UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE CIENCIAS EXACTAS Y NATURALES

Modelos de programación lineal entera para problemas de asignación de turnos con condiciones de estabilidad

Tesis de Licenciatura en Ciencias de Datos

Dafne Sol Yudcovsky

Director: Javier Marenco

Codirector: Esteban Lanzarotti

Buenos Aires, 2025

MODELOS DE PLE PARA PROBLEMAS DE ASIGNACIÓN DE TURNOS CON CONDICIONES DE ESTABILIDAD

Dentro del campo de la Optimización, la Programación Lineal Entera (PLE) se destaca por su capacidad para modelar problemas de decisión en los que las variables deben asumir valores enteros, cuya flexibilidad expresiva puede reflejar restricciones inherentes a muchos sistemas de la vida real. Esta rama de la programación matemática permite abordar una amplia variedad de aplicaciones en logística, como la planificación de turnos y horarios.

Como caso de estudio se tiene una Estación de Servicio que busca mejorar su atención al cliente priorizando la rentabilidad y la estabilidad en el inicio de los turnos de los empleados. En esta tesis de licenciatura se buscará encontrar un cronograma de empleados que permita satisfacer estas cuestiones. A partir de este objetivo, se presentarán distintas variaciones de un modelo matemático cuya formulación cubra las necesidades del problema a resolver y luego se procederá con una experimentación computacional para evaluar la performance de cada variación. Estas variaciones se corresponden con activar o desactivar una restricción elástica dentro del modelo PLE. Por último, se realizará un análisis de resultados a partir de distintos parámetros comparativos entre los modelos promedio de cada variación.

Palabras claves: Programación Lineal Entera, Optimización, Asignación de Turnos, Rostering, solver, Restricciones Deseables/Elásticas, Predicción de Arribos de clientes, Modelo PLE.

AGRADECIMIENTOS

No solo no hubiera sido nada sin ustedes,
sino con toda la gente que estuvo a mi
alrededor desde el comienzo.
Algunos siguen hasta hoy.

Este camino no lo recorrí sola. Hoy cierro una etapa muy importante y no dejo de pensar en todas aquellas personas que estuvieron ahí. En primer lugar, agradezco a mi mamá, por estar desde el minuto uno de mi escolaridad apoyando cada proyecto nuevo que emprendía, cada idea loca que quería llevar adelante y cada verdad difícil de digerir. A todas las instituciones que me formaron, y me inculcaron el valor de la palabra libertad. A mis maestros de matemática, que me ayudaron a trazar un camino que luego se convertiría en mi forma de ver el mundo. A mis facu-amigos, por tolerar juntos cada frustración y desafío que acarrea ser alumno de esta facultad. A mis amigos, o mejor dicho a mis hermanos de la vida (ustedes saben a quiénes me refiero) por llorar y reír conmigo cuando buscaba un refugio seguro. A mi familia, que intenta hasta hoy en día comprender muchas cosas y me reciben en cada reunión con una sonrisa. A la persona que amo. Y por último, gracias a la música originaria de este país maravilloso, que llenó mis momentos de amparo y respuestas cuando lo necesitaba. Gracias Totales.

Índice general

1.. Introducción 1

1.1. Motivación y estado del arte 1

1.2. Objetivos 3

1.3. Estructura de la tesis 4

2.. Planteo del problema 5

2.1. Planteo general 5

2.2. Descripción de los datos 5

2.3. Estimación de la tasa de atención 6

3.. Modelo de PLE para asignación de turnos 8

3.1. Modelo general 8

3.2. Variables, Restricciones y Función Objetivo 8

3.2.1. Variables 9

3.2.2. Función Objetivo 9

3.2.3. Restricciones duras 9

3.3. Restricciones deseables 10

3.3.1. Restricción Deseable 1 11

3.3.2. Restricción Deseable 2 12

3.3.3. Restricción Deseable 3 13

3.3.4. Restricción Deseable 4 16

3.3.5. Restricción Deseable 5 16

3.3.6. Comentarios Finales del Modelo 17

4.. Experimentación y resultados 20

4.1. Resultados 20

5.. Conclusiones y trabajo futuro 29

1. INTRODUCCIÓN

Dentro del campo de la Investigación Operativa, los problemas de Asignación de Turnos al personal constituyen un desafío clásico. Tradicionalmente, en entornos como las Estaciones de Servicio con operación continua (abiertas las 24hs), la gestión de horarios se realiza de forma manual, basándose en la experiencia cotidiana. Este enfoque muchas veces no dimensiona cuantitativamente la afluencia real de clientes y el personal necesario para satisfacer esa demanda. Esto puede generar ineficiencia operativa, como tener más empleados de los necesarios en ciertos momentos y faltantes en otros; o generar colas de autos muy extensas esperando a ser atendidos, impactando negativamente en los costos y calidad del servicio.

