor distinto. Existen varias razones por las cuales se elige hacer este seccionado, y entre estas se encuentran que la cantidad de alumnos sea superior a la capacidad del salón, que haya más alumnos que los que un profesor pueda atender, o que se quiera disponer más de un período de horario para los cursos. El problema de seccionado de estudiantes consiste en asignar estudiantes a secciones, respetando algunas restricciones. En general, en dicho problema se tiene la restricción de que un estudiante no puede ser asignado a dos secciones del mismo curso, y que no haya un conflicto de secciones en el cronograma del estudiante, o sea que no sea asignado a dos secciones cuyos horarios se superpongan.

Existen distintas variantes al problema dependiendo del contexto de cada uno. Por ejemplo, como las distintas secciones pueden ser dictadas en salones con restricciones de capacidad, cada una de ella podría ser asignada a una cantidad diferente de estudiantes. En ocasiones los factores preferencias y seccionado aparecen juntos, y existen variantes sobre cómo se dan las preferencias estudiantiles. En algunos casos el estudiante tiene preferencia sobre los cursos (y no sobre los horarios), y en ese caso se asignan a los estudiantes a secciones de los cursos. Además, el estudiante puede tener preferencia sobre horarios, y en ese caso esto es tomado en cuenta al momento de crear la asignación. También puede suceder que los estudiantes ya estén inscriptos a los cursos, y en ese caso lo que se quiere resolver es asignar dichos estudiantes a las distintas secciones de estos cursos.

Según Carter (2001), cuando los cursos son asignados en múltiples secciones se crea una paradoja de *timetabling*. Los estudiantes solicitan cursos, pero hasta que no se realice el *timetabling*, no se sabe qué día y horario tienen las secciones de los cursos. Por un lado, no se puede asignar tiempos a las secciones hasta que se sepa qué estudiantes hay en cada sección, porque se quiere evitar los conflictos. Por otro lado, tampoco se puede asignar estudiantes a las secciones hasta que se sepa cuándo se van a llevar a cabo.

Carter (2001) realiza un *clustering* de estudiantes con solicitudes de cursos similares como primer paso del algoritmo de *timetabling*. Una vez que crean los *clusters*, se asignan los grupos a secciones para minimizar el número esperado de conflictos entre secciones de cursos. El artículo muestra un ejemplo con el fin de explicar por qué se crean los grupos de esta manera. El autor supone un contexto donde se tiene estudiantes de distintas carreras (tres estudiantes de matemática, tres de psicología y tres de ingeniería mecánica) que se inscriben al curso opcional de psicología PSY 101. En dicho ejemplo se crean los grupos con estudiantes que cursan la misma carrera (porque sus solicitudes son similares), reduciendo considerablemente la cantidad de potenciales conflictos. En la Figura 2.3 se muestra el grafo de conflictos potenciales entre cursos. En dicha figura los nodos son los cursos a los cuales los estudiantes mencionados se inscribieron, donde se tiene: MAT 1 a 4 los cursos de los la carrera de matemática, PSY 1 a 4 son los de psicología, MEC 1 a 4 los de ingeniería mecánica y PSY 101- 1 a 3 las secciones del curso PSY 101. Los nodos están conectados por una arista si hay estudiantes que tengan un potencial conflicto, o sea que estén inscriptos a ambos. Al separar los estudiantes en los grupos mencionados, se tienen 30 potenciales conflictos, mientras que si se crean los grupos con un estudiante de

cada carrera, cada sección de PSY 101 estaría en conflicto con todos los otros cursos, aumentando el número de conflictos potenciales a 54. Cabe destacar que esta asignación de grupos a secciones es una asignación inicial, y que luego que se crea el cronograma se asignan los estudiantes individualmente a las secciones.

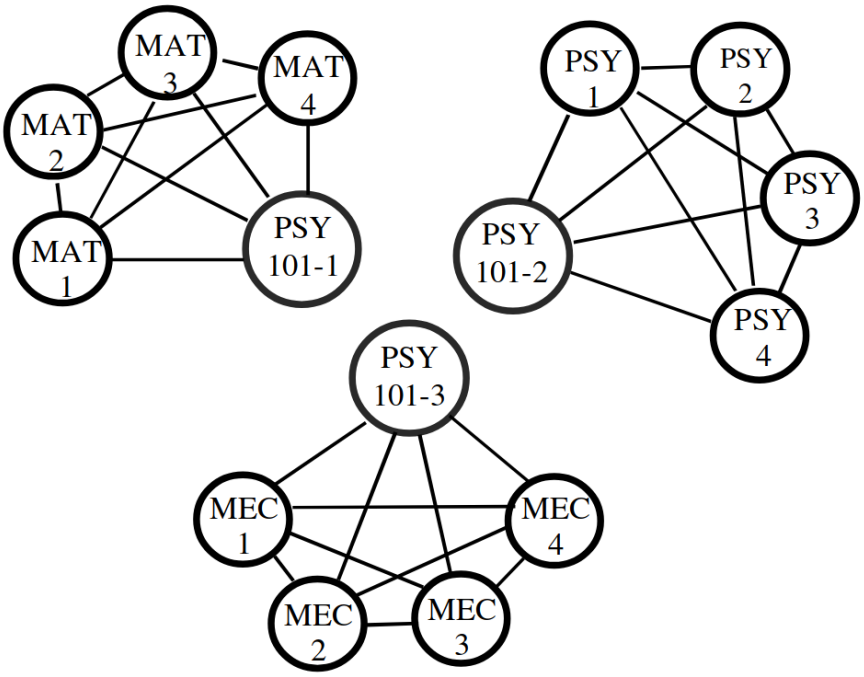

Figura 2.3: Grafo de conflictos potenciales (Carter, 2001).

2.7.2. Problema de perturbación minimal

Varios artículos estudian el problema de perturbación minimal (*Minimal Perturbation Problem, MPP*) en el contexto del problema UCTP. Este problema consiste en tomar un cronograma que tenga infactiblidades, y construir un cronograma factible lo más cercano al original (Lemos, Monteiro, y Lynce, 2021). Por ejemplo, en Lemos y cols. (2021) al momento de crear un cronograma para un año en especifico, se toma como cronograma inicial el del año anterior. Dicho cronograma puede tener alguna infactibilidad en el año cursante, pero como en términos generales es un cronograma bueno, se busca una solución cercana al mismo.

Para poder encontrar un cronograma cercano al original, es necesario definir el concepto de distancia entre cronogramas. Como en ambos cronogramas se tienen los mismos eventos, para ver la distancia entre estos se considera para cada

evento las diferencias en la asignación espacial y temporal en cada uno. Aunque esto sea abarcativo a todos los problemas de perturbación minimal, cada instancia puede determinar cómo valorizar dichas diferencias. Una vez que se define la distancia entre cronogramas, el objetivo es encontrar un cronograma que la minimice. Por ejemplo, Phillips y cols. (2017) penalizan con una unidad por cada período que un evento cambie, y con dos unidades por cada día, mientras que le da una pequeña penalización si el evento cambia de salón, ya que no es un problema que los eventos se realicen en salones distintos. Por otro lado, Lemos y cols. (2021) definen dos métricas: Distancia de Hamming, que se calcula contando los eventos que cambian de horario o de salón entre los dos cronogramas, y Distancia de Hamming con peso, que es la distancia de Hamming donde se puede ponderar cada cambio de una forma distinta. Esta última distancia es la misma que se utiliza en Phillips y cols. (2017), donde los pesos están dados por una matriz de pesos calculada utilizando los criterios explicados anteriormente, mientras que en Lemos y cols. (2021) el peso es definido como la cantidad de estudiantes afectados por el cambio.

Capítulo 3

Modelado del Problema

En este capítulo se presenta el modelo de programación entera construido para resolver el problema presentado en el Capítulo 1, con el detalle de conjuntos, parámetros, variables y restricciones definidas. Luego se detalla la implementación del modelo en el lenguaje algebraico AMPL, así como la aplicación de consola construida para poder ejecutar el mismo a partir de diversas fuentes de datos.

3.1. Modelo

En esta sección se presenta el modelo de programación entera construido. Para alcanzar una resolución satisfactoria se identifican tres objetivos: en primer lugar, se quiere respetar el orden de las preferencias de los estudiantes, es decir asignarlos a los cursos y turnos más preferidos. Luego se busca lograr la mayor cantidad de estudiantes asignados a al menos un curso. Por último, se quiere lograr una cierta equidad en las asignaciones con el fin de que no haya estudiantes muy beneficiados mientras otros sean muy desfavorecidos. Como se quiere una solución que satisfaga todos estos objetivos, se puede ver el problema como un problema multi-objetivo.

Como se describe en la Sección 2.6, en este tipo de problemas no es posible alcanzar todos los objetivos simultáneamente. Por ejemplo, si se enfoca en un objetivo en particular, como puede ser el de maximizar las preferencias globales, nada garantiza que un segundo objetivo se cumpla, como el de maximizar la cantidad de estudiantes asignados a un curso. Por el contrario, puede quedar una gran cantidad de estudiantes sin ser asignados. Sin embargo, en un contexto donde existan varias soluciones posibles al problema de maximizar la preferencias globales, se puede introducir el segundo objetivo de forma de buscar una solución que maximice la cantidad de estudiantes asignados a algún curso, dentro del subconjunto de soluciones que maximizan las preferencias globales. El mismo razonamiento es válido para el objetivo relacionado a la equidad, ya que nada nos garantiza que no existan estudiantes más favorecidos que otros,

y se puede aplicar el mismo concepto y buscar soluciones que adicionalmente cumplan el tercer objetivo.

Una forma de resolver este tipo de problema es la optimización lexicográfica, que exige ordenar los objetivos en orden de importancia. En nuestro caso se considera que en primer lugar se busca maximizar prioridades, luego maximizar la cantidad de estudiantes asignados a algún curso y finalmente alcanzar una solución equitativa. Se utiliza la resolución secuencial como mecanismo de resolución de optimización lexicográfica, utilizando tres fases, donde en cada una se busca optimizar un objetivo de los previamente mencionados sin perjudicar los objetivos alcanzados previamente. La formulación de cada una de las fases se encuentra en las Secciones 3.1.1, 3.1.2 y 3.1.3.

Con el fin de modelar el problema, se tiene el conjunto de estudiantes E, el conjunto de cursos C y el conjunto de turnos T. Debido a la estructura del modelo, este soporta dos alternativas a la forma de modelar los turnos. La primera alternativa es modelar los turnos como entidades compartidas entre los distintos cursos, y la segunda es modelarlos como entidades subordinadas al curso. Dado un contexto donde se tienen los cursos c y c' que se dictan en el mismo horario matutino, en el primer caso dichos cursos tienen el mismo turno M, que representa el turno matutino. En el segundo caso los turnos no son compartidos entre distintos cursos, es decir un turno t pertenece a un único curso c. Bajo el mismo contexto, en esta alternativa se tienen dos turnos: $M - c$ asociado al curso c, y $M - c'$ asociado al curso c'. Para soportar ambas alternativas, se define para cada curso $c \in C$ el conjunto de turnos del curso TC_c.

Para modelar las superposiciones entre turnos, que suceden cuando dos turnos comparten espacio temporal, se tiene el conjunto S_t, que se define como las superposiciones del turno t. En dicho conjunto se tienen todos los turnos que se superponen con t. Como la superposición es una relación simétrica, se cumple que si $t' \in S_t$, entonces $t \in S_{t'}$.

Por último se tienen las preferencias estudiantiles por curso y por turnos de curso. Los cursos preferidos del estudiante e se representan mediante el conjunto CP_e, y los turnos preferidos del estudiante e en el curso c se representan mediante el conjunto TP_{ec}. A partir de la unión de todos los conjuntos TP_{ec} se obtienen los estudiantes que prefieren el turno t del curso c, representados por el conjunto ET_{ct}.

Para las preferencias de un estudiante e sobre un curso c y las preferencias sobre un turno t de curso c, se plantean ordinales con los parámetros pc_{ec} y pt_{ect} respectivamente. Ambos ordenan las preferencias de forma que a valores menores, mayor es la preferencia. Por lo tanto, el curso c más prioritario para un estudiante e es el que tiene $pc_{ec} = 1$, el segundo más prioritario, c', tiene $pc_{ec'} = 2$, y así sucesivamente. Lo mismo sucede con los turnos: dado un curso c, el turno t más prioritario cumple que $pt_{ect} = 1$, mientras que el segundo, t' cumple que $pt_{ect'} = 2$. Una particularidad de pt es que para un estudiante, dos turnos de distintos cursos pueden tomar el mismo valor, por ejemplo si t es el turno más preferido de c, y t' es el turno más preferido de c', entonces $pt_{ect} = pt_{ec't'} = 1$. Otra particularidad de las preferencias de turnos es que no son comparables entre turnos de cursos distintos. En otras palabras, para los

turnos t del curso c y t' del curso c', pt_{ect} no es comparable con $pt_{ec't'}$ sino que el componente que permite decir si t es más o menos preferido que t' viene dado por la comparación entre preferencias de cursos, pc_{ec} y $pc_{ec'}$.

Con el propósito de dar valor a la preferencia de un estudiante e hacia un turno t de curso c, se combina la preferencia hacia el curso, pc_{ec}, con la preferencia hacia el turno, pt_{ect}, generando la prioridad combinada hacia un turno:

$$p\_comb_{ect} = Kpc_{ec} + pt_{ect} \tag{3.1}$$

En la ecuación (3.1) la función del parámetro K es evitar la superposición en las prioridades de turnos de cursos distintos. En otras palabras, para un estudiante e y dos turnos t y t' de cursos distintos (c y c' respectivamente), siempre se va a cumplir $Kpc_{ec} + pt_{ect} \neq Kpc_{ec'} + pt_{ec't'}$. Adicionalmente, como la prioridad sobre el curso toma más importancia que la prioridad sobre el turno, si se tiene que $pc_{ec} > pc_{ec'}$, entonces $Kpc_{ec} + pt_{ect} > Kpc_{ec'} + pt_{ec't'}$.

Para decidir si asignar un estudiante a un turno de curso se utiliza la variable binaria x_{ect}, que toma valor 1 en caso que el estudiante e es asignado al turno t del curso c, y 0 en el caso contrario. Dado que no tiene sentido asignar estudiantes a turnos que no son de su preferencia, esta variable solo está definida para los casos donde el estudiante indica una preferencia sobre el turno. Con esta restricción de dominio se busca reducir el número de variables x_{ect}, disminuyendo así la complejidad del problema para que la resolución sea más eficiente.

Un estudiante debe quedar en lista de espera de un curso en caso que el mismo no pueda ser asignado a alguno de sus turnos preferidos. Para modelar esto, se define la variable z_{ec} que toma valor 1 si el estudiante e es asignado a la lista de espera del curso c, y 0 en caso contrario. Para asegurar que z_{ec} valga 1 si y solo si el estudiante no es asignado a ningún turno del curso se incluye una restricción. Para garantizar que la lista de espera del curso c sea la peor opción posible para cada estudiante e, se penaliza su asignación con un valor elevado. Por lo tanto, se pondera z_{ec} con el parámetro M, el cual debe ser mayor que cualquier ponderación de x_{ect} para todo $t \in TP_{ec}$. Así, se debe tomar M mayor estricto a $\max_{e\in E, c\in CP_e, t\in TP_{ec}}\{(K * pc_{ec} + pt_{ect})^2\}$.

3.1.1. Fase 1: maximizar prioridades

En esta sección se presentan los componentes presentes en la primera fase del modelo. En esta primera fase se busca encontrar la o las soluciones que logran asignar de mejor manera a los estudiantes en función de sus prioridades. A continuación se reiteran sucintamente los componentes de conjuntos, parámetros y variables, definidos previamente.

Conjuntos

E: estudiantes
C: cursos
T: turnos

TC_c: turnos del curso $c \in C$, tal que $TC_c \subseteq T$

S_t: turnos que se superponen con el turno $t \in T$, tal que $S_t \subseteq T$

TP_{ec}: turnos del curso $c \in C$ preferidos por el estudiante $e \in E$, tal que $TP_{ec} \subseteq TC_c$

CP_e: cursos preferidos por el estudiante $e \in E$, tal que $CP_e \subseteq C$

ET_{ct}: estudiantes que prefieren el turno $t \in TC_c$ del curso $c \in C$, tal que $ET_{ct} \subseteq E$

Parámetros

pt_{ect}: preferencia del estudiante $e \in E$ sobre turno $t \in TC_c$ del curso $c \in C$, $pt_{ect} \in \{1, 2, ..., |TC_c|\}$

pc_{ec}: preferencia del estudiante $e \in E$ sobre el curso $c \in CP_e$, $pc_{ec} \in \{1, 2, ..., |CP_e|\}$

K: coeficiente que toma como valor la cantidad de turnos del curso con más disponibilidad de turnos ($\max_{c \in C}\{|TC_c|\}$). Es utilizado para dar peso a la preferencia del curso en la función objetivo

cap_{ct}: capacidad, en cantidad de estudiantes, del turno $t \in TC_c$ del curso $c \in C$

M: coeficiente de penalización a las asignaciones a la lista de espera

Variables de decisión

x_{ect}: vale 1 si el estudiante $e \in E$ es asignado al turno $t \in TP_e$ del curso $c \in CP_e$, y vale 0 en otro caso

z_{ec}: vale 1 si el estudiante $e \in E$ es asignado a la lista de espera del curso $c \in CP_e$, y vale 0 en otro caso

Formulación

$$\min \sum_{e \in E} \sum_{c \in CP_e} \sum_{t \in TP_{ec}} (Kpc_{ec} + pt_{ect})^2 x_{ect} + \sum_{e \in E} \sum_{c \in CP_e} M z_{ec} \tag{3.2}$$

s.a

$$\sum_{t \in TP_{ec}} x_{ect} + z_{ec} = 1, \quad e \in E,\ c \in CP_e, \tag{3.3}$$

$$x_{ect} + x_{ec't'} \leq 1, \quad e \in E,\ c \in CP_e,\ t \in TP_{ec},\ t' \in S_t, \\ c' \in CP_e : c' \neq c \wedge t' \in TC_{c'} \cap TP_{ec'}, \tag{3.4}$$

$$\sum_{e \in ET_{ct}} x_{ect} \leq cap_{ct}, \quad c \in C,\ t \in TC_c, \tag{3.5}$$

$$x_{ect} \in \{0, 1\}, \quad e \in E,\ c \in CP_e,\ t \in TP_{ec}, \tag{3.6}$$

$$z_{ec} \in \{0, 1\}, \quad e \in E,\ c \in CP_e. \tag{3.7}$$

Función objetivo

La función objetivo definida en (3.2) toma valores más chicos cuando se realizan más asignaciones y estas son realizadas a turnos de mayor preferencia. Dicha función se puede separar en dos sumandos, los cuales son analizados de forma independiente. Cada sumando tiene un rol diferente: el primer sumando busca cumplir lo mejor posible las prioridades, mientras que el segundo sumando busca maximizar la cantidad de asignaciones realizadas. Para lograr esto se separa por un lado las asignaciones de estudiantes a turnos, y por otro las asignaciones a lista de espera.

En el primer término se realiza una sumatoria ponderada de las asignaciones de estudiantes a turnos de curso, donde el peso es el cuadrado de la prioridad combinada. Como no se puede asignar un estudiante a un turno de curso no preferido por él, se crea una triple sumatoria que no itera sobre todos los elementos del conjunto C y T, sino que solamente incluye aquellos elementos preferidos por el estudiante. Para cada estudiante se itera sobre sus cursos preferidos, luego se itera sobre sus turnos preferidos del curso, y se multiplica la asignación del estudiante por el cuadrado de la prioridad combinada del estudiante hacia el turno del curso.

La prioridad combinada se encuentra elevada al cuadrado con el objetivo de penalizar las asignaciones con preferencia "extrema", ya que si el modelo asigna a un turno de curso con valor de prioridad grande, elevar el término al cuadrado incrementa el impacto que tiene esta asignación sobre la función objetivo. Esto tiene como efecto que las asignaciones se concentren en estudiantes-turnos con prioridades de valor intermedio, evitando caer en extremos, es decir, que hayan algunas asignaciones muy buenas y otras muy malas.

En el segundo sumando se trabaja con las asignaciones a la lista de espera de los cursos que prefiere el estudiante, y como se quiere penalizar enviar estudiantes a lista de espera, se pondera cada una de estas asignaciones con un valor M positivo muy grande que aumente el valor objetivo. El valor de M depende de la instancia, y debe ser mayor al cuadrado del valor más grande de prioridades combinadas, que es igual a $\max_{e \in E, c \in CP_e, t \in TP_{ec}} \{(Kpc_{ec} + pt_{ect})^2\}$. Al ser una función de minimización, esto tiene como consecuencia realizar la menor cantidad de asignaciones a lista de espera.

Restricciones

A continuación se presenta una explicación de las restricciones presentadas. La restricción (3.3) indica que cada estudiante debe ser asignado a exactamente un turno preferido por él en cada curso preferido, o a la lista de espera de dicho curso. Adicionalmente, la restricción (3.4) es utilizada para que se respeten las superposiciones entre turnos. Por ejemplo, si los turnos t y t' se superponen, no se debe asignar ningún estudiante a ambos turnos. Por otro lado, no se debe asignar más estudiantes a un turno de curso de lo que establece el parámetro de capacidad, lo cual queda definido en la restricción (3.5). Por último, la variables x_{ect} y z_{ec} quedan definidas como binarias en las restricciones (3.6) y (3.7)

respectivamente.

3.1.2. Fase 2: maximizar cantidad de estudiantes asignados

A continuación se presentan los componentes utilizados en la segunda fase del modelo, que utiliza como base lo desarrollado en la Fase 1, presentada en la Sección 3.1.1. En esta segunda fase el modelo busca soluciones que alcancen valores cercanos al óptimo obtenido en la fase anterior, y además busca tener la mayor cantidad de estudiantes asignados.

Conjuntos

Los definidos en la Sección 3.1.1

Parámetros

Y^1_{obj}: valor obtenido en la función objetivo de la Fase 1 (3.2)
α: coeficiente de relajación del valor del objetivo obtenido en la Fase 1, $\alpha \geq 1$
Los definidos en la Sección 3.1.1

Variables de decisión

y_e: vale 1 si el estudiante $e \in E$ es asignado a algún turno $t \in TP_c$ de algún curso $c \in CP_e$, vale 0 en cualquier otro caso
Las definidas en la Sección 3.1.1

Formulación

$$\max \sum_{e \in E} y_e \tag{3.8}$$

s.a

$$\sum_{c \in CP_e} \sum_{t \in TP_{ec}} x_{ect} \leq |CP_e| y_e, \quad e \in E, \tag{3.9}$$

$$y_e \leq \sum_{c \in CP_e} \sum_{t \in TP_{ec}} x_{ect}, \quad e \in E, \tag{3.10}$$

$$\sum_{e \in E} \sum_{c \in CP_e} \sum_{t \in TP_{ec}} (Kpc_{ec} + pt_{ect})^2 x_{ect} + \sum_{e \in E} \sum_{c \in CP_e} Mz_{ec} \leq \alpha Y^1_{obj}, \tag{3.11}$$

$$y_e \in \{0, 1\}, \quad e \in E, \tag{3.12}$$

Incluyendo restricciones (3.3) ... (3.6).

Función objetivo

La función objetivo definida en (3.8) maximiza la suma de y_e, o sea maximiza la cantidad de estudiantes asignados a por lo menos un curso.

Restricciones

A continuación se presenta una explicación de las restricciones presentadas anteriormente.

Las restricciones (3.9) y (3.10) son para dar valor 1 a y_e en caso que el estudiante sea asignado a algún curso y 0 en otro caso. Si un estudiante es asignado por lo menos a un curso, entonces va a existir un x_{ect} con valor 1, y en ese caso la restricción (3.9) obliga que y_e valga 1. En el caso que un estudiante no sea asignado a ningún curso, entonces la variable x_{ect} va a tomar valor 0 para todos los cursos y todos los turnos. En ese caso, la restricción (3.10) establece que y_e valga 0. Por otro lado, la restricción (3.11) establece que el valor de la Fase 2 no se aleje del valor objetivo obtenido en la Fase 1. Con el fin de buscar una mejor solución en función del objetivo de la Fase 2, se relaja el valor obtenido en la Fase 1 con el parámetro α. El valor de α determina cuánto se le permite al modelo desviarse de lo obtenido anteriormente con el fin de asignar más estudiantes. Esta relajación es de gran utilidad en casos donde existe una única solución a la Fase 1. Si no se relaja, o sea se toma $\alpha = 1$, la Fase 2 no puede alterar la solución obtenida, pero si se toma $\alpha > 1$ se amplía el conjunto de soluciones posibles, pudiendo encontrar una mejor solución en función del objetivo de la Fase 2. Por último, la restricción (3.12) define la variable y_e como binaria.

3.1.3. Fase 3: aumentar equidad entre asignaciones

A continuación se presentan los componentes utilizados en la tercera fase del modelo. En esta fase se buscan soluciones que reduzcan la brecha entre los estudiantes mejor y peor asignados. Para lograr esto es necesario valorizar el resultado de la asignación a un estudiante, y con este valor comparar si un estudiante recibe una mejor o peor asignación respecto a otro. Con este propósito se define la insatisfacción del estudiante e como

$$\sum_{c \in CP_e} \sum_{t \in TP_{ec}} (Kpc_{ec} + pt_{ect})^2 x_{ect} + \sum_{c \in CP_e} M z_{ec}$$

Una vez se logra expresar la insatisfacción se busca minimizar la mayor de estas. En esta fase se utiliza como base lo desarrollado en la Fase 1; Sección 3.1.1 y Fase 2; Sección 3.1.2.

Conjuntos:

Los definidos en la Sección 3.1.2

Parámetros:

Y^2_{obj}: valor obtenido en la función objetivo de la Fase 2; Sección 3.1.2
Los definidos en la Sección 3.1.2

Variables de decisión:

w: representa el máximo de las insatisfacciones individuales de estudiantes, $w \geq 0$
Las definidas en la Sección 3.1.2

Formulación:

$$\min\ w \tag{3.13}$$

s.a

$$\sum_{c \in CP_e} \sum_{t \in TP_{ec}} (Kpc_{ec} + pt_{ect})^2 x_{ect} + \sum_{c \in CP_e} M z_{ec} \leq w, \quad e \in E, \tag{3.14}$$

$$\sum_{e \in E} y_e = Y^2_{obj}, \tag{3.15}$$

$$w \geq 0, \tag{3.16}$$

Incluyendo restricciones (3.3) ... (3.12).

Función objetivo

En la función (3.13) se busca minimizar w, que como se expresa en la restricción (3.14), toma el valor del máximo de las insatisfacciones individuales. Esto hace que la solución final sea igual o más equitativa, ya que busca reducir la diferencia entre los estudiantes más y menos insatisfechos.

Restricciones

En la restricción (3.14) se define w como cota superior de las insatisfacciones individuales. En el lado izquierdo de la inecuación se tiene la representación de la insatisfacción individual de un estudiante, mientras que del lado derecho de esta se tiene w. Como esta restricción es válida para todos los estudiantes, w toma un valor superior a todas las insatisfacciones individuales. Por otro lado, con el fin de no empeorar lo obtenido en la fase anterior, en la restricción (3.15) se establece que la función objetivo de la Fase 2 sea igual al óptimo obtenido. Por último, la variable w queda definida como positiva en la restricción (3.16).

3.2. Implementación

Con el fin de llevar a la práctica el modelo formulado anteriormente, este se implementa en un lenguaje de modelado algebraico para posteriormente ser resuelto utilizando un *solver*. El lenguaje empleado es AMPL (*AMPL Options — AMPL Resources*, s.f.), y es resuelto haciendo uso de Gurobi (*Gurobi Optimization*, s.f.).

Con el objetivo de crear una interfaz amigable para la ejecución del modelo, se desarrolla una aplicación de consola que permite su ejecución a partir de diversas fuentes de datos. La aplicación es implementada en Python, aprovechando las ventajas de un lenguaje flexible, con un amplio ecosistema de bibliotecas que facilita la extensión de su funcionalidad (*Python*, s.f.). Además, su versatilidad y compatibilidad con diversas tecnologías facilitan su futura integración con otros sistemas.

Para interactuar con AMPL se utiliza la biblioteca *amplpy*, que ofrece una interfaz eficiente para acceder a las funcionalidades de AMPL directamente desde Python (*amplpy: Python API for AMPL*, s.f.).

En esta sección se profundiza sobre la implementación del modelo y la aplicación desarrollada.

3.2.1. Implementación del modelo

El modelo presentado en la sección anterior es implementado utilizando el lenguaje de modelado algebraico AMPL. El archivo *.mod*, incluido en el Anexo A.1, contiene la definición de conjuntos, parámetros, variables, restricciones y las funciones objetivos de las fases uno, dos y tres.

Los datos a ser utilizados en la ejecución del modelo pueden ser provistos en un archivo *.dat*. Con el fin de mejorar la consistencia de los mismos, se preprocesan los datos en la definición del modelo para reducir la cantidad de datos que se deben proveer al modelo en dicho archivo. En primer lugar se decide que las matrices de preferencias no sean definidas de manera explícita en el *.dat*, sino que sean calculadas en base a otros datos. Por ejemplo, no se debe especificar la matriz de preferencias de curso pc_{ec}, ya que esta, se calcula según el orden en que los cursos son definidos en el conjunto de cursos preferidos por el estudiante e, CP_e. Un comportamiento análogo ocurre para las preferencias de turno pt_{ect}, que se calculan según el orden en que se definen los turnos en el conjunto de turnos preferidos por el estudiante e en el curso c, TP_{ec}.

Tampoco se deben especificar los conjuntos que contienen la lista de estudiantes que prefieren un turno t del curso c, ET_{ct}. Se decide generar estos conjuntos mediante una expresión en el archivo *.mod*, ya que son fácilmente calculables utilizando los cursos y turnos preferidos por el estudiante.

Con el mismo propósito, no es necesario definir explícitamente el conjunto de los turnos T en el archivo *.dat*, ya que los subconjuntos TC_c son una partición de T. Para evitar delegar la mantención de la consistencia al usuario o al componente de software encargado de cargar los datos al modelo, se define T como la unión de todos los subconjuntos TC_c.

Se opta por la utilización de Gurobi como *solver* por su potencia que permite resolver instancias del porte de las instancias de SeCIU en tiempos razonables. Adicionalmente, cabe destacar que la primera opción considerada fue HiGS (Huangfu y Hall, 2018) pero los tiempos de ejecución son altamente superiores en comparación a la utilización de una licencia estudiantil de Gurobi.

3.2.2. Descripción de la aplicación en Python

La aplicación de consola implementada permite ejecutar las tres fases del modelo a partir de datos en distintos formatos. Esta puede recibir datos de tres formas distintas: archivos de datos *.dat* que se alineen con la especificación del modelo, archivos *.dat* que cumplan con la especificación del modelo de SeCIU, y por último planillas con formato *.xlsx*, cuyo formato es especificado en el Anexo A.2. La salida de la aplicación son tres archivos *.xlsx*, donde se presentan las salidas de cada fase en un formato legible para el usuario. En la Figura 3.1 se presenta un ejemplo de formato de salida, y la especificación completa de la misma se encuentra en el Anexo A.3.

	A	B	C	D	E	F
1		estudiante	curso	turno	x	var.rc
2	0	517898	830201	830201_45025	0	0
3	1	517898	830201	830201_45026	1	0
4	2	517898	830201	830201_45027	0	0
5	3	517898	830202	830202_45028	0	0
6	4	517898	830202	830202_45029	0	0
7	5	517898	830202	830202_45030	0	0
8	6	517898	830202	830202_45031	0	0
9	7	517898	830203	830203_45038	0	0
10	8	517898	830203	830203_45040	0	0
11	9	517898	830203	830203_45042	0	0
12	10	517898	830203	830203_45044	0	0
13	11	517898	830289	830289_45835	0	0
14	12	517898	830289	830289_45838	0	0
15	13	517898	830289	830289_45839	0	0
16	14	517898	830289	830289_45840	0	0
17	15	517902	830201	830201_45025	1	0
18	16	517902	830201	830201_45026	0	0
19	17	517902	830201	830201_45027	0	0
20	18	517902	830202	830202_45028	0	0
21	19	517902	830202	830202_45029	0	0
22	20	517902	830202	830202_45030	0	0

Figura 3.1: Captura de pantalla de la planilla de salida luego de ejecutar el modelo

Para la ejecución de la aplicación el usuario debe ejecutar el *script main.py* utilizando los siguientes argumentos:

Parámetro	Tipo	Descripción
--tipo-entrada	String	El tipo de la entrada. Puede ser seciu, dat o xlsx
--ruta-entrada	String	Ruta del archivo de entrada
--alpha	Float	Valor de α que se utiliza en las Fases 2 y 3 del modelo
--m	Integer	Valor de M que será utilizado en el modelo
--mipgap	Float	Valor de la opción MIPGap de Gurobi, utilizado para flexibilizar la ejecución

Tabla 3.1: Parámetros de la aplicación

El parámetro --mipgap permite modificar el valor de la opción de Gurobi MIPGap, que controla la diferencia que se permite entre la solución actual y la mejor cota conocida antes de detenerse (*MIPGap*, s.f.). Cuánto más pequeño este valor, más cercano a la cota se requiere la solución. Para determinar cuando culminar se utiliza un *gap*, que es calculado utilizando la cota primal z_P y la cota dual z_D. El valor de la cota primal corresponde a la mejor solución obtenida por el solver, la cual representa una cota superior en un problema de minimización. La cota dual z_D corresponde a la mejor cota inferior obtenida por el solver en caso de minimización. El cálculo utilizado se encuentra en (3.17). Cuando este valor es menor al valor de la opción especificada, se culmina la ejecución y se devuelve la solución. Si el parámetro de la aplicación no es especificado, se utiliza el valor por defecto de la opción: 1×10^{-4}.

$$gap = \frac{|z_P - z_D|}{|z_P|} \tag{3.17}$$

3.2.3. Componentes

En la Figura 3.2 se presenta el diagrama de componentes de la aplicación, creado utilizando el lenguaje de modelado UML, que ilustra los componentes desarrollados y sus dependencias para el funcionamiento de la aplicación. Además, se muestran los archivos utilizados por el sistema, que son los archivos de datos de entrada (en formato *.xlsx* o *.dat*), y el archivo de definición de modelo *base_model.mod*, así como el componente que hace uso de ellos.

main

Este archivo es el punto de entrada de la aplicación; sus responsabilidades principales son procesar los argumentos de entrada descriptos en la Tabla 3.1 y orquestar todos los pasos de la ejecución, comunicándose con el resto de los componentes.

Dicho componente no contiene ninguna lógica propia de la aplicación sino que delega acciones al resto de los componentes. Para empezar hace uso de la clase *Model* para cargar el modelo desde un archivo *.mod*. Luego carga los datos utilizando la clase *DataLoader*, quién contiene la lógica para cargar los datos según el formato. Una vez que los datos son cargados en el modelo, utiliza la

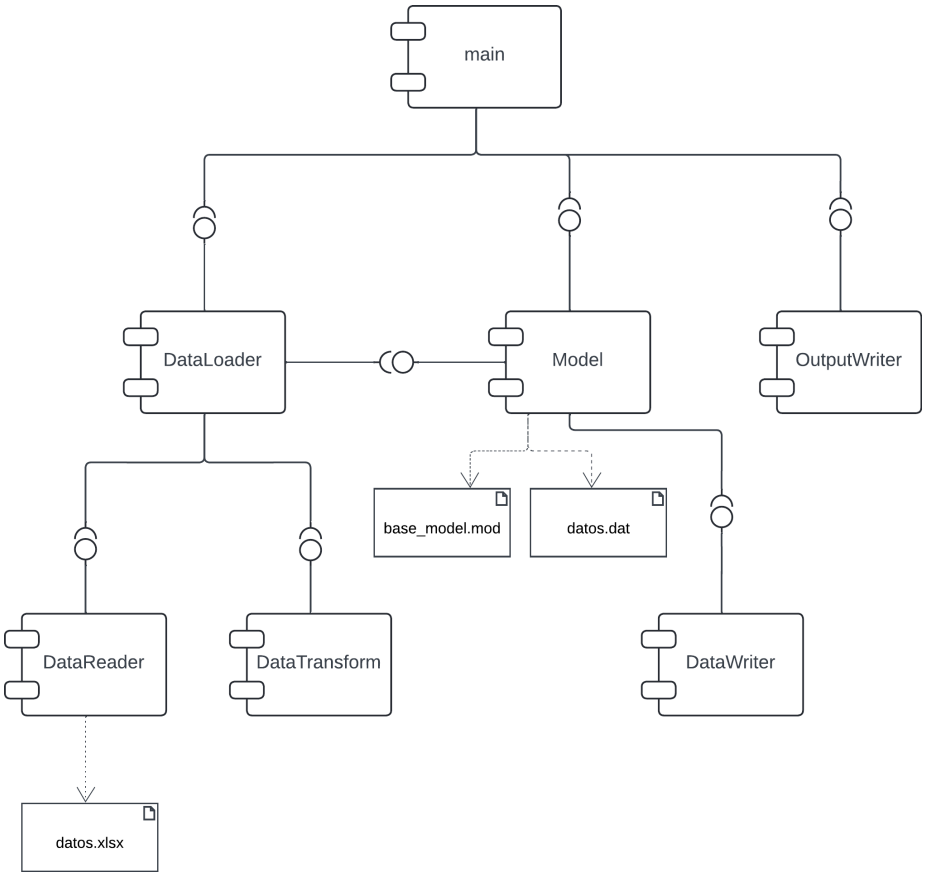

Figura 3.2: Diagrama de componentes UML de la aplicación

clase *Model* nuevamente para la ejecución de las tres fases, y finalmente genera la salida en formato *.xlsx* utilizando *OutputWriter*.

DataLoader

La clase *DataLoader* es utilizada por *main* y tiene como responsabilidad cargar los datos en el modelo, considerando el formato de entrada. Si los datos se encuentran en un archivo *.dat*, estos se cargan directamente en el modelo mediante la clase *Model*. En caso de que la fuente de datos corresponda a un archivo *.dat* alineado con la especificación del modelo SeCIU, los datos son transformados conforme a la especificación del modelo implementado, empleando la clase *DataTransform*, y luego se incorporan en el modelo a través de la interfaz proporcionada por *Model*. Finalmente, si el archivo de entrada está en formato *.xlsx*, se utiliza la clase *DataReader* para la lectura de los datos, que

posteriormente se cargan en el modelo utilizando *Model*.

DataTransform

La clase *DataTransform* se encarga de la transformación de los datos que se encuentran en formato del modelo de SeCIU al formato del modelo implementado.

Ambos modelos son muy similares. Sin embargo, existen diferencias entre ellos que deben ser solucionadas para ejecutar el modelo implementado con los datos provistos por SeCIU. En primer lugar, el modelo de SeCIU contiene información que no es de interés en el modelo implementado, por lo que es ignorada. Por otro lado, hay conjuntos que son especificados de forma distinta, por lo que se requiere una transformación.

La primera diferencia entre ambos modelos es la forma en la que son representadas las listas de espera. Mientras que en el modelo implementado las listas de espera son representadas con la variable z_{ec}, en el modelo de SeCIU lo son utilizando un turno extra que tiene el sufijo 99999. Por lo tanto, esta clase realiza esta transformación, eliminando el turno de lista de espera de la lista de turnos del curso. Además se transforman las superposiciones, ya que en la fuente original de datos son dados a modo de pares ordenados de la forma (t_1, t_2), representando que el turno $t1$ se superpone con el turno $t2$. Estos pares se utilizan para la generación de los conjuntos S_t del modelo implementado. Finalmente, se construyen los conjuntos CP_e y TP_{ec} haciendo uso de las matrices de preferencia y los conjuntos de cursos preferidos por el estudiante y turnos preferidos por el estudiante.

DataReader

Esta clase tiene como responsabilidad leer los datos que están almacenados en formato *.xlsx*, y transformarlos para que se pueda ejecutar el modelo con los mismos. En el Anexo A.2 se especifica la estructura que el archivo debe seguir para que pueda ser procesado por esta clase.

Model

Es la clase principal de la aplicación, es la que mantiene el estado de la ejecución del modelo y la que provee interfaces para realizar distintas acciones en él.

Esta provee métodos que permiten cargar la definición del modelo desde un archivo *.mod*, cargar los datos desde un archivo *.dat* y ejecutar cualquiera de las tres fases del modelo. Para implementar estas funcionalidades se hace uso de las facilidades provistas por la librería *amplpy*.

Además de proveer estas funcionalidades, esta clase se encarga de configurar las opciones de ejecución de AMPL y del *solver* (*AMPL Options — AMPL Resources*, s.f.). Las opciones más importantes establecidas son Gurobi como *solver*, y el valor de MIPGap, el cual es establecido con el parámetro de entrada

de la aplicación *--mipgap*. Luego se establecen otras configuraciones que son útiles para tareas de depuración.

DataWriter

La clase *DataWriter* es generada con el fin de cargar en un objeto de tipo AMPL (clase provista por la librería amplpy) todos los conjuntos y parámetros necesarios para la ejecución del modelo. Por esto, se provee un método que recibe un objeto cuyas propiedades contienen toda la información a ser cargada, y dicha clase los asigna a conjuntos y parámetros de la clase AMPL.

Como se presenta en la descripción de la clase *Model*, esta utiliza a *DataWriter* para poder cargar los datos en el modelo.

OutputWriter

OutputWriter es diseñada para generar las salidas del modelo de forma persistente. Para lograrlo, se decide almacenar dichas salidas en planillas en formato *.xlsx* con el fin de poder procesarlas fácilmente luego de la ejecución del modelo. En las planillas de cada instancia se guardan dos hojas, llamadas x y z, donde se guardan los valores de las variables x_{ect} y z_{ec} respectivamente. La especificación completa de la salida se encuentra en el Anexo A.3.

3.2.4. Implementación del cálculo de métricas

Además del desarrollo de la aplicación principal, se desarrolla un *notebook* llamado *experimentacion.ipynb* en JupyterLab (*jupyterlab: JupyterLab computational environment*, s.f.). Se decide utilizar JupyterLab porque es una tecnología que provee un ambiente interactivo, permitiendo una rápida visualización de los comandos ejecutados, y una visualización eficiente de gráficas y tablas útiles a la hora de presentar métricas de forma eficaz. En el archivo nombrado se implementan todas las métricas presentadas en la sección 4.2, utilizando las planillas en formato *.xlsx* obtenidos como salida de la ejecución del modelo.

3.2.5. Tecnologías utilizadas

Para la implementación de la aplicación se utiliza el lenguaje de programación *Python*, en su versión 3.11 (*Python*, s.f.). Este lenguaje es escogido debido a la fácil integración existente con AMPL mediante la librería *amplpy*, que es utilizada en su versión 0.14.0 (*amplpy: Python API for AMPL*, s.f.). A su vez, para la manipulación y transformación de datos se utiliza la librería *pandas* en su versión 2.1.3 (*pandas: Powerful data structures for data analysis, time series, and statistics*, s.f.). Por último, para visualizar las métricas se utiliza el ambiente de desarrollo basado en *notebooks* JupyterLab, en su versión 4.2.3 (*jupyterlab: JupyterLab computational environment*, s.f.), junto con la librería de visualización de gráficas *matplotlib*, en su versión 3.9.2

(*matplotlib: Python plotting package*, s.f.). Todos los requerimientos para ejecutar el programa están especificados en el archivo *requirements.txt*. En el repositorio https://gitlab.fing.edu.uy/maximiliano.diaz/tesis-grado-diaz-waltes se encuentra el código fuente de la aplicación.

Capítulo 4

Experimentación

En este capítulo se muestran los resultados de la experimentación realizada utilizando el modelo desarrollado en la Sección 3.1. En primer lugar, en la Sección 4.1 se presenta la validación del modelo realizada con casos de prueba de pequeña escala utilizados para estudiar y validar distintas características del mismo. Posteriormente, en la Sección 4.2 se describe la experimentación con los datos obtenidos de SeCIU, donde se hace un estudio cualitativo y cuantitativo con instancias de mediana y gran escala.

4.1. Validación del modelo

Con el fin de validar que el modelo de tres fases construido funciona como se espera, se diseñaron nueve casos de prueba que buscan evaluar distintos aspectos. Los casos de prueba utilizados en cada una de las validaciones se encuentran en el Anexo B.1.

4.1.1. Parámetros generales utilizados

A continuación se detallan los valores de los parámetros utilizados en las validaciones del modelo. En los casos donde se utilice un valor diferente a los especificados en la Tabla 4.1, se indicará explícitamente.

Parámetro	Valor
α	1
K	$\max_{c \in C} \lvert TC_c \rvert$
M	$(\lvert C \rvert \times K + \max_{e \in E} TP_e)^2 + 1$
MIPGap	1×10^{-4}

Tabla 4.1: Parámetros utilizados

4.1.2. Ponderación de preferencias de cursos sobre turnos

Objetivo: Comprobar que las preferencias sobre cursos tienen mayor relevancia que la preferencia de turnos.

Datos: Se presenta una instancia de datos con dos estudiantes ($E = \{E1, E2\}$), dos cursos ($C = \{C1, C2\}$) y un turno ($T = \{M\}$), donde ambos cursos son dictados en dicho turno. Todos los turnos de curso tienen capacidad dos.

Se presentan en la Tabla 4.2 las preferencias sobre los turnos de cada curso, y en la Tabla 4.3 las preferencias sobre los cursos.

	C1	C2
	M	M
E1	1	1
E2	1	1

Tabla 4.2: Preferencias de turno

	C1	C2
E1	1	2
E2	2	1

Tabla 4.3: Preferencias de curso

En este caso, el modelo solo podrá asignar a cada estudiante a un único curso, ya que los turnos disponibles de ambos cursos se superponen. Dado que las preferencias de turnos de ambos estudiantes son idénticas, se pretende demostrar que la preferencia por el curso es el factor decisivo en la asignación. Como el estudiante $E1$ prefiere $C1$ sobre $C2$, y lo prefiere más que el estudiante $E2$, la solución esperada es que el estudiante $E1$ sea asignado a $C1$, independientemente de los valores de preferencia de turnos. De manera análoga, el estudiante $E2$ prefiere $C2$ y, por lo tanto, debe ser asignado a este curso.

La definición de los datos utilizados en la ejecución (en formato *.dat*) se encuentra en el Anexo B.1.1.

Solución: Luego de la ejecución del modelo con esta instancia se obtuvo la asignación presentada en la Tabla 4.4.

	C1	C2
	M	M
E1	1	0
E2	0	1

Tabla 4.4: Valores de x_{ect}

Estos resultados son los esperados, ya que el modelo asigna el estudiante $E1$ al curso $C1$ y el estudiante $E2$ al curso $C2$. Esto prueba que el modelo cumple que las preferencias de cursos tienen mayor relevancia que las preferencias de turnos.

4.1.3. Asignación de estudiantes a sus cursos más preferidos

Objetivo: Comprobar que el modelo asigna cada estudiante a su mejor elección de curso siempre y cuando no exista conflictos entre los turnos de curso.

Datos: Se presenta una instancia de datos con tres estudiantes ($E = \{E1, E2, E3\}$), tres cursos ($C = \{C1, C2, C3\}$) y un turno ($T = \{M\}$), donde todos los cursos son dictados en dicho turno. Todos los turnos de curso tienen capacidad tres.

Todos los estudiantes tienen preferencia de ir a todos los cursos, pero con distinta prioridad: el estudiante $E1$ prefiere más a $C1$, el estudiante $E2$ a $C2$ y el estudiante $E3$ a $C3$. Como todos los turnos de curso tienen capacidad tres, en cualquiera de ellos podrían ser asignados todos los estudiantes. Por otro lado, como todos los cursos se dictan en el mismo turno, cada estudiante puede ser asignado únicamente a un curso. El modelo debe escoger a qué curso asignar a cada estudiante, y si este se comporta como se quiere, debe tomar en consideración las preferencias planteadas, asignando cada estudiante a su curso más preferido.

Se presentan en la Tabla 4.5 las preferencias sobre los turnos de cada curso, y en la Tabla 4.6 las preferencias sobre los cursos.

	C1	C2	C3
	M	M	M
E1	1	2	3
E2	2	1	3
E3	3	2	1

Tabla 4.5: Preferencias de turno

	C1	C2	C3
E1	1	2	3
E2	2	1	3
E3	3	2	1

Tabla 4.6: Preferencias de curso

La definición de los datos utilizados en la ejecución (en formato *.dat*) se encuentra en el Anexo B.1.2.

Solución: La solución para este caso de prueba se muestra en la Tabla 4.7.

	C1	C2	C3
	M	M	M
E1	1	0	0
E2	0	1	0
E3	0	0	1

Tabla 4.7: Valores de x_{ect}

Como se puede ver en la Tabla 4.7, cada estudiante es asignado a su curso más preferido.

4.1.4. Asignación de estudiantes a sus turnos más preferidos

Objetivo: Comprobar que el modelo busca asignar a cada estudiante a su turno de curso de mayor preferencia. Esta característica del modelo asegura que cuando dos estudiantes tienen la misma preferencia para un curso, la preferencia de turno determina qué turno se le asigna a cada uno.

Datos: Se presenta una instancia de datos con tres estudiantes ($E = \{E1, E2, E3\}$), un curso ($C = \{C1\}$) y tres turnos ($T = \{M, V, N\}$), cada uno con capacidad para los tres estudiantes. Todos los estudiantes tienen la misma preferencia por el curso $C1$, pero difieren en la prioridad que establecen a los turnos. En esta instancia, el estudiante $E1$ prefiere el turno M, seguido por el turno V, y finalmente el turno N. El estudiante $E2$ prefiere el turno V en primer lugar, luego el turno M, y después el turno N. Por su parte, el estudiante $E3$ prefiere el turno N, seguido por el turno V, y en último lugar el turno M.

Se presenta en la Tabla 4.8 las preferencias sobre turnos de cada curso, y en la Tabla 4.9 las preferencias sobre cursos.

	C1		
	M	V	N
E1	1	2	3
E2	2	1	3
E3	3	2	1

Tabla 4.8: Preferencias de turnos

	C1
E1	1
E2	1
E3	1

Tabla 4.9: Preferencias de cursos

Para esta instancia el modelo debe escoger a que turno de curso asignar a cada estudiante. La hipótesis indica que el modelo debe asignar cada estudiante

a su turno más preferido del curso $C1$.

La definición de los datos utilizados en la ejecución (en formato *.dat*) se encuentra en el Anexo B.1.3.

Solución:

	C1		
	M	V	N
E1	1	0	0
E2	0	1	0
E3	0	0	1

Tabla 4.10: Valores de x_{ext}

Como se puede ver en la Tabla 4.10, cada estudiante es asignado a su turno más preferido.

4.1.5. Asignación de estudiantes a sus preferencias menos deseadas antes que a la lista de espera

Objetivo: Comprobar que el modelo prefiere asignar estudiantes a su peor elección antes que asignarlos a lista de espera.

Datos: Se presenta una instancia de datos con tres estudiantes ($E = \{E1$, $E2$,$E3\}$), tres cursos ($C = \{C1, C2, C3\}$) y tres turnos ($T = \{M, V, N\}$), donde todos los cursos son dictados en todos los turnos. Además, todos los turnos de curso tienen capacidad uno.

Se presentan en la Tabla 4.11 las preferencias sobre los turnos de cada curso, y en la Tabla 4.12 las preferencias sobre los cursos.

	C1			C2			C3		
	M	V	N	M	V	N	M	V	N
E1	1	2	3	1	2	3	1	2	3
E2	1	2	3	1	2	3	1	2	3
E3	1	2	3	1	2	3	1	2	3

Tabla 4.11: Preferencias de turno

	C1	C2	C3
E1	1	2	3
E2	1	2	3
E3	1	2	3

Tabla 4.12: Preferencias de curso

En este caso las peores asignaciones para cada estudiante corresponde a que sean asignados al turno N del curso $C3$.

La definición de los datos utilizados en la ejecución (en formato *.dat*) se encuentra en el Anexo B.1.4.

Solución: Luego de la ejecución del modelo con esta instancia se obtiene la asignación presentada en la Tabla 4.13.

	C1			C2			C3		
	M	V	N	M	V	N	M	V	N
E1	0	1	0	1	0	0	0	0	1
E2	0	0	1	0	1	0	1	0	0
E3	1	0	0	0	0	1	0	1	0

Tabla 4.13: Valores de x_{ect}

Como se puede ver en esta solución se lograron realizar 9 asignaciones ya que cada estudiante es asignado a los 3 cursos. En este caso el estudiante $E1$ es asignado a su peor elección, el turno menos preferido de su curso menos preferido $(C3 - N)$. Este resultado permite concluir que el modelo prefiere asignar un estudiante a su peor elección antes que asignarlo a la lista de espera.

4.1.6. Cumplimiento de la capacidad de turnos

Objetivo: Comprobar que la restricción definida en la inecuación (3.5) es construida correctamente, es decir que esta restricción asegura que el modelo respeta la capacidad de los turnos. Con el fin de comprobar esto, se genera una instancia en la que se pueda verificar fácilmente que la mejor solución sería asignar todos los estudiantes a todos sus cursos preferidos. Sin embargo, se requiere comprobar que el modelo respeta la capacidad del turno de curso, y no realiza más asignaciones que dicha capacidad.

Datos: Se presenta una instancia de datos con tres estudiantes ($E = \{E1, E2, E3\}$), un curso ($C = \{C1\}$) y un turno ($T = \{M\}$), donde $C1$ es dictado en el turno M, y este tiene capacidad para un estudiante.

Se presentan en la Tabla 4.14 las preferencias sobre los turnos de cada curso, y en la Tabla 4.15 las preferencias sobre los cursos.

	C1
	M
E1	1
E2	1
E3	1

Tabla 4.14: Preferencias de turno

	C1
E1	1
E2	1
E3	1

Tabla 4.15: Preferencias de curso

La solución esperada es que el modelo realice únicamente una asignación, ya que esta es la capacidad disponible.

La definición de los datos utilizados en la ejecución (en formato *.dat*) se encuentra en el Anexo B.1.5.

Solución: La solución para este caso es la siguiente

	C1
	M
E1	1
E2	0
E3	0

Tabla 4.16: Valores de x_{ect}

La asignación que optimiza la función objetivo de la Fase 1 (3.1.1), sin tener en cuenta las restricciones, es asignar todos los estudiantes al turno de curso disponible. Sin embargo, esta no es una solución factible ya que el modelo respeta la capacidad de los turnos de cursos. Por esto el modelo no tiene otra alternativa que asignar un único estudiante al curso, y el resto a la lista de espera de dicho curso, perjudicando el valor objetivo. Esto permite concluir que para este caso de prueba las restricciones de capacidad son satisfechas.

4.1.7. Exclusión de asignación a turnos superpuestos

Objetivo: Comprobar que la restricción definida en la inecuación (3.4) está hecha de forma correcta, es decir, que esta restricción asegura que no se asigna ningún estudiante a dos turnos de curso superpuestos.

Datos: Se presenta una instancia de datos con tres estudiantes ($E = \{E1, E2, E3\}$), dos cursos ($C = \{C1, C2\}$) y tres turnos ($T = \{M1, M2, V\}$), donde ambos cursos son dictados en todos los turnos. Adicionalmente, todos los turnos de curso tienen capacidad tres. Los turnos $M1$ y $M2$ representan dos turnos matutinos, y en esta instancia están superpuestos. Además, el turno V está superpuesto únicamente consigo mismo.

Se presentan en la Tabla 4.17 las preferencias sobre los turnos de cada curso, y en la Tabla 4.18 las preferencias sobre los cursos.

	C1			C2		
	M1	M2	V	M1	M2	V
E1			1			1
E2	1				1	
E3		1		1		

Tabla 4.17: Preferencias de turno

	C1	C2
E1	1	2
E2	1	2
E3	1	2

Tabla 4.18: Preferencias de curso

Se ve para cada estudiante cuál es la asignación esperada. El estudiante $E1$ tiene preferencia sobre el mismo turno, V, en ambos cursos.

Como se demostró en la Sección 4.1.5, el modelo prefiere asignar los estudiantes a su peor elección antes que asignarlos a lista de espera, por lo que si no existiera la restricción (3.4), este asignaría al estudiante $E1$ al turno V en ambos cursos. Sin embargo, si dicha restricción cumple el cometido que se busca, esta asignación no es posible, por lo que este estudiante sólo podrá ser asignado a uno de ellos. El caso del estudiante $E2$ es ligeramente distinto, ya que declaró preferencia sobre turnos distintos en ambos cursos (sobre el turno $M1$ en $C1$ y $M2$ en $C2$). Pero como estos turnos están superpuestos, el modelo tampoco puede asignarlo a ambos cursos. Un razonamiento análogo se puede realizar sobre el estudiante $E3$ y los turnos $M2$ de $C1$ y $M1$ de $C2$.

La definición de los datos utilizados en la ejecución (en formato *.dat*) se encuentra en el Anexo B.1.6.

Solución: El resultado de la corrida del modelo en este caso se presenta en la Tabla 4.19.

	C1			C2		
	M1	M2	V	M1	M2	V
E1			1			0
E2	1				0	
E3		1		0		

Tabla 4.19: Valores de x_{ect}

Aunque todos los turnos de curso tienen capacidad suficiente y la función objetivo busca maximizar la cantidad de asignaciones, la restricción (3.4) funciona correctamente, asegurando que ningún estudiante sea asignado a turnos superpuestos.

4.1.8. Maximizar la cantidad de estudiantes asignados a al menos un curso

Objetivo: Comprobar que la Fase 2 cumple con el objetivo planteado, que es maximizar la cantidad de estudiantes asignados a por lo menos un curso. Además, se busca experimentar con distintos niveles de α para relajar el valor obtenido en la Fase 1.

Datos: Se presenta una instancia de datos con tres estudiantes ($E = \{E1, E2, E3\}$, dos cursos ($C = \{C1, C2\}$) y tres turnos ($T = \{M, V, N\}$, donde el curso $C1$ es dictado en los tres turnos, mientras que el curso $C2$ es únicamente dictado en el turno M. Todos los turnos de curso tienen capacidad uno.

La idea de la instancia es que su solución óptima de la Fase 1 deje sin asignar al estudiante $E2$ o $E3$. Sin embargo, para cumplir con el objetivo del caso de prueba, la instancia debe permitir soluciones factibles donde todos los estudiantes sean asignados, de modo que al ajustar el parámetro $\alpha > 1$ en la Fase 2 se puedan encontrar estas soluciones, y así cumplir con el objetivo de dicha fase.

Se presentan en la Tabla 4.20 las preferencias sobre los turnos de cada curso, y en la Tabla 4.21 las preferencias sobre los cursos.

	C1		C2
	V	N	M
E1		1	1
E2	1		1
E3	1		

Tabla 4.20: Preferencias de turno

	C1	C2
E1	2	1
E2	1	2
E3	1	

Tabla 4.21: Preferencias de curso

La definición de los datos utilizados en la ejecución (en formato *.dat*) se encuentra en el Anexo B.1.7.

Solución: Luego de correr únicamente la Fase 1, se obtiene la solución presente en la Tabla 4.22.

	C1		C2
	V	N	M
E1		1	1
E2	0		0
E3	1		

Tabla 4.22: Valores de x_{ect} luego de correr la Fase 1

Como se puede ver, el estudiante $E1$ es asignado a dos cursos, el estudiante $E2$ no es asignado a ningún curso, y el estudiante $E3$ es asignado a un curso. Cabe destacar que otra solución óptima es asignar al estudiante $E2$ al turno V del curso $C1$, ya que tiene la misma prioridad sobre este turno de curso que el estudiante $E3$, y en este caso sería el estudiante $E3$ que quedaría sin asignar.

Luego de correr la Fase 2 con $\alpha = 1{,}5$, se obtienen los resultados presentes en la Tabla 4.23.

	C1		C2
	V	N	M
E1		1	0
E2	0		1
E3	1		

Tabla 4.23: Valores de x_{ext} luego de correr la Fase 2

En este escenario, el estudiante $E1$ pasa de ser asignado a dos cursos a ser asignado solo a uno, el estudiante $E2$ pasa de no ser asignado a ningún curso a ser asignado a uno, y el estudiante $E3$ no varía en sus asignaciones. Por lo tanto, en esta solución todos los estudiantes son asignados a por lo menos un curso.

Para esta instancia es necesario tomar un valor de α mayor a 1 porque la solución obtenida en la Fase 2 no es óptima según la función objetivo de la Fase 1. Esto ocurre porque se reemplaza la asignación de $E1$ a su curso más preferido, $C2$, por la asignación de $E2$ a este curso, el cual es su menos preferido, lo que incrementa la función (3.2).

En pruebas experimentales se detectó que para que la Fase 2 arroje este resultado, el valor de α escogido no puede ser muy cercano a 1, porque en esos casos la solución presentada en la Tabla 4.23 no es factible. Por otro lado, α no debe ser muy grande porque se deteriora en mayor proporción lo obtenido en la Fase 1, llegando a una solución no deseada.

Finalmente, se ve que para esta instancia el modelo prefiere soluciones con la mayor cantidad de estudiantes asignados, ya que en este caso se obtiene una solución donde todos los estudiantes son asignados en lugar de una solución en la que queden estudiantes sin asignar.

4.1.9. Equilibrio en la satisfacción estudiantil

Objetivo: Comprobar que la Fase 3 cumple con el objetivo planteado, que es equilibrar la satisfacción estudiantil. Para mostrar esto, se revisa la solución intermedia obtenida por el modelo para una instancia y se comprueba que si bien es óptima en los objetivos de las Fases 1 y 2, esta no es equitativa en la cantidad de asignaciones. La solución final aplicando Fase 3 mantiene la optimalidad según las Fases 1 y 2, siendo además más equitativa.

Datos: Se presenta una instancia de datos con tres estudiantes ($E = \{E1, E2, E3\}$), seis cursos ($C = \{C1, C2, C3, C4, C5, C6\}$) y tres turnos ($T = \{M, V, N\}$), y en la Tabla 4.24 se presenta el turno disponible en cada curso. Además, todos los turnos de curso tienen capacidad uno. Los turnos M, V y N representan los turnos matutino, vespertino y nocturno respectivamente, los cuales solo están superpuestos consigo mismo.

Cursos	Turnos
C1	M
C2	M
C3	M
C4	V
C5	V
C6	N

Tabla 4.24: Valores de turno por curso

En esta instancia todos los estudiantes establecen preferencias sobre todos los cursos. Las preferencias estudiantiles sobre turnos son presentadas en la Tabla 4.25.

	C1	C2	C3	C4	C5	C6
	M	M	M	V	V	N
E1	1	1	1	1	1	1
E2	1	1	1	1	1	1
E3	1	1	1	1	1	1

Tabla 4.25: Preferencias de turno

Las preferencias sobre los cursos se presentan en la tabla 4.26.

	C1	C2	C3	C4	C5	C6
E1	1	2	3	4	5	6
E2	2	1	3	5	4	6
E3	3	2	1	4	6	5

Tabla 4.26: Preferencias de curso

La definición de los datos utilizados en la ejecución (en formato *.dat*) se encuentra en el Anexo B.1.8.

Solución: Para estos datos luego de ejecutar el modelo se obtiene como solución intermedia de la Fase 1 y Fase 2 la presente en la Tabla 4.27.

	C1	C2	C3	C4	C5	C6
	M	M	M	V	V	N
E1	1	0	0	1	0	0
E2	0	1	0	0	0	0
E3	0	0	1	0	1	1

Tabla 4.27: Valores de x_{ect} luego de las Fase 1 y Fase 2

Esta solución no es equitativa entre la asignación de los estudiantes. Para medir esto, definimos la insatisfacción de cada estudiante como:

$$I(e) = \sum_{c \in CP_e} \sum_{t \in TP_{ec}} (Kpc_{ec} + pt_{ect})^2 x_{ect} + \sum_{c \in CP_e} M z_{ec} \tag{4.1}$$

Tomando $K = 1$ y $M = 50$, se llega a que la insatisfacción por estudiante es: $I(E1) = 229$, $I(E2) = 254$, $I(E3) = 215$.

Por lo tanto en esta solución el estudiante $E2$ es más perjudicado que el estudiante $E3$. Esto se ve también reflejado directamente en la cantidad de cursos a los que es asignado un estudiante y otro. Mientras que los estudiantes $E1$ y $E3$ son asignados a dos y tres cursos respectivamente, el estudiante $E2$ es asignado únicamente a un curso. Esta no es la única solución a este problema, existen más soluciones que mantienen los valores objetivos alcanzados en la Fases 1 y 2, donde se puede buscar una mayor equidad.

Ahora contemplando la Fase 3 la solución obtenida se presenta en la Tabla 4.28.

	C1	C2	C3	C4	C5	C6
	M	M	M	V	V	N
E1	1	0	0	1	0	0
E2	0	1	0	0	1	0
E3	0	0	1	0	0	1

Tabla 4.28: Valores de x_{ect} luego de la Fase 3

Para esta solución los valores de insatisfacción de cada estudiante son: $I(E1) = 229$, $I(E2) = 229$, $I(E3) = 240$.

Estos son más equitativos que los vistos anteriormente. También se ve reflejado en la solución ya que todos los estudiantes son asignados a dos cursos. Cabe destacar que la Fase 3 busca mejorar al estudiante más desfavorecido, a expensas de uno o más estudiantes más favorecidos. Por ejemplo, en este caso el estudiante $E3$ aumenta su insatisfacción (de 215 a 240), pero igualmente el máximo de las insatisfacciones disminuye (254 a 240). Esto permite ver que siempre que sea posible la Fase 3 asegura soluciones más equitativas que si solo existieran las Fases 1 y 2.

Se puede corroborar que esta nueva solución es óptima en términos de Fase 1 comparando las sumas globales de las insatisfacciones. Al ser iguales, con valor 698, se puede decir que ambas soluciones son óptimas. La solución es óptima en Fase 2 ya que todos los estudiantes son asignados al menos a un curso. Esto es el máximo posible de estudiantes asignados al menos a un curso, por lo tanto es óptimo en dicho aspecto.

Concluimos entonces que la Fase 3 hace que el modelo prefiera soluciones con equilibrio en la satisfacción estudiantil.

4.1.10. Penalización de los estudiantes con mayor disponibilidad

Objetivo: Analizar las asignaciones del modelo en el caso de estudiantes con diferentes niveles de disponibilidad de preferencias.

Datos: Se presenta una instancia de datos con cuatro estudiantes ($E = \{E1, E2, E3, E4\}$), tres cursos ($C = \{C1, C2, C3\}$) y dos turnos ($T = \{M, V\}$). El curso $C1$ se dicta en los turnos M y V, mientras que los cursos $C2$ y $C3$ se dictan únicamente en el turno M, y todos los turnos de curso tienen capacidad uno.

Los estudiantes $E1$ y $E2$ tienen disponibilidad de ir únicamente al curso $C1$, pero el estudiante $E1$ muestra más disponibilidad con respecto a los turnos de curso, ya que puede asistir a ambos turnos, mientras que el estudiante $E2$ solo puede asistir al turno M. Por otro lado, el estudiante $E3$ muestra más disponibilidad de cursos a los cuales quiere asistir, $C2$ y $C3$, mientras que el estudiante $E4$ solo puede asistir al curso $C2$. En la Tabla 4.29 se presentan las preferencias de turno de los estudiantes, y en la Tabla 4.30 se presentan las preferencias de curso.

	C1		C2	C3
	M	V	M	M
E1	1	2		
E2	1			
E3			1	1
E4			1	1

Tabla 4.29: Prioridades de turno

	C1	C2	C3
E1	1		
E2	1		
E3		1	2
E4		1	

Tabla 4.30: Prioridades de curso

La definición de los datos utilizados en la ejecución (en formato *.dat*) se encuentra en el Anexo B.1.9.

Solución: En la Tabla 4.31 se muestra la solución obtenida.

	C1		C2	C3
	M	V	M	M
E1	0	1		
E2	1			
E3			0	1
E4			1	

Tabla 4.31: Valores de x_{ect}

La solución se analiza tomando en cuenta por un lado a los estudiantes $E1$ y $E2$, y por otro a los estudiantes $E3$ y $E4$, ya que estos estudiantes tienen preferencia sobre distintos cursos.

El estudiante $E2$ es asignado al turno M del curso $C1$, que representa su opción más prioritaria, mientras que el estudiante $E1$ es asignado al turno V del curso $C1$, su opción menos prioritaria. El modelo busca maximizar la satisfacción global, buscando el mayor número posible de asignaciones. Por ejemplo, si el estudiante $E1$ hubiera sido asignado a su opción más prioritaria, el turno M del curso $C1$, debido a que este turno tiene una capacidad de solo un estudiante, el estudiante $E2$ no habría podido ser asignado al curso. Esto resultaría en un menor número de asignaciones y, por lo tanto, en un aumento en el valor de la función objetivo. Acá se puede ver como mostrar una menor disponibilidad en turnos puede resultar en una ventaja frente a estudiantes con mayor disponibilidad.

Por otro lado, el estudiante $E4$ es asignado al turno M del curso $C2$, su opción más prioritaria, mientras que el estudiante $E3$ es asignado al turno M del curso $C3$, su opción menos prioritaria. De forma análoga a lo desarrollada en el párrafo anterior, se puede ver que la asignación que maximiza la satisfacción global es asignar a ambos estudiantes, aunque el estudiante $E3$ sea asignado a un curso menos prioritario. En este caso se puede ver como mostrar una menor disponibilidad en cursos puede resultar ventajoso.

Cabe destacar que, aunque mostrar menor disponibilidad puede ofrecer cierta ventaja, también conlleva un riesgo para el estudiante, ya que reduce las opciones disponibles para el modelo al momento de asignarlo, lo que podría resultar en un menor número de asignaciones para el mismo.

4.2. Experimentación con datos de prueba de SeCIU

Esta sección tiene como objetivo describir la experimentación con instancias reales que se recibieron. En la Sección 4.2.2 se realiza una descripción y análisis de la instancias, así como las dimensiones del modelo de optimización en cada

una de ellas. Luego en la Sección 4.2.3 se desarrolla sobre distintas métricas las cuales dan distintas visiones sobre las soluciones obtenidas: en primer lugar se da una visión global, luego se analiza la calidad, y por último se estudia la equidad entre los estudiantes obtenida. A continuación, en la Sección 4.2.4 se comparan indicadores a lo largo de las fases, lo que permite observar cómo influye cada una de estas. Posteriormente, en la Sección 4.2.5 se analiza la influencia del parámetro α en las Fases 2 y 3. Finalmente, en Sección 4.2.6 se comenta sobre los tiempos de ejecución obtenidos.

4.2.1. Parámetros generales utilizados

A continuación se detallan los valores de los parámetros utilizados en las siguientes secciones. En los casos donde se utilice un valor diferente a los especificados en la Tabla 4.32, se indicará explícitamente.

Parámetro	Valor
α	1
K	$\max_{c \in C} \lvert TC_c \rvert$
M	$(\lvert C \rvert \times K + \max_{e \in E} TP_e)^2 + 1$
MIPGap	0,05

Tabla 4.32: Parámetros utilizados

4.2.2. Dimensiones y análisis de instancias

A continuación se describen las instancias recibidas indicando la cardinalidad de conjuntos, cardinalidad de variables y restricciones generadas.

Se utilizan seis instancias de datos generadas por SeCIU en entornos reales de la Facultad de Derecho de la Universidad de la República. Los datos provienen de encuestas realizadas a estudiantes, donde estos señalan sus preferencias de cursos y turnos a cursar. Los estudiantes tienen la posibilidad de escoger los cursos que desean asistir, junto con su orden de prioridad. Asimismo para cada curso escogido deben elegir cuatro turnos ordenados por preferencia, a menos que el curso no disponga de tal cantidad, y en este caso el estudiante debe elegirlos todos.

Como se menciona en la Sección 3.2, cada una de estas instancias es transformada acorde a la definición de parámetros y conjuntos del modelo presentado en la Sección 3.1.

Cardinalidad de conjuntos por instancia

La Tabla 4.33 muestra información detallada de cada instancia. Para cada una, se incluyen las siguientes columnas: cantidad de cursos (C), cantidad de turnos (T), y cantidad de superposiciones (S), que representa la cantidad de turnos diferentes que se superponen entre sí (contabilizando las simetrías una única vez). También se presenta la cantidad de preferencias estudiante-curso (E-C), que corresponde al número total de cursos seleccionados por los estudiantes,

y la cantidad de preferencias estudiante-curso-turno (E-C-T), que indica el total de turnos elegidos por los estudiantes. Finalmente, se muestra la cantidad total de cupos disponibles (cap).

Para cada instancia se generó un nombre nemotécnico el cual indica la cantidad de estudiantes y cantidad de cursos de la misma.

Instancia	E	C	T	S	E-C	E-C-T	cap
E300-C4	300	4	26	27	1.200	4.500	1.122
E300-C10	300	10	77	154	3.000	11.700	395
E401-C94	401	94	863	38.453	28.200	110.400	72.522
E6457-C86	6.457	86	258	1.129	22.824	76.229	57.271
E7963-C94	7.963	94	506	10.706	108.263	295.660	58.711
E8208-C111	8.208	111	678	19.982	175.514	506.387	59.519

Tabla 4.33: Datos de las instancias

Las instancias abarcan un gran rango de dimensiones, desde la más chica que tiene 300 estudiantes y 4 cursos, hasta la más grande que tiene 8.208 estudiantes y 111 cursos. A partir de estos datos se puede evaluar cuán difícil es satisfacer la demanda total de los estudiantes. Por ejemplo, en aquellas instancias donde haya menos cupos totales (cap) que demanda de cursos por parte de estudiantes (E-C), que son las instancias *E300-C4*, *E300-C10*, *E7963-C94* y *E8208-C111*, resulta imposible que el modelo asigne todos los estudiantes a todos sus cursos preferidos.

Dimensiones del problema

Como se menciona en la Sección 1, este tipo de problemas son de complejidad NP-Completo, lo que hace que el número de variables impacte fuertemente en los tiempos de resolución en la práctica. En nuestro problema los factores que determinan esta dificultad no van ligados únicamente con la cantidad de estudiantes o cursos de la instancia, sino con la cantidad de preferencias estudiante-curso-turno, lo cual es a grandes rasgos una referencia de cuantas variables de decisión tiene nuestro problema.

Otro punto que en algunos problemas aumenta directamente la dificultad de resolución del mismo es la cantidad de restricciones; cuanto más restricciones se tienen el problema a resolver se puede volver más complejo. Cabe destacar que esto no siempre sucede, ya que en algunos problemas agregar restricciones puede facilitar la resolución. Por este motivo se considera importante presentar el número de restricciones.

En la Tabla 4.34 se presenta la cantidad de variables (#V) y restricciones (#R) para cada fase en cada una de las instancias.

	Fase 1		Fase 2		Fase 3	
Instancia	#V	#R	#V	#R	#V	#R
E300-C4	5.700	8.130	6.000	8.731	6.001	9.032
E300-C10	14.700	25.277	15.000	25.878	15.001	26.179
E401-C94	138.600	4.220.257	139.001	4.221.060	139.002	4.221.462
E6457-C86	99.123	69.252	105.580	82.167	105.581	88.625
E7963-C94	403.923	1.178.081	411.886	1.194.008	411.887	1.201.972
E8208-C111	681.901	3.400.438	690.109	3.416.855	690.110	3.425.064

Tabla 4.34: Cantidad de variables y restricciones creadas en cada fase

Como se observa en la tabla anterior, todas las instancias presentan un gran número de variables y restricciones. Sin embargo, la cantidad de variables de decisión no está únicamente relacionada con el número de cursos y estudiantes. Por ejemplo, aunque la instancia *E6457-C86* tiene significativamente más estudiantes y cursos que la instancia *E401-C94*, esta última resulta más compleja. Esto se debe a la gran cantidad de preferencias entre estudiantes, cursos y turnos, además del alto volumen de superposiciones, lo que incrementa considerablemente el número de variables y restricciones.

4.2.3. Análisis de métricas para cada instancia

En esta sección se presentan diversas métricas que son aplicadas a las instancias, y se analizan los resultados obtenidos para cada una de ellas. En la Sección *Indicadores generales de los resultados* se presentan algunos indicadores que analizan la solución desde diversas perspectivas. Luego, en la Sección *Indicadores sobre la calidad de las asignaciones* se examina el grado en que se satisfacen las preferencias en las asignaciones. Finalmente, en la Sección *Indicadores de equidad estudiantil* se analiza la equidad lograda entre los estudiantes.

Indicadores generales de los resultados

Cantidad de asignaciones: En todo problema de asignación, uno de los objetivos fundamentales es realizar una cantidad elevada de asignaciones. Para el modelo construido, este no es un objetivo explícito, ya que su propósito principal es maximizar las preferencias de los estudiantes. No obstante, se evalúa el número de asignaciones efectivamente realizadas entre estudiantes y cursos. Este valor es relevante, ya que proporciona un indicador de cuántos estudiantes fueron asignados a cursos de su preferencia.

Instancia	# Asignaciones	Máx. Asignaciones	E-C
E300-C4	423	423	1.200
E300-C10	395	395	3.000
E401-C94	5.988	7.357	28.200
E6457-C86	22.472	22.472	22.824
E7963-C94	25.328	26.085	108.263
E8208-C111	31.154	31.983	175.514

Tabla 4.35: Cantidad de asignaciones realizadas

Para poder determinar si los valores obtenidos son satisfactorios se desarrolla una variante de la Fase 1 del modelo, presentada en la Sección 3.1.1, donde se utilizan algunos conjuntos, parámetros y restricciones, pero se optimiza por la máxima cantidad de asignaciones[1]. Con este modelo se evalúan las mismas instancias, y se obtiene la cantidad de asignaciones realizadas, disponible en la columna *Máx. Asignaciones*.

Para las instancias *E300-C4*, *E300-C10* y *E6457-C86* se puede ver que el Modelo 3.1 alcanza la máxima cantidad de asignaciones posibles, y para las instancias *E7963-C94* y *E8208-C111* este alcanza un valor cercano al máximo, existiendo una diferencia de un 3 %. Sin embargo, para la instancia *E401-C94* la solución alcanza 5.988 asignaciones, quedando 1.370 asignaciones por debajo del máximo posible, lo cual representa un 18 % de diferencia entre ambos. Si bien las otras instancias tienen una diferencia menor, este tipo de escenarios son posibles debido al diseño del modelo. Este admite que el modelo pueda preferir realizar menos asignaciones de muy buena calidad frente a más asignaciones de calidad menor.

Sin embargo, los resultados presentados en la Tabla 4.35 permiten visualizar que esto no sucede en todas las instancias, y cuando ocurre no lo hace de forma significativa. En los casos donde se busque obtener valores más cercanos al máximo número de asignaciones posibles, se puede ajustar el valor de M de forma ascendente. Esto penalizará más la asignación a la lista de espera en la función objetivo (3.2), lo que a su vez incentivará al modelo a maximizar las asignaciones.

Además de comparar la cantidad de asignaciones realizadas con la cantidad máxima de asignaciones, se incluye en la Tabla 4.35 una columna con la cantidad de cursos preferidos por los estudiantes. Esto ayuda a contextualizar la cantidad de asignaciones porque representa el número ideal de estas. Este valor refleja un escenario ideal en el que todos los estudiantes son asignados a todos sus cursos preferidos.

En varias de las instancias se observa una diferencia significativa al comparar la cantidad máxima de asignaciones con el valor ideal de asignaciones. Esta se debe a que, aunque en un escenario ideal todos los estudiantes serían asignados a los cursos de su preferencia, en la práctica, esto no es posible debido a limitaciones como capacidades y superposiciones. Por ejemplo, un estudiante

[1]La formulación matemática de este modelo se encuentra en el Anexo B.2

que es asignado a un curso puede no ser asignado a otro curso preferido si los horarios de sus turnos se superponen. Asimismo, en cursos con alta demanda, la capacidad limitada de sus turnos puede impedir que todos los estudiantes interesados sean asignados. La combinación de estos factores genera una variedad de situaciones que impiden alcanzar el ideal de asignación.

Porcentaje de cupos libres: Un segundo indicador que ayuda a determinar si la capacidad de los turnos es una limitante en la solución obtenida es el porcentaje de cupos libres, que muestra cuántos cupos quedaron disponibles. En la Tabla 4.36 se presentan los porcentajes de cupos libres para cada instancia.

Instancia	Asignaciones	Cap total	Cupos libres (%)
E300-C4	423	1.122	62,30
E300-C10	395	395	0
E401-C94	5.988	72.522	91,74
E6457-C86	22.472	57.271	50,58
E7963-C94	25.328	58.711	56,86
E8208-C111	31.154	59.519	47,66

Tabla 4.36: Porcentaje de cupos libres

Este indicador aporta información valiosa principalmente en los casos extremos: aquellos cuyo valor es cercano a 0 o a 100.

En el caso que el porcentaje sea cercano a 0 indica claramente que la capacidad es una limitante. Un ejemplo de esto es la instancia *E300-C10*, la cual tiene un 0 % de cupos libres y la cantidad de asignaciones es significativamente menor al valor de E-C. Para esta instancia un incremento en la capacidad posiblemente permita que el modelo alcance una mejor solución con mayor cantidad de asignaciones.

Por otro lado, en aquellas instancias que tengan un alto porcentaje de cupos libres, la capacidad global no es un problema. Un ejemplo de esto es la instancia *E401-C94*, con un 94,74 % de cupos libres. Los dos motivos principales para tener un valor tan alto es que existen cursos con mucha oferta pero que no son preferidos por los estudiantes, o que existe otra limitante al momento de realizar la asignación, como por ejemplo las superposiciones. En esta instancia se tiene que en promedio cada estudiante escoge 70 cursos (de los 94 disponibles), por lo que la primera hipótesis queda descartada. En cambio, esta es la instancia con mayor cantidad de superposiciones, lo que hace más probable que estos sean la limitante.

Cantidad de estudiantes sin cursos asignados: Otra propiedad de interés en las soluciones del modelo es la cantidad de estudiantes que no han sido asignados a ningún curso. Este aspecto se relaciona directamente con uno de los objetivos del modelo, ya que en la Fase 2 se busca minimizar dicha cantidad. Por esto se calcula el número de estudiantes sin asignación a ningún curso tras la ejecución de las tres fases del modelo.

El resultado obtenido es que en todas las instancias no hay estudiantes que no hayan sido asignados. Esto es un resultado positivo, ya que garantiza que todos los estudiantes han sido considerados en la solución y se les ha asignado al menos un curso.

Es importante destacar que, aunque la solución final logra asignar a todos los estudiantes, este resultado no necesariamente se obtiene al término de la Fase 1. La Fase 1 se enfoca en minimizar las preferencias de los estudiantes en las asignaciones iniciales. Sin embargo, la Fase 2 juega un papel clave al mantener ese objetivo de minimización de preferencias mientras busca, además, asegurar que la mayoría de los estudiantes estén asignados a al menos un curso. De este modo, el modelo avanza hacia soluciones más equilibradas y completas, donde se optimizan múltiples objetivos. Dicho esto, los detalles de las soluciones parciales alcanzadas después de cada fase se pueden consultar en la Sección 4.2.4.

Porcentaje de asignación promedio por estudiante: A continuación, se llevará a cabo un análisis centrado en el estudiante, que se enfoca en el porcentaje promedio de asignaciones por estudiante. Para cada estudiante, se calcula el porcentaje de cursos asignados en función de sus preferencias, y se obtiene el promedio de estos valores. Los datos están presentados en la Tabla 4.37.

Esta medida puede ser de utilidad para ver qué tan satisfecho están los estudiantes con la asignación. Se puede suponer que si se tiene un porcentaje de asignación promedio muy alto, la mayoría de los estudiantes están satisfechos. En cambio, si el porcentaje es bajo, es necesario analizar la instancia para determinar la razón. Por ejemplo, los estudiantes pueden declarar preferencia de más cursos que los que realmente van a cursar, o puede haber una demanda insatisfecha por los cupos disponibles, entre otras razones.

Instancia	Asignación promedio p. est. (%)
E300-C4	35,25
E300-C10	12,17
E401-C94	40,33
E6457-C86	98,93
E7963-C94	36,59
E8208-C111	32,88

Tabla 4.37: Porcentaje de asignación promedio por estudiante

En un extremo se tiene la instancia *E6457-C86* con un 98,93 %, que muestra que casi todos los estudiantes son asignados a casi todos sus cursos. En el otro extremo se tiene la instancia *E300-C4* con un 12,17 %. En esta última instancia todos los estudiantes declaran preferencia sobre todos los cursos, y como se tiene una capacidad total de 395, es natural que cada estudiante sea asignado a pocos cursos. El resto de las instancias tienen valores entre 32 y 40 %, lo que significa que en promedio los estudiantes son asignados a aproximadamente un tercio de los cursos a los cuales establecen preferencia.

Listas de espera y ocupación por curso: En el siguiente indicador se cambia la perspectiva, pasando de la perspectiva del estudiante, a la perspectiva del curso. Como se mencionó anteriormente, el modelo busca asignar a los estudiantes a sus cursos preferidos, pero cuando esto no es posible, los asigna a su lista de espera. Por lo tanto, resulta importante analizar las listas de espera de los distintos cursos, ya que da una visión más completa sobre la solución obtenida, y da soporte a la toma de decisiones para mejorar escenarios no deseados.

En este indicador se presenta un gráfico de barras que muestra cuántos estudiantes hay en lista de espera para cada curso. Dichas barras son coloreadas con un color diferente según el porcentaje de ocupación del curso, que es el porcentaje de estudiantes asignados al curso sobre el cupo del mismo. Se utiliza un color más azulado cuando este porcentaje es bajo, y un color verdoso cuando es alto.

En la Figura 4.1 se muestra esta gráfica para la instancia *E300-C4*. En este caso se tiene que tres de los cuatro cursos tienen estudiantes en lista de espera, con cantidad de estudiantes similares. Además, todos presentan un alto nivel de ocupación. Como se puede ver en la Tabla 4.36, esta instancia tiene un 62,30 % de cupos libres. Si se analiza esto de forma aislada, se podría llegar a concluir que se tienen cupos extra que no son utilizados. Pero, al observar la gráfica, se puede apreciar como los cursos 2, 3 y 4 [2] tienen un elevado porcentaje de ocupación, se puede deducir que los cupos libres son del curso 1 ya que los demás cursos no tienen capacidad disponible según el color del gráfico.

Por lo tanto, analizar el porcentaje de cupos libres de forma exclusiva es insuficiente ya que es un resultado global que no toma las consideraciones individuales de los cursos. Por ejemplo, en caso que exista la posibilidad de ajustar los cupos disponibles, incrementar las capacidades de los cursos 2, 3 y 4 puede impactar positivamente en la solución, cosa que no sucede si se aumenta en el curso 1.

En la instancia *E300-C10*, graficada en la Figura 4.2, se muestra que todos los cursos tienen estudiantes en lista de espera, y que lo tienen en cantidades similares. Además, todos los cursos tienen 100 % de porcentaje de ocupación, lo que se corresponde con el 0 % de cupos libres presentado en la Tabla 4.36. Esta información confirma que la solución fue limitada fuertemente por la capacidad de los cursos.

[2] El numeral del curso corresponde con la posición del curso en la gráfica

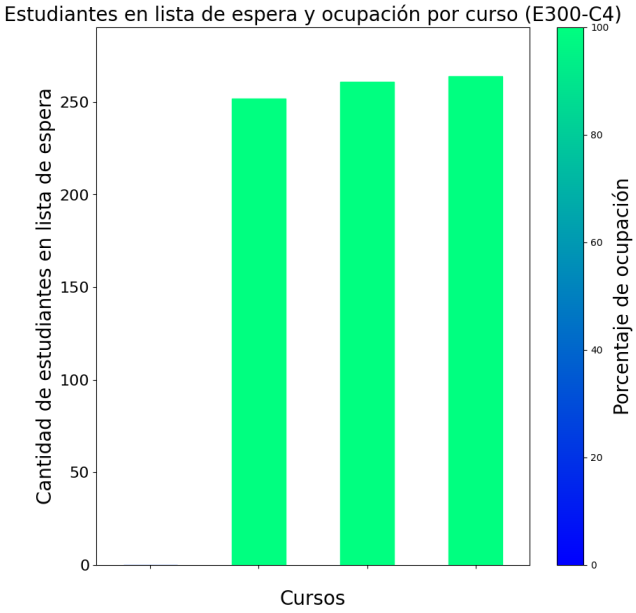

Figura 4.1: Estudiantes en lista de espera y ocupación por curso para instancia E300-C4

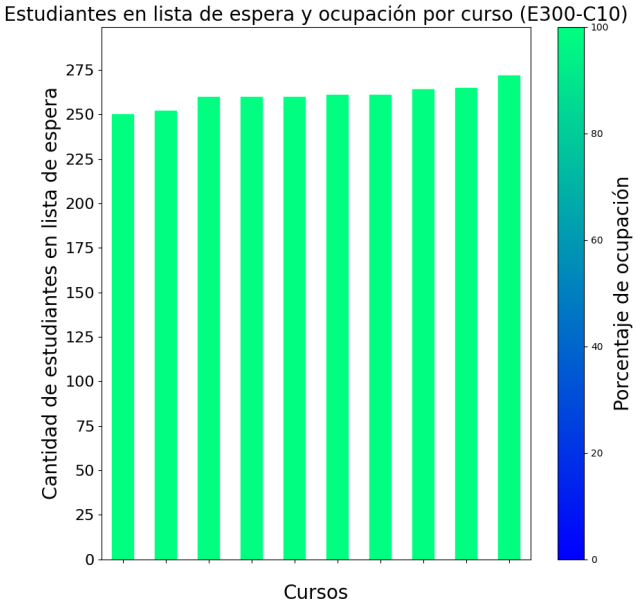

Figura 4.2: Estudiantes en lista de espera y ocupación por curso para instancia E300-C10

Para la instancia *E401-C94*, cuya gráfica está presente en la Figura 4.3, el número de cursos crece en gran medida, lo que dificulta realizar un análisis particular por curso, y es por este motivo que se analiza desde un panorama general.

Los cursos que tienen menos estudiantes en lista de espera coinciden en que tienen un nivel de ocupación bajo. Existe una concordancia entre estos sucesos, ya que una menor demanda de un curso reduce la probabilidad de que hayan estudiantes disponibles para inscribirse sin que el horario entre en conflicto con otros cursos de mayor preferencia. Esto resulta en que el modelo tiene menor cantidad de alternativas para poder asignar a estos estudiantes, lo que resulta en un porcentaje de ocupación bajo.

En los cursos con un bajo porcentaje de ocupación pero con más de 125 estudiantes en lista de espera, la alta demanda sugiere que la probabilidad de asignar estudiantes no debería ser un problema. Para estos cursos puede estar sucediendo alguna de las siguientes situaciones. La primera posibilidad consiste en que estos cursos sean elegidos por los estudiantes con baja prioridad, y además tengan un grado alto de superposición, de manera que si bien los estudiantes los eligen, la mayoría prefieren en mayor medida otros cursos que se superponen. En otra situación, los estudiantes seleccionan estos cursos con alta prioridad; sin embargo, debido a múltiples superposiciones en sus elecciones, el modelo opta por asignarlos a más de un curso superpuesto de menor prioridad, en la medida en que esto optimiza la función objetivo. Estas situaciones se presentan en los cursos con bajo porcentaje de ocupación que tienen entre 125 y 225 estudiantes en lista de espera (situados a la izquierda), y en los cursos con bajo porcentaje de ocupación con más de 275 estudiantes en lista de espera (situados a la derecha).

Luego, en la gráfica se ven también cursos con extensas listas de espera pero con un alto porcentaje de ocupación. Para aquellos donde el porcentaje es cercano al 100 %, se trata de cursos demandados, limitados por su capacidad. Aquellos otros cursos con un porcentaje más cercano al 80 % y cuya lista de espera supera los 250 estudiantes, dan indicio de tener un grado considerable de superposición.

En términos generales, el hecho de que la mayoría de los cursos tengan un bajo porcentaje de ocupación concuerda con la gran cantidad de superposiciones existentes en esta instancia, ya que esta tiene la mayor cantidad entre todas las instancias, 38.435, y un promedio de 44 superposiciones por turno. Esto dificulta asignar estudiantes a todos los cursos deseados, bajando los porcentajes de ocupación de los cursos.

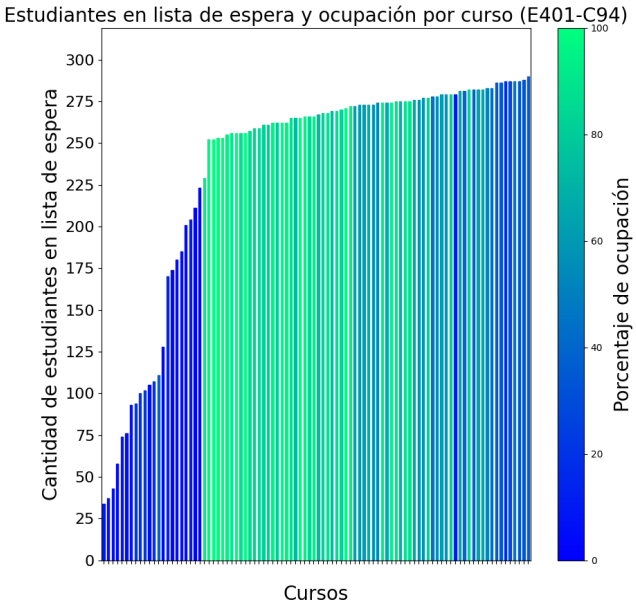

Figura 4.3: Estudiantes en lista de espera y ocupación por curso para instancia E401-C94

Para la instancia *E6457-C86*, cuya gráfica se muestra en la Figura 4.4, se puede visualizar que la mayoría de los cursos no tienen lista de espera. Los resultados indican que 72 cursos presentan una lista de espera vacía, y esto hace que se pierda mucha información en el gráfico. Sin embargo a continuación se analiza la información presentada.

El hecho de que pocos cursos tengan lista de espera se debe a que la solución presenta un alto porcentaje de asignación, alcanzando un 98 %, según los resultados de la Tabla 4.37. Este porcentaje señala que en la mayoría de los casos, el estudiante es asignado a sus cursos elegidos. Dicho suceso se debe a que existen pocas superposiciones con respecto a otras instancias, y en un contexto con buenas capacidades aumenta la probabilidad de alcanzar soluciones con estas características.

En cuanto a los 14 cursos que tienen lista de espera se puede ver que la mayoría tienen un alto porcentaje de ocupación, dado que tienen una mayor demanda que oferta. El curso atípico es el que se encuentra a la derecha del gráfico, que tiene una lista de espera significativamente más grande que el resto y no tiene un nivel de ocupación alto. Cabe destacar que esta tiene un tamaño de 273 estudiantes de un total de 6.457, lo que es un caso normal en las otras instancias de tamaño similar. Este curso es escogido por 1.018 estudiantes, y el modelo asigna a 751, un 73,77 %, pero como tiene una capacidad de 2.322, el porcentaje de ocupación es de 32,35 %.

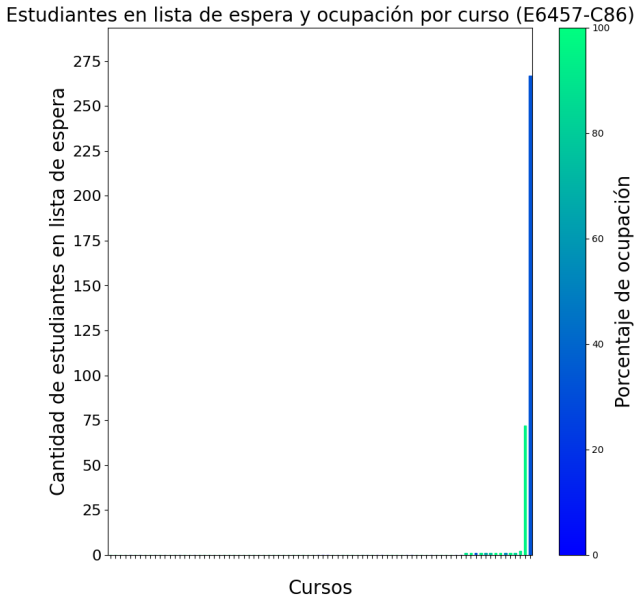

Figura 4.4: Estudiantes en lista de espera y ocupación por curso para instancia E6457-C86

Por otro lado, en la Figura 4.5 se ve este indicador para la instancia *E7963-C94*. La mayoría de los cursos tienen una lista de espera alta, de más de 1.000 estudiantes. Esto se puede explicar porque la capacidad total disponible, 58.711, es muy inferior a la demanda de los estudiantes, 108.263, como se muestra en la Tabla 4.33. Aunque las listas de espera sean extensas, casi todos los cursos tienen un porcentaje de ocupación alto. Si se compara con la instancia *E401-C94*, se puede ver que en esa se tienen más cursos con baja ocupación. Esto puede ser explicado porque la instancia *E7963-C94* tiene un promedio de 29 superposiciones con turnos, mientras que la instancia *E401-C94* tiene un promedio de 44, y tener menos superposiciones resulta en una asignación más sencilla, lo que significa que se pueden asignar más estudiantes.

En cambio, los cursos con lista de espera más chica tienen un porcentaje de ocupación menor. Este comportamiento se puede explicar de la misma manera que se explicó para la instancia *E401-C94*: a menor demanda, más difícil es completar el cupo del curso.

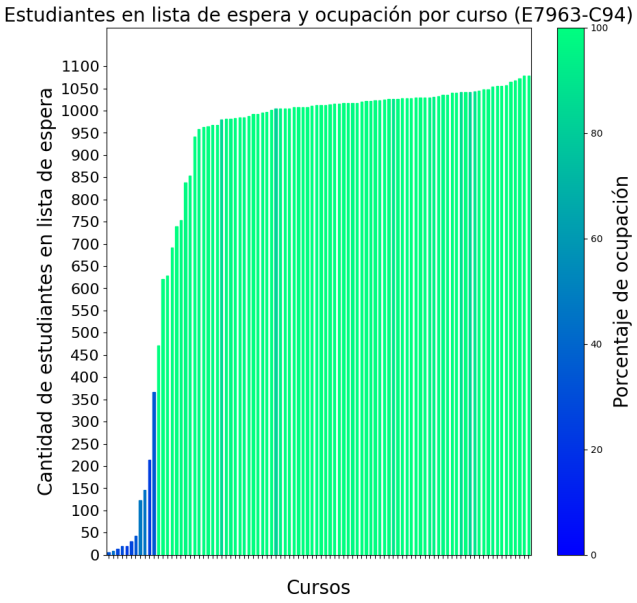

Figura 4.5: Estudiantes en lista de espera y ocupación por curso para instancia E7963-C94

Por último, se tiene la gráfica para la instancia *E8208-C111* en la Figura 4.6. Este caso es muy similar a los anteriores, los cursos con menor lista de espera tienen baja ocupación porque es difícil completar el cupo, los cursos con alta lista de espera tienen un alto nivel de ocupación, lo que significa que son muy demandados. Además las listas de esperas son extensas al igual que para la instancia anterior, debido a que la demandas de los estudiantes es sumamente superior a la capacidad general ofrecida.

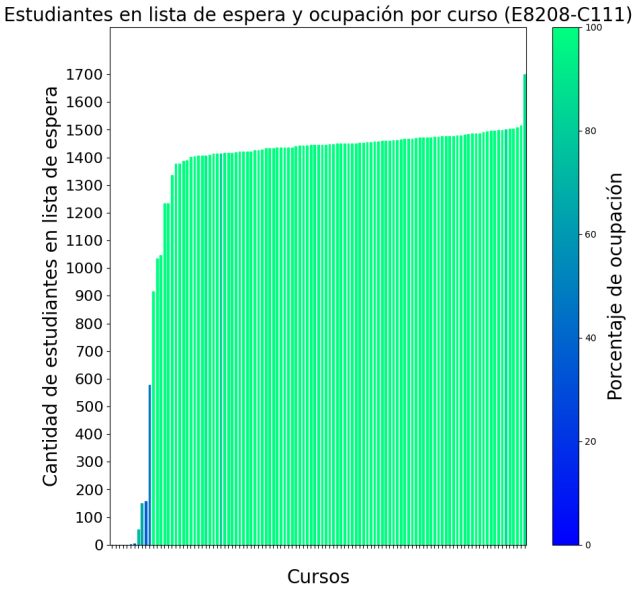

Figura 4.6: Estudiantes en lista de espera y ocupación por curso para instancia E8208-C111

Indicadores sobre la calidad de las asignaciones

En la sección anterior se presentaron indicadores generales que proporcionan información sobre los resultados obtenidos. En esta sección se presentan indicadores que evalúan la calidad de las asignaciones realizadas por el modelo.

Distribución de preferencias de curso en asignaciones: Una forma de evaluar la calidad de la solución es analizar la distribución de las preferencias de curso en las asignaciones realizadas. Específicamente, se debe examinar cómo se llevaron a cabo las asignaciones y cuál es la preferencia del curso *pc* que el estudiante registra para ese curso. Dado que uno de los objetivos del modelo es asignar a los estudiantes a sus cursos preferidos, este análisis permite evaluar en qué medida el modelo cumple con dicho objetivo. Para esto se realizan dos gráficos. El primero muestra la distribución de valores de *pc* presentes en las asignaciones, es decir se analizan todas las asignaciones estudiante-curso, y se registra el valor de *pc* que establece el estudiante sobre dicho curso. También se grafica la mejor asignación de curso por estudiante, es decir, qué valor de *pc* tiene el curso más preferido asignado al estudiante.

Para algunas instancias de gran escala, se puede observar que en su primer gráfico existen asignaciones a cursos con nivel de preferencia bajo. Además, en todos los casos el nivel más bajo coincide con la máxima cantidad de cursos elegidos por un estudiante. Si bien esto significa que se asigna por lo menos a un estudiante a la peor preferencia de curso posible, y a priori puede ser visto

de forma negativa, no es así. Aunque estas asignaciones no son realizadas a los cursos más preferidos por los estudiantes, igualmente son a cursos que desean asistir, por lo que asignarlos es una mejor opción que ubicarlos en lista de espera.

En la Figura 4.7 se muestra las gráficas mencionadas anteriormente para la instancia *E300-C4*. En este caso son 423 las asignaciones totales, y casi la mitad de estas, 211, son del curso más preferido por los estudiantes. Como se puede ver en la gráfica de la izquierda, el resto de las asignaciones se distribuye de forma casi por igual en el segundo, tercer y cuarto curso más preferido. Como se ve en la gráfica de la derecha, no hay estudiantes cuya mejor asignación es su cuarta elección de curso, lo que significa que todos los estudiantes fueron asignados a por lo menos su tercera elección.

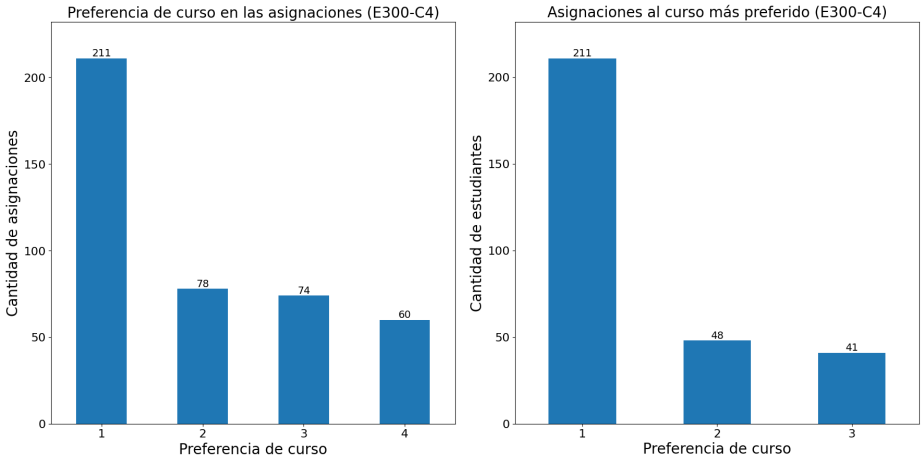

Figura 4.7: Valores de prioridad de curso en las asignaciones y mejor asignación de curso a estudiantes, instancia E300-C4

Luego, para la instancia *E300-C10*, cuyas gráficas se encuentran en la Figura 4.8, todos los estudiantes fueron asignados a su primera elección, lo que es muy favorable para ellos. Además, el resto de las asignaciones, que fueron 95, son hechas a la segunda elección, y no hay asignaciones para valores de preferencia de curso más bajas.

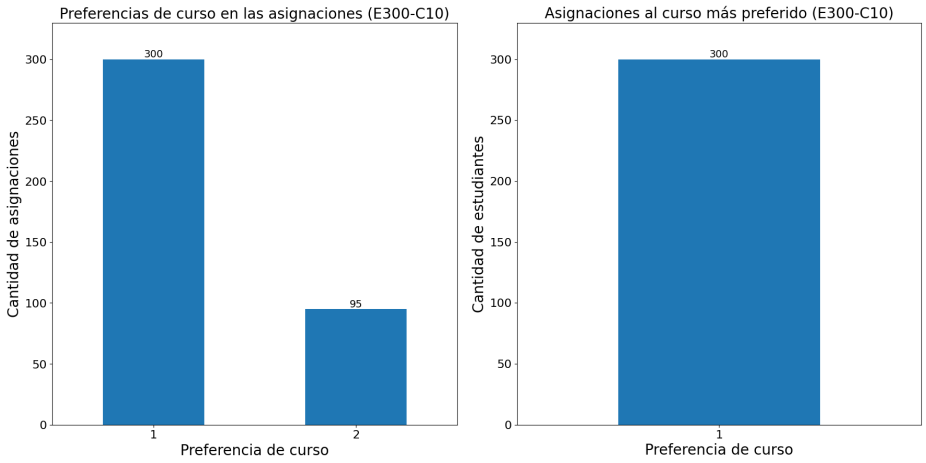

Figura 4.8: Valores de prioridad de curso en las asignaciones y mejor asignación de curso a estudiantes, instancia E300-C10

La instancia *E401-C94* es de mayor dimensión que las instancias analizadas previamente y sus gráficos se presentan en la Figura 4.9.

El gráfico de la izquierda muestra que las asignaciones están distribuidos en cursos que van hasta su elección número 94. Debido a la gran cantidad de elecciones de curso que hacen los estudiantes, el modelo tiene una mayor cantidad de alternativas para asignarlos a una mayor cantidad de estos, aunque no sean necesariamente de sus primeras opciones. Dicho gráfico muestra una tendencia decreciente en la cantidad de asignaciones, lo que indica que la mayor cantidad de elecciones se encuentran en cursos de mayor preferencia, y luego cada vez menos se asignan a cursos de menor preferencia. Esto es positivo ya que ante elecciones en conflicto, el modelo tiende a asignar aquellas con mejor valor de preferencia de curso.

Luego, en el gráfico de la derecha se puede ver que el rango de preferencias de curso es significativamente más chico que en la gráfica anterior. Aunque hayan asignaciones realizadas al curso número 94 de un estudiante, esta gráfica indica que no es la mejor asignación de este, sino que todos los estudiantes son asignados al menos a su decimaquinta opción. Además, se tiene que 384 de 401 estudiantes son asignados hasta su cuarta opción, y 256 de estos son a su primera.

Al igual que el caso anterior, en la instancia *E6457-C86* se tiene una gráfica de distribución de preferencia de curso con tendencia decreciente, como se puede ver en la Figura 4.10. La mayor cantidad de las asignaciones se concentran entre cursos con preferencia de hasta valor 8. Luego, en el gráfico de la derecha se muestra que todos los estudiantes son asignados a su mejor opción de curso.

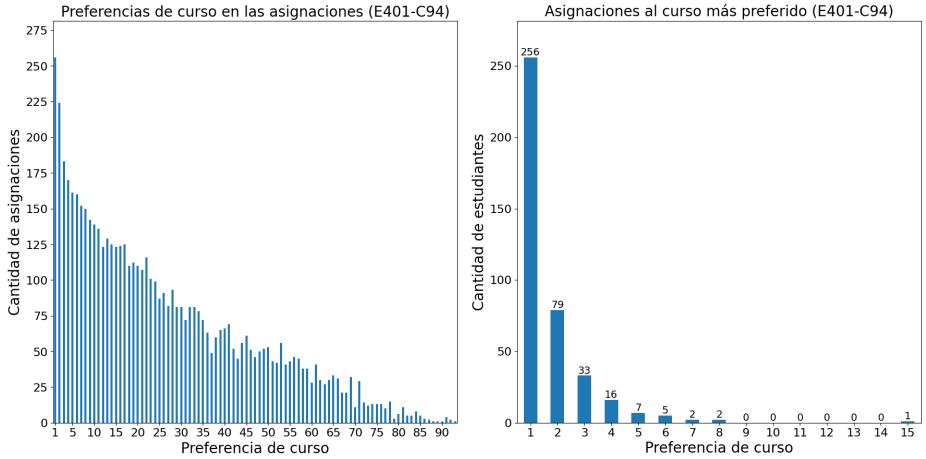

Figura 4.9: Valores de prioridad de curso en las asignaciones y mejor asignación de curso a estudiantes, instancia E401-C94

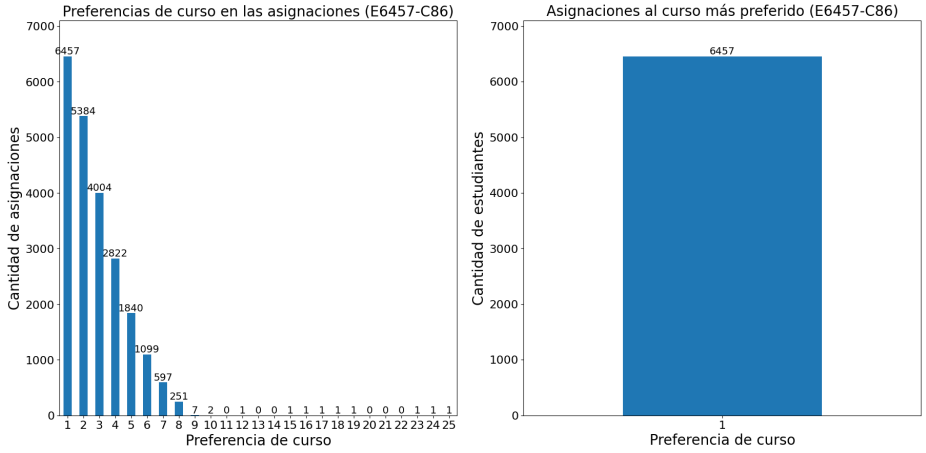

Figura 4.10: Valores de prioridad de curso en las asignaciones y mejor asignación de curso a estudiantes, instancia E6457-C86

En la Figura 4.11 se presentan los gráficos para *E7963-C94*. El primer gráfico, ubicado a la izquierda, no muestra una tendencia decreciente tan marcada como las instancias anteriores, ya que se puede ver un pico en el valor 15. Esto puede parecer contraintuitivo, dado que el modelo busca asignar a las mejores elecciones. A pesar de esto, una de las razones por la que puede suceder es que el modelo elija realizar más asignaciones netas de calidad menor (con valores entre 13 y 17), a realizar menos asignaciones de calidad mayor (con preferencia menor a 12).

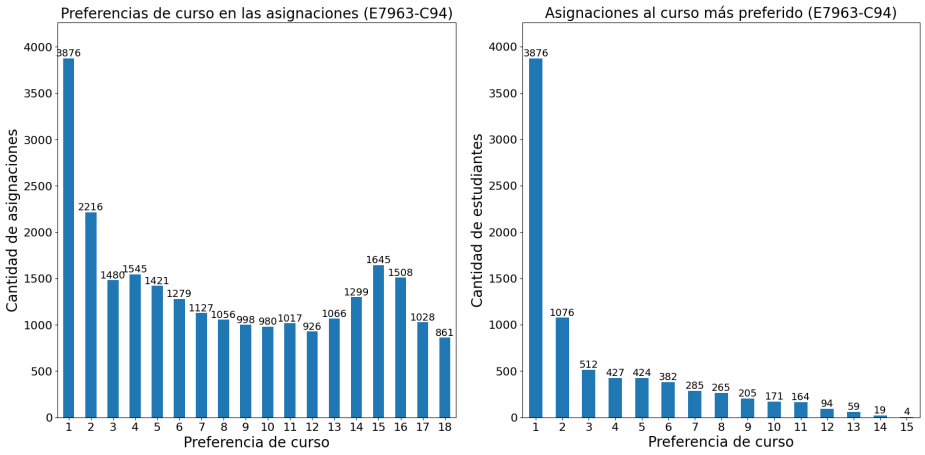

Figura 4.11: Valores de prioridad de curso en las asignaciones y mejor asignación de curso a estudiantes, instancia E7963-C94

Las gráficas para la instancia *E8208-C111* se muestran en la Figura 4.12, y son muy similares a *E7962-C94*. Para empezar, las asignaciones se distribuyen entre las primeras 30 ya que hay estudiantes que eligen hasta 30 cursos. Además la distribución de las asignaciones es también similar marcando un pico en el valor 27. En referencia al gráfico de la derecha se muestra que la mayoría de los estudiantes, 5560 de 8208, son asignados a alguna de sus primeras tres opciones.

Para concluir, las gráficas presentadas demuestran que el comportamiento práctico del modelo se alinea con las expectativas deseadas. En general, la mayoría de los estudiantes son asignados a cursos dentro de sus principales preferencias. De manera consistente, se observa que por lo menos un 67 % de los estudiantes recibe alguna de sus primeras 3 opciones. Las tendencias decrecientes en la distribución de asignaciones reflejan la preferencia del modelo por asignar estudiantes a cursos con un alto valor de *pc*. Además, los valles observados en algunos gráficos indican un comportamiento esperado del modelo, que tiende a preferir realizar muchas asignaciones a cursos de calidad moderada en lugar de pocas asignaciones a cursos de excelente calidad. Este comportamiento se vuelve más pronunciado debido a la significativa penalización aplicada a las asignaciones que resultan en la lista de espera, ya que cada vez que no se realiza una asignación afecta considerablemente el valor objetivo del modelo.

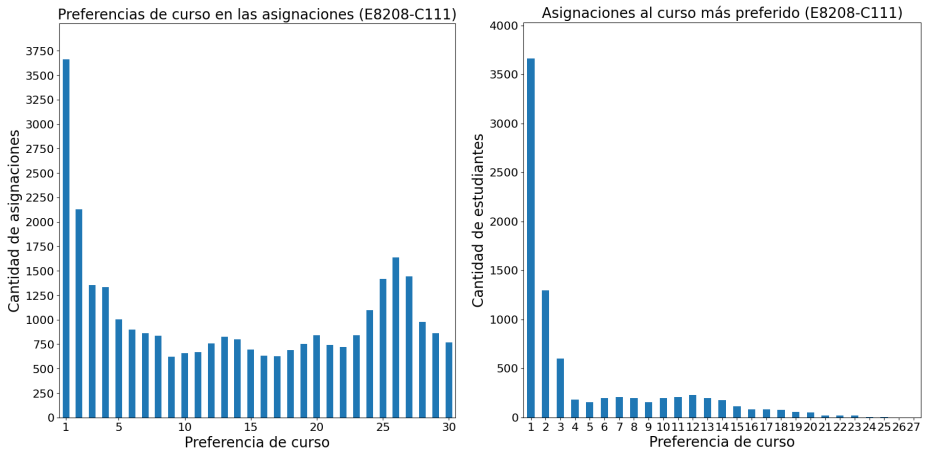

Figura 4.12: Valores de prioridad de curso en las asignaciones y mejor asignación de curso a estudiantes, instancia E8208-C111

Análisis general de preferencias en asignaciones: Además de analizar las asignaciones tomando en cuenta la distribución de las preferencias de curso, se puede hacerlo mirando la preferencia combinada y las preferencias de turno. En la Tabla 4.38 se presenta para cada instancia el promedio, mínimo y máximo de preferencia combinada en las asignaciones. La preferencia combinada, como es la definida en la ecuación (3.1), es dependiente del parámetro K. Como este parámetro depende de la instancia, es necesario conocerlo para poder comparar los valores de preferencia combinada entre estas, y por lo tanto se lo incluye en la tabla. Además, en esta tabla se presenta el promedio de preferencia de curso pc y el promedio de preferencia de turno pt en las asignaciones.

		Preferencia combinada				
Instancia	K	Prom.	Mín.	Máx.	Prom. pc	Prom. pt
E300-C4	13	26,53	14	53	1,96	1,05
E300-C10	13	17,22	14	28	1,33	1,17
E401-C94	15	388,78	16	1411	25,76	2,37
E6457-C86	39	110,15	40	976	2,79	1,23
E7963-C94	39	317,10	40	706	8,09	1,53
E8208-C111	39	546,50	40	1174	13,97	1,69

Tabla 4.38: Promedio, mínimo y máximo de preferencia combinada, promedio de prioridad de curso, promedios de prioridad de turno

En todas las instancias analizadas, el mínimo de la preferencia combinada corresponde a $K + 1$, lo que indica que la mejor asignación siempre se realiza a un curso-turno con $pc = 1$ y $pt = 1$, es decir, al turno más preferido del curso más deseado por el estudiante. Este resultado no es sorprendente porque

el modelo está diseñado para minimizar las preferencias asignadas, aunque sigue siendo importante confirmarlo. Por otro lado, la asignación con la preferencia combinada más alta se da en *E401-C94*, con un valor de 1.411, que corresponde a un curso con preferencia de valor 94 y un turno con preferencia de valor 1.

Como se observa en la columna *Prom. pc*, el promedio de preferencia de curso varía significativamente. El promedio más bajo se encuentra en *E300-C10*, con un valor de 1,33. En esta instancia, los estudiantes son asignados a su primer o segundo curso preferido, como se muestra en 4.8, lo que justifica este bajo promedio.

La siguiente instancia con el promedio de *pc* más bajo es *E300-C4*, con un valor de 1,96. Este resultado es esperado dado que, al haber solo 4 cursos disponibles, las asignaciones tienden a ser a cursos con niveles de preferencia bajos. La siguiente instancia con un promedio bajo es *E6457-C86*, con un valor de 2,79. Este valor es coherente con la distribución de *pc* favorable al estudiante observada en 4.10, ya que casi todas las asignaciones se sitúan entre el primer y el octavo curso más preferido.

En el otro extremo, *E401-C94* muestra el promedio de preferencia de curso más alto, llegando a un valor de 25,76. En primer lugar, esta instancia es en la que más elecciones de curso establecen los estudiantes, llegando a tener estudiantes con 94 elecciones, lo que permite que existan valores de *pc* más altos. Además, este resultado es coherente con las dificultades anteriormente comentadas que presenta esta instancia para realizar buenas asignaciones, como el alto nivel de superposiciones.

A diferencia del promedio de preferencia de curso, el promedio de preferencia de turno varía significativamente menos. Esto se debe en parte a la limitación en la elección de turnos, ya que los estudiantes pueden elegir hasta 4 turnos. En todas las instancias, excepto una, el promedio de preferencia de turno se encuentra entre los valores 1 y 2, lo que indica que las asignaciones en términos de turno son generalmente muy adecuadas para los estudiantes. La única instancia que supera el valor 2 es *E401-C94*, con un promedio de valor 2,37.

Indicadores de equidad estudiantil

Un factor importante a medir para el problema descrito en la Sección 3.1 es la equidad estudiantil, ya que una solución satisfactoria involucra cierto grado de justicia entre los estudiantes. Esto refiere a que todos los estudiantes sean considerados de la misma manera por el modelo, es decir que evite caer en situaciones donde algunos estudiantes sean altamente favorecidos en comparación con otros.

Equidad en felicidad: El primer indicador que se define evalúa la equidad en las asignaciones a los estudiantes en función de sus preferencias. Anteriormente, se analizó la distribución de las preferencias en las asignaciones, pero no se considera a qué estudiante se le asignaban esos cursos. En esta etapa, el enfoque se centrará en el estudiante, examinando tanto las preferencias de curso y turno en las asignaciones recibidas, como los cursos en los que quedó en lista de espera. Para ello, se define la felicidad estudiantil para un estudiante e, deno-

tada como F_e, que se conforma del cociente entre la satisfacción del estudiante obtenida y el máximo de satisfacción que este podría obtener.

Previo a presentar la formulación de F_e, es necesario definir la felicidad que le aporta a un estudiante ser asignado a un turno de curso. Para esto, se utiliza el valor de preferencia combinada del estudiante e sobre el turno t del curso c, p_comb_{ect}, como es definido en (3.1). Dicho esto, la felicidad que le aporta al estudiante e la asignación al turno t del curso c, denotada como f_{ect}, queda de la siguiente manera:

$$f_{ect} = \frac{1}{p\_comb_{ect}} \tag{4.2}$$

Además, también es necesario definir cuál es el máximo de felicidad que un estudiante podría tener si es asignado al curso. Para esto, se utiliza la prioridad combinada del turno más preferido de cada curso. De esta forma, el máximo de felicidad que el estudiante e puede obtener para el curso c es denotado como f_max_{ec} y queda como:

$$f\_max_{ec} = \frac{1}{min_{t \in TC_c}\{p\_comb_{ect}\}} \tag{4.3}$$

Además, se define el conjunto A_e de parejas de curso-turno a las que es asignado el estudiante e.

Una vez se presentados estos conceptos, se define la felicidad de un estudiante e de la siguiente manera:

$$F_e = \frac{\sum_{(c,t) \in A_e} f_{ect}}{\sum_{c \in CP_e} f\_max_{ec}} \tag{4.4}$$

La función de felicidad F_e tiene como característica que si el estudiante es asignado a todos sus cursos preferidos, entonces $F_e = 1$; si no es asignado a ningún curso, $F_e = 0$; y toma un valor intermedio si es asignado a alguno de sus cursos. Además, la felicidad del estudiante aumenta proporcionalmente a la asignación de cursos o turnos más preferidos por él.

Una vez definida la felicidad por estudiante, se puede evaluar la equidad en la felicidad de todos ellos. En un escenario ideal, todos los estudiantes tienen niveles de felicidad similares, lo que resulta en una gran equidad. Una medida que evalúa la disparidad entre los valores de una distribución es la desviación estándar. Con el fin de poder comparar los resultados entre distintas instancias, se divide la desviación estándar entre la media. Dicho cociente es conocido en la literatura como el cociente de variación. A partir de este, se define la equidad en la felicidad de la siguiente manera:

$$EquidadFelicidad = 1 - \frac{DesviaciónEstándar(F_1, F_2, ...)}{Media(F_1, F_2, ...)} \tag{4.5}$$

En el caso ideal, donde los valores de F_e son todos iguales, lo que significa que todos los estudiantes tienen la misma felicidad, la desviación estándar de esta

distribución es igual a 0, y por lo tanto la equidad en felicidad alcanza su valor máximo, que es 1.

En la Tabla 4.39 se muestra la equidad en felicidad para todas las instancias.

Instancia	Equidad en felicidad
E300-C4	0,6069
E300-C10	0,7098
E401-C94	0,4687
E6457-C86	0,9691
E7963-C94	0,2080
E8208-C111	0,1166

Tabla 4.39: Equidad en felicidad por estudiante, para todas las instancias

Para realizar un análisis de los resultados de equidad obtenidos, clasificamos los resultados en tres franjas: aquellas instancias las cuales obtuvieron una equidad por encima de 0,6, las que tienen equidad entre 0,4 y 0,6, y aquellas que se encuentran por debajo de 0,4.

En la Tabla 4.39 se puede ver como las instancias *E300-C4*, *E300-C10* y *E6457-C86* tienen un muy buen nivel de equidad superando el valor de 0,6. Esto se traduce en que la mayoría de los estudiantes fueron tomados en cuenta por el modelo de manera igualitaria. Para las instancias más pequeñas (*E300-C4* y *E300-C10*) sus valores de equidad son similares, lo que se condice con los datos presentados en las Figuras 4.7 y 4.8, las cuales muestran que las asignaciones se distribuyen entre los cursos más prioritarios. Adicionalmente, en ambas se logra que la mayoría de los estudiantes tengan como su asignación más prioritaria aquellas de primer o segunda prioridad. A su vez, aproximadamente el 80 % de los estudiantes están en al menos alguna lista de espera ya que las listas de espera son de casi 250 estudiantes. Lo mencionado lleva a que exista un buen nivel de equidad en términos generales ya que no existen estudiantes completamente favorecidos y otros demasiado desfavorecidos.

En el caso de la instancia *E6457-C86* se obtuvo un valor de 0,96 de equidad, por lo que todos los estudiantes fueron satisfechos de forma muy similar, lo cual se alinea con el 98,93 % de asignación promedio por estudiante que se muestra en la Tabla 4.37. En esta instancia se logra que en promedio los estudiantes sean asignados a casi todas sus elecciones logrando listas de espera muy pequeñas. Además todos los estudiantes fueron asignados a su curso más preferido. Toda esta información condice con que todos los estudiantes fueron satisfactoriamente considerados, logrando buenas asignaciones para todos, y por lo tanto equidad en estas.

Las instancias *E7963-C94* y *E8208-C111* tienen un muy bajo nivel de equidad el cual se refleja en los resultados del indicador *Distribución de preferencias de curso en asignaciones*. Por ejemplo, para la instancia *E8208-C111*, en la Figura 4.12 se puede ver que existen aproximadamente 5.560 estudiantes cuya mejor asignación son cursos de preferencia uno, dos o tres, mientras que existen 1.549 estudiantes cuya mejor asignación son preferencias de 10 o más. Esto

habla de cierta disparidad entre grupos grandes de estudiantes, lo que se alinea con el resultado de equidad.

Luego, para la instancia *E7963-C94* se tiene un mayor nivel de equidad, pero igualmente es bajo. En este caso en la Figura 4.11 se puede ver algo muy similar a la instancia anterior, donde un grupo de 1.648 de estudiantes tiene su mejor asignación a cursos de preferencia uno, dos o tres mientras que para casi la misma cantidad de estudiantes su mejor asignación es de 6 o más. Esto indica una disparidad menor pero que igualmente existe y es reflejada en el bajo valor de equidad.

Por último, la instancia *E401-C94* tiene un nivel de equidad intermedio, que indica que los estudiantes obtienen asignaciones más equilibradas que los casos descritos en el párrafo anterior, pero no lo suficiente como para que estos tengan niveles de felicidad similares.

A modo de conclusión, los resultados obtenidos en la equidad en felicidad mostraron cierta disparidad en las distintas instancias. Mientras que algunas presentaron valores altamente positivos, otras evidenciaron desempeños significativamente inferiores.

Equidad en porcentaje de asignación: Otra manera de evaluar la equidad entre los estudiantes es ignorar sus preferencias individuales y centrarse únicamente en cuántos de los cursos seleccionados les han sido asignados. Por este motivo, en esta sección se analiza la equidad en la distribución de porcentaje de asignación por estudiante. Definiendo el porcentaje de asignación del estudiante e como:

$$PA_e = 100\frac{\sum_{c \in CP_e} \sum_{t \in TP_{ec}} x_{ect}}{|CP_e|} \tag{4.6}$$

La formulación de la equidad en porcentaje de asignación queda como sigue:

$$EquidadPA = 1 - \frac{DesviaciónEstándar(PA_1, PA_2, ...)}{Media(PA_1, PA_2, ...)} \tag{4.7}$$

En la Tabla 4.40 se presenta la equidad en porcentaje de asignación por estudiante, para todas las instancias.

Instancia	Equidad porc. asignación
E300-C4	0,65060
E300-C10	0,64611
E401-C94	0,14238
E6457-C86	0,95142
E7963-C94	0,19577
E8208-C111	0,09144

Tabla 4.40: Equidad en porcentaje de asignación, para todas las instancias

Análogamente a la equidad en felicidad, valores más cercanos al 1 indican que la instancia presenta una mayor equidad, y más cercanos al 0 una menor.

En particular, se puede ver que las instancias *E300-C4*, *E300-C10* y *E6457-C86* son las que mayor equidad tienen, lo que se condice con los resultados de estas instancias en el indicador previo.

Las instancias *E7963-C94* y *E8208-C111* son las que obtienen el menor nivel de equidad en porcentaje de asignación. Además, estas instancias son las que tienen un peor nivel de equidad en felicidad, como se mencionó anteriormente. Por lo tanto, se puede concluir que hay disparidad entre los estudiantes de estas instancias, ya que no se logró alcanzar equidad en ninguna de las dos formas de evaluación.

En cambio, en la instancia *E401-C94* existe una aparente contradicción entre los resultados obtenidos. Por un lado, tiene una baja equidad en porcentaje de asignación, pero no así según el indicador previo. El motivo de esta aparente contradicción es que los indicadores miden la equidad con enfoques distintos, el primero utiliza la satisfacción de los estudiantes mientras que el segundo utiliza los porcentajes de cursos a los que se asigna el estudiante. Según el porcentaje de cursos obtenido se puede decir que existen varios estudiantes que son asignados a muy pocos cursos, frente a otros que son asignados a muchos cursos. Sin embargo, el nivel de equidad en felicidad obtenido no es tan bajo, por lo que la felicidad entre los estudiantes es similar. A partir de estos hechos, se deduce que los estudiantes con pocas asignaciones son asignados a cursos de alta preferencia, mientras que aquellos asignados a muchos cursos son asignados a cursos de baja preferencia. De esta manera, se equilibra la felicidad de estudiantes con porcentajes de asignación dispar.

Como conclusión de ambas equidades, se obtienen resultados dispares. Cuatro de las seis instancias tienen un nivel de equidad en felicidad medio a alto, mientras que dos lo tienen bajo. Por otro lado, tres instancias tienen un nivel alto de equidad en porcentaje de asignación, que coinciden con tres de las cuatro instancias que tienen alto nivel de equidad en felicidad. Por lo tanto, tenemos tres instancias con alto nivel de equidad según ambos enfoques, y una con alto nivel únicamente en equidad de felicidad. Esto indica que en la mayoría de las instancias se alcanza un grado de equidad satisfactorio, balanceando felicidad y porcentaje de asignación entre los estudiantes.

Asimismo, las instancias *E7963-C94* y *E8208-C111*, que son las que tienen equidad baja visto desde ambos enfoques, tienen una disparidad alta entre los estudiantes, siendo unos favorecidos más que otros. Las causas pueden ser muy variadas debido a la complejidad del problema. Además, siendo instancias de gran porte, resulta difícil analizar los motivos de dichos resultados. Cabe destacar que el objetivo principal del modelo es maximizar las preferencias estudiantiles, y este objetivo condiciona fuertemente la asignación final.

Por último, ambos indicadores nos dan una medida de equidad en la solución pero su combinación nos da una visión más completa sobre lo que ocurre en las instancias.

4.2.4. Comparación entre los resultados parciales de las fases

Hasta ahora se ha realizado el estudio de la solución final del modelo, obtenida luego de ejecutar las tres fases. En esta sección se toma un enfoque distinto al centrarse en la evaluación de la solución intermedia luego de las Fases 1 y 2, con el objetivo de evaluar cómo progresan distintas métricas a lo largo de la ejecución. Se hace énfasis en métricas asociadas a los objetivos de la Fases 2 y 3, ya que se quiere ver si estas fases cumplen los objetivos establecidos.

El objetivo de la Fase 2 es minimizar la cantidad de estudiantes que no están asignados a ningún curso. Como se mencionó anteriormente, en la solución final todos los estudiantes en todas las instancias son asignados a por lo menos un curso. Para evaluar la evolución de esta métrica, se grafica en la Figura 4.13 el porcentaje de estudiantes sin ser asignados a ningún curso al terminar las Fases 1, 2 y 3 en cada instancia.

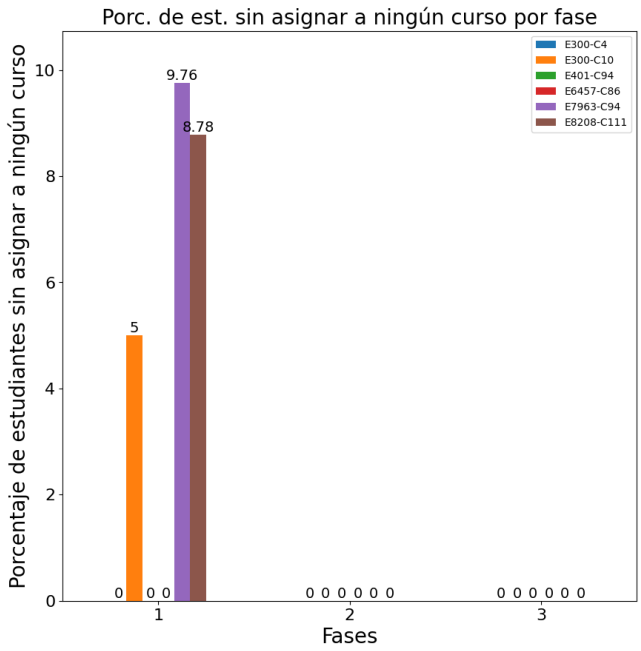

Figura 4.13: Porcentaje de estudiantes sin ser asignados a ningún curso, analizado por fase

Al finalizar la Fase 1, tres de las seis instancias tienen estudiantes sin ningún curso asignado, alcanzando un 9,76 % en la instancia *E7963-C94*, lo que corresponde a 777 estudiantes. Sin embargo, luego de que se ejecute la segunda fase, que tiene como objetivo minimizar este número, el valor en todas las instancias desciende a 0, lo que comprueba que la Fase 2 lo cumple. Cabe destacar que

se obtiene este resultado utilizando el parámetro α con valor 1, por lo que en la Fase 2 no se modifica el valor obtenido en la función objetivo de la Fase 1. Es decir, la Fase 2 busca, dentro del mismo espacio de soluciones óptimas, una solución alternativa que no deje a ningún estudiante sin asignaciones.

A continuación se analiza la evolución de aspectos relacionados a la equidad a lo largo de la ejecución.

En primer lugar se analiza la evolución del valor máximo de la insatisfacción estudiantil (definida en (3.1.3)) a lo largo de las fases. Este valor representa al estudiante con mayor insatisfacción, por lo que si se disminuye, se reduce la brecha entre los estudiantes mejor y peor asignados. Además, como la Fase 3 tiene como objetivo reducir este valor, resulta relevante estudiar el alcance de dicha reducción. En la Tabla 4.41 se muestran estos valores para todas las instancias.

	Máx. insatisfacción estudiantil		
Instancia	Fase 1	Fase 2	Fase 3
E300-C4	15.487	15.487	14.278
E300-C10	204.500	185.650	185.650
E401-C94	167.040.706	167.385.373	156.281.198
E6457-C86	86.215.219	86.215.219	86.215.219
E7963-C94	247.086.468	233.857.878	206.845.449
E8208-C111	572.382.750	554.681.601	499.124.376

Tabla 4.41: Máxima insatisfacción estudiantil, analizado por fase

En todas las instancias menos en *E6457-C86*, se obtiene una mejoría. Cabe destacar que en la instancia *E6457-C86* se efectúan asignaciones con alto nivel de equidad, como se analiza en las Tablas 4.39 y 4.40, por lo que se puede concluir que estas asignaciones equitativas son resultado de la Fase 1 y no por las otras fases, ya que se obtuvo el valor óptimo en la primera fase.

Por el contrario, los cambios más grandes se encuentran en las instancias *E7963-C94* y *E8208-C111*, donde se efectúa una mejora del 16 % y 13 % respectivamente.

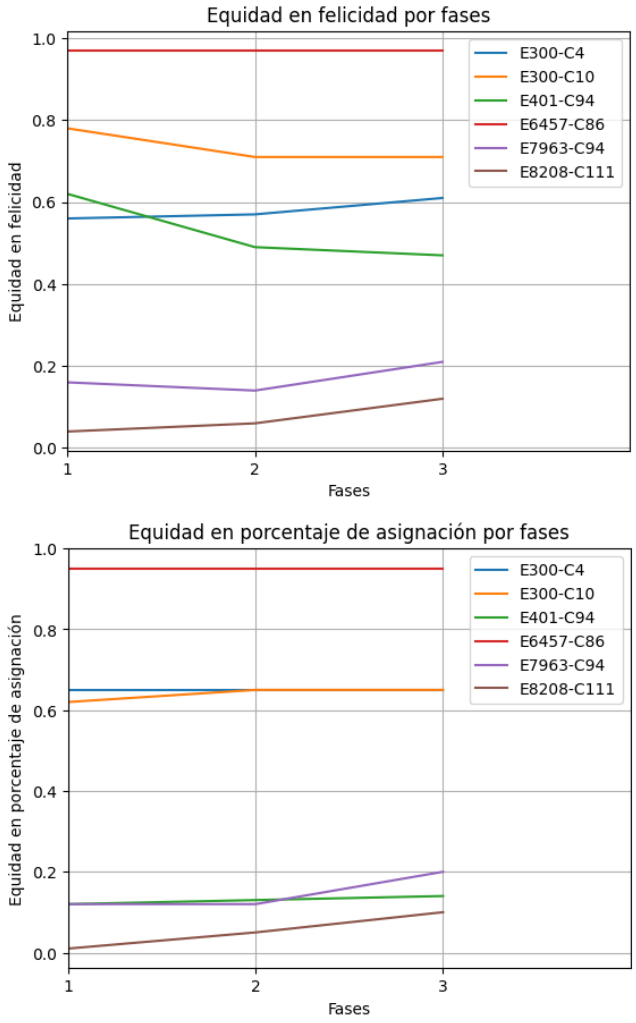

Figura 4.14: Indicadores de equidad por fase

En la Figura 4.14 se muestran dos gráficos, el primero muestra cómo varía la equidad desde el enfoque de la felicidad de los estudiantes, mientras que el segundo muestra la variación de la equidad de porcentaje de asignación.

En los gráficos se puede ver que para las instancias *E7963-C94* y *E8208-C111* la equidad mejora en ambos aspectos, es decir, la Fase 3 logra que los estudiantes sean asignados a proporciones de cursos similares y que las preferencias de estos sean de calidad similar. Estas instancias son las que tienen valores de equidad bajos en ambas perspectivas, pero se puede ver que los resultados de las Fases 1 y 2 tienen niveles muy bajos de equidad, y la Fase 3 logra mejorarlas.

Luego, la instancia *E300-C4* a pesar de ser una instancia pequeña con baja

flexibilidad, logra mejorar la equidad en felicidad mientras que mantiene los buenos niveles de equidad en porcentaje de asignación obtenidos en las fases anteriores. Esto significa que para dicha instancia las fases permiten mejorar la justicia en cuanto a niveles de preferencia sin alterar el hecho que los estudiantes sean asignados a proporciones de cursos similares.

Para la instancia *E6457-C86* no se obtienen mejoras en ninguno de los dos aspectos, pero esto se debe a que ya cuenta con excelentes niveles de equidad luego de la Fase 1.

Aunque la mayoría de las instancias mencionadas muestran mejoras tras las diferentes fases, en algunos casos el impacto positivo no es tan evidente. Por ejemplo, en las instancias *E300-C10* y *E401-C94*, se observan ligeros retrocesos en uno de los dos aspectos evaluados. En el caso de la primera, si bien se mantiene la justicia en cuanto a la proporción de cursos asignados a los estudiantes, se reduce la equidad en función de la felicidad. Al considerar que en esta instancia el modelo logra reubicar a estudiantes que se encontraban en lista de espera durante la Fase 2, es posible inferir que dicha reubicación se realiza a costa de una disminución en los niveles de equidad. Además, dado que en la Fase 3 se debe mantener esta reubicación para no afectar el valor objetivo alcanzado en la Fase 2, el nivel de equidad obtenido en la Fase 1 no puede recuperarse, ni se consigue reducir la insatisfacción máxima de los estudiantes.

En la instancia *E401-C94* sucede algo similar. En dicha instancia la Fase 1 ya alcanza la optimalidad en términos de Fase 1 y Fase 2, es decir, logra que todos los estudiantes sean asignados a por lo menos un curso. Por este motivo, la ligera disminución en la equidad en felicidad puede explicarse por el hecho que la Fase 2 encuentra otra solución con el mismo valor de función objetivo pero con peor nivel de equidad. Sin embargo, esta segunda solución no modifica significativamente la equidad entre los estudiantes respecto a la proporción de cursos a los que se los asigna. Finalmente, en una tercera fase se mejora el objetivo de Fase 3 pero este no tiene un impacto relevante en ninguna de las dos equidades; la equidad en felicidad disminuye ligeramente y la equidad por porcentaje de asignación aumenta ligeramente. Este tipo de resultados son posibles ya que la función objetivo y los indicadores de equidad no son iguales.

Como conclusión, se tiene que la Fase 3 logra una mejora en la equidad en felicidad en tres de las seis instancias, y una mejora o mantención de la equidad en porcentaje de asignación en todas las instancias. Sin embargo, los incrementos de equidad no son significativos, lo que indica que la Fase 3 no es eficaz para alcanzar buenos niveles de equidad bajo las perspectivas establecidas. Una posible razón para esta limitación es que la Fase 3 no está diseñada para optimizar los valores de equidad determinados como criterio.

4.2.5. Análisis del parámetro α

En el modelo se utiliza el parámetro α para flexibilizar el valor objetivo obtenido en la Fase 1 durante las fases subsiguientes, con el propósito de obtener un mejor resultado en estas. En los resultados mostrados anteriormente se utiliza el valor de $\alpha = 1$, y en esta sección el propósito es experimentar con distintos

valores de α y evaluar cómo afecta los distintos objetivos.

En la Fase 2, el objetivo es maximizar la cantidad de estudiantes asignados a algún curso. Como se observa en la Figura 4.13, tras la ejecución de esta fase, en todas las instancias, todos los estudiantes han sido asignados a al menos un curso. Por lo tanto, variar el valor de α no tendrá impacto en este resultado, ya que se ha alcanzado la cota superior para este objetivo.

En cambio, en la Fase 3 sí se puede evaluar la influencia de α en la búsqueda de su objetivo, que se recuerda que es mejorar la equidad estudiantil. Se analizan distintos resultados con los valores 1, 1,05, 1,1 y 1,3 de α. Se evalúa cómo este parámetro influye en la optimización de la función objetivo de dicha fase. En la Tabla 4.42 se muestra el valor máximo de la insatisfacción estudiantil para los distintos valores.

	Valores de α			
Instancia	1	1,05	1,1	1,3
E300-C4	14.278	14.278	14.278	14.278
E300-C10	185.650	184.246	184.246	184.246
E401-C94	156.281.198	156.906.043	156.123.897	157.198.155
E6457-C86	86.215.219	86.215.219	86.215.219	86.215.219
E7963-C94	206.845.449	206.607.123	206.598.423	206.617.580
E8208-C111	499.124.376	499.656.492	499.650.564	499.332.087

Tabla 4.42: Valor máximo de insatisfacción estudiantil para los distintos valores de α, para todas las instancias

En las instancias *E300-C4* y *E6457-C86* no se observa un cambio en el valor del objetivo al variar α. En contraste, en las instancias *E300-C10*, *E401-C94* y *E7963-C94* se aprecia una disminución leve para los valores de $\alpha = 1{,}05$ y $\alpha = 1{,}1$ en comparación con la solución base, donde $\alpha = 1$. No obstante, en las instancias *E401-C94* y *E7963-C94* se detecta un incremento sutil respecto a los valores anteriores con $\alpha = 1{,}3$, aunque sin superar la solución base. Este comportamiento podría explicarse por el uso de la opción *MIPGap* de Gurobi, la cual interrumpe la búsqueda de la solución antes de alcanzar el óptimo exacto [3]. Si bien esta configuración permite obtener una solución en un tiempo razonable, tiene como consecuencia el retorno de una solución subóptima. Un comportamiento similar se observa en la instancia *E8208-C111*, donde el mejor valor se obtiene con α=1.

Como conclusión, aunque en algunas instancias se mejore el valor obtenido en la Fase 3, esta relajación parametrizada por α no es suficiente para hacerlo de forma general en todas las instancias. En estas el valor de insatisfacción estudiantil máxima no puede ser disminuido sin afectar significativamente el valor obtenido en la Fase 1, probablemente por la realidad de la instancia y por los datos. Esto nos indica que si se quisiera utilizar este parámetro de relajación de forma más efectiva, se debería buscar una alternativa a la función objetivo

[3] En la Sección 3.2.2 se explica el comportamiento de la opción *MIPGap*

de la Fase 3, de manera que esta sea más flexible y permita mejorar el valor objetivo fácilmente.

4.2.6. Tiempos de ejecución de instancias

Las instancias fueron ejecutadas utilizando una computadora personal con un procesador AMD Ryzen 9 6900HS 3.30 GHz, 16 GB de RAM, con el sistema operativo Windows 11, utilizando AMPL en su versión 20231031, y el *solver* Gurobi en su versión 11.0.2. Se utilizó la opción *MIPGap* de Gurobi con un valor de 0,05, para que las instancias más grandes finalicen en un tiempo prudente. Este parámetro permite que la ejecución termine antes que se encuentre una solución óptima (en la Sección 3.2.2 se explica el comportamiento de esta opción). Con un valor de 0,05, la ejecución culmina cuando se encuentra una solución cuyo valor objetivo está a no más del 5 % de la mejor cota conocida.

Instancia	Tiempo F1	Tiempo F2	Tiempo F3	Tiempo total
E300-C4	0,13	0,21	0,68	1,02
E300-C10	0,31	0,56	3,00	3,86
E401-C94	1.376,02	1.032,14	11.973,88	14.382,05
E6457-C86	14,69	11,74	7,94	34,37
E7963-C94	70,10	72,17	305,32	447,59
E8208-C111	155,91	115,53	624,86	896,31

Tabla 4.43: Tiempo de ejecución por fase, expresado en segundos

Se puede ver que para las instancias *E300-C4*, *E300-C10* y *E6457-C86* el modelo finaliza en tiempos pequeños. Si se ve la Tabla 4.34, estas instancias son las que tienen la menor cantidad de restricciones y variables, por lo que es coherente que sean las que demoran menos.

Las instancias *E7963-C94* y *E8208-C111* presentan una duración de resolución superior, aunque se encuentran dentro de los parámetros razonables para el problema que se está abordando. En contraste, la instancia *E401-C94* es la que muestra el mayor tiempo de resolución, con una duración cercana a las 4 horas. Esta alta duración se debe a la gran cantidad de restricciones generadas, que ascienden a 4 millones 220 mil. Cabe destacar que, al ejecutar el modelo con esta instancia sin utilizar la opción de *MIPGap* de Gurobi, es decir, obligando a que el modelo alcance la solución óptima, la ejecución no finalizó pasado los tres días. En consecuencia, se optó por utilizar dicha opción para obtener un resultado cercano al óptimo.

En casi todas las instancias la Fase 3 es la que más tiempo demora, ocupando la mayor parte del tiempo total de ejecución. Esta alta duración no se debe a la cantidad de variables y restricciones utilizadas, porque es casi la misma que en las otras fases, sino al diseño de dicha fase. La única instancia que esto no sucede es en *E6457-C86*, que se puede explicar porque es la única instancia donde el óptimo de la Fase 3 es alcanzado en la Fase 2, como se muestra en la Tabla 4.41.

Capítulo 5

Conclusiones y Trabajo Futuro

A continuación se presentan los resultados principales obtenidos, las conclusiones y las oportunidades para futuros trabajos.

En el proyecto se cumplió el objetivo principal, que consistía en construir un modelo de programación entera que resuelva el problema de asignación de estudiantes a turnos en cursos con cupos, considerando preferencias y superposiciones de horarios.

En primer lugar se llevó a cabo un estudio sobre cómo se resuelven problemas similares al que enfrenta la Universidad de la República. Se clasificó la familia de problemas relacionados, identificando al problema UCTP como la categoría del problema a resolver. A continuación, se recopiló cómo se modelan en la literatura las restricciones más comunes, las preferencias entre las entidades y las funciones objetivo en las soluciones a este problema. Además, se analizaron varios artículos que modelan este problema como un problema de programación entera, así como otras alternativas, como el coloreo de grafos y los algoritmos genéticos. Por último, se estudiaron los problemas de optimización multi-objetivo y cómo estos se aplican para resolver el problema UCTP.

Se logró modelar el problema como un caso de programación entera. Se identificaron tres objetivos a optimizar: maximizar las preferencias globales de los estudiantes, maximizar la cantidad de alumnos que son inscritos en algún curso y aumentar la equidad en las asignaciones. A partir de estos objetivos, se formuló el problema como un problema de optimización multi-objetivo, utilizando la optimización lexicográfica para priorizar los objetivos en el orden que son señalados, y la resolución secuencial para resolverlo. El proceso se lleva a cabo en tres fases, en las cuales se optimiza un objetivo diferente en cada una, conservando los valores obtenidos en las fases anteriores. Además, se desarrolla un mecanismo que permite flexibilizar el valor óptimo alcanzado en la primera fase para las fases subsiguientes.

El modelo desarrollado gestiona correctamente las restricciones de capacidad

y la superposición de horarios, garantizando así la validez de las asignaciones. De este modo, se asegura que la cantidad de estudiantes asignados a un salón no exceda su capacidad, y que los estudiantes puedan asistir a todos sus cursos sin conflictos de horario.

Se implementó el modelo utilizando el lenguaje de programación algebraico AMPL, y se resolvió utilizando el *solver* Gurobi. Además, se construyó la definición de modelo en código con el objetivo de minimizar la cantidad de datos que se le debe proveer al modelo, reduciendo errores por inconsistencias.

Se construyó una aplicación de consola, utilizando el lenguaje de programación *Python*, que permite ejecutar el modelo mediante una interfaz sencilla. Asimismo, se proveen distintos parámetros para que el usuario pueda modificar valores de la ejecución. Esta aplicación procesa datos provenientes de distintas fuentes de datos, como archivos *.dat* de AMPL o planillas de cálculo *.xlsx* en un formato especificado. Además, guarda el resultado de la ejecución en una planilla de cálculo para que pueda ser analizada.

Luego, se diseñaron nueve casos de prueba para validar el modelo. Estos lo evalúan bajo diversos escenarios, lo que comprobó que se comporta como es requerido.

Se ejecutó el modelo utilizando seis instancias de datos provenientes de estudiantes de Facultad de Derecho de la UdelaR, provistos por el Servicio Central de Informática de la UdelaR (SeCIU). Estas instancias son de mediano y gran porte, y se desarrollaron métricas para evaluar el comportamiento del modelo.

En primer lugar, se observó que el modelo logra realizar al menos el 97 % de la máxima cantidad de asignaciones posibles en cinco de las seis instancias. Si bien alcanzar el número máximo de asignaciones no es un objetivo explícito de la formulación algebraica, los resultados muestran que el modelo se aproxima consistentemente a este valor óptimo.

Luego, se analizaron las listas de espera de los cursos para las distintas instancias. Estas fueron clasificadas por porcentaje de ocupación, lo que permitió observar que dichas listas de espera pueden generarse por dos razones: la limitación en la capacidad de los turnos del curso y/o la superposición de estos turnos con otras asignaciones realizadas. En la mayoría de las instancias, se observó que la capacidad de los cursos son una limitante para alcanzar una mayor cantidad de asignaciones. Asimismo, se concluyó que dicho indicador puede ser utilizado para asistir en la toma de decisiones, ya que permite visualizar los cursos en los que sería más beneficioso ampliar la capacidad para alcanzar mejores resultados.

El primer objetivo del modelo era asignar a los estudiantes a sus cursos más preferidos. Por esto, se analizaron las soluciones mirando la calidad de las asignaciones, o sea, dada una asignación se evaluó la preferencia de curso que el estudiante registró para dicho curso. Para esto, se construyeron gráficos de barras con la cantidad de asignaciones realizadas para cada valor de preferencia de curso. En términos generales, se vio que en todas las instancias las gráficas tienen un comportamiento decreciente. A partir de esto, se concluyó que el modelo prefiere hacer asignaciones de mayor calidad, beneficiando a los estudiantes. Sin embargo, se vieron casos donde el modelo prefirió realizar más asignaciones netas de menor calidad, antes que realizar menos asignaciones de mayor calidad.

Finalmente se vio que la mayoría de los estudiantes son asignados a cursos dentro de sus principales preferencias. De manera consistente, se observó que por lo menos un 67 % de los estudiantes recibe alguna de sus primeras 3 opciones.

Además de asignar a los cursos más preferidos por el estudiante, el modelo debía asignar también a los turnos más preferidos de este. Se verificó que el modelo cumple exitosamente con esta preferencia: en cinco de las seis instancias se alcanza un promedio de preferencia de turno en las asignaciones entre los valores 1 y 2, lo que significa que en promedio los estudiantes son asignados al primer o segundo turno más preferido del curso.

El segundo objetivo del modelo planteado era maximizar la cantidad de estudiantes asignados a por lo menos un curso. Se comprobó que en todas las instancias, todos los estudiantes son asignados a por lo menos a un curso, y por lo tanto este objetivo es cumplido.

Del mismo modo, se analizó el alcance del tercer objetivo del modelo, que era aumentar la equidad en las asignaciones. Se estudió la equidad en las asignaciones del modelo desde dos puntos de vista: en primer término, la equidad en felicidad, que toma en consideración la preferencia de los cursos a los que son asignados los estudiantes. En segundo término, la equidad en porcentaje de asignación por estudiante, que únicamente tiene en cuenta la cantidad de cursos a los que es asignado un estudiante. Se observó que en cuatro de las seis instancias se obtuvo altos niveles de equidad en felicidad, y en tres se obtuvo altos niveles de equidad en porcentaje de asignación. Esto se traduce en que en tres de las seis instancias el modelo realizó asignaciones equitativas tanto del punto de vista de calidad (valor de preferencia combinada de sus asignaciones), como del punto de vista de cantidad de cursos a los que un estudiante es asignado respecto a cuantos prefirió.

Además, se llevó a cabo un estudio sobre otras métricas de interés que permitieron analizar las soluciones. Una de ellas es el porcentaje de cupos libres, que compara la cantidad de asignaciones realizadas con la capacidad total. Otra métrica es el porcentaje de asignación promedio por estudiante, que evalúa la cantidad de cursos asignados a cada estudiante en relación con la cantidad de cursos que este declaró como preferencia. Finalmente, se concluye que todos los indicadores generados fueron de utilidad para el análisis de los resultados.

Se estudiaron las soluciones parciales luego de cada fase, y se evaluaron los objetivos de la Fase 2 y la Fase 3 en estas, con el propósito de evaluar el efecto de las fases sobre los objetivos. En primer lugar, se observó que en tres instancias la Fase 1 arrojó una asignación con estudiantes sin asignaciones a cursos. Una vez corrida la Fase 2, todos estos estudiantes son asignados a por lo menos un curso. Por lo tanto, se concluyó que la Fase 2 es eficaz en lograr su objetivo. Por otro lado, se verificó que la Fase 3 realizó una mejora en la equidad de felicidad en tres de las seis instancias, y una mejora o mantención de equidad en porcentaje de asignación en todas las instancias. Sin embargo, los incrementos de equidad no son significativos, y por consiguiente, la Fase 3 no es eficaz para alcanzar buenos niveles de equidad bajo las perspectivas establecidas.

Por otro lado, se descubrió que la variación del parámetro α, que flexibiliza el valor de función objetivo de la Fase 1 en las fases posteriores, no mejora el

valor de la función objetivo de la Fase 3 de forma general en todas las instancias. Se deduce que la función objetivo de la Fase 3 no es flexible y no permite que se modifique su valor fácilmente.

Por último, se realizó un análisis de los tiempos de ejecución. Se logró ejecutar el modelo en una computadora personal en tiempos razonables. Cinco de las seis instancias culminaron en menos de quince minutos, y la instancia que demoró más lo hizo en menos de cuatro horas. Se utilizó la opción *MIPGap* de Gurobi con un valor de 0,05 para que las instancias grandes puedan culminar en un tiempo prudente.

Como trabajo a futuro se identifican varias líneas de investigación. La primera es buscar alternativas a la Fase 3 para modelar mejor la equidad, y para que la función objetivo de dicha fase sea más flexible al modificarse el parámetro α.

La segunda línea de investigación consiste en incorporar una etapa posterior al proceso de asignación. Una vez realizada la asignación inicial, se informan a cada estudiante los cursos a los que han sido asignados, y estos pueden elegir cambiar sus elecciones o mantenerlas. Luego que este proceso es realizado por todos los estudiantes, se realiza una reasignación que considera las nuevas preferencias expresadas.

Otra línea de investigación es ampliar el modelo para que se consideren otras necesidades de la UdelaR. Por ejemplo, se puede considerar tomar en cuenta la cantidad de créditos de los estudiantes para priorizar a los estudiantes con más créditos. Otro elemento a agregar puede ser que los estudiantes elijan la máxima cantidad de cursos que quieren ser asignados, para que puedan declarar preferencias a más cursos de los que desea inscribirse.

Por último, para que el modelo sea utilizado por los estudiantes de la universidad, se debe integrarlo con el Sistema de Gestión Administrativa de la Enseñanza (SGAE). Una posible línea de trabajo es crear una interfaz gráfica para que los estudiantes elijan los cursos y turnos que desean cursar, y una vez que todos los estudiantes declaren sus preferencias, se ejecute el modelo para realizar la asignación.

Referencias

Abdallah, K. S. (2016). Multi-objective Optimization for Exam Scheduling to Enhance the Educational Service Performance. *Journal of Management & Engineering Integration*, *9*(1), 14–23. (Publisher: Journal of Management & Engineering Integration)

Alghamdi, H., Alsubait, T., Alhakami, H., y Baz, A. (2020, diciembre). A Review of Optimization Algorithms for University Timetable Scheduling. *Engineering, Technology & Applied Science Research*, *10*(6), 6410–6417. (Publisher: D. G. Pylarinos) doi: 10.48084/etasr.3832

AMPL Options — AMPL Resources. (s.f.). Descargado 2024-09-18, de https://dev.ampl.com/ampl/options.html

amplpy: Python API for AMPL. (s.f.). Descargado 2024-09-24, de http://ampl.com/

Anwar, A., y Bahaj, A. (2003, agosto). Student project allocation using integer programming. *IEEE Transactions on Education*, *46*(3), 359–367. (Conference Name: IEEE Transactions on Education) doi: 10.1109/TE.2003.811038

Babaei, H., Karimpour, J., y Hadidi, A. (2015, agosto). A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, *86*, 43–59. doi: 10.1016/j.cie.2014.11.010

Burke, E., de Werra, D., y Kingston, J. (Eds.). (2004). Applications to timetabling. *Handbook of Graph Theory*. (Publisher: CRC Press London)

Carter, M. W. (2001). A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo. En E. Burke y W. Erben (Eds.), *Practice and Theory of Automated Timetabling III* (pp. 64–82). Berlin, Heidelberg: Springer. doi: 10.1007/3-540-44629-X_5

Chartier, T. P., Ellison, V., y Langville, A. N. (2014, diciembre). A Davidson College multi-objective assignment problem: a case study. *4OR*, *12*(4), 379–401. doi: 10.1007/s10288-014-0259-2

Costabel, S. (2005). Metaheurísticas aplicadas aun problema de asignación a un problema de salones y horarios de asignaturas. (Accepted: 2014-11-24T22:37:28Z Publisher: UR. FI-INCO,)

Faudzi, S., Abdul-Rahman, S., y Abd Rahman, R. (2018, mayo). An Assignment Problem and Its Application in Education Domain: A Review and Potential Path. *Advances in Operations Research*, 1–19. (Publisher: Hindawi Limited) doi: 10.1155/2018/8958393

Ganguli, R., y Roy, S. (2017). A Study on Course Timetable Scheduling using Graph Coloring Approach.

Gartner, D., y Kolisch, R. (2021, agosto). Mathematical programming for nominating exchange students for international universities: The impact of stakeholders' objectives and fairness constraints on allocations. *Socio-Economic Planning Sciences*, *76*, 100974. doi: 10.1016/j.seps.2020.100974

Gosselin, K., y Truchon, M. (1986, junio). Allocation of Classrooms by Linear Programming. *Journal of the Operational Research Society*, *37*(6), 561–569. doi: 10.1057/jors.1986.98

Gozali Alfian Akbar, y Fujimura Shigeru. (2020, enero). Solving University Course Timetabling Problem Using Multi-Depth Genetic Algorithm. *SHS Web of Conferences*, *77*, 01001–01001. (Publisher: EDP Sciences) doi: 10.1051/shsconf/20207701001

Gurobi Optimization. (s.f.). Descargado 2024-09-24, de https://www.gurobi.com/

Hassim, W., Haniza, W., Shibghatullah, A., y Shahbodin, F. (2014, enero). Solving new student allocation problem (NSAP) with analytical hierarchy process (AHP). *Science International*, *26*.

Huangfu, Q., y Hall, J. A. J. (2018, marzo). Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, *10*(1), 119–142. doi: 10.1007/s12532-017-0130-5

jupyterlab: JupyterLab computational environment. (s.f.). Descargado 2024-09-24, de https://jupyter.org

Lemos, A., Monteiro, P. T., y Lynce, I. (2021, febrero). Disruptions in timetables: a case study at Universidade de Lisboa. *Journal of Scheduling*, *24*(1), 35–48. (Publisher: Springer Nature) doi: 10.1007/s10951-020-00666-3

Lewis, R. (2008, enero). A survey of metaheuristic-based techniques for University Timetabling problems. *OR Spectrum*, *30*(1), 167–190. (Publisher: Springer Nature) doi: 10.1007/s00291-007-0097-0

matplotlib: Python plotting package. (s.f.). Descargado 2024-09-24, de https://matplotlib.org

Miettinen, K. (1998). *Nonlinear multiobjective optimization* (Vol. 12). Boston, USA: Kluwer Academic Publishers. doi: 10.1007/978-1-4615-5563-6

MIPGap. (s.f.). Descargado 2024-09-24, de https://www.gurobi.com/documentation/current/refman/mipgap2.html

pandas: Powerful data structures for data analysis, time series, and statistics. (s.f.). Descargado 2024-09-24, de https://pandas.pydata.org

Phillips, A. E., Walker, C. G., Ehrgott, M., y Ryan, D. M. (2017, mayo). Integer programming for minimal perturbation problems in university course timetabling. *Annals of Operations Research*, *252*(2), 283–304. doi: 10.1007/s10479-015-2094-z

Python. (s.f., septiembre). Descargado 2024-09-24, de https://www.python.org/

Teo, Y., y Ho, D. (1998, febrero). A systematic approach to the implementation of final year project in an electrical engineering undergraduate cour-

se. *IEEE Transactions on Education, Education, IEEE Transactions on, IEEE Trans. Educ.*, *41*(1), 25–30. (Place: USA Publisher: IEEE) doi: 10.1109/13.660783

Wren, A. (1996). Scheduling, timetabling and rostering — A special relationship? En E. Burke y P. Ross (Eds.), *Practice and Theory of Automated Timetabling* (pp. 46–75). Berlin, Heidelberg: Springer. doi: 10.1007/3-540-61794-9_51

Zykina, A. V. (2004, marzo). A Lexicographic Optimization Algorithm. *Automation & Remote Control*, *65*(3), 363–368. (Publisher: Springer Nature) doi: 10.1023/B:AURC.0000019366.84601.8e

Anexo A

Anexo 1

A.1. Modelo en AMPL

```
# estudiantes
set E;
# cursos
set C;
# los turnos del curso c
set TC{C};

# turnos
set T = union {c in C} TC[c];

# los turnos que se superponen con t
set S{T} within T;

# los turnos preferidos por e del curso c
set TP{E, C} within T ordered;
# los cursos preferidos por e
set CP{E} within C ordered;
# los estudiantes que prefieren el turno t del curso c
set ET{c in C, t in T} =  {e in E : c in CP[e] and t in TP[e,c]};
# preferencia de estudiante e hacia el turno t del curso c, definicion de matriz
# optimiza memoria definiendo de forma subordinada en e
param pt{e in E, c in CP[e], t in TP[e, c]} = ord(t, TP[e,c]);
# preferencia de estudiante e hacia el curso c, definicion de matriz
# optimiza memoria definiendo de forma subordinada en e
param pc{e in E, c in CP[e]} = ord(c, CP[e]);
# la capacidad del turno t del cuso c
param cap{c in C, TC[c]};
# k es equivalente al maximo numero de turnos de todos los cursos
```

```
param K = max{c in C} card(TC[c]);
# M es numero muy grande
param M;
# coeficiente de relajacion
param a default 1;


# variable de desicion que indica si el estudiante e es asignado al turno t
# optimiza cantidad variables
var x{e in E,c in CP[e], TP[e, c]} binary;

# variable de desicion que indica si el estudiante e es asignado a la lista de espera
var z{e in E,c in CP[e]} binary;

# variable que vale 1 si el estudiante es asignado a algun curso
var y{e in E} binary;

# cota superior de todas las insatisfacciones indivuales
var w >= 0 integer;


# variable creada para reutilizar la expresion en ambas funciones objetivos
var insatisfaccion_estudiante {e in E} = sum {c in CP[e], t in TP[e, c]} ((((K*pc[e,c])

# fase 1
var funcion_objetivo1 = sum {e in E} (insatisfaccion_estudiante[e]);

minimize valor_preferencias:
funcion_objetivo1;

# fase 2

var funcion_objetivo2 = sum {e in E} y[e];
maximize cantidad_estudiantes_asignados:
funcion_objetivo2;

# fase 3

minimize maximo_insatisfaccion_individual:
    w;

# fase 1
# Un estudiante debe ser asignado a un turno para cada curso preferido por el
s.t. ctr_est_asignado_pref{e in E,c in CP[e]}:
(sum {t in {TP[e,c]}} x[e,c,t]) + z[e,c] = 1;
# Para un estudiante el cual es asignado a dos turnos de cursos distintos, estos no del
```

```
s.t. ctr_solap{e in E, c in CP[e], t in TP[e, c], t2 in S[t],
               c2 in CP[e]: c2 <> c and t2 in TC[c2] inter TP[e,c2] }:
    x[e,c,t] + x[e,c2,t2] <= 1;
# La capacidad de los turnos no debe ser sobrepasada por la cantidad de estudiantes asignados.
s.t. ctr_cap{c in C, t in TC[c]}:
sum {e in ET[c, t]} x[e,c,t] <= cap[c,t];

# fase 2

s.t. ctr_asignado_algun_curso{e in E}:
   sum {c in CP[e], t in TP[e,c]}(x[e,c,t]) <= card(CP[e]) * y[e];

s.t. ctr_cota_inferior_y {e in E}:
   y[e] <= sum {c in CP[e], t in TP[e,c]}(x[e,c,t]);


# fase 3
# definicion por construccion de la variable i
s.t. ctr_insatisfaccion_individual{e in E}:
    insatisfaccion_estudiante[e] <= w;

param valor_optimal1;
s.t. ctr_fijar_obj1:
funcion_objetivo1 <= a * valor_optimal1;

param valor_optimal2;
s.t. ctr_fijar_obj2:
funcion_objetivo2 = valor_optimal2;
```

A.2. Formato de entrada .xslx

A continuación se especifica el formato que debe tener el archivo *.xslx* para que pueda ser utilizado en la ejecución del modelo.

La planilla debe tener 5 hojas, con los nombres *Cursos*, *PreferenciasCursos*, *PreferenciasTurnos*, *Capacidades* y *Superposiciones*. Se presenta un ejemplo de una instancia para mostrar la estructura a seguir en cada hoja.

	A	B	C
1	C1	C2	C3
2	T1	T1	T1
3	T2	T2	T2
4	T3		

Figura A.1: Ejemplo de hoja *Cursos*

Para especificar los cursos, se ingresan en la primera fila se tienen todos estos, y debajo de cada uno se tiene los turnos del mismo. En este ejemplo se tienen los cursos *C1*, *C2* y *C3*, donde el curso *C1* es dictado en los turnos *T1*, *T2* y *T3*, y los cursos *C2* y *C3* son dictados en *T1* y *T2*.

	A	B	C	D
1	Estudiante	C1	C2	C3
2	1	0	1	0
3	2	0	0	1
4	3	0	1	0
5	4	1	0	0
6	5	0	1	0
7	6	2	3	1

Figura A.2: Ejemplo de hoja *PreferenciasCursos*

Por otro lado, para especificar la preferencia de cursos de los estudiantes, en la primera columna se ingresan todos los estudiantes, y en el resto de columnas se tienen los cursos. En la entrada de la matriz se tiene la preferencia de curso pc_{ec} del estudiante hacia el curso. Por ejemplo, el estudiante *1* prefiere únicamente al curso *C2*, mientras que el estudiante *6* prefiere a los cursos *C3*, *C1* y *C2*, en ese orden.

	A	B	C	D	E
1	Estudiante	Curso	T1	T2	T3
2	1	C2	0	1	
3	2	C3	1	0	
4	3	C2	0	1	
5	4	C1	1	0	1
6	5	C2	0	1	
7	6	C1	1	0	0
8	6	C2	1	0	
9	6	C3	1	0	

Figura A.3: Ejemplo de hoja *PreferenciasTurnos*

Para definir las preferencias de turnos se hace algo similar, con la diferencia es que es una matriz de doble índice, los cuales son las columnas Estudiante y Curso. Luego, en las otras columnas se encuentran todos los turnos disponibles. En la entrada de la matriz se especifica la preferencia de turno pt_{ect} para este estudiante, curso y turno.

	A	B	C
1	Turno	Curso	Capacidad
2	T1	C1	1
3	T2	C1	1
4	T3	C1	1
5	T1	C2	1
6	T2	C2	1
7	T1	C3	1
8	T2	C3	1

Figura A.4: Ejemplo de hoja *Capacidades*

Pasando a las capacidades, estas se definen también utilizando es una matriz

de doble índice, y en este caso son las columnas Turno y Curso. En la columna Capacidad se establece la capacidad cap_{ct} de este curso y turno. En este ejemplo todos los turnos de curso tienen capacidad 1.

	A	B	C
1	T1	T2	T3
2		T3	T2

Figura A.5: Ejemplo de hoja *Superposiciones*

Por último, para establecer las superposiciones se dispone en la primera fila todos los turnos, y debajo de cada uno se tienen los turnos que se superponen con este. Cabe destacar que no es necesario incluir al propio turno en su lista de superposiciones, y que si dos turnos se superponen, ambos deben ser incluidos en la lista de superposiciones del otro. Por ejemplo, los turnos *T2* y *T3* se superponen, y tanto el turno *T3* está en la lista de turnos superpuestos de *T2*, y *T2* está en la de *T3*.

A.3. Formato de salida

A continuación se especifica el formato de las planillas de salida de la aplicación de consola, que es donde se guardan los resultados del modelo.

Se generan tres archivos llamados *fase-1.xlsx*, *fase-2.xlsx* y *fase-3.xlsx*, donde en cada uno se guarda la solución terminada dicha fase, y por lo tanto la solución final se encuentra en *fase-3.xlsx*. Todos los archivos tienen el mismo formato, por lo que se especifica el mismo de forma genérica.

El archivo contiene dos hojas, llamadas x y z, donde se guarda el resultado de las variables x_{ect} y z_{ec} respectivamente. En la Figura A.6 se muestra un ejemplo de salida de la hoja x. Esta contiene una tabla con las columnas *estudiante*, *curso*, *turno*, x y *var.rc*. Las primeras tres columnas corresponden a los valores de estos elementos, mientras que la columna x tiene el valor 1 dependiendo si el estudiante es asignado al turno de curso, y 0 en caso contrario. Por último, la columna *var.rc* tiene el costo reducido de esta variable, que toma el valor de $(Kpc_{ec} + pt_{ect})^2$, donde e, c y t son los estudiante, curso y turno de esta fila.

La hoja z tiene un formato similar, con una tabla con columnas *estudiante*, *curso*, z y *var.rc*. Las primeras dos columnas corresponden a los valores de estos elementos, y en la columna z tiene el valor 1 si el estudiante es asignado a la lista de espera del curso, y 0 en caso contrario. De manera análoga a la hoja x, en la columna *var.rc* se tiene el costo reducido de esta variable, que tiene el valor de M en todas las filas. En la Figura A.7 se muestra en un ejemplo de salida de esta hoja.

	A	B	C	D	E	F
1		**estudiante**	**curso**	**turno**	**x**	**var.rc**
2	**0**	1	C2	T2	1	16
3	**1**	1	C2	T3	0	25
4	**2**	2	C3	T1	1	16
5	**3**	2	C3	T3	0	25
6	**4**	3	C2	T2	0	16
7	**5**	3	C2	T3	1	25
8	**6**	4	C1	T1	0	16
9	**7**	4	C1	T3	1	25
10	**8**	5	C2	T2	0	16
11	**9**	5	C2	T3	1	25
12	**10**	6	C1	T1	1	49
13	**11**	6	C2	T1	0	100
14	**12**	6	C2	T3	1	121
15	**13**	6	C3	T1	0	16
16	**14**	6	C3	T3	1	25

Figura A.6: Ejemplo de hoja x

	A	B	C	D	E
1		**estudiante**	**curso**	**z**	**var.rc**
2	**0**	1	C2	0	145
3	**1**	2	C3	0	145
4	**2**	3	C2	0	145
5	**3**	4	C1	0	145
6	**4**	5	C2	0	145
7	**5**	6	C1	0	145
8	**6**	6	C2	0	145
9	**7**	6	C3	0	145

Figura A.7: Ejemplo de hoja z

Anexo B

Anexo 2

B.1. Datos utilizados en la validación del modelo

Como parte de la experimentación, en la Sección 4.1 se realizan distintas validaciones al modelo para comprobar que este se comporte se espera. Para ello, se utilizan los casos de pruebas presentados a continuación, utilizando el formato *.dat* de AMPL.

B.1.1. Datos de ponderación de pc_{ec} sobre pt_{ect}

```
set E := 1, 2;

set C := 'C1', 'C2';

set TC['C1'] := 'M';
set TC['C2'] := 'M';

set S['M'] := 'M';

set TP[1, 'C1'] := 'M';
set TP[2, 'C1'] := 'M';
set TP[1, 'C2'] := 'M';
set TP[2, 'C2'] := 'M';

set CP[1] := 'C1', 'C2';
set CP[2] := 'C2', 'C1';

param cap := 'C1', 'M' 2,
```

```
            'C2', 'M' 2;

end;
```

B.1.2. Datos de asignación a los estudiantes a los cursos más preferidos

```
set E := 1, 2, 3;

set C := 'C1', 'C2', 'C3';

set TC['C1'] := 'M';
set TC['C2'] := 'M';
set TC['C3'] := 'M';

set S['M'] := 'M';

set TP[1, 'C1'] := 'M';
set TP[1, 'C2'] := 'M';
set TP[1, 'C3'] := 'M';

set TP[2, 'C1'] := 'M';
set TP[2, 'C2'] := 'M';
set TP[2, 'C3'] := 'M';

set TP[3, 'C1'] := 'M';
set TP[3, 'C2'] := 'M';
set TP[3, 'C3'] := 'M';


set CP[1] := 'C1', 'C2', 'C3';
set CP[2] := 'C2', 'C1', 'C3';
set CP[3] := 'C3', 'C2', 'C1';

param cap := 'C1', 'M' 3,
             'C2', 'M' 3,
             'C3', 'M' 3;

param M := 10000000;
end;
```

B.1.3. Datos de asignación a los estudiantes a su turno más preferido

```
set E := 1, 2, 3;
set C := 'C1';

set TC['C1'] := 'M', 'V', 'N';

set S['M'] := 'M';
set S['V'] := 'V';
set S['N'] := 'N';

set TP[1, 'C1'] := 'M', 'V', 'N';
set TP[2, 'C1'] := 'V', 'M', 'N';
set TP[3, 'C1'] := 'N', 'V', 'M';


set CP[1] := 'C1';
set CP[2] := 'C1';
set CP[3] := 'C1';

param cap := 'C1', 'M' 10,
             'C1', 'V' 10,
             'C2', 'M' 10,
             'C2', 'V' 10;

param M := 10000000;
end;
```

B.1.4. Datos de peor asignación antes que lista de espera

```
set E := 1, 2, 3;
set C := 'C1', 'C2', 'C3';

set TC['C1'] := 'M', 'V', 'N';
set TC['C2'] := 'V', 'N', 'M';
set TC['C3'] := 'M','V', 'N';

set S['M'] := 'M';
set S['V'] := 'V';
set S['N'] := 'N';

set TP[1, 'C1'] := 'M','V', 'N';
set TP[2, 'C1'] := 'M','V', 'N';
set TP[3, 'C1'] := 'M','V', 'N';
```

```
set TP[1, 'C2'] := 'M', 'V', 'N';
set TP[3, 'C2'] := 'M', 'V', 'N';
set TP[2, 'C2'] := 'M', 'V', 'N';

set TP[1, 'C3'] := 'M','V', 'N';
set TP[2, 'C3'] := 'M','V', 'N';
set TP[3, 'C3'] := 'M','V', 'N';

set CP[1] := 'C1', 'C2', 'C3';
set CP[2] := 'C1', 'C2', 'C3';
set CP[3] := 'C1', 'C2', 'C3';


param cap:= 'C1', 'M' 1,
            'C1', 'V' 1,
            'C1', 'N' 1,
            'C2', 'V' 1,
            'C2', 'N' 1,
            'C2', 'M' 1,
            'C3', 'N' 1,
            'C3', 'M' 1,
            'C3', 'V' 1;

end;
```

B.1.5. Datos de capacidad de los turnos de curso

```
set E := 1, 2, 3, 4, 5;
set C := 'C1', 'C2', 'C3','C4', 'C5';

set TC['C1'] := 'M', 'V';
set TC['C2'] := 'M', 'V';
set TC['C3'] := 'M', 'V';
set TC['C4'] := 'M', 'V';
set TC['C5'] := 'M', 'V';

set S['M'] := 'M';
set S['V'] := 'V';

set TP[1, 'C1'] := 'V';
set TP[2, 'C1'] := 'V';
set TP[3, 'C1'] := 'V';
set TP[4, 'C1'] := 'V';
set TP[5, 'C1'] := 'V';

set TP[1, 'C2'] := 'M';
```

```
set TP[2, 'C2'] := 'M';
set TP[3, 'C2'] := 'M';
set TP[4,'C2'] := 'M';
set TP[5,'C2'] := 'M';

set TP[1, 'C3'] := 'M';
set TP[2, 'C3'] := 'M';
set TP[3, 'C3'] := 'M';
set TP[4, 'C3'] := 'M';
set TP[5, 'C3'] := 'M';

set TP[1, 'C4'] := 'M';
set TP[2, 'C4'] := 'M';
set TP[3, 'C4'] := 'M';
set TP[4, 'C4'] := 'M';
set TP[5, 'C4'] := 'M';

set TP[1, 'C5'] := 'V';
set TP[2, 'C5'] := 'V';
set TP[3, 'C5'] := 'V';
set TP[4, 'C5'] := 'V';
set TP[5, 'C5'] := 'V';

set CP[1] := 'C1', 'C2', 'C3', 'C4', 'C5';
set CP[2] := 'C1', 'C2', 'C3', 'C4', 'C5';
set CP[3] := 'C1', 'C2', 'C3', 'C4', 'C5';
set CP[4] := 'C1', 'C2', 'C3', 'C4', 'C5';
set CP[5] := 'C1', 'C2', 'C3', 'C4', 'C5';

param cap := 'C1', 'M' 1,
             'C1', 'V' 1,
             'C2', 'M' 1,
             'C2', 'V' 1,
             'C3', 'M' 1,
             'C3', 'V' 1,
             'C4', 'M' 1,
             'C4', 'V' 1,
             'C5', 'M' 1,
             'C5', 'V' 1;

end;
```

B.1.6. Datos de superposiciones entre turnos

```
set E := 1, 2, 3;
```

```
set C := 'C1', 'C2';

set TC['C1'] := 'M1', 'M2', 'V';
set TC['C2'] := 'M1', 'M2', 'V';

set S['M1'] := 'M1', 'M2';
set S['M2'] := 'M2', 'M1';
set S['V'] := 'V';


set TP[1, 'C1'] := 'V';
set TP[1, 'C2'] := 'V';

set TP[2, 'C1'] := 'M1';
set TP[2, 'C2'] := 'M2';

set TP[3, 'C1'] := 'M2';
set TP[3, 'C2'] := 'M1';

set CP[1] := 'C1', 'C2';
set CP[2] := 'C1', 'C2';
set CP[3] := 'C1', 'C2';

param cap := 'C1', 'M1' 3,
             'C1', 'M2' 3,
             'C1', 'V'  3,
             'C2', 'M1' 3,
             'C2', 'M2' 3,
             'C2', 'V'  3;

end;
```

B.1.7. Datos de soluciones con la mayor cantidad de estudiantes asignados

```
set E := 1, 2, 3;
set C := 'C1', 'C2';

set TC['C1'] := 'V', 'N';
set TC['C2'] := 'M';

set S['M'] := 'M';
set S['V'] := 'V';
set S['N'] := 'N';
```

```
set TP[1, 'C1'] := 'N';
set TP[2, 'C1'] := 'V';
set TP[3, 'C1'] := 'V';

set TP[1, 'C2'] := 'M';
set TP[2, 'C2'] := 'M';

set CP[1] := 'C2', 'C1';
set CP[2] := 'C1', 'C2';
set CP[3] := 'C1';

param cap := 'C1', 'V' 1,
             'C1', 'N' 1,
             'C2', 'M' 1;

param a := 1.5; # este valor es utilizado una vez en 1 y otra vez en 1.5
end;
```

B.1.8. Datos de soluciones con equilibrio en la satisfacción estudiantil

```
set E := 1, 2, 3;
set C := 'C1', 'C2', 'C3', 'C4', 'C5', 'C6';

set TC['C1'] := 'M';
set TC['C2'] := 'M';
set TC['C3'] := 'M';
set TC['C4'] := 'V';
set TC['C5'] := 'V';
set TC['C6'] := 'N';

set S['M'] := 'M';
set S['V'] := 'V';
set S['N'] := 'N';

set TP[1, 'C1'] := 'M';
set TP[2, 'C1'] := 'M';
set TP[3, 'C1'] := 'M';

set TP[1, 'C2'] := 'M';
set TP[2, 'C2'] := 'M';
set TP[3, 'C2'] := 'M';

set TP[1, 'C3'] := 'M';
set TP[2, 'C3'] := 'M';
set TP[3, 'C3'] := 'M';
```

```
set TP[1, 'C4'] := 'V';
set TP[2, 'C4'] := 'V';
set TP[3, 'C4'] := 'V';

set TP[1, 'C5'] := 'V';
set TP[2, 'C5'] := 'V';
set TP[3, 'C5'] := 'V';

set TP[1, 'C6'] := 'N';
set TP[2, 'C6'] := 'N';
set TP[3, 'C6'] := 'N';

set CP[1] := 'C1', 'C2','C3', 'C4', 'C5', 'C6';
set CP[2] := 'C2', 'C1','C3', 'C5', 'C4', 'C6';
set CP[3] := 'C3', 'C2','C1', 'C5', 'C6', 'C4';

param cap := 'C1', 'M' 1,
             'C2', 'M' 1,
             'C3', 'M' 1,
             'C4', 'V' 1,
             'C5', 'V' 1,
             'C6', 'N' 1;

end;
```

B.1.9. Datos de penalización de estudiantes con mayor disponibilidad

```
set E := 1, 2, 3, 4;
set C := 'C1', 'C2', 'C3';

set TC['C1'] := 'M', 'V';
set TC['C2'] := 'M';
set TC['C3'] := 'M';

set S['M'] := 'M';
set S['V'] := 'V';

set TP[1, 'C1'] := 'M', 'V';
set TP[2, 'C1'] := 'M';

set TP[3, 'C2'] := 'M';
set TP[3, 'C3'] := 'M';

set TP[4, 'C2'] := 'M';
```

```
set CP[1] := 'C1';
set CP[2] := 'C1';
set CP[3] := 'C2', 'C3';
set CP[4] := 'C2';


param cap := 'C1', 'M' 1,
             'C1', 'V' 1,
             'C2', 'M' 1,
             'C3', 'M' 1;

end;
```

B.2. Modelo de maximización de asignaciones

Como parte de la experimentación, en la Sección 4.2.3 se compara el modelo propuesto en este informe con una variante que, en lugar de minimizar la insatisfacción global, maximiza la cantidad de asignaciones. A continuación, se presenta la formulación matemática de esta variante, la cual se basa en un subconjunto de los conjuntos, variables y restricciones definidos en la Sección 3.1.1.

Conjuntos

E: estudiantes
C: cursos
T: turnos
TC_c: turnos del curso $c \in C$, tal que $TC_c \subseteq T$
S_t: turnos que se superponen con el turno $t \in T$, tal que $S_t \subseteq T$
TP_{ec}: turnos del curso $c \in C$ preferidos por el estudiante $e \in E$, tal que $TP_{ec} \subseteq TC_c$
CP_e: cursos preferidos por el estudiante $e \in E$, tal que $CP_e \subseteq C$
ET_{ct}: estudiantes que prefieren el turno $t \in TC_c$ del curso $c \in C$, tal que $ET_{ct} \subseteq E$

Parámetros

cap_{ct}: capacidad del turno $t \in TC_c$ del curso $c \in C$

Variables de decisión

x_{ect}: vale 1 si el estudiante $e \in E$ es asignado al turno $t \in TP_e$ del curso $c \in CP_e$, y vale 0 en otro caso

Formulación

$$\max \sum_{e \in E} \sum_{c \in CP_e} \sum_{t \in TP_{ec}} x_{ect} \tag{B.1}$$

s.a

$$\sum_{t \in TP_{ec}} x_{ect} <= 1, \quad e \in E,\ \forall c \in CP_e, \tag{B.2}$$

$$x_{ect} + x_{ect'} \leq 1, \quad e \in E,\ c \in CP_e,\ t \in TP_{ec},\ t' \in S_t, \\ c' \in CP_e : c' \neq c \wedge t' \in TC_{c'} \cap TP_{ec'}, \tag{B.3}$$

$$\sum_{e \in ET_{ct}} x_{ect} \leq cap_{ct}, \quad c \in C,\ t \in TC_c, \tag{B.4}$$

$$x_{ect} \in \{0, 1\}, \quad e \in E,\ c \in CP_e,\ t \in TP_{ec}. \tag{B.5}$$