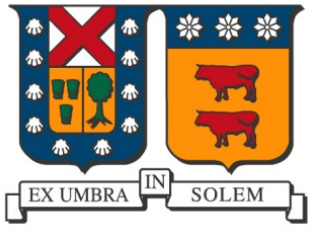
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“ESTUDIO DE CRITERIOS Y AUTOMATIZACIÓN DEL PROCESO DE ASIGNACIÓN DE AYUDANTES EN EL DEPARTAMENTO DE INFORMÁTICA”

JOSÉ IGNACIO MARINO GARCÍA PEREIRA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Carlos Castro
Profesor Correferente: José Luis Martí

Septiembre - 2020

DEDICATORIA

A todos los que me importan.

AGRADECIMIENTOS

Agradezco a los profesores del departamento que me han podido ayudar en este largo periodo. También quiero incluir a la industria del entretenimiento, por ayudarme a mantener la cordura durante la cuarentena. Y a mi papá, por motivarme a concluir este trabajo.

RESUMEN

Resumen— Todos los semestres se pasa por el proceso de asignar ayudantes a las asignaturas. Éste es un problema de asignación el cual se realiza mediante un sistema de postulación informático. Esto genera distintos problemas, lo que produce interés en determinar si es posible mejorar el proceso actual. Se plantea estudiar distintos criterios presentes en este proceso de asignación de ayudantes, simulando el impacto de éstos mediante un modelo matemático, logrando proponer modificaciones al proceso actual con el objetivo de optimizar el resultado.

Palabras Clave— Problema de asignación; Problema de transporte; Programación entera; Ayudantes; Optimización.

ABSTRACT

Abstract— Every semester we go through the process of assigning teaching assistants to subjects. This is an assignment problem which is done through an application system. This creates a number of problems, leading to an interest in determining whether the current process can be improved. It is proposed to study different criteria present in this process of assigning teaching assistants, simulating the impact of the system using a mathematical model, managing to propose modifications to the current process in order to optimize the result.

Keywords— Assignment problem; Transportation problem; Integer programming; Teaching assistants; Optimization.

GLOSARIO

GAP: Generalized Assignment Problem.
GA: Genetic Algorithm.
SA: Simulated Annealing.
TP: Transportation problem.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los problemas asociados a la distribución de recursos, que satisfacen múltiples criterios de diferentes entes, son un tópico bastante interesante para los revisores y creadores de procesos, ya que suelen encontrarse altos beneficios al lograr optimizaciones dentro de estas distribuciones de recursos. Estos procesos de optimización suelen ser triviales para una persona si la cantidad de variables y parámetros asociados son muy pequeñas, pero cuando empezamos a tener una cantidad un poco mayor de elementos a considerar, nuestro rendimiento como humanos no es el mejor. Algo que hemos aprendido con el paso de los años, es que los humanos no somos buenos optimizando, pero sí reconociendo patrones. Los procesos de optimización se pueden realizar de manera más eficiente y certera si usamos modelos matemáticos y herramientas de computación.

Es bajo este tipo de pensamiento que, en el capítulo 1, se describe la necesidad de resolver algunos problemas asociados al proceso de asignación de ayudantes mediante la incorporación de un modelo dedicado a la optimización de la asignación asociada. La asignación sub-óptima que se obtiene en el proceso actual presenta ciertos problemas, los cuales son reconocidos y se busca poder resolver mediante una mejora en la asignación final del proceso.

Para poder realizar el proceso de optimización de forma inteligente con un algoritmo, es necesario establecer un modelo, el cual debe incorporar criterios y parámetros que permitan representar al proceso real de la manera más general posible, es decir, cubrir una gran cantidad de casos útiles y de interés a optimizar. En el capítulo 2 se hace una pequeña revisión de métodos y modelos asociados al problema propuesto, y como estos son abordados y resueltos.

En el capítulo 3 se propone un nuevo proceso, incluyendo el modelamiento asociado a la asignación de ayudantes. Este proceso de modelamiento concluye con la postulación de un modelo de programación lineal, el cual es capaz de modelar la asignación de ayudantes y el recurso asociado, las horas de ayudantía, optimizando estos valores.

La validación de este modelo propuesto pasa por comprobar el funcionamiento correcto del mismo, lo que luego permite validar una modificación al proceso de asignación inicial para crear un nuevo proceso, en donde la asignación inteligente es considerada como una etapa más del proyecto. En el capítulo 4 se hace un análisis del comportamiento del modelo final propuesto, y como este cumple con lo propuesto para los criterios de optimización.

Finalmente, en el capítulo 5 se establecen varias proposiciones encontradas a lo largo de este trabajo, tanto al respecto de los resultados obtenidos, como de la forma de trabajo y modelamiento, y del cumplimiento de los objetivos propuestos. Además, se plantean diferentes caminos a seguir en posibles trabajos a futuro, incluyendo acercamientos interesante como la inclusión de la imprecisión del problema en el modelo final.

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

Antes de poder entrar en los subprocesos de postulación y selección es necesario definir los siguientes conceptos:

- Proceso de postulación: Proceso en el cual los alumnos entran al sistema de postulación, ven la oferta de ramos y vacantes de ayudantía, e ingresan las postulaciones que deseen con preferencias específicas y distintas por asignatura. Este proceso suele realizarse durante 1 mes antes del inicio del correspondiente semestre académico, dejando espacio para el proceso de selección y el proceso de vacantes no cubiertas.
- Proceso de selección: En este proceso, los profesores encargados de cada asignatura eligen un alumno candidato, para que este alumno sea asignado como ayudante de su asignatura. Este proceso se realiza luego del proceso de postulación y suele tomar 1 semana.
- Proceso de asignación: Proceso en el cual el sistema de postulación asigna a un alumno a una ayudantía. Actualmente, este proceso considera como input al proceso de selección anterior, es decir, se asignan los alumnos seleccionados por los profesores durante el proceso de selección.
- Criterios de selección: Se definen como criterios de selección a aquellos criterios que el profesor tiene en consideración al momento de elegir al alumno que será ayudante de su asignatura, y funcionan como parámetros que permiten diferenciar a un candidato de otro, y saber cuál se ajusta mejor a las consideraciones del profesor. Actualmente, los criterios considerados por cada profesor suelen ser distintos.

Actualmente, el macro-proceso de “programación de ayudantías” en el DI se realiza todos los semestres a través del Sistema de Postulación a Ayudantías (SPA). Este tiene muchos subprocesos, pero en este trabajo consideraremos solo 3: postulación, selección y asignación.

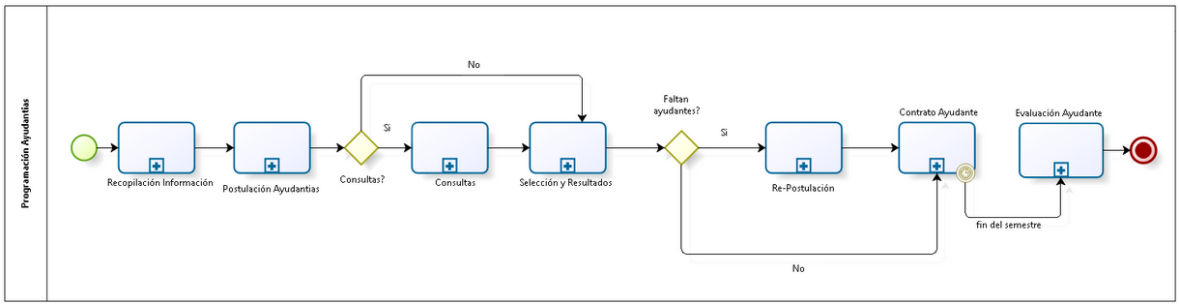


Figura 1: Macroproceso Programación Ayudantías.
Fuente: Modelado de Procesos de la Subdirección de Pregrado del Departamento de Informática de la UTFSM, Mauricio Muñoz.

Para poder aclarar los subprocesos de postulación y selección, podemos ver los siguientes diagramas:

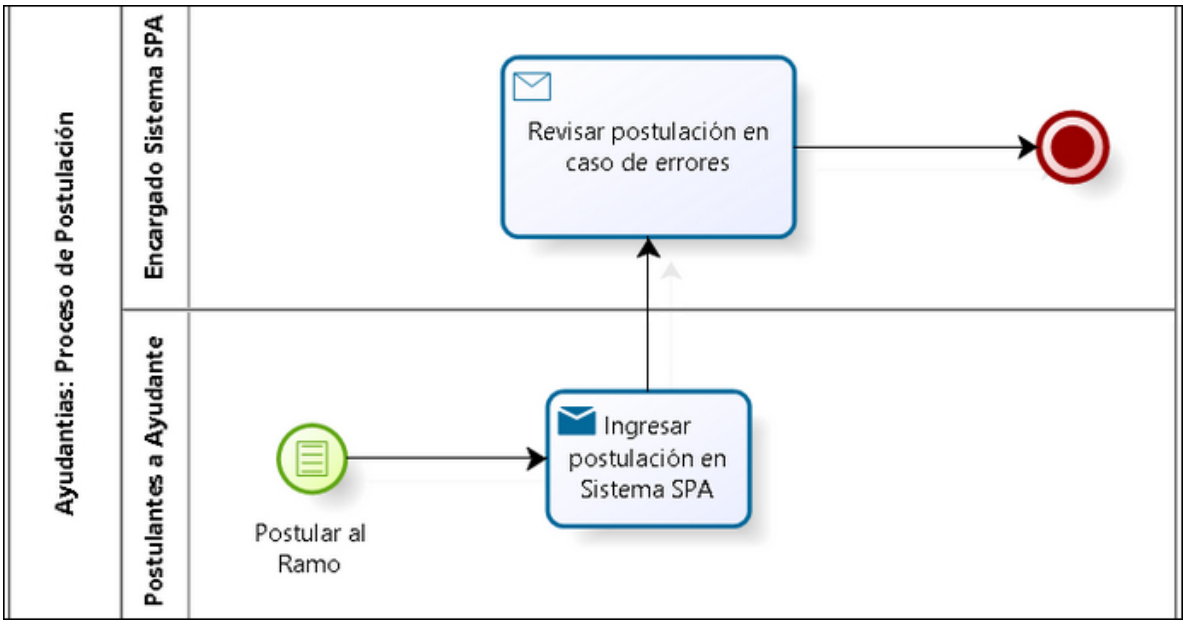


Figura 2: Proceso Postulación Ayudantías.
Fuente: Modelado de Procesos de la Subdirección de Pregrado del Departamento de Informática de la UTFSM, Mauricio Muñoz.

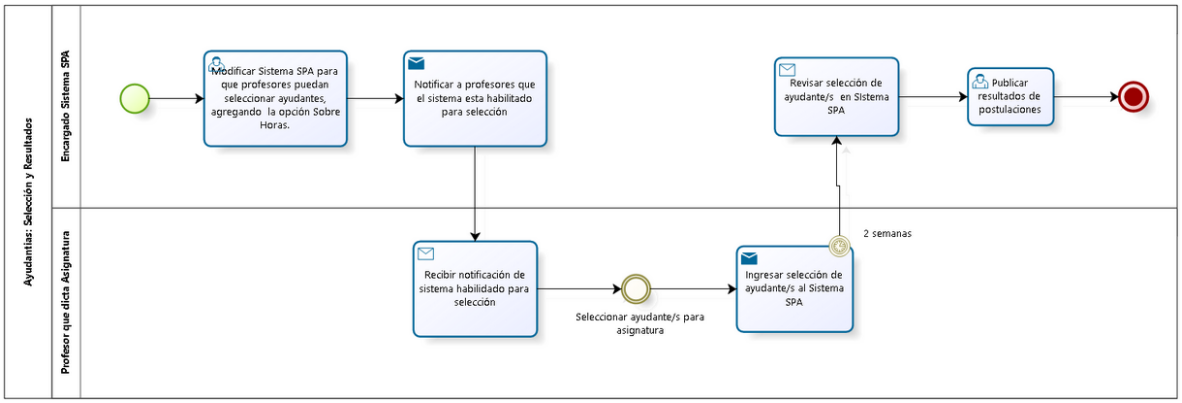


Figura 3: Proceso Selección y Resultados.
Fuente: Modelado de Procesos de la Subdirección de Pregrado del Departamento de Informática de la UTFSM, Mauricio Muñoz.

En otras palabras, luego de que los alumnos postulan a distintas ayudantías con distintas preferencias, los profesores de cada asignatura eligen al ayudante que les parezca más adecuado según sus propios criterios. Con esta definición del proceso surgen varios problemas:

- Asignación final de ayudantes no cubre todas las vacantes abiertas, por lo que se debe generar un segundo proceso de asignación.
- Asignación final considera elecciones según asignaturas, por lo que la solución final es un óptimo local del problema.

En el siguiente dibujo podemos ver una situación ficticia, en donde se pueden ver aquellos casos donde la asignación no es óptima para todas las asignaturas y, además, donde no son cubiertas todas las vacantes, siendo que quedan alumnos postulantes sin asignar.

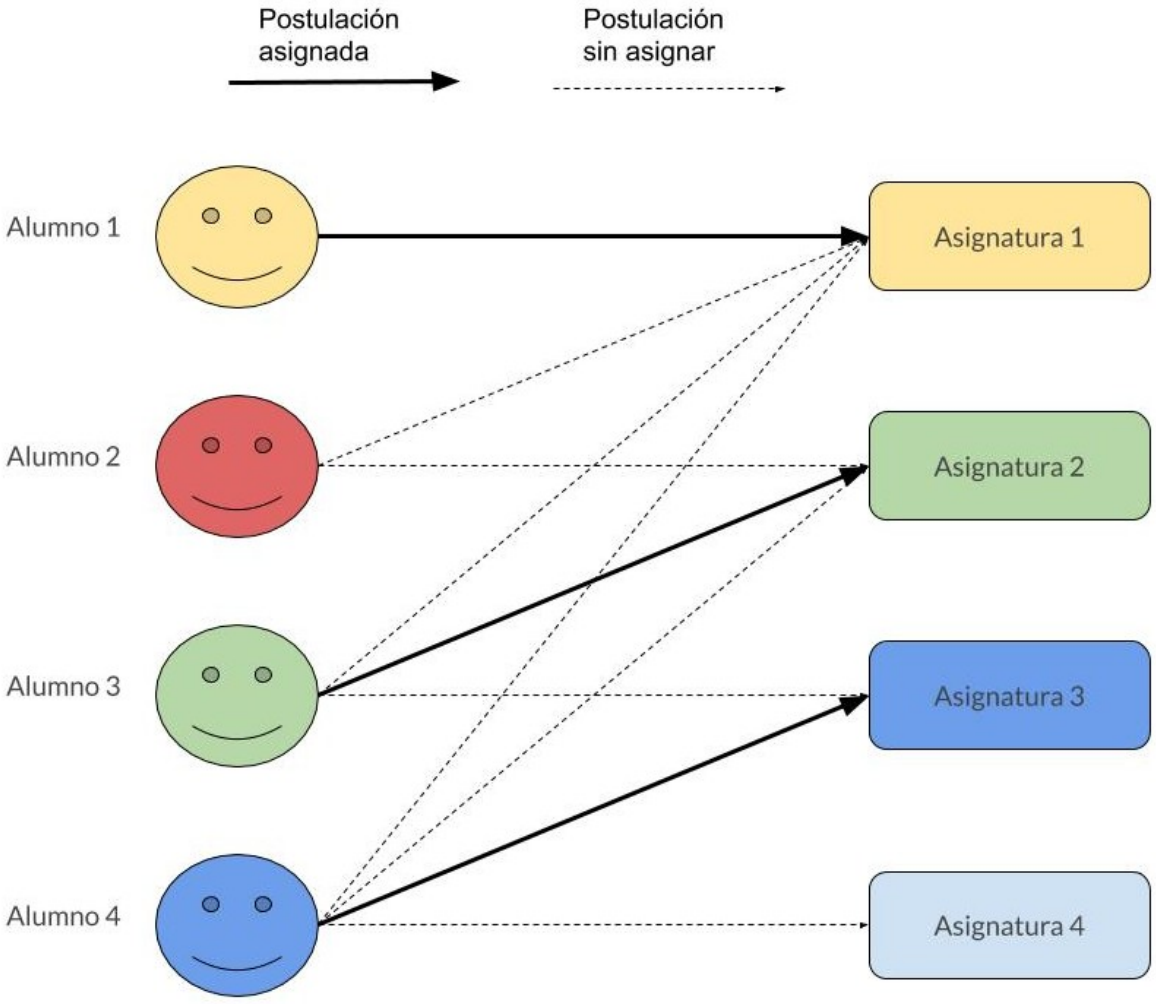


Figura 4: Asignación de ayudantes sub-óptima.
Fuente: Elaboración propia.

Es así como surgen las preguntas:

- ¿Cuál es la mejor forma de realizar el proceso de asignación de ayudantes, tomando en cuenta una visión global de las asignaturas, los criterios de selección y la influencia de preferencias de alumnos y profesores?
- ¿Es posible reformular los procesos de selección y asignación, considerando un modelo matemático para realizar el proceso de asignación de forma inteligente?

Podemos resumir las distintas aristas del problema con el siguiente árbol del problema:

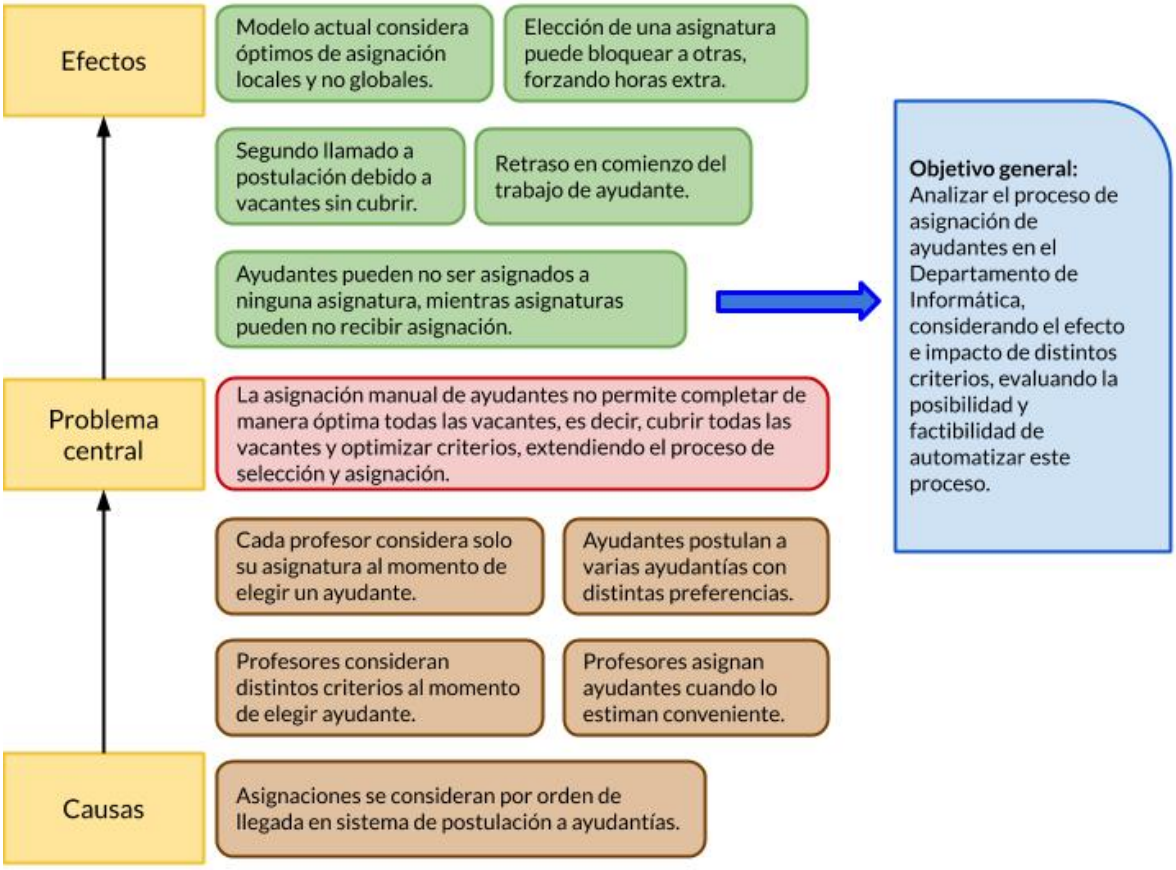


Figura 5: Árbol del problema.
Fuente: Elaboración propia.

1.1. Objetivo General

Analizar el proceso de asignación de ayudantes en el Departamento de Informática, considerando el efecto e impacto de distintos criterios, evaluando la posibilidad y factibilidad de automatizar este proceso.

1.2. Objetivos Específicos

1. Compilar criterios de asignación de ayudantes desde reglamentos de la universidad y experiencia de profesores involucrados en el sistema.
2. Simular el impacto de los distintos criterios de asignación mediante un modelo de matemático.
3. Proponer un nuevo proceso de asignación que incorpore los resultados obtenidos en la simulación.

CAPÍTULO 2
MARCO CONCEPTUAL

En general, el problema actual se puede modelar como un problema de asignación de recursos. En esta línea, nos encontramos con los problemas conocidos como AP (Asignation Problem), en los cuales se busca ver como asignar recursos de la mejor manera posible. A continuación veremos el problema de asignación generalizado.

2.1. Generalized Assignment Problem (GAP)

El problema de asignación generalizado examina el mínimo costo de asignar **m** tareas a **n** máquinas, de tal forma que cada tarea sea asignada a una única máquina, sujeto a las distintas restricciones en particular que pueda tener cada máquina (Capacidad de trabajo de la máquina). El GAP es un problema de optimización combinatorial NP-Hard. Los GAPs tienen muchas aplicaciones en problemas reales, por ejemplo, como subproblema en problemas de enrutamiento de vehículos [Fisher y Jaikumar, 1981], como problema de ubicación de instalaciones [Ross y Soland, 1977] o como problema de programación de recursos, programación de redes de proyectos, asignación de espacio de almacenamiento y programación de comerciales televisivos en ventanas de tiempo [Cattrysse y Van Wassenhove, 1992], entre muchos otros.

2.1.1. Modelo matemático

Existen bastantes definiciones para el GAP, siendo la más general la siguiente:

$$\text{(GAP) min} \quad \sum_{i \in I} \sum_{j \in J} c_{i,j} x_{i,j} \tag{1}$$

$$\text{sujeto a} \quad \sum_{j \in J} a_{i,j} x_{i,j} \leq b_i \qquad \forall i \in I \tag{2}$$

$$\sum_{i \in I} x_{i,j} = 1 \qquad \forall j \in J \tag{3}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall i \in I, j \in J \tag{4}$$

En donde I es el conjunto de **m** tareas y J es el conjunto de **n** máquinas.

$$I = \{1, 2, \ldots, m\} \tag{5}$$

$$J = \{1, 2, \ldots, n\} \tag{6}$$

Para esta definición tenemos la función objetivo (1) que busca minimizar el costo total de la asignación, siendo $c_{i,j}$ el costo asociado. Para $c_{i,j}$ existe una matriz de costos $C_{m \times n}$ que

almacena todos los costos de cada tarea para cada máquina.

$$C_{m\times n} = \begin{pmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,n} \\ c_{2,1} & c_{2,2} & \dots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & \dots & c_{m,n} \end{pmatrix}$$

Luego, la capacidad de cada máquina dada por (2), en donde se puede modelar si todas las máquinas son iguales, o si alguna puede tener asignada más de 1 tarea. Estas **n** restricciones generan la siguiente matriz y vector de parámetros:

$$A_{m\times n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix} \wedge \vec{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Para el caso más básico del problema de asignación, cuando la relación de tareas asignadas a máquinas es $1:1$, todos los parámetros tienen valor 1.

En seguida, las **m** restricciones (3), que cumplen con limitar la asignación de 1 tarea a 1 única máquina. Es necesario tener estas restricciones ya que una tarea no es divisible, solo puede ser realizada por aquella máquina a la cual fue asignada. Finalmente, tenemos la restricción (4), que señala que las variables son binarias, lo que categoriza al problema como un problema de programación entera.

Debido a que este problema es un problema de programación entera, se deben aplicar métodos de resolución específicos, ya que los métodos comunes para problemas lineales no pueden manejar la restricción (4). A continuación se explicará el como funciona la programación entera y como esta resuelve el GAP.

2.2. Transportation problem (TP)

El problema de transporte es un problema jerárquicamente superior al problema de asignación descrito anteriormente, en cuanto el problema de asignación describe un caso particular del problema de transporte. Este problema examina también el mínimo costo asociado a distribuir recursos de **m** fábricas a **n** sucursales, en donde las rutas de distribución tienen un costo asociado de transporte, el que se desea minimizar. La principal diferencia de este problema con el problema de asignación es que los recursos distribuidos no son unitarios, es decir, se pueden distribuir más de 1 recurso por sucursal (el problema anterior fijaba que cada máquina sólo podía tener asignada una única tarea). Al ser un problema más general que el problema de asignación, es posible modelar aún más situaciones con este tipo de problema. En general, el problema de transporte es usado para modelar problemas de redes de distribución.

2.2.1. Modelo matemático

Una definición común del TP es la siguiente:

$$\text{(TP) min} \quad \sum_{i \in I} \sum_{j \in J} c_{i,j} y_{i,j} \tag{7}$$

$$\text{sujeto a} \quad \sum_{i \in I} y_{i,j} \geq S_j \qquad \forall j \in J \tag{8}$$

$$\sum_{j \in J} y_{i,j} \leq F_i \qquad \forall i \in I \tag{9}$$

$$y_{i,j} \geq 0 \qquad \forall i \in I, j \in J \tag{10}$$

En donde I es el conjunto de **m** fábricas y J es el conjunto de **n** sucursales.

$$I = \{1, 2, \ldots, m\} \tag{11}$$

$$J = \{1, 2, \ldots, n\} \tag{12}$$

Para esta definición tenemos la función objetivo (7) que busca minimizar el costo total de la distribución, siendo $c_{i,j}$ el costo asociado. Para $c_{i,j}$ existe una matriz de costos $C_{m \times n}$ que almacena todos los costos de distribuir los recursos de cada fábrica a cada sucursal.

$$C_{m \times n} = \begin{pmatrix} c_{1,1} & c_{1,2} & \ldots & c_{1,n} \\ c_{2,1} & c_{2,2} & \ldots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & \ldots & c_{m,n} \end{pmatrix}$$

Luego, la capacidad de cada máquina dada por (2), en donde se puede modelar si todas las máquinas son iguales, o si alguna puede tener asignada más de 1 tarea. Estas **n** restricciones generan la siguiente matriz y vector de parámetros:

$$\vec{F} = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_m \end{pmatrix} \wedge \vec{S} = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_n \end{pmatrix}$$

El problema de transporte se puede entender también en términos de oferta y demanda, en donde las fábricas y sucursales modelan estas cantidades, respectivamente. Un problema de transporte se considera equilibrado si se cumple que la oferta y demanda totales son iguales, es decir:

$$\sum_{j \in J} S_j = \sum_{i \in I} F_j$$

Las restricciones (8) y (9) permiten representar un problema equilibrado, pero a la vez dan holgura en caso de tener oferta y demanda distintas. Finalmente, la restricción (10) es una de las principales diferencias del TP con el GAP, ya que el tipo de variable es entera.

TP también es un problema de programación lineal entera, por lo que los métodos usados para su resolución suelen ser similares a los usados en GAP.

2.3. Programación lineal entera

Los modelos discretos son una extensión de los modelos lineales, donde algunas variables toman valores enteros. Podemos clasificar los modelos de optimización lineal en 3 tipos según el dominio de sus variables:

- Si $x_{i,j} \in \mathbf{R}$, el problema es lineal.
- Si $x_{i,j} \in \mathbf{Z}$, el problema es lineal entero mixto.
- Si $x_{i,j} \in \{0, 1\}$, el problema es binario.

A diferencia de los modelos lineales, el cálculo de soluciones es mucho más costoso, debido a la cantidad de restricciones que agregan al modelo, incluso invalidando técnicas como simplex. Es por esto que los problemas enteros usan otros métodos para ser resueltos, los que enumeraremos a continuación:

2.3.1. Resolver problemas relajados

Consiste en resolver el problema ignorando la condición de dominio de las variables enteras. Luego, se redondea las variables que deben tener un valor entero como resultado. Esto trae los siguientes problemas:

- Muchas alternativas de redondeo.
- No garantiza resultados óptimos.
- No garantiza resultados factibles.

2.3.2. Planos de corte de Gomory

En esta técnica, se resuelve el problema original relajado en el que se incluyen restricciones adicionales, que reducen la región factible sin excluir soluciones que cumplen las condiciones de optimalidad. En cada iteración se añade una restricción que se denomina corte de Gomory. El método es descrito por el algoritmo a continuación:

- Paso 1. (iniciación). Se resuelve el problema original sin restricciones de integralidad. Si la solución no está acotada o el problema es infactible, se para el problema original o no está acotado o es infactible.
- Paso 2. (control de optimalidad). Si la solución obtenida cumple las condiciones de integralidad, se para dado que esta solución es óptima. En otro caso, se continúa con el paso siguiente.
- Paso 3. (generación de corte). Se emplea una variable básica que ha de ser entera pero no lo es para generar un corte de Gomory.
- Paso 4. (resolución). Se añade el corte de Gomory obtenido al problema previamente resuelto, se resuelve el problema resultante y se continúa con el paso 2.

Cabe destacar que el número de restricciones crece con el número de iteraciones. Puesto que en cada iteración se añade una nueva restricción, debe emplearse para la resolución del problema el MPE (método simplex dual). Esto es así puesto que al añadir una nueva restricción, el problema primal correspondiente se hace infactible pero su dual permanece factible, aunque no óptimo. Por tanto, para resolver el nuevo problema puede partirse de la solución dual del problema previo (sin restricción adicional), lo que supone una ventaja computacional importante [Castillo et al.,]. Este procedimiento genera progresivamente una envoltura convexa de la región factible entera-mixta, lo que origina soluciones que cumplen las condiciones de integralidad. Podemos ver este efecto en el siguiente dibujo:

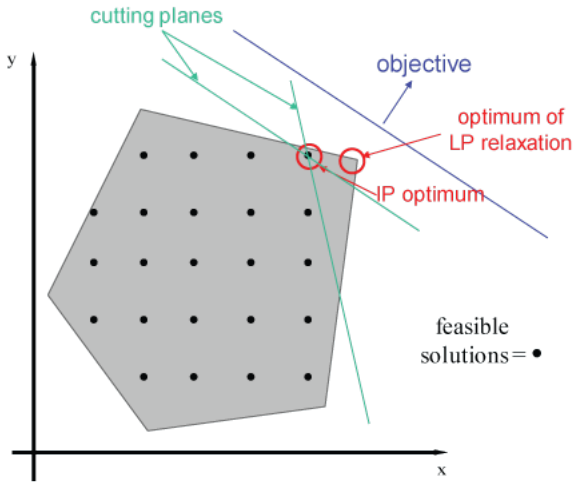


Figura 6: Planos de corte.
Fuente: [Gurobi Optimization, 2019].

2.3.3. Branch and bound

La técnica de Ramificación y Poda (Branch and bound) divide el problema introduciendo condiciones simples sobre los valores de las variables. Esto fuerza a las soluciones de las distintas partes a ser enteras, permitiendo enumerar implícitamente todas las posibles soluciones enteras para conocer la que posea un valor óptimo. Para ganar eficiencia, se calculan cotas sobre la función objetivo para eliminar subproblemas donde no puede estar la solución. El objetivo de este método es poder encontrar un valor x que minimice o maximice el valor de la función objetivo $f(x)$ dentro de un conjunto S de soluciones candidatas. En esencia, el método posee 2 etapas:

- **Branching:** También conocida como "ramificación", consiste en dividir el problema en subproblemas más pequeños. Por ejemplo, acotar por cada variable usando las funciones techo y piso (ceiling and floor) de un valor decimal como regla de ramificación. Este proceso se hace de forma recursiva, minimizando en cada etapa $f(x)$.
- **Bounding:** También conocida como "poda", consiste en descartar aquellos nodos que no cumplen con el criterio de optimización buscado. Esto se logra almacenando los mejores valores obtenidos en nodos anteriores para $f(x)$.

Se puede ver como el proceso de ramificación ayuda a explorar alternativas cercanas a la solución relajada, aplicando las restricciones de carácter entero. A su vez, el proceso de poda es igualmente importante, ya que de no existir la poda, la técnica se transformaría en un algoritmo de búsqueda completa (Backtracking). Además, evita que se revisen soluciones que no son óptimas, logrando reducir el tiempo en que la técnica encuentra una solución. El algoritmo formal para este método se describe de la siguiente forma:

- Sea $f(x)$: función objetivo, S : conjunto de soluciones candidatas e I : instancia del problema. A su vez, S_I : conjunto de soluciones candidatas para la instancia I.
- $branch(I)$: Produce 2 o más instancias, las cuales representan un subconjunto de S_I.
- $bound(I)$: calcula una cota inferior para el valor de cualquier solución candidata factible en la instancia I, es decir:

$$bound(I) \leq f(x) \qquad \forall x \in S_I$$

- $solution(I)$: determina si I representa a una única solución candidata. En caso contrario, retorna soluciones factibles dentro de S_I.

Podemos ver un ejemplo del funcionamiento del método para siguiente ejemplo, en el cual se puede apreciar la representación de árbol asociada a la optimización:

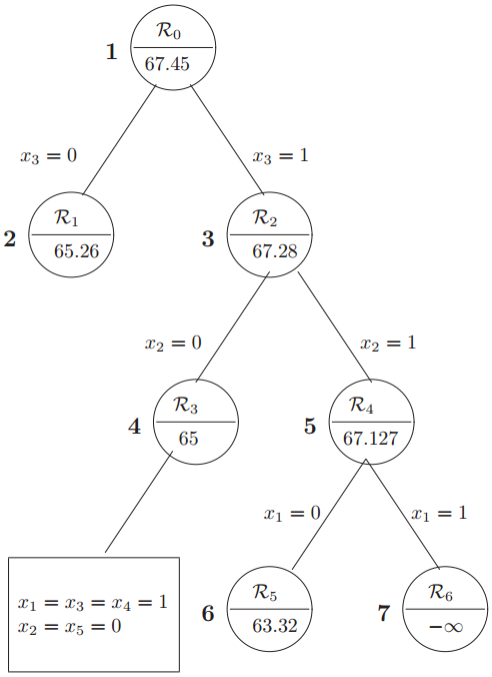


Figura 7: Resultado de Branch and Bound.
Fuente: [Gao, 2014].

En este ejemplo (problema de maximización) es posible ver como el método realiza la poda en la rama izquierda, llegando a una solución con el nodo 4 debido a que los otros nodos candidatos finales (6 y 7) no poseen un valor mejor.

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

Podemos ver el proceso actual de forma resumida en la siguiente imagen, remarcando los subprocesos Recopilación de información, Postulación de ayudantes, Selección de ayudantes y Extras.

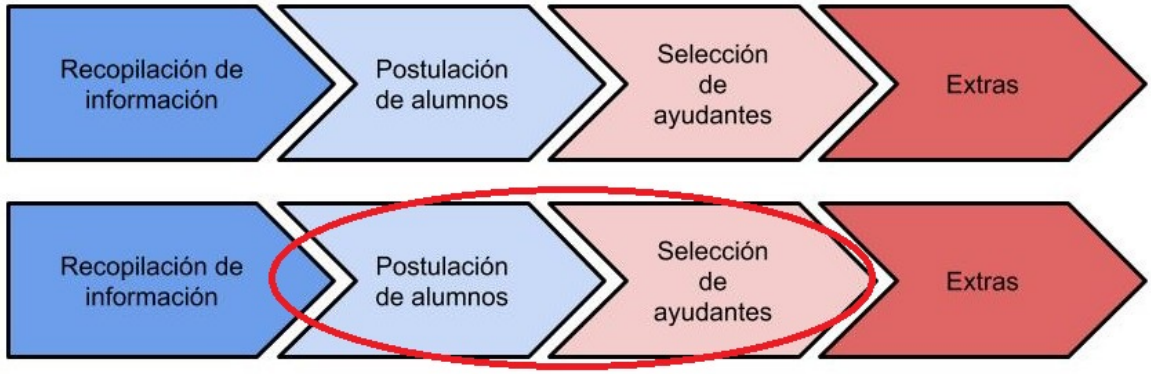


Figura 8: Versión resumida del proceso actual.
Fuente: Elaboración propia.

La propuesta de nuevo proceso se enfoca en alterar el proceso anterior, separando el proceso de selección en 2 nuevos subprocesos, Especificación de preferencias y Asignación inteligente, y modificando el subproceso Postulación de alumnos para incluir la asignación de preferencias de los alumnos. Podemos ver el cambio en la siguiente figura:

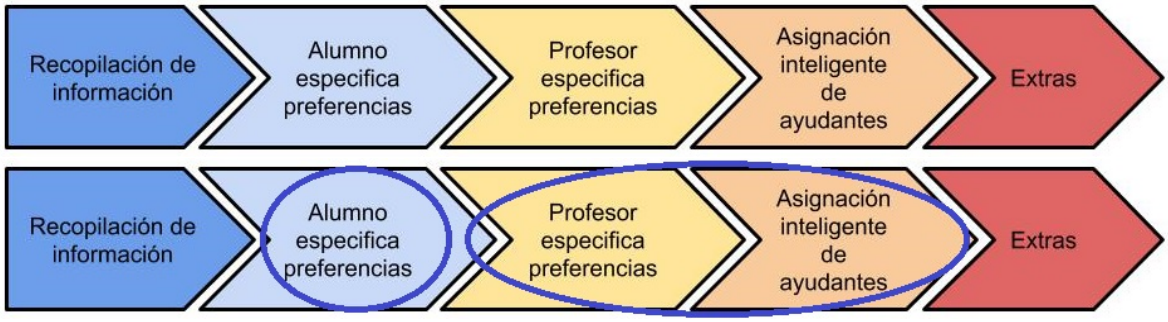


Figura 9: Versión resumida del proceso propuesto.
Fuente: Elaboración propia.

En síntesis, se propone el proceso con el siguiente flujo:

1. Se abre proceso de postulación a ayudantías.

2. Alumnos postulan a ayudantías, ingresando preferencias.
3. Se cierra proceso de postulación y se pasa a proceso de elección por parte de los profesores.
4. Profesor ingresa preferencias de alumnos que postularon a la ayudantía, asignando estas preferencias en base a los criterios cualitativos que considere más importantes.
5. Luego de que los profesores ingresen sus preferencias, se entregan los datos al modelo, y se obtiene la asignación óptima para las ayudantías, considerando ambos tipos de criterios.

La solución propuesta consta de 3 partes:

1. Alumno especifica preferencias: Cuando el alumno entra al sistema asigna distintas preferencias a las asignaturas a las cuales desea postular, indicando alta preferencia por aquella(s) asignatura(s) en la que desearía trabajar como ayudante, y baja preferencia para aquellas asignaturas que le sean indiferentes. Estos niveles de preferencia son variables, y por defecto una asignatura tiene preferencia 0, que significa que una asignación no es posible.
2. Profesor especifica preferencias: Luego de que los alumnos ingresen sus postulaciones al sistema, cada profesor entra al sistema y, en vez de elegir a un único ayudante postulante, asigna distintas preferencias a todos los postulantes, indicando alta preferencia por aquel o aquellos postulantes con los cuales le gustaría trabajar, y baja preferencia a aquellos con los que no desea trabajar. Estos niveles de preferencia son variables.
3. Asignación inteligente de ayudantes: Luego de que todos los profesores especifiquen las distintas preferencias, el sistema procederá a asignar a los alumnos postulantes a cada asignatura a través del uso de un modelo matemático lineal, el cual obtendrá una asignación que optimizará distintos criterios, logrando una asignación que toma en cuenta todas las asignaturas del sistema.

Pasaremos a dar más detalles de cada etapa en la siguientes secciones.

3.1. Criterios

3.1.1. Levantamiento de criterios

Con 14 respuestas, de mayor a menor, aquellos criterios con más del 50 % de preferencia:

1. Nota con la que aprobó el ramo: 14 (100 %).
2. Prioridad académica: 11 (79 %).
3. Semestres con experiencia como ayudante: 8 (57 %).

3.1.2. Criterios cualitativos

A continuación, enumeramos los criterios cualitativos recuperados de la encuesta, con 14 respuestas, de mayor a menor, aquellos con más del 50 % de preferencia:

1. Motivación del alumno: 13 (93 %).
2. Conocer al alumno: 11 (79 %).
3. Responsabilidad del alumno: 9 (64 %).
4. Experiencia previa como ayudante con el profesor: 8 (57 %).

Estos criterios recopilados son útiles para identificar y estandarizar la forma de preferencia sobre un estudiante o otro en la asignación. Se propone establecer un parámetro de preferencia (definido en detalle posteriormente), el que agrupará todos estos criterios de forma implícita.

3.2. Alumno especifica preferencias

En primera instancia, cada alumno que desea postular a alguna asignatura ingresa al sistema de postulación de ayudantías, elige la asignatura en la que desea trabajar durante el semestre y registra una postulación. El nuevo proceso, llamado **Alumno especifica preferencias**, plantea que el alumno al momento de hacer la postulación a la asignatura asignará un valor de preferencia por aquella asignatura, y así la postulación será registrada en el sistema. La preferencia es un variable categórica que puede variar entre los siguientes estados:

- **Máxima Preferencia:** Si el alumno especifica que cierta asignatura tiene **Máxima Preferencia**, esto significa que el sistema ponderará está preferencia como el mayor valor respecto de las otras, lo que favorece la asignación de este alumno. Esta preferencia se diseña con el objetivo de que el alumno la aplique en las asignaturas en las que desea trabajar guiado por alguna motivación.
- **Preferencia Media:** Si el alumno especifica que cierta asignatura tiene **Preferencia Media**, esto significa que el sistema ponderará está preferencia como un valor neutro

respecto de las otras, lo que mantiene la probabilidad base de que este alumno sea asignado como ayudante. Esta preferencia se diseña con el objetivo de que el alumno la aplique en las asignaturas en las que desea trabajar, pero sin una motivación profunda, reflejando un pensamiento como: “no me molestaría ser ayudante en este ramo”.

- **Preferencia Nula:** Una asignatura con **Preferencia Nula** significa que el sistema no considerará esta preferencia, lo que tiene como efecto que el sistema no tome en cuenta las asignaturas asociadas a esta preferencia. Esta preferencia se diseña para ser usada como valor por defecto para el alumno en las asignaturas, evitando así que el sistema considere al alumno para la asignatura. La única forma de cambiar esta preferencia es que el alumno especifique una preferencia distinta para la asignatura, registrando así una postulación.

Este nuevo proceso presenta distintas ventajas y desventajas. A continuación se detallan las ventajas del proceso nuevo:

- Es una mejora en todo aspecto respecto al proceso de postulación: Esto debido a que en el peor de los casos, este proceso se comporta como el proceso anterior de postulación, ya que la existencia de la **Preferencia Nula** permite a los alumnos dejar fuera de la postulación a las asignaturas que ellos deseen, y volver a la forma de postulación anterior, postulando a una o varias asignaturas usando solo la **Preferencia Máxima**. Debido a esta particularidad, cualquier otro caso que no sea el mencionado será una mejora, debido a que la postulación tendrá en cuenta un mayor número de alumnos candidatos para ser asignados como ayudantes.
- Favorece la interacción de los alumnos con el sistema: El proceso anterior suele tener problemas por falta de participación por parte del alumnado. Con este nuevo proceso, se pueden obtener combinaciones de alumnos y asignaturas que antes no existían debido a la rigidez de la postulación. En vez de postular a 1 o 2 asignaturas, con el sistema de preferencias, un alumno podría postular a más asignaturas, favoreciendo un mayor número de combinaciones para que el sistema pueda asignar.
- Permite manifestar criterios de selección de cada alumno: En este nuevo proceso, el alumno puede manifestar su nivel de preferencia para trabajar en una asignatura, valor que puede ser usado dentro de los parámetros que el sistema busca optimizar.

3.3. Profesor especifica preferencias

Como se mencionaba anteriormente, una vez terminado el proceso previo, cada profesor pasa a seleccionar al alumno postulante que quiere que sea su ayudante por el semestre. El nuevo proceso, llamado **Profesor especifica preferencias**, plantea que el profesor ya no debe hacer la selección de los alumnos, sino que debe especificar el nivel de preferencia respecto

de cada alumno que postula a su asignatura. La preferencia es un variable categórica que puede variar entre los siguientes estados:

- **Máxima Preferencia:** Si el profesor especifica que cierto alumno tiene **Máxima Preferencia**, esto significa que el sistema ponderará está preferencia como el mayor valor respecto de las otras, lo que aumenta la probabilidad de que este alumno sea asignado como ayudante. Esta preferencia se diseña con el objetivo de que el profesor la aplique en los postulantes que cumplen con todos los criterios que él busca en su ayudante.
- **Alta Preferencia:** Si el profesor especifica que cierto alumno tiene **Alta Preferencia**, esto significa que el sistema ponderará está preferencia como el segundo mayor valor respecto de las otras, lo que favorece la asignación de este alumno, pero no por sobre uno de **Máxima Preferencia**. Esta preferencia se diseña con el objetivo de que el profesor la aplique en los postulantes que cumplen con la mayoría de criterios que él busca en su ayudante, considerándolo como segunda mejor opción.
- **Preferencia Media:** Si el profesor especifica que cierto alumno tiene **Preferencia Media**, esto significa que el sistema ponderará está preferencia como un valor neutro respecto de las otras, lo que mantiene la probabilidad inicial de que este alumno sea asignado como ayudante. Esta preferencia se diseña con el objetivo de que el profesor la aplique en los postulantes que cumplen con algunos de criterios que él busca en su ayudante, considerando al alumno postulante como una opción aceptable, pero peor que los postulantes anteriores.
- **Baja Preferencia:** Si el profesor especifica que cierto alumno tiene **Baja Preferencia**, esto significa que el sistema ponderará está preferencia como un valor negativo, lo que reduce la probabilidad de que este alumno sea asignado como ayudante. Esta preferencia se diseña con el objetivo de que el profesor la aplique en los postulantes que no cumplen con la mayoría de criterios que él busca en su ayudante, considerando al alumno postulante como una opción no aceptable, pero que igual está dispuesto a aceptar en caso de que sea asignado.
- **Preferencia Nula:** Si el profesor especifica que cierto alumno tiene **Preferencia Nula**, esto significa que el sistema no considerará esta preferencia, lo que tiene como efecto que el sistema no tome en cuenta las postulaciones asociadas a esta preferencia. Esta preferencia se diseña con el objetivo de que el profesor la aplique en los postulantes que no cumplen con ninguno de los criterios que él busca en su ayudante, considerando al alumno postulante como una opción no aceptable, pero que además el profesor no está dispuesto a aceptar como asignación para su asignatura.

Podemos ver en la siguiente imagen como se compara el proceso de selección con el proceso de especificación de preferencias:

Figura 10: Selección de ayudantes v/s Selección propuesta, especificando preferencias. Fuente: Elaboración propia.

Este nuevo proceso presenta distintas ventajas y desventajas. A continuación se detallan las ventajas del proceso nuevo:

- Es una mejora en todo aspecto respecto al proceso de selección: Esto debido a que en el peor de los casos, este proceso se comporta como el proceso anterior de selección, ya que la existencia de la **Preferencia Nula** permite a los profesores dejar fuera de la asignación a los postulantes que ellos deseen, y volver a la forma de selección anterior, eligiendo solo a un postulante. Debido a esta particularidad, cualquier otro caso que no sea el mencionado será una mejora, debido a que la asignación tendrá en cuenta un mayor número de postulantes candidatos para ser asignados como ayudantes.
- Aumenta el número de postulantes candidatos a asignación: Si en el proceso anterior se tenía en cuenta solo 1 alumno por asignatura como candidato para la asignación, ahora se puede tener 1 o más alumnos por asignatura como candidatos a asignación.
- Permite manifestar criterios de selección de cada profesor: En este nuevo proceso, el profesor puede manifestar el cumplimiento de los criterios a través del sistema de preferencias, lo que incluso abre la posibilidad de trabajos futuros respecto al análisis global de preferencias.

3.4. Asignación inteligente de ayudantes

Para realizar la asignación inteligente, se modela el problema para ser resuelto usando técnicas de programación lineal, aprovechando la nueva definición de preferencias para representar la influencia de los distintos criterios existentes.

Existen distintas alternativas de modelos, que se basan en como es planteado el problema. En este trabajo se proponen 4 modelos, los que fueron diseñados de forma iterativa, logrando obtener en el último modelo una representación general de los modelos anteriores. En las siguientes secciones veremos los parámetros asociados a los modelos, la definición de los modelos propuestos, y un detalle sobre el modelo final y sus distintas configuraciones.

3.4.1. Parámetros

Los parámetros definidos a continuación son válidos para todos los modelos. Los primeros 3 modelos no hacen uso de todos los parámetros, debido a la especificidad con la que fueron hechos. Se incluye además quien define los parámetros en caso de que involucren ser definidos por alguien quien interactúe con el sistema.

- m: Cantidad de estudiantes postulantes a ayudantías.
- n: Cantidad de asignaturas.
- ha_j: Cantidad total de horas de ayudantía requeridas para la asignatura j (especificado por profesor).
- na_max_j: Cantidad máxima de ayudantes deseada para la asignatura j (especificado por profesor).
- na_min_j: Cantidad mínima de ayudantes deseada para la asignatura j (especificado por profesor).
- $pp_{i,j}$: Preferencia del profesor j por el alumno i (especificada por el profesor).
- he_max_i: Cantidad máxima de horas de ayudantía que puede realizar el estudiante i (especificado según tipo de estudiante).
- he_min_i: Cantidad mínima de horas de ayudantía que desea realizar el estudiante i (especificado por estudiante).
- ne_max_i: Cantidad máxima de ayudantías deseadas por el estudiante i (especificado por estudiante).
- ne_min_i: Cantidad mínima de ayudantías deseadas por el estudiante i (especificado por estudiante).
- $pe_{i,j}$: Preferencia del estudiante i por la asignatura j (especificada por el estudiante).

3.4.2. Modelo 1

El modelo de asignación más simple, pensado en el problema de asignación de ayudantes, es definido de la siguiente forma:

1. Variables:
   - $x_{i,j}$: Se asigna alumno i a asignatura j.
   - $x \in \{1, 0\} \; \forall \, i = 1, \ldots, m; \; j = 1, \ldots, n$
2. Restricciones:
   - Se asignan ayudantes según necesidades de cada asignatura:
     - Cantidad máxima de ayudantes a asignar para cada asignatura:

$$\sum_{i=1}^{m} x_{ij} \leq na\_max_j \quad \forall \, j = 1, \ldots, n$$

     - Cantidad mínima de ayudantes a asignar para cada asignatura:

$$\sum_{i=1}^{m} x_{ij} \geq na\_min_j \quad \forall \, j = 1, \ldots, n$$

   - Se asignan ayudantes según necesidades de cada estudiante:
     - Cantidad máxima de ayudantías para cada estudiante:

$$\sum_{j=1}^{n} x_{ij} \leq ne\_max_i \quad \forall \, i = 1, \ldots, m$$

     - Cantidad mínima de ayudantías para cada estudiante:

$$\sum_{j=1}^{n} x_{ij} \geq ne\_min_i \quad \forall \, i = 1, \ldots, m$$

3. Función Objetivo:

$$\text{Maximizar: } \sum_{j=1}^{n} \sum_{i=1}^{m} x_{i,j} \cdot pe_{i,j} + \sum_{j=1}^{n} \sum_{i=1}^{m} x_{i,j} \cdot pp_{i,j}$$

Respecto de este primer modelo, es un modelo bastante simple, pero nos ayuda a modelar la asignación de estudiante a asignatura, maximizando preferencias.

3.4.3. Modelo 2

Un segundo modelo a considerar es aquel en el que la variable a asignar es la cantidad de horas de ayudantía para cada asignación:

1. Variables:
   - $y_{i,j}$: Cantidad de horas de ayudantía asignadas para el estudiante i en la asignatura j; $y_{i,j} \in [0, ..., h_i] \; \forall \; i = 1, \ldots, m; j = 1, \ldots, n$
2. Restricciones:
   - Se asignan ayudantes según necesidades de cada asignatura:
     - Cantidad total máxima de horas de ayudantía a asignar para cada asignatura:
     $$\sum_{i=1}^{m} y_{ij} \leq ha_j \quad \forall \; j = 1, \ldots, n$$
   - Se asignan ayudantes según necesidades de cada estudiante:
     - Cantidad máxima de horas de ayudantía para cada estudiante:
     $$\sum_{j=1}^{n} y_{ij} \leq he\_max_i \quad \forall \; i = 1, \ldots, m$$
     - Cantidad mínima de horas de ayudantía para cada estudiante:
     $$\sum_{j=1}^{n} y_{ij} \geq he\_min_i \quad \forall \; i = 1, \ldots, m$$
3. Función Objetivo:

$$\text{Maximizar: } \sum_{j=1}^{n} \sum_{i=1}^{m} y_{i,j} \cdot pe_{i,j} + \sum_{j=1}^{n} \sum_{i=1}^{m} y_{i,j} \cdot pp_{i,j}$$

Respecto de este segundo modelo, no modela directamente la asignación ,lo que trae problemas al considerar la importancia de las preferencias, pero nos ayuda a modelar la asignación de horas de ayudantía, que es un factor muy importante a tomar en cuenta cuando el problema se ve como un problema de distribución de recursos.

3.4.4. Modelo 3

Como tercer modelo se propone un acercamiento híbrido, tomando ideas de ambos modelos anteriores. Este modelo se encarga de manejar asignaciones y cantidad de horas a la vez. Este modelo es más general que los anteriores, ya que, usando restricciones específicas sobre el mismo, se pueden obtener los modelos anteriores.

1. Variables:
   - $x_{i,j}$: Se asigna alumno i a asignatura j; $x \in \{1, 0\} \; \forall \, i = 1, \ldots, m; \; j = 1, \ldots, n$
   - $y_{i,j}$: Cantidad de horas de ayudantía asignadas para el estudiante i en la asignatura j; $\; y_{i,j} \in [0, ..., h_i] \; \forall \, i = 1, \ldots, m; j = 1, \ldots, n$
2. Restricciones:
   - Se asignan ayudantes según necesidades de cada asignatura:
     - Cantidad total máxima de horas de ayudantía a asignar para cada asignatura:

$$\sum_{i=1}^{m} y_{ij} \leq ha_j \quad \forall \, j = 1, \ldots, n$$

     - Cantidad máxima de ayudantes a asignar para cada asignatura:

$$\sum_{i=1}^{m} x_{ij} \leq na\_max_j \quad \forall \, j = 1, \ldots, n$$

     - Cantidad mínima de ayudantes a asignar para cada asignatura:

$$\sum_{i=1}^{m} x_{ij} \geq na\_min_j \quad \forall \, j = 1, \ldots, n$$

   - Se asignan ayudantes según necesidades de cada estudiante:
     - Cantidad máxima de horas de ayudantía para cada estudiante:

$$\sum_{j=1}^{n} y_{ij} \leq he\_max_i \quad \forall \, i = 1, \ldots, m$$

     - Cantidad mínima de horas de ayudantía para cada estudiante:

$$\sum_{j=1}^{n} y_{ij} \geq he\_min_i \quad \forall \, i = 1, \ldots, m$$

     - Cantidad máxima de ayudantías para cada estudiante:

$$\sum_{j=1}^{n} x_{ij} \leq ne\_max_i \quad \forall \, i = 1, \ldots, m$$

     - Cantidad mínima de ayudantías para cada estudiante:

$$\sum_{j=1}^{n} x_{ij} \geq ne\_min_i \quad \forall \, i = 1, \ldots, m$$

- Relaciones entre variables:
  - No se pueden asignar horas de ayudantías a un estudiante no asignado, cota superior:

$$y_{ij} \leq x_{ij} \cdot he\_max_i \quad \forall\, i = 1, \ldots, m; j = 1, \ldots, n$$

  - No se pueden asignar horas de ayudantías a un estudiante no asignado, cota inferior:

$$y_{ij} \geq x_{ij} \cdot he\_min_i \quad \forall\, i = 1, \ldots, m; j = 1, \ldots, n$$

3. Función Objetivo:

$$\text{Maximizar: } \sum_{j=1}^{n}\sum_{i=1}^{m} y_{i,j} + \sum_{j=1}^{n}\sum_{i=1}^{m} x_{i,j} \cdot pe_{i,j} + \sum_{j=1}^{n}\sum_{i=1}^{m} x_{i,j} \cdot pp_{i,j}$$

Este modelo ya es bastante general por si mismo, incluye ambos enfoques anteriores, y nos permite representar una gran cantidad de situaciones al incorporar todos los parámetros definidos. Este modelo define una base sólida para el modelo final.

3.4.5. Modelo 4

El modelo 3 cumple con modelar una buena cantidad de situaciones. Trabajando sobre este mismo modelo, se propone un cuarto modelo final, que trabaja incluyendo modificaciones sobre el modelo 3. Este modelo 4 es el modelo usado para realizar las asignaciones y, por ende, validar la propuesta.

1. Variables:
   - $x_{i,j}$: Se asigna alumno i a asignatura j; $x \in \{1, 0\}\ \forall\, i = 1, \ldots, m;\ j = 1, \ldots, n$
   - $y_{i,j}$: Cantidad de horas de ayudantía asignadas para el estudiante i en la asignatura j; $y_{i,j} \in [0, ..., h_i]\ \forall\, i = 1, \ldots, m; j = 1, \ldots, n$
   - ya_j : Cantidad de horas de ayudantía satisfechas en la asignatura j; $\forall\ j = 1, \ldots, n$
   - za_j : Cantidad de horas de ayudantía no satisfechas en la asignatura j; $\forall\ j = 1, \ldots, n$
   - xpe : Valor total de las preferencias de alumnos asignadas.
   - xpp : Valor total de las preferencias de profesores asignadas.
2. Restricciones:
   - Se asignan ayudantes según necesidades de cada asignatura:

- Cantidad total máxima de horas de ayudantía a asignar para cada asignatura:

$$\sum_{i=1}^{m} y_{ij} \leq ha_j \quad \forall\, j = 1, \ldots, n$$

- Cantidad máxima de ayudantes a asignar para cada asignatura:

$$\sum_{i=1}^{m} x_{ij} \leq na\_max_j \quad \forall\, j = 1, \ldots, n$$

- Cantidad mínima de ayudantes a asignar para cada asignatura:

$$\sum_{i=1}^{m} x_{ij} \geq na\_min_j \quad \forall\, j = 1, \ldots, n$$

- Se asignan ayudantes según necesidades de cada estudiante:
  - Cantidad máxima de horas de ayudantía para cada estudiante:

$$\sum_{j=1}^{n} y_{ij} \leq he\_max_i \quad \forall\, i = 1, \ldots, m$$

  - Cantidad mínima de horas de ayudantía para cada estudiante:

$$\sum_{j=1}^{n} y_{ij} \geq he\_min_i \quad \forall\, i = 1, \ldots, m$$

  - Cantidad máxima de ayudantías para cada estudiante:

$$\sum_{j=1}^{n} x_{ij} \leq ne\_max_i \quad \forall\, i = 1, \ldots, m$$

  - Cantidad mínima de ayudantías para cada estudiante:

$$\sum_{j=1}^{n} x_{ij} \geq ne\_min_i \quad \forall\, i = 1, \ldots, m$$

- Relaciones entre variables:
  - Preferencias de estudiantes:

$$xpe = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j} \cdot pe_{i,j} \quad \forall\, i = 1, \ldots, m; j = 1, \ldots, n$$

  - Preferencias de profesores:

$$xpp = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j} \cdot pp_{i,j} \quad \forall\, i = 1, \ldots, m; j = 1, \ldots, n$$

- Cantidad total de horas de ayudantía asignadas a la asignatura:

$$ya_j = \sum_{i=1}^{m} y_{ij} \quad \forall\, j = 1, \ldots, n$$

- Cantidad de horas de ayudantía no satisfechas en la asignatura:

$$za_j = ha_j - ya_j \quad \forall\, j = 1, \ldots, n$$

- No se pueden asignar horas de ayudantías a un estudiante no asignado, cota superior:

$$y_{ij} \leq x_{ij} \cdot he\_max_i \quad \forall\, i = 1, \ldots, m; j = 1, \ldots, n$$

- No se pueden asignar horas de ayudantías a un estudiante no asignado, cota inferior:

$$y_{ij} \geq x_{ij} \cdot he\_min_i \quad \forall\, i = 1, \ldots, m; j = 1, \ldots, n$$

3. Posibles funciones objetivo:

$$\text{Maximizar: } \alpha \sum_{j=1}^{n} ya_j + \beta \cdot xpp + \gamma \cdot xpe$$

$$\text{Maximizar: } -\alpha \sum_{j=1}^{n} za_j + \beta \cdot xpp + \gamma \cdot xpe$$

$$\text{Maximizar: } -\alpha \sum_{j=1}^{n} \frac{za_j}{ha_j} + \beta \cdot xpp + \gamma \cdot xpe$$

Este modelo final propone varias funciones objetivo, estas serán analizadas en la siguiente sección, para determinar como funciona cada función objetivo, que optimiza y que tan buenas soluciones nos entrega.

Además de esto, vemos que este modelo incorpora nuevas variables. Todas estas variables se definieron con el objetivo de poder saber que es lo que se intenta optimizar en el problema, además de poder tener algún indicador de la calidad de la asignación que permita poder comparar modelos. En las siguiente sección se realizarán distintas pruebas con distintos casos, para poder comparar y obtener un modelo final concreto válido para el problema. Estos modelos presentan algunos problemas, a continuación estos se detallarán y se verá como resolverlos.

- **Modelo considera que todos los alumnos postulan a todas las asignaturas:** En la realidad esto no aplica, ya que los alumnos postulan a aquellas asignaturas que les interesan. Para resolver este problema se trabaja sobre la matriz de preferencias del estudiante, asignando una preferencia cercana a $-\infty$ a aquellas asignaturas a las cuales el alumno no postula. Esto permite al modelo no considerar aquellas alternativas para la asignación final, de forma similar a lo que sucede con aquellas postulaciones con **Prioridad Nula.**

CAPÍTULO 4
ANÁLISIS DE RESULTADOS

Para poder probar los modelos propuestos como parte de la solución, se propone un análisis basado en una grilla de corridas del modelos, variando parámetros de interés como los casos a resolver y las funciones objetivo usadas, además de analizar varias métricas asociadas al desempeño del modelo. En una segunda etapa, se procede a validar el comportamiento de los modelos mediante la interpretación de las variables designadas como métricas.

4.1. Método de grillado

4.1.1. Casos a resolver

Los casos a resolver serán aquellos casos considerados como relevantes para tener en consideración el desempeño del modelo. Un caso se define como un conjunto específico de parámetros que modela una situación. Se proponen los siguientes elementos mínimos a considerar en la generación de los casos:

- Tamaño del espacio de búsqueda: Es necesario considerar el comportamiento del modelo en asignaciones con una cantidad de variables distinta, y explorar el tiempo de solución. Considerando como ejemplo la asignación del semestre 2019-2 en el DI, en donde se asignan alrededor de 30 ayudantes, es necesario también abordar casos con parámetros de alumnos y asignaturas que superen este valor.
- Flexibilidad de restricciones: El modelo propuesto incluye parámetros asociados a flexibilidad de restricciones, como la cantidad de ayudantes mínima y máxima. Es necesario comprobar que sucede en casos en que se usan efectivamente estos parámetros, casos en los que no.

Tomando en cuenta estos elementos y priorizando sólo considerar los factores que sean necesarios por motivos de simpleza, se proponen los siguientes casos.

1. Tamaño del espacio de búsqueda: Podemos notar que la dimensionalidad del problema está ligada a los parámetros m y n. Se probarán 4 casos con los valores $(20; 20), (40; 40), (40; 20), (20; 40)$ para $(m; n)$, respectivamente.
2. Flexibilidad de restricciones: Se propone probar 2 casos, el primero en donde todos los parámetros que incluyan min y max sean iguales y el otro en donde todos sean distintos.

Esto nos deja con un total de $2 \times 4 = 8$ casos basados en las combinaciones de los anteriores, que serán probados junto con las distintas funciones objetivo.

4.1.2. Funciones objetivo

Como se propuso anteriormente, serán 3 funciones objetivos las que se probaran en esta sección

$$\text{Maximizar: } f1 = \alpha \sum_{j=1}^{n} ya_j + \beta \cdot xpp + \gamma \cdot xpe$$

$$\text{Maximizar: } f2 = -\alpha \sum_{j=1}^{n} za_j + \beta \cdot xpp + \gamma \cdot xpe$$

$$\text{Maximizar: } f3 = -\alpha \sum_{j=1}^{n} \frac{za_j}{ha_j} + \beta \cdot xpp + \gamma \cdot xpe$$

Nombramos a estas funciones como $f1$, $f2$ y $f3$ para poder diferenciarlas posteriormente. En particular, cada función cumple un objetivo en específico además de maximizar las preferencias del problema (común para todas), detallado a continuación:

- $f1$: La función $f1$ busca maximizar la cantidad de horas de ayudantía asignadas a cada asignatura mediante la variable y_a. Esto cumple con buscar una asignación que, junto con maximizar preferencias, pueda efectivamente asignar un óptimo evitando que no se cumplan requisitos de horas.
- $f2$: La función $f2$ tiene un rol similar a la anterior, solo que ésta busca ahora minimizar la cantidad de horas de asignatura no asignadas mediante la variable z_a. Esta aborda la optimización de las asignaciones desde otro punto de vista comparado con $f1$.
- $f3$: La función $f3$ se basa en $f2$ para minimizar las horas no asignadas, pero busca hacerlo de forma normalizada, es decir, se busca la razón de cantidad de horas faltantes para una asignatura con su cantidad de horas totales, para evitar casos en donde la minimización se haga para asignaturas con una gran cantidad de horas, por ejemplo, dejando de lado a asignaturas con pocas horas totales.

Además, podemos ver que estas 3 funciones poseen parámetros de peso α, β, γ. Estos parámetros también serán contrastados en el análisis. El objetivo de estos parámetros es alterar la ponderación de cada factor en la evaluación de la función objetivo. En particular:

- α: Pondera el término asociado a la demanda de horas del problema. Se dejará fijo en 1, ya que aumentar o no la ponderación de la demanda no debería tener mayor efecto en la maximización de las preferencias.
- β y γ: Ponderan los términos asociados a la preferencia de los profesores. Se probarán valores que sean complementarios sumando 1, específicamente, los valores $[(0;1),(0{,}5;0{,}5),(1;0)]$ para β y γ, respectivamente.

Considerando los valores por parámetro, esto nos deja un total de $1 \times 3 = 3$ combinaciones de parámetros. Además, cada una de estas combinaciones se realizará para cada función objetivo, por lo tanto, se tendrán $3 \times 3 = 9$ combinaciones de funciones objetivo y parámetros.

4.1.3. Métricas

Para poder obtener una interpretación de los resultados y poder compararlos y validar nuestro modelo son necesarias métricas. Se consideran métricas para casos individuales y métricas para conjuntos de casos. A continuación proponemos las siguientes métricas para casos individuales:

- Variable xpe: Está variable nos permite ver como han sido consideradas las preferencias de los estudiantes en la asignación, sin la influencia del coeficiente asociado de la función objetivo γ.
- Variable xpp: De forma análoga, está variable nos permite ver como han sido consideradas las preferencias de los profesores en la asignación, sin la influencia del coeficiente asociado de la función objetivo β.
- Variable z_a: Esta variable también nos indica si la demanda de horas es resuelta, y que tanto déficit terminar por poseer.
- Tiempo de solución: El tiempo en que toma al modelo encontrar una solución al problema. Se fija un timeout de 5 minutos por modelo, por lo que tiempos mayores a este no serán considerados.

Para conjuntos de casos se proponen las siguientes métricas:

- Optimalidad de las soluciones: Ya que probaremos casos agrupados por categorías como tamaño, es interesante considerar cuantas de las soluciones encontradas por el modelo son consideradas óptimas.
- Soluciones dentro del timeout: También es interesante considerar si el solver puede encontrar un número de soluciones relevantes dentro del timeout especificado anteriormente.

- Agregación de métricas individuales: Además, podemos reportar métricas agregadas, como tiempo promedio, xpp y xpe promedio, y z_a promedio.

4.2. Resultados

Si consideramos ambas subsecciones anteriores, la cantidad de casos que fueron probados son $8 \times 9 = 72$, por lo que será posible reportar los resultados en tablas de forma ordenada. Evidentemente la cantidad de casos probados es bastante pequeña, revisar la sección TRABAJO FUTURO para más detalles. Se reportarán las métricas señaladas anteriormente agrupando casos por tamaño, además de detalles al respecto de la naturaleza de los casos probados específicamente.

4.2.1. 20x20

Podemos ver los resultados del caso $20x20$ en la tabla 1. Queda claro que este es un caso relativamente sencillo de resolver, en donde las distintas configuraciones suelen terminar en un tiempo bajo. Se ve un peculiar comportamiento de la función $f3$ que tiene tiempos muy disparejos, y también altos en los casos extremos. En general, se puede ver que el caso que pondera ambas preferencias no logra los valores máximos de preferencia, a diferencia de los casos que prefieren una preferencia específica, pero ambos valores de preferencia encuentran valores altos, por lo que se puede esperar un alto grado de satisfacción para ambos estudiantes y profesores.

4.2.2. 40x40

Los resultados del caso $40x40$ se encuentran en la tabla 2. Parece ser que el tiempo de la mayoría de las configuraciones es mayor a la anterior, lo que es totalmente esperable dado que el aumento de dimensión hace crecer la cantidad de variables y restricciones. Para la configuración de $\beta = \gamma$ se puede ver que los casos óptimos son menores, estos es debido a que el solver encuentra una solución subóptima.

4.2.3. 40x20

El caso $40x20$, en la tabla 3, es particularmente especial ya que independiente de la configuración probada, siempre se encuentra un caso en el que el modelo no encuentra solución por timeout. El parámetro de timeout fue fijado en 5 minutos. A pesar de esto, para el otro caso si es posible encontrar soluciones en tiempos razonables, y estas soluciones mantienen el comportamiento de los casos anteriores respecto de las preferencias y sus maximizaciones.

En particular, pareciera que la función $f1$ posee tiempos más altos para este caso. A la vez, la configuración que privilegia a β sobre γ también posee tiempos mayores que las otras.

4.2.4. 20x40

Finalmente, el caso $20x40$ y sus resultados en la tabla 4, nos presenta un fenómeno que no se aprecia en los anteriores, en donde el modelo encuentra soluciones óptimas para las que no se cumple toda la demanda de horas de ayudantía. Esto es posible ya que el diseño del caso puede incluir una mayor demanda de horas (asignaturas) y no suficiente estudiantes para cubrir esta demanda. También presenta algunas configuraciones en donde en vez de encontrar una solución termina en timeout. Respecto de los tiempos, se puede ver que en general son bajos independiente de la configuración.

CAPÍTULO 5
CONCLUSIONES Y TRABAJO FUTURO

5.1. Conclusiones

De forma general, en este trabajo se puede ver el proceso de modelamiento de un problema real, así como varias decisiones de diseño asociadas a procesos y parámetros. Además, pudimos comprobar la utilidad de generar un modelo de forma iterativa, el cual a medida que se agrega complejidad permite representar situaciones más completas para el problema de asignación de ayudantes. También se pudo realizar un proceso de validación del modelo mediante un método de grillado en el que se prueba que efectivamente se cumple con la intención que se plantea como solución al problema, esto es, obtener una maximización global de las preferencias asociadas a las postulaciones, tanto de estudiantes como profesores.

De forma particular, se pueden obtener las siguientes conclusiones:

- Fue posible analizar el proceso de selección de ayudantes (objetivo general del trabajo) y cumplir con plantear distintos criterios asociados, probar el impacto de estos criterios mediante asignaciones simuladas con el diseño de un modelo matemático, y proponer un nuevo proceso en donde se crean 2 nuevos subprocesos para poder incorporar al modelo propuesto.
- Inicialmente se propone modelar el proceso de asignación mediante un problema de asignación clásico, pero de forma iterativa y para poder darle al modelo más capacidad de modelar distintas situaciones, se termina con un modelo híbrido entre el problema de asignación y el problema de transporte.
- Agregar variables auxiliares al modelo para obtener valores asociados a distintas cantidades (como por ejemplo, el caso de la variable y_a que modela la demanda de horas, mientras que z_a modela el incumplimiento de la demanda) resulta ser útil para poder interpretar los resultados del modelo, además de lograr distintos resultados en la optimización misma.
- La construcción de las preferencias fue hecha correctamente, de tal forma que en su posterior uso en el modelo matemático es posible notar que los modelos pueden efectivamente maximizar esta cantidad. El caso más notorio de esto es el asociado a la **Preferencia Nula**, en donde en caso de tener una asignación con ésta, el valor de las preferencias sería un número negativo de alto valor, caso que no ocurre en ninguna de las situaciones probadas.
- Se puede notar que plantear varias funciones objetivo permite agregar variantes del modelo, en donde el objetivo de optimización y posterior resultado de asignación varía

según la función. Esto enriquece al modelo en cuanto le agrega más flexibilidad para poder implementarlo en distintos escenarios.

- En la resolución fue usado un solver basado en el método de branch and bound, lo que se conoce como una técnica completa (revisa todo el espacio de búsqueda). Esto permite obtener soluciones óptimas globales para el problema, evitando caer en soluciones locales. Incluso si algunos casos presentan problemas debido a la cantidad de variables y restricciones asociadas, el solver entrega una buena cantidad de soluciones óptimas para una dimensionalidad razonable con respecto a la naturaleza del proceso de asignación de ayudantías del DI.
- Mayor trabajo con más postulantes: El trabajo de cada profesor, especificar las preferencias por cada alumno, aumenta linealmente con el número de postulantes. A modo de ejemplo, si a una asignatura postularan 20 alumnos, esto implica que el profesor debe especificar 20 preferencias distintas. Es posible mitigar esto incluyendo preferencias por defecto, por ejemplo, todos los postulantes parten con preferencia media o nula.
- Más trabajo para el alumno: Un alumno que quiera reflejar de forma adecuada sus intenciones en el sistema tendrá que ingresar un mayor número de preferencias, pero este proceso puede simplificarse dependiendo de como sea diseñado el formulario de ingreso de las preferencias para los alumnos al momento de postular a las ayudantías para las asignaturas.

5.2. Trabajo futuro

En general, este trabajo se ha destacado por ser un acercamiento inicial a tratar de entender, manifestar y modelar un problema práctico, por lo que las simplificaciones hechas en su desarrollo son abundantes. Detallaré a continuación varios aspectos que me parece interesante considerar en futuros acercamientos a esta problemática.

5.2.1. Modelamiento de los parámetros

En este trabajo se propone un modelo con parámetros asociados para modelar restricciones y preferencias. Si bien estos nos ayudan a entender el fenómeno, siempre es posible agregar más detalle al modelo mediante la inclusión de más parámetros. Por ejemplo, en el caso de la preferencia de los profesores y estudiantes, esta fue expresada mediante un único valor que agrega la preferencia general al momento de postular o seleccionar, agregando distintos criterios de forma indiscriminada. Se propone a futuro poder reemplazar el valor de las preferencias de modelo por un sistema de parámetros en los que los criterios de selección estén detallados de forma explícita (y no implícita como en este trabajo), por ejemplo, algo

del estilo:

$$pp_{i,j} = \theta \cdot Nota\_alumno + \sigma \cdot Prioridad\_acadmica + \lambda \cdot Experiencia + ...$$

Esto obviamente encarece el modelamiento y las posteriores pruebas del mismo, pero permitiría modelar de forma explícita los criterios de selección, logrando determinar combinaciones óptimas para la asignación de los criterios.

5.2.2. Acercamiento borroso

La investigación de modelos de programación lineal fuzzy nos ha dejado algunos métodos interesantes para abordar problemas en donde la especificación de los parámetros en si misma es compleja, debido a la variabilidad y poca especificidad de los recursos disponibles. En el caso de este problema, de forma paralela se desarrollo un trabajo asociado en el que se busca resolver el problema mediante un acercamiento borroso, en donde algunos parámetros que presentan un cierto nivel de imprecisión se modelan de forma borrosa. El acercamiento propuesto en la SECCIÓN ANEXO trabaja sobre la simplificación del problema como problema de asignación básico (Modelo 1). Una extensión por esta rama puede ser un interesante problema a considerar a futuro.

5.2.3. Dimensionalidad y validación

Se menciona anteriormente que los casos reportados en la sección ANALISIS DE RESULTADOS son una evidente simplifiación de las pruebas. Esto se debe a que la dimensionalidad del modelo crece a medida que se incrementan los parámetros m y n. En específico, el espacio de búsqueda del modelo y las restricciones crecen con las siguientes expresiones:

$$EB(m,n) = 2(m \cdot n + n + 1)$$

$$RES(m,n) = 2m \cdot n + 6n + 4m + 2$$

Las expresiones consideran la cantidad de variables y restricciones (sin considerar restricciones del dominio de las variables, es decir, variables enteras o binarias) para los parámetros m y n. Se puede ver un claro crecimiento cuadrático en el factor $m \cdot n$, por lo que para casos de dimensión $40x40$ ya es posible ver modelos con aproximadamente 3000 variables y 3000 restricciones.

Además de este problema de dimensionalidad, se simplificaron las variantes al momento de generar los casos de prueba. Por ejemplo, los valores de α, β y γ fueron acotados para reducir el número de combinaciones, así como también solo se incluye en las pruebas las variantes de casos estrictos v/s casos flexibles. Se puede agregar más valor si consideramos más combinaciones de valores para los pesos de las funciones, o si consideramos otras variaciones del problema. Por ejemplo, ¿Qué pasa con el modelo en caso de que el problema no tenga soluciones factibles? es una pregunta válida que se plantea como trabajo a futuro, debido a la cantidad de combinaciones extra que esto agrega al modelo.

5.2.4. Solver

En este trabajo se plantea un acercamiento a la resolución del modelo mediante técnicas completas. El principal problema del uso de técnicas completas es justamente su escalabilidad debido a su diseño de trabajar en todo el espacio de búsqueda. Posibles trabajos futuros incluyen el intentar resolver el modelo propuesto con técnicas incompletas, lo que, si bien no garantiza encontrar la solución óptima global, ayuda a escalar la dimensión del problema a instancias más grandes en tiempos razonables. Cabe mencionar que en acercamiento fuzzy antes mencionado se usa una técnica incompleta, puede usarse como inspiración para este mismo trabajo.

ANEXOS

En los Anexos se incluye todo aquel material complementario que no es parte del contenido de los capítulos de la memoria, pero que permiten a un lector contar con un contenido adjunto relacionado con el tema.

5.1. Tablas

		Beta - Gamma		
		0 - 1	0.5 - 0.5	1 - 0
Función objetivo	$f1$	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 115 $\overline{xpe}$ = 54 $\overline{za}$ = 0 $\bar{t}$ = 0.342	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 97 $\overline{xpe}$ = 108 $\overline{za}$ = 0 $\bar{t}$ = 0.427	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 33 $\overline{xpe}$ = 120 $\overline{za}$ = 0 $\bar{t}$ = 1.315
	$f2$	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 115 $\overline{xpe}$ = 54 $\overline{za}$ = 0 $\bar{t}$ = 0.553	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 101 $\overline{xpe}$ = 104 $\overline{za}$ = 0 $\bar{t}$ = 0.222	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 14 $\overline{xpe}$ = 120 $\overline{za}$ = 0 $\bar{t}$ = 0.250
	$f3$	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 115 $\overline{xpe}$ = 52 $\overline{za}$ = 0 $\bar{t}$ = 10.812	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 95 $\overline{xpe}$ = 110 $\overline{za}$ = 0 $\bar{t}$ = 0.249	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 9 $\overline{xpe}$ = 120 $\overline{za}$ = 0 $\bar{t}$ = 2.374

Tabla 1: Resultados agrupados para casos (2) de tamaño 20x20 variando flexibilidad de las restricciones.

		Beta - Gamma		
		0 - 1	0.5 - 0.5	1 - 0
Función objetivo	$f1$	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 240 $\overline{xpe}$ = 98 $\overline{za}$ = 0 $\overline{t}$ = 2.305	Optimalidad = 1 timeout = 0 $\overline{xpp}$ = 222 $\overline{xpe}$ = 240 $\overline{za}$ = 0 $\overline{t}$ = 2.948	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 45 $\overline{xpe}$ = 240 $\overline{za}$ = 0 $\overline{t}$ = 6.474
	$f2$	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 240 $\overline{xpe}$ = 76 $\overline{za}$ = 0 $\overline{t}$ = 0.937	Optimalidad = 1 timeout = 0 $\overline{xpp}$ = 219 $\overline{xpe}$ = 238 $\overline{za}$ = 0 $\overline{t}$ = 2.960	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 51 $\overline{xpe}$ = 240 $\overline{za}$ = 0 $\overline{t}$ = 6.023
	$f3$	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 240 $\overline{xpe}$ = 126 $\overline{za}$ = 0 $\overline{t}$ = 1.086	Optimalidad = 1 timeout = 0 $\overline{xpp}$ = 219 $\overline{xpe}$ = 238 $\overline{za}$ = 0 $\overline{t}$ = 3.092	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 23 $\overline{xpe}$ = 240 $\overline{za}$ = 0 $\overline{t}$ = 5.068

Tabla 2: Resultados agrupados para casos (2) de tamaño 40x40 variando flexibilidad de las restricciones.

		Beta - Gamma		
		0 - 1	0.5 - 0.5	1 - 0
Función objetivo	$f1$	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 158 $\overline{xpe}$ = 60 $\overline{za}$ = 0 $\overline{t}$ = 7.411	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 146 $\overline{xpe}$ = 152 $\overline{za}$ = 0 $\overline{t}$ = 6.458	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 20 $\overline{xpe}$ = 160 $\overline{za}$ = 0 $\overline{t}$ = 23.565
	$f2$	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 158 $\overline{xpe}$ = 92 $\overline{za}$ = 0 $\overline{t}$ = 0.594	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 156 $\overline{xpe}$ = 142 $\overline{za}$ = 0 $\overline{t}$ = 0.688	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 20 $\overline{xpe}$ = 160 $\overline{za}$ = 0 $\overline{t}$ = 1.369
	$f3$	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 158 $\overline{xpe}$ = 64 $\overline{za}$ = 0 $\overline{t}$ = 0.678	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 156 $\overline{xpe}$ = 142 $\overline{za}$ = 0 $\overline{t}$ = 0.662	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 40 $\overline{xpe}$ = 160 $\overline{za}$ = 0 $\overline{t}$ = 4.413

Tabla 3: Resultados agrupados para casos (2) de tamaño 40x20 variando flexibilidad de las restricciones.

		Beta - Gamma		
		0 - 1	0.5 - 0.5	1 - 0
Función objetivo	$f1$	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 160 $\overline{xpe}$ = 80 $\overline{za}$ = 3.75 $\bar{t}$ = 0.550	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 142 $\overline{xpe}$ = 156 $\overline{za}$ = 3.75 $\bar{t}$ = 0.560	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 30 $\overline{xpe}$ = 160 $\overline{za}$ = 3.75 $\bar{t}$ = 0.971
	$f2$	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 160 $\overline{xpe}$ = 64 $\overline{za}$ = 3.75 $\bar{t}$ = 1.618	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 142 $\overline{xpe}$ = 156 $\overline{za}$ = 0 $\bar{t}$ = 0.280	Optimalidad = 1 timeout = 1 $\overline{xpp}$ = 0 $\overline{xpe}$ = 160 $\overline{za}$ = 0 $\bar{t}$ = 0.329
	$f3$	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 160 $\overline{xpe}$ = 70 $\overline{za}$ = 3.75 $\bar{t}$ = 0.430	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 142 $\overline{xpe}$ = 156 $\overline{za}$ = 3.75 $\bar{t}$ = 0.446	Optimalidad = 2 timeout = 0 $\overline{xpp}$ = 6 $\overline{xpe}$ = 160 $\overline{za}$ = 3.75 $\bar{t}$ = 1.092

Tabla 4: Resultados agrupados para casos (2) de tamaño 20x40 variando flexibilidad de las restricciones.

5.2. Casos de prueba

Indicaciones de como construir los casos de prueba usados para obtener los resultados se disponen a continuación.

Para todos los casos, la matriz pe se construye mediante un muestreo con reemplazo para los valores $[-999, 0, 4]$, considerando el vector de probabilidades $[0{,}2; 0{,}4; 0{,}4]$. La matriz pp se construye mediante un muestreo con reemplazo para los valores $[-999, -2, 0, 2, 4]$, considerando el vector de probabilidades $[0{,}1; 0{,}225; 0{,}225; 0{,}225; 0{,}225]$. Se fija la semilla aleatoria en 42. El tamaño de la muestra es la cantidad de variables del caso.

- Caso 1:

$$m : 20$$

$$n : 20$$

$$he\_max : [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]$$

$$he\_min : [7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7]$$

$$ha : [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]$$

$$ne\_max : [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]$$
$$ne\_min : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$
$$na\_max : [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]$$
$$na\_min : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

- Caso 2:

$$m : 20$$
$$n : 20$$
$$he\_max : [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]$$
$$he\_min : [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]$$
$$ha : [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]$$
$$ne\_max : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$
$$ne\_min : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$
$$na\_max : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$
$$na\_min : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

5.2.1. 40x40

El caso 40x40 es idéntico al 20x20, solo que aumentan las dimensiones de los parámetros.

5.2.2. 40x20

- Caso 1:

$$m : 40$$
$$n : 20$$
$$he\_max : \{15\} \text{ repetido 40 veces}$$
$$he\_min : \{7\} \text{ repetido 40 veces}$$
$$ha : [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]$$
$$ne\_max : \{2\} \text{ repetido 40 veces}$$
$$ne\_min : \{1\} \text{ repetido 40 veces}$$
$$na\_max : [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]$$
$$na\_min : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

- Caso 2:

$$m : 40$$

$$n : 20$$

$$he\_max : \{15\} \text{ repetido 40 veces}$$

$$he\_min : \{5\} \text{ repetido 40 veces}$$

$$ha : [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]$$

$$ne\_max : \{1\} \text{ repetido 40 veces}$$

$$ne\_min : \{1\} \text{ repetido 40 veces}$$

$$na\_max : [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]$$

$$na\_min : [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]$$

5.2.3. 20x40

- Caso 1:

$$m : 20$$

$$n : 40$$

$$he\_max : [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]$$

$$he\_min : [7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7]$$

$$ha : \{15\} \text{ repetido 40 veces}$$

$$ne\_max : [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]$$

$$ne\_min : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

$$na\_max : \{3\} \text{ repetido 40 veces}$$

$$na\_min : \{1\} \text{ repetido 40 veces}$$

- Caso 2:

$$m : 20$$

$$n : 40$$

$$he\_max : [30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30]$$

$$he\_min : [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]$$

$$ha : \{15\} \text{ repetido 40 veces}$$

$$ne\_max : [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]$$

$$ne\_min : [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]$$

$$na\_max : \{1\} \text{ repetido 40 veces}$$

$$na\_min : \{1\} \text{ repetido 40 veces}$$

5.3. Acercamiento fuzzy

5.3.1. Descripción del problema

1. **Contexto:** El problema que se aborda en este trabajo es la automatización del proceso de asignación de ayudantes en el DI. El proceso de selección actual consta de 2 etapas, una etapa de postulación, en donde los estudiantes postulan a distintas asignaturas, y una etapa de selección, en donde los profesores eligen alumnos dentro de los que han postulado a dicha asignatura. Este tipo de proceso puede presentar los siguientes problemas:
    - La selección por parte de los profesores se realiza por orden de llegada, en donde si un alumno postulara a 2 ayudantías, quedaría seleccionado en la que sea electo en primera instancia (podría hacer las 2 considerando reglamentos y solicitudes).
    - Como la selección la realiza cada profesor de forma individual, la elección final puede caer en asignaciones óptimas locales, es decir, es posible que exista otra asignación que maximice de mejor forma variables como la cantidad de horas cubiertas, o la compatibilidad entre estudiante y profesor.
    - Este proceso actual suele tener problemas cubriendo todas las vacantes de ayudantías, ya sea por falta de postulantes, o por la situación descrita anteriormente en donde un alumno no puede hacer múltiples ayudantías de forma simultanea.
    - Cada profesor decide que estudiante escoger como ayudante según criterios propios (Nota del alumno, motivación, prioridad académica, etc). A su vez, cada estudiante elige una asignatura específica basado en otros criterios (Experiencia, motivación, etc.). Esta etapa es poco transparente en este sentido, y si conociéramos estos criterios sería posible lograr asignaciones que satisfagan tanto a profesores como estudiantes envueltos en el proceso.
2. **Solución crisp:** Para este problema se propone crear un modelo matemático de optimización basado en restricciones, el cual fue diseñado como parte del trabajo de memoria en la sección Asignación inteligente de ayudantes. Este modelo simula correctamente el proceso de asignación de ayudantías, incorporando restricciones sobre la cantidad de horas y ayudantes para cada asignatura y estudiante, además de declarar de forma explícita las preferencias de ambas partes involucradas en la asignación. Con una configuración adecuada para la función objetivo, el modelo puede obtener una asignación que maximiza de forma simultanea el tiempo (evitando que queden vacantes) y las preferencias de profesores y estudiantes.
3. **Imprecisión:** Si bien el modelo anterior resuelve el problema planteado anteriormente mediante alguna técnica de programación lineal, es posible notar fuentes de información que son bastante estrictas para el modelo, y en un problema real estas podrían definirse de forma imprecisa, dejando al modelo anterior con la labor de asumir supuestos para poder trabajar con la imprecisión. En particular:

- Los parámetros $pe_{i,j}$ y ppi,j definen las preferencias para estudiantes y profesores, respectivamente. Estas preferencias son poco precisas, ya que se construyen en base a distintos criterios, lo cuales no son declarados de forma directa. Que un profesor prefiera a un estudiante más que a otro, o que un estudiante prefiera postular a una asignatura más que a otra es difícil de cuantificar directamente asignando valores numéricos.
- Parámetros como na_max_j, na_min_j, ne_max_i y ne_min_i son entregados como cotas superiores e inferiores para varias restricciones, por lo que, en caso de que sean distintos, significa esto que las cantidades asociadas a estos parámetros no están definidas de forma precisa.

Es aquí en donde surge el verdadero problema de este trabajo, que involucra en encontrar alguna forma de poder modelar esta imprecisión de forma correcta para el problema de asignación descrito anteriormente. Veremos en secciones posteriores acercamientos con métodos que involucran el uso de técnicas borrosas.

5.3.2. Propuesta

Lo que se propone en este trabajo para poder modelar la imprecisión dentro de este problema es seguir modelando la situación usando programación lineal, pero incluyendo técnicas del mundo borroso que ayuden a agregar representaciones que sean capaces de hacerse cargo de la imprecisión. El problema pasa entonces de un problema de programación lineal crisp a un problema de programación lineal fuzzy, problema para el cual existen diferentes variantes asociadas al modelamiento y posterior resolución.

1. **Simplificación:** El problema original es resuelto con un modelo de programación lineal entera, que combina ideas asociadas tanto al problema de asignación como al problema transporte, lo que lo transforma en modelo más interesante con el que trabajar, pero a su vez aumenta el número de variables y restricciones asociadas, incluyendo la dificultad de contar con variables binarias y enteras. Para poder realizar un primer acercamiento se decide trabajar en base a un modelo simplificado, el cual está basado sólo en el problema de asignación. Este modelo simplificado es una iteración previa del modelo final, que permite modelar la asignación de ayudantes maximizando las preferencias asociadas, pero no toma en consideración la cantidad de horas de cada asignación. El modelo simplificado es el modelo 1 propuesto anteriormente

5.3.3. Trabajos relacionados

Existen diversos trabajos asociados a como es posible enfrentar el problema de programación lineal fuzzy. En esta sección se describen algunos de los métodos más relevantes que

puedan ser aplicados al problema de asignación, debido a su particularidad en cuanto a como se definen las variables y restricciones. También se tiene en cuenta cómo se modela la imprecisión asociada al modelo.

Es importante señalar que la mayoría de trabajos encontrados trabajan bajo el supuesto de que los números borrosos usados son triangulares.

1. **Tipos de modelos:**

   Un trabajo muy completo acerca de estado del arte de esta disciplina es el hecho por [Ghanbari et al., 2019], en donde se realiza una exploración de distintas formas de modelar y resolver un problema de programación lineal fuzzy. Los autores logran clasificar los distintos problemas en 3 grandes clases, definidas a continuación:

   a) Fuzzy Variable Linear Programming Problem (FVLPP): Esta clase de problemas destaca por poseer una combinación de variables borrosas y parámetros crisp. La definición formal de los problemas de esta clase es:

   $$FVLPP \begin{cases} \text{max(min) } \tilde{z} \approx c^T \tilde{x} \\ s.t. \\ A\tilde{x}\{\leq, \approx, \geq\}\tilde{b} \\ x \geq \tilde{0} \end{cases}$$

   En donde $\tilde{x}$ representa a las variables borrosas, mientras que el resto de parámetros son crisp. Es interesante ver que esta definición considera que el vector $\tilde{b}$ también es borros, debido a su relación con las variables borrosas. La definición del tipo de problema acepta ciertas variaciones, buscando una propuesta general.

   b) Fuzzy Number Linear Programming Problem (FNLPP): Esta clase de problemas posee una combinación de variables crisp y parámetros borrosos. La definición formal de los problemas de esta clase es:

   $$FNLPP \begin{cases} \text{max(min) } \tilde{z} \approx \tilde{c}^T x \\ s.t. \\ \tilde{A}x\{\leq, \approx, \geq\}\tilde{b} \\ x \geq 0 \end{cases}$$

   Esta clase es aún más flexible que la anterior, en donde es posible definir algunos parámetros crips y otros fuzzy, y hacer combinaciones con estos.

   c) Fully Fuzzy Linear Programming Problem (FFLPP): En esta última clase de problemas, todos los componentes del modelo se consideran como número borrosos, tanto variables como parámetros. La definición formal de los problemas de esta clase es:

   $$FFLPP \begin{cases} \text{max(min) } \tilde{z} \approx \tilde{c}^T \tilde{x} \\ s.t. \\ \tilde{A}\tilde{x}\{\leq, \approx, \geq\}\tilde{b} \\ \tilde{x} \geq \tilde{0} \end{cases}$$

Esta clase es de gran utilidad ya que mezcla ambas clases anteriores, y considerando supuestos adecuados podría modelar problemas asociados a clases anteriores mediante una defuzzyficación de los valores.

Lo que plantean los autores para diferenciar entre una clase y otra tiene relación con el uso de los parámetros imprecisos de una situación específica, y si esta imprecisión es usada para condicionar una decisión, que a su vez también puede ser imprecisa.

2. **Método posibilístico:** Este método es propuesto en el trabajo de [Buckley, 1988], en donde el autor propone una solución para un modelo FNLPP, basado en la siguiente serie de pasos:

   a) No es posible maximizar (o minimizar) la cantidad $\tilde{z}$, debido a que es una variable fuzzy (de forma similar a como en un problema estocástico no es posible maximizar el valor de una variable aleatoria).

   b) A pesar de lo anterior, $\tilde{z}$ tiene una función de posibilidad asociada, y es posible evaluar inecuaciones como $Poss[\tilde{z} \geq \lambda]$ o $Poss[\lambda_1 \leq \tilde{z} \leq \lambda_2]$. Si es posible construir una expresión para la función de posibilidad de $\tilde{z}$, se pueden evaluar estas cantidades.

   c) Se definen las posibilidades asociadas a cada restricción i del problema de la siguiente forma:

   $$Poss[x \in \mathcal{F}_i] = sup\{min\{\mu(a_{i,1}), \mu(a_{i,2}), ..., \mu(b_i)\}\}$$

   En donde μ es una función de pertenencia borrosa asociada a los parámetros fuzzy.

   d) Luego, una solución factible debe cumplir con:

   $$Poss[x \in \mathcal{F}] = min_i\{Poss[x \in \mathcal{F}_i]\}$$

   e) Además, se define una función de posibilidad asociada a la función objetivo de la siguiente forma:

   $$Poss[\tilde{z}|x] = min_i\{\mu(c_i)\}$$

   f) Finalmente, la expresión deseada se obtiene agregando las posibilidades:

   $$Poss[\tilde{z}] = sup_x\{min\{Poss[\tilde{z}|x], Poss[x \in \mathcal{F}]\}\}$$

   Este acercamiento es muy interesante, y se plantea concepto interesantes que serán usados en un posterior modelamiento. Su resolución teórica no es trivial, ya que involucra funciones no lineales, como muestra el autor en el trabajo.

3. **FBLPP:** Fuzzy Binary Linear Programming Problem (FBLPP) es propuesto como una subcategoría de FNLPP, en donde se tienen problemas con variables de decisión binarias. En el trabajo de [Yu y Li, 2001], los autores proponen un algoritmo que permite adaptar un FBLPP a un problema de programación lineal entera clásico. Los autores proponen el traspaso del FBLPP al problema crisp mediante las siguientes propuestas:

a) Cada parámetro borroso asociado al problema es representado de la siguiente forma en el modelo:

$$s_{i,k} = \frac{\mu(c_{i,k+1}) - \mu(c_{i,k})}{c_{i,k+1} - c_{i,k}}$$

$$\mu(c_i) = \mu(c_{i,1}) + s_{i,1}(c_i - c_{i,1}) + \frac{s_{i,2} - s_{i,1}}{2}(|c_i - c_{i,2}| + c_i - c_{i,2})$$

para este caso, $\tilde{c}_i$ es un coeficiente de la función objetivo (manteniendo la notación previa).

b) Si además de parámetros borrosos en la función objetivo el problema incluye parámetros borrosos en las restricciones, se incluyen variables w_i y δ_i que son optimizadas junto con las variables de decisión.

c) Además, se agregan restricciones para mantener la propiedad binaria de las variables de decisión.

En general, este método es interesante porque mantiene la linealidad del problema, y se puede seguir resolviendo con un solver linear (como branch and bound). El principal problema es que agrega demasiadas variables y restricciones al problema, como se puede ver en el trabajo original.

4. **Método de Zimmermann:** En el trabajo de [Zimmermann, 1975], se trabajan con 2 conceptos muy relevantes para el trabajo actual. Estos son:

   - Restricciones borrosas: Se busca poder representar la capacidad de violar una restricción o no mediante una función de pertenencia asociada al parámetro fuzzy, escrita de la siguiente forma:

$$\mu_i : \mathbf{R} \to [0,1], \quad \mu_i(x) = \begin{cases} 1 & \text{si} \quad x \leq b_i \\ f_i(x) & \text{si} \quad b_i \leq x \leq b_i + t_i \\ 0 & \text{si} \quad x \geq b_i + t_i \end{cases}$$

     En donde f_i es una función continua incremental. Se puede ver que esta función de pertenencia evalúa que tanto es cumplida la restricción, considerando un margen tolerado. La expresión detallada es válida para restricciones con la forma:

$$a_i x \leq b_i$$

     En el caso de tener restricciones con $\geq$ es necesario hacer unos pequeños ajustes.

   - Función objetivo: Se plantea que es posible plantear a la solución objetivo como una restricción del problema de la siguiente forma:

$$\text{max } c^T x \to c^T x \geq z_0$$

     En donde el valor de z_0 es una cota, que puede ser la solución crisp. Este método es particularmente útil para problemas borrosos ya que permite aplicar la técnica anterior basada en el modelamiento de restricciones incluyendo la función objetivo.

Una implementación interesante en R fue hecha en el trabajo de [Villacorta et al., 2017], donde además se incluyen detalles teóricos de este método y otros.

5.3.4. Modelamiento fuzzy

Luego de revisar los trabajos relacionados, se decide elegir el modelamiento basado en restricciones borrosas para poder representar el problema. En particular, el problema de asignación propuesto consta con las siguientes restricciones:

$$\sum_{i=1}^{m} x_{ij} \leq na\_max_j \quad \forall\, j = 1, \ldots, n$$

$$\sum_{i=1}^{m} x_{ij} \geq na\_min_j \quad \forall\, j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} \leq ne\_max_i \quad \forall\, i = 1, \ldots, m$$

$$\sum_{j=1}^{n} x_{ij} \geq ne\_min_i \quad \forall\, i = 1, \ldots, m$$

$$\sum_{j=1}^{n}\sum_{i=1}^{m} x_{i,j} \cdot pe_{i,j} + \sum_{j=1}^{n}\sum_{i=1}^{m} x_{i,j} \cdot pp_{i,j} \geq z_0$$

Para poder pensar en traspasar estas restricciones a su equivalente borroso, se definen las siguientes transformaciones asociadas a los parámetros del modelo:

- Preferencias crisp: Las preferencias crisp originales con las que se modela el problema son definidas mediante valores lingüísticos, a los que se le asocia un valor numérico dentro de la optimización:
  - $pe_{i,j}$: {Máxima=4, Media=0, Nula=-999}
  - $pp_{i,j}$: {Máxima=4, Alta=2, Media=0, Baja=-2, Nula=-999}
- Preferencias fuzzy: Para construir las nuevas preferencias borrosas es necesario considerar algún valor de desviación d, el que luego se usa para construir un número borroso triangular centrado en el valor numérico declarado en la optimización crisp:
  - $\widetilde{pe}_{i,j} : \{(pe_{i,j} - d; pe_{i,j}; pe_{i,j} + d)\}$
  - $\widetilde{pp}_{i,j} : \{(pp_{i,j} - d; pp_{i,j}; pp_{i,j} + d)\}$

- Cantidades crisp: Para el problema crisp el modelo define cotas superiores e inferiores asociadas a la cantidad de estudiantes que pueden ser asignados y la cantidad de asignaturas por estudiante:
  - ne_min_i/ne_max_i: $\{\mathbb{N} : ne\_min_i \leq ne\_max_i\}$
  - na_min_j/na_max_j: $\{\mathbb{N} : na\_min_j \leq na\_max_j\}$
- Cantidades fuzzy: Estas cotas se representan mediante un único número borroso, el cual incorpora ambos valores en un número triangular:
  - $\widetilde{ne}_i : \{(ne\_min_i; \frac{ne\_max_i + ne\_min_i}{2}; ne\_max_i)\}$
  - $\widetilde{na}_j : \{(na\_min_j; \frac{na\_max_j + na\_min_j}{2}; na\_max_j)\}$

Considerando estas transformaciones, las restricciones finales, se obtiene un modelo que conserva las mismas variables de decisión, pero incluye estas nuevas restricciones:

$$\sum_{i=1}^{m} x_{ij} \preceq \widetilde{ne}_i \quad \forall\, j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} \preceq \widetilde{na}_j \quad \forall\, i = 1, \ldots, m$$

En particular, para la función objetivo se tiene:

$$\sum_{j=1}^{n}\sum_{i=1}^{m} x_{i,j} \cdot pe_{i,j} + \sum_{j=1}^{n}\sum_{i=1}^{m} x_{i,j} \cdot pp_{i,j} \succeq \tilde{z}_0$$

Esto permite aplicar la transformación de restricción borrosa a la restricción, en donde lo borroso se modela dentro de la cota $\tilde{z}_0$, además de poder comparar la asignación con esta cantidad. Notar que los comparadores son especiales debido a que se compara un número crisp con uno fuzzy, lo que se termina realizando en la función de pertenencia respectiva.

Siguiendo las ideas del trabajo de [Ribeiro y Pires, 1999], se terminan por modelar las restricciones con las siguientes funciones de pertenencia:

$$\mu_{\preceq}(k) = \begin{cases} 0 & \text{si} \quad R_k(x) > c_k + d_k \\ 1 - \frac{R_k(x) + c_k}{d_k} & \text{si} \quad c_k < R_k(x) \leq c_k + d_k \\ 1 & \text{si} \quad R_k(x) \leq c_k \end{cases}$$

$$\mu_{\succeq}(k) = \begin{cases} 0 & \text{si} \quad R_k(x) < c_k + d_k \\ 1 - \frac{R_k(x) - c_k}{d_k} & \text{si} \quad c_k > R_k(x) \geq c_k + d_k \\ 1 & \text{si} \quad R_k(x) \geq c_k \end{cases}$$

En donde se tiene $R_k = \sum_j a_{ij}x_j$ ó $\sum_j g_j x_j$. En este trabajo se plantea también el siguiente problema de optimización asociado a la siguiente expresión:

$$max[min_k \mu(k)]$$

en donde se elige como t-norma agrupadora la función mínimo. Los autores proponen abordar este problema de optimización usando la metaheurística Simulated Annealing (SA).

5.3.5. Implementación

Se proponen 3 métodos para buscar soluciones para el problema. Esta decisión surge debido a la naturaleza del SA, que es considerada más una técnica reparadora de soluciones, y por ende es muy sensible a decisiones como el movimiento escogido o la calidad de la solución inicial.

1. **1er método:** El primer método probado es un SA simple, el que se programa bajo las siguientes decisiones de diseño:
   - Temperatura: Inicial de 800, termina de iterar con temperaturas menores a 10^{-5}. El decaimiento de la temperatura se da por la formula
     $$T(t) = 0{,}9^{t-1} \cdot T_0$$
   - Movimiento: Se decide usar como movimiento un bit flip, es decir, elegir una variable y cambiarla por su valor contrario. Dependiendo del caso se puede modificar la cantidad de bits que se cambian. Para problemas con 100 o más variables se decide usar 4-bit flip.
   - Solución inicial: Se considera como solución inicial un vector aleatorio, lo que normalmente entrega soluciones en el espacio de soluciones infactibles.

   El método sigue, de forma general, los siguientes pasos para obtener la solución:

   a) Se obtiene una solución inicial aleatoria.
   b) Se calculan las cantidades $\mu(k)$ asociadas a cada restricción.
   c) Se obtiene la t-norma de estas funciones de pertenencia.
   d) Si la solución es la mejor encontrada, se guarda. Si no, se obtiene un valor aleatorio $Unif(0, 1)$ y se comparar con la cantidad $exp(\frac{\Delta}{T})$, en donde Δ es la calidad de la solución. Esta comparación permite la exploración a soluciones de menor calidad.
   e) El algoritmo termina cuando la temperatura se reduce a un valor menor al valor mínimo fijado.

2. **2do método:** El segundo método busca abordar la debilidad del método anterior respecto de la solución inicial. Se plantea un método que combina secuencialmente 2 instancias de SA, en donde en la primera etapa se maximiza el número de restricciones satisfechas (para obtener soluciones factibles), y en la segunda instancia se aplica el método 1.

3. **3er método:** El tercer método busca abordar un problema presente en la restricción asociada a la función objetivo, que no es trivial de optimizar con un movimiento sencillo como un bit flip. Se plantea un método que combina secuencialmente 3 instancias de SA, en donde en la primera etapa se maximiza el valor de la función objetivo (independiente de si la solución obtenida no es factible). En ambas segunda y tercera instancias se aplican los métodos 2 y 1 respectivamente.

5.3.6. Resultados

Se procede a probar el método con un caso 10x10, lo que involucra 100 variables de decisión y 40 restricciones (+1 considerando la función objetivo). Los parámetros asociados al modelo son los siguientes:

$$m : 10$$

$$n : 10$$

$$ne\_max : [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]$$

$$ne\_min : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

$$na\_max : [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]$$

$$na\_min : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

Respecto de la preferencias, la matriz pe se construye mediante un muestreo con reemplazo para los valores $[-999, 0, 4]$, considerando el vector de probabilidades $[0{,}1; 0{,}5; 0{,}4]$. La matriz pp se construye mediante un muestreo con reemplazo para los valores $[-999, -2, 0, 2, 4]$, considerando el vector de probabilidades $[0{,}1; 0{,}15; 0{,}15; 0{,}4; 0{,}2]$. Se fija la semilla aleatoria en 24. El tamaño de la muestra es la cantidad de variables del caso. Se considera una desviación para el problema fuzzy $d = 2$.

El problema crisp se resuelve de manera sencilla con estos parámetros, encontrando una solución con valor óptimo 124.

Para el problema fuzzy, este valor óptimo se usará como referencia para fijar el valor de cota $\tilde{z_0}$. Se proponen 2 cotas para la función objetivo difusa, en donde se considera una cota alta y una más flexible, para evaluar la capacidad de los métodos de satisfacer la restricción asociada a la función objetivo.

Se ejecutan los 3 métodos propuestos anteriormente, y se grafican los resultados considerando 2 medidas de desempeño. Primero, la calidad de la solución a través del valor de la función objetivo defuzzyficada, y en segundo lugar la factibilidad de la solución considerando el número de soluciones satisfechas.

1. **Umbral z = 100:** Podemos ver que este umbral es un tanto estricto, pero no tanto como el valor óptimo encontrado en el problema crisp. Esto hace que el método 2 termine por satisface la mayoría de restricciones, excepto la restricción asociada a la función objetivo, debido a su alto valor. El método 3 no presenta problemas con esto, mientras que el método 1 no puede moverse por espacios no factibles de soluciones debido al comportamiento de la t-norma.

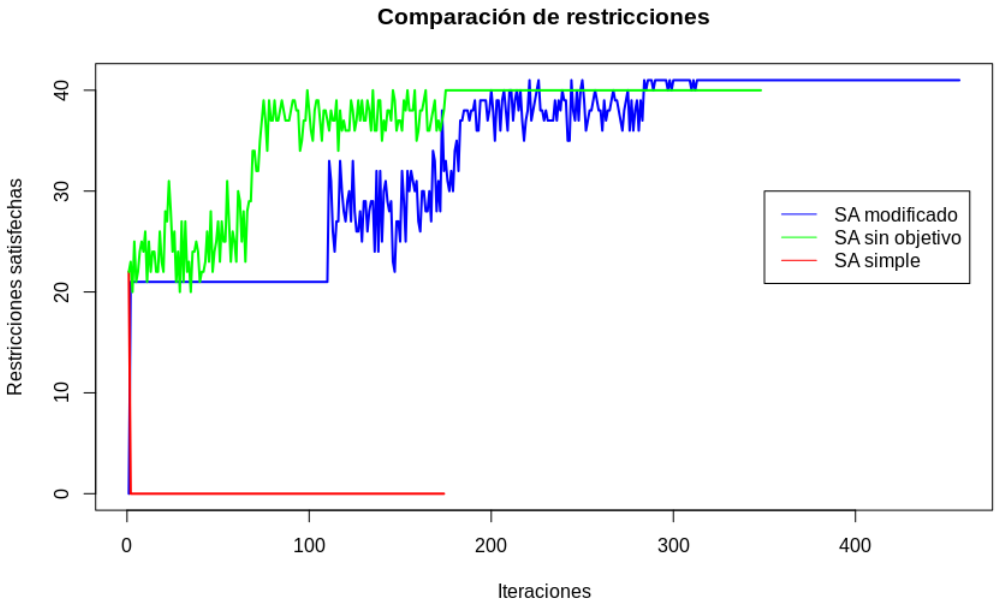


Figura 11: Resultados caso 10x10 con cota $z_0 = 100$, viendo cantidad de restricciones satisfechas según el número de iteraciones. En el gráfico, método 1 (rojo), método 2 (verde) y método 3 (azul).

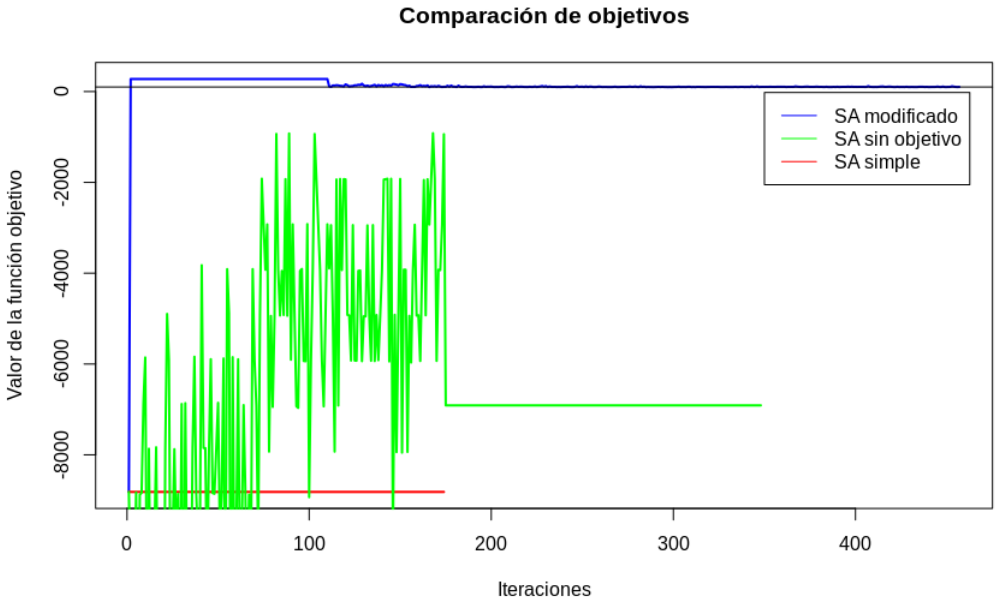


Figura 12: Resultados caso 10x10 con cota $z_0 = 100$, viendo valor de la función objetivo según el número de iteraciones. En el gráfico, método 1 (rojo), método 2 (verde) y método 3 (azul).

2. **Umbral z = 0:** Podemos ver que este umbral es un poco más permisivo, en donde las soluciones encontradas no serán las mejores, pero permite que sea más fácil satisfacer la restricción asociada a la función objetivo. Esto hace que el método 2 pueda satisfacer esta restricción correctamente, logrando equipararse al método 3. El método 1 nuevamente no puede moverse por espacios no factibles, lo que genera resultados horribles.

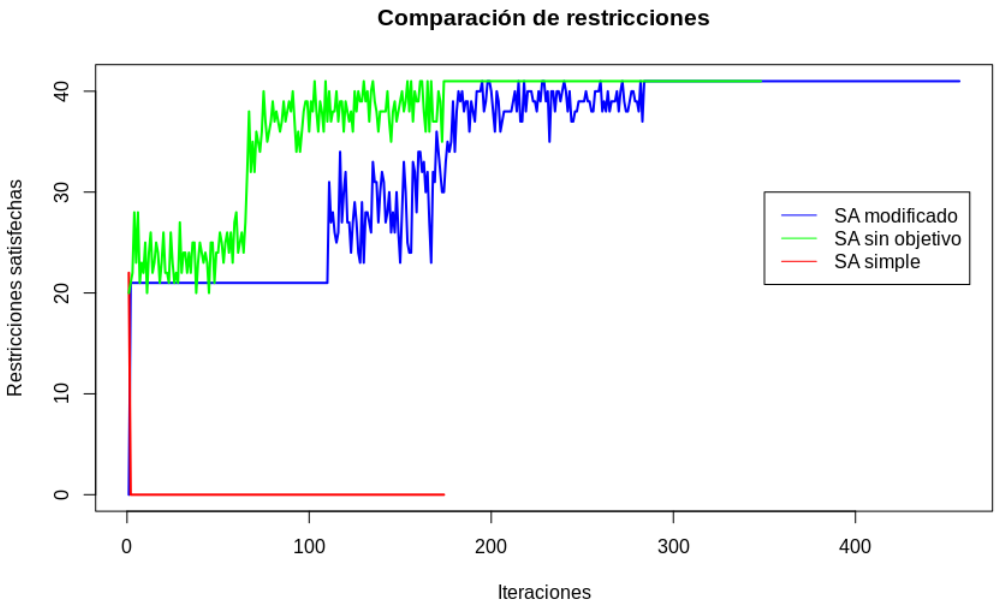


Figura 13: Resultados caso 10x10 con cota $z_0 = 0$, viendo cantidad de restricciones satisfechas según el número de iteraciones. En el gráfico, método 1 (rojo), método 2 (verde) y método 3 (azul).

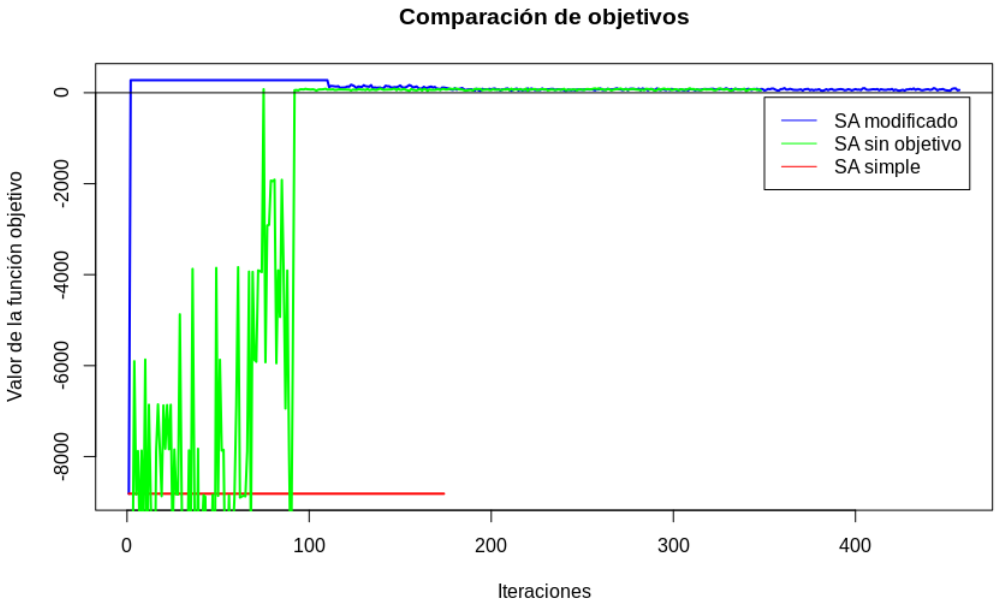


Figura 14: Resultados caso 10x10 con cota $z_0 = 0$, viendo valor de la función objetivo según el número de iteraciones. En el gráfico, método 1 (rojo), método 2 (verde) y método 3 (azul).

5.3.7. Conclusiones

En este trabajo se puede ver que existen muchos acercamientos a la programación lineal fuzzy, en donde termino por escoger una técnica específica para poder abordar una instancia relativamente sencilla del problema. La representación escogida para incorporar la imprecisión dentro del modelo pasa por el uso de las restricciones borrosas, en donde se usan los conceptos de función de pertenencia y t-norma para crear una cantidad optimizable. Si bien la técnica que se propone en el trabajo de referencia no es la más adecuada, se pueden definir métodos como los propuestos para atacar las debilidades de la metaheurística SA y obtener soluciones factibles.

Además, se concluye de forma particular:

- SA puede proveer soluciones interesantes, pero requiere de bastante trabajo extra debido a la naturaleza del problema mismo.
- Los métodos propuestos corrigen falencias del SA simple, pero en ningún caso alcanzan el desempeño del problema crisp.
- SA (y cualquier otra metaheurística con movimientos) permiten mantener restricciones binarias asociadas al problema de asignación.

- El costo de modelar un problema de programación lineal con parámetros fuzzy es mayor, pero se compensa en el hecho de poder modelar problemas con información imprecisa.
- Sospecho que GA tendrá mejor desempeño que SA, pensando en la mayor capacidad de exploración introducida por los cruzamientos y las múltiples soluciones iniciales dentro de la población.

REFERENCIAS BIBLIOGRÁFICAS

[Buckley, 1988] Buckley, J. (1988). Possibilistic linear programming with triangular fuzzy numbers. Fuzzy sets and Systems, 26(1):135–138.

[Castillo et al.,] Castillo, E., Conejo, A. J., Pedregal, P., Garcıa, R., y Alguacil, N. Formulación y resolución de modelos de programación matemática en ingenierıa y ciencia.

[Cattrysse y Van Wassenhove, 1992] Cattrysse, D. G. y Van Wassenhove, L. N. (1992). A survey of algorithms for the generalized assignment problem. European journal of operational research, 60(3):260–272.

[Fisher y Jaikumar, 1981] Fisher, M. L. y Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. Networks, 11(2):109–124.

[Gao, 2014] Gao, J. (2014). Branch and bound (bb).

[Ghanbari et al., 2019] Ghanbari, R., Ghorbani-Moghadam, K., Mahdavi-Amiri, N., y De Baets, B. (2019). Fuzzy linear programming problems: models and solutions. Soft Computing, pp. 1–31.

[Gurobi Optimization, 2019] Gurobi Optimization, L. (2019). Mixed-integer programming (mip) – a primer on the basics.

[Ribeiro y Pires, 1999] Ribeiro, R. A. y Pires, F. M. (1999). Fuzzy linear programming via simulated annealing. Kybernetika, 35(1):57–67.

[Ross y Soland, 1977] Ross, G. T. y Soland, R. M. (1977). Modeling facility location problems as generalized assignment problems. Management Science, 24(3):345–357.

[Villacorta et al., 2017] Villacorta, P. J., Rabelo, C. A., Pelta, D. A., y Verdegay, J. L. (2017). Fuzzylp: an r package for solving fuzzy linear programming problems. En Granular, Soft and Fuzzy Approaches for Intelligent Systems, pp. 209–230. Springer.

[Yu y Li, 2001] Yu, C.-S. y Li, H.-L. (2001). An algorithm for generalized fuzzy binary linear programming problems. European Journal of Operational Research, 133(3):496–511.

[Zimmermann, 1975] Zimmermann, H.-J. (1975). Description and optimization of fuzzy systems. International journal of general System, 2(1):209–215.