Se propone abordar esta problemática mediante la formulación y análisis de un Modelo de Programación Lineal Entera (PLE) para la Asignación de Turnos Laborales [1] en una Estación de Servicio que opera 24 horas. Específicamente, se busca determinar una asignación óptima de turnos minimizando la cantidad de empleados que se necesitan para cubrir la demanda de los clientes, teniendo en cuenta *condiciones de estabilidad en el inicio de turnos* [2], [3], [4].

Para ello, se define un *Modelo General de PLE:* las variables (binarias y enteras), la función objetivo y las restricciones correspondientes. Luego, iterativamente, se van agregando las Restricciones de Estabilidad en los turnos (cada una corresponde a un modelo distinto). Posteriormente, se implementa el modelo en lenguaje Python y se resuelve con el solver SCIP [5].

Para analizar la performance del modelo, se realizará una experimentación computacional, haciendo foco en el impacto de las distintas restricciones deseables en el rendimiento del solver [6] (tiempos de ejecución, gap de optimalidad) y la calidad de las soluciones obtenidas en términos de la estabilidad de los turnos.

Esta Tesis de Licenciatura se plantea como una continuación del trabajo presentado en el artículo *"Predicción de arribos y programación de turnos en una estación de servicio"* [4], utilizando información sobre la distribución de los datos y la predicción de demanda generada mediante técnicas de aprendizaje automático.

1.1. Motivación y estado del arte

Dentro del campo de la Optimización, la Programación Lineal Entera (PLE) se presenta como una rama con una destacada capacidad para modelar problemas de decisión en los que las variables toman valores enteros. Esta característica le confiere una flexibilidad expresiva y semántica, permitiendo reflejar fielmente las restricciones inherentes a muchos sistemas del mundo real. Su aplicación se extiende a una amplia variedad de ámbitos, siendo especialmente relevante en áreas de logística como los *Problemas de Rostering* [7] [8] [9]. Dentro de esta categoría se encuentran aquellos problemas en donde se desea crear un cronograma de trabajo para el personal de una organización, como pueden ser los problemas de asignación de turnos o los de planificación de horarios del personal. Ejemplos de trabajos anteriores en Rostering aparecen en áreas tan diversas como asignación de turnos en medicina [1] [10] [11] [12] [13] [14] [15] [16] [17] [18], organización del personal en aerolíneas y otras áreas del transporte [19] [20] [21], y también con enfoques diferentes como una aplicación de algoritmos genéticos [22].

El problema general de PLE es **NP-hard** [23], [6]. Esto implica que –a la fecha– encontrar la solución óptima exacta requiere un tiempo de cálculo exponencial en el tamaño de la entrada. Los solvers moder-

nos para estos problemas utilizan heurísticas [23], [6] para encontrar buenas soluciones [3]. La mayor parte de los solvers se basa principalmente en técnicas de relajación de la integralidad de las variables, también conocidas como técnicas poliédricas [24]. Éstas consisten en buscar la solución óptima del PLE ignorando temporalmente los requisitos de integralidad de las variables (las variables solo toman valores numéricos enteros), resultando en un problema de programación lineal (PL) que es más fácil de resolver computacionalmente. Sin embargo, la solución óptima de esta relajación lineal a menudo no cumple con dicha condición de integralidad.

Con el fin de resolver esta cuestión, en la década de 1950, Ralph Gomory desarrolla Algoritmos de Planos de Corte [25]. Estos algoritmos buscan identificar *desigualdades válidas* (los “cortes") que son satisfechas por todas las soluciones enteras factibles [6], [2], [3], pero que excluyen la solución fraccionaria obtenida de la relajación lineal. En general, se separa en $y^i_j \leq \lfloor \bar{y}^i_j \rfloor$ e $y^i_j \geq \lceil \bar{y}^i_j \rceil$ donde $\bar{y}^i_j$ es la solución para la variable i-ésima en el estado j. Al añadir repetidamente estos cortes al problema de PL, se restringe progresivamente el conjunto factible de dicha relajación, acercándolo a la cápsula convexa del problema original (espacio factible) [6]. Si se pudiera describir a la perfección dicha cápsula convexa mediante desigualdades lineales, el problema entero podría resolverse simplemente con técnicas de programación lineal (por lo que podríamos usar Simplex o métodos de punto interior [6], que son exactos).

Si bien los algoritmos de planos de corte pueden garantizar terminación del método en teoría, en la práctica su rendimiento puede ser lento. Por lo tanto, en primera instancia se usaba un esquema más amplio conocido como *Branch and Bound* [6], [3], que consiste en podar bajo algún criterio de **factibilidad u optimalidad** algunas ramas del árbol binario de búsqueda para encontrar la solución óptima. Hoy en día los solvers más utilizados como GORUBI, CPLEX o SCIP utilizan variantes del algoritmo Branch-and-Cut, una combinación de branch-and-bound que genera dinámicamente los planos de corte. Esta estrategia se complementa con heurísticas y numerosos preprocesamientos para achicar el tamaño de la formulación.

Esto se hace seleccionando la mejor subrama con algún criterio/métrica que tenga que ver con la **optimalidad** (cuando el valor del *lower bound* obtenido de su relajación lineal es peor que la mejor solución entera encontrada hasta el momento) o la **factibilidad** (una rama que no va a llegar a una solución factible). El éxito computacional de estas técnicas ha permitido resolver instancias de problemas combinatorios más velozmente que antes [3].

La cantidad de iteraciones de un algoritmo *Branch-and-Cut* necesarias para encontrar una solución óptima depende significativamente de la formulación del Modelo PLE. Un factor crucial en el rendimiento es la *fuerza de la relajación lineal* de la formulación. Idealmente, se busca que la región factible de la relajación (obtenida al ignorar las restricciones de integralidad) sea lo más “cercana"posible al conjunto de soluciones enteras factibles. Formalmente, esto se relaciona con que la proyección de la región factible de la relajación coincida con la cápsula convexa de las soluciones enteras factibles.

Definición 1. Una formulación PLE de un conjunto $S \subset \mathbb{Q}^n$ es *sharp* si y solo si la proyección en las variables x de su relajación lineal es exactamente *conv(S)*, la cápsula convexa de la relajación. [3]

Por lo tanto, el *sharpness* es crucial porque una formulación *sharp* proporciona una cota inferior lo más estricta posible a partir de la relajación LP entre todas las formulaciones para ese conjunto específico S. Un lower bound más fuerte generalmente reduce el número de nodos que el solver de Branch and Cut necesita explorar, mejorando así el tiempo de cómputo. Diversas técnicas de formulación PLE influyen en la fuerza y el tamaño de la formulación. Estas incluyen el uso de variables auxiliares [3] para construir formulaciones más fuertes sin ser excesivamente grandes, el agregado de restricciones redundantes, y la descomposición del modelo en más de una componente, entre otras técnicas interesantes.

Aplicación a la asignación de turnos y objetivos del trabajo

El problema de la Asignación de Turnos comparte características con problemas de asignación de tareas y planificación de personal estudiados en la literatura [1]. El modelo PLE desarrollado busca capturar los requerimientos esenciales para la empresa, como cubrir la demanda a lo largo de un período de tiempo definido (una semana o un mes), respetando las reglas de los turnos (duración, no solapamiento, días francos), y asegurando una capacidad de atención suficiente basada en una tasa de servicio estimada.

Además de las restricciones estáticas (*hard*), el modelo incorpora **restricciones deseables o elásticas**. Estas reflejan la búsqueda de **estabilidad en los horarios de inicio de los turnos**, un aspecto muy valorado por el personal de una empresa. La tesis explora varias formulaciones para estas restricciones deseables, penalizando su incumplimiento en la función objetivo de manera que se priorice siempre la minimización del número de empleados. La modelización de estas condiciones flexibles, como maximizar la coincidencia de horarios de inicio con el día anterior o el último día trabajado, o lograr horarios fijos, introduce variables de holgura y penalizaciones en la función objetivo. Consecuentemente, esto añade complejidad a la formulación matemática y podría ser un factor importante en el rendimiento computacional del solver.

1.2. Objetivos

El presente trabajo tiene como objetivo general formular un modelo PLE que aborde el problema de asignación de turnos considerando tanto las restricciones duras (*hard*) como las restricciones elásticas (*soft*) de Estabilidad en el inicio de turnos para empresas de operación continua.

En primer lugar, dado un dataset de los arribos de los clientes en función de las horas, se quiere generar un cronograma de empleados que permita satisfacer la demanda del público, minimizando la cantidad de empleados contratados. Esta minimización de empleados es un factor clave para controlar el costo de personal y consecuentemente, mejorar la rentabilidad y la capacitación de los empleados. Este primer objetivo debe ser logrado en la formulación del modelo general, que incluya las variables, restricciones y función objetivo adecuadas. Se busca que el modelo sea correcto, legible y computable, y de forma deseable que sea minimal para facilitar su comprensión y proporcionar una formulación elegante.

En segundo lugar, se quiere evaluar la performance computacional de cada una de las Restricciones de Estabilidad en el inicio en los turnos (cada una corresponde a un modelo distinto). Para esto, se llevará a cabo un análisis de los resultados a través de las instancias disponibles. En la Bibliografía [3], [24] se menciona que las restricciones deseables inciden en el *sharpness* de una formulación, por lo que, en ese sentido, es interesante observar cómo afecta cada restricción deseable (RD) en esos tiempos de cómputo. Veremos que esto se puede estudiar con varios parámetros del resultado que arroja el solver.

Por último, se plantean preguntas sobre la flexibilidad de una RD afecta o no los tiempos de cómputo, en el sentido de la dificultad empírica de que esa restricción se cumpla en alguna asignación. En otras palabras, si una restricción impone horarios de entrada más amplios que otra, entonces la primera es más flexible que la segunda. Además, se busca comparar las RD entre sí con el fin de dar alguna respuesta concreta a la empresa que planteó el problema, exponiendo ventajas y desventajas de cada una y cuáles son aquellas condiciones de estabilidad que se prevee llevar a la práctica. Un agregado al trabajo será experimentar con cuestiones de rompimiento de simetría en el modelo general inspirado en la literatura, para analizar si estas formulaciones que agrandan el tamaño del modelo logran mejorar los tiempos de cómputo.

1.3. Estructura de la tesis

La presente Tesis de Licenciatura se organiza de la siguiente manera. El Capítulo 2 presenta el planteo detallado del problema de asignación de turnos, describiendo los datos de entrada y la estimación de parámetros clave como la tasa de atención. El Capítulo 3 desarrolla formalmente el Modelo de Programación Lineal Entera, detallando el modelo general con sus variables, función objetivo y restricciones obligatorias, y posteriormente introduciendo iterativamente las restricciones deseables. El Capítulo 4 expone el proceso de experimentación computacional, incluyendo la implementación del modelo en el solver y el análisis de los resultados obtenidos para las distintas configuraciones del modelo. Finalmente, el Capítulo 5 presenta las conclusiones del trabajo y discute posibles lineamientos de investigación futura.

2. PLANTEO DEL PROBLEMA

En este capítulo, se expondrá una descripción detallada del problema a resolver, junto con sus requerimientos y sus principales objetivos. Luego se presentarán los datos a analizar, explicando cuál es su formato y distribución para poder utilizarlos en el análisis del modelo del siguiente capítulo. Para finalizar con esta sección, se deriva la *tasa de atención*, que es la cantidad de servicios promedio que un empleado puede realizar en una hora. Este número es determinístico y tiene que ver con particularidades de la empresa, como la cantidad de surtidores disponibles y el flujo de personas que se puede generar.

Para la extracción de datos, se utiliza la Sección 3 del Paper *"Predicción de arribos y programación de turnos en una estación de servicio"*[4], reduciendo el problema en cuanto al tamaño del dataset (por los alcances computacionales) y utilizando la información del documento en cuanto a la distribución de los datos.

2.1. Planteo general

Se busca satisfacer la demanda de clientes que tiene una Estación de Servicio a lo largo de un período de tiempo (medido en horas). En primera instancia, el problema que queremos resolver consiste en la Asignación de Turnos Laborales para los empleados de la empresa. El objetivo principal es determinar qué días debe trabajar y a qué hora debe comenzar su turno cada trabajador, de modo tal de sólo contar con empleados que sean necesarios para satisfacer la demanda de clientes. Concretamente, se cuenta con el siguiente escenario:

- El horario de atención es 24hs (es decir, se atiende de corrido).
- Sea H $= \{1, ..., \#Horas\}$ el conjunto de horas de varias semanas (donde #Horas es múltiplo de 168), comenzando desde las 0:00 del lunes. Para cada $h \in H$, se conoce una estimación de C_h que denota la cantidad de clientes que ingresan en la hora h, y se conoce el tiempo promedio que se demora en atender la orden de un cliente.
- La empresa tiene una cantidad máxima de trabajadores que puede emplear. Cada trabajador debe realizar exactamente cinco turnos (de ocho horas cada uno) a lo largo de toda la semana. En principio, no hay restricciones sobre el horario en que debe empezar cada turno (Sección *Restricciones deseables*). Una observación es que no hay límites en la cantidad de surtidores que tenemos disponibles a priori, siendo un asunto a resolver más adelante.
- En cuanto a los turnos, cada trabajador puede empezar hasta un turno por día, y además no pueden solaparse, por lo que hasta que no termina el turno, no puede comenzar otro.

2.2. Descripción de los datos

Se generaron 25 instancias sintéticas con una longitud de una semana (168 horas) con la distribución sugerida en [4], luego de procesar los datos eliminando los outliers.

En la *Figura 2.1* se ilustra la demanda en función de los datos disponibles. Desde las 00:00 hasta las 05:00 inclusive, los arribos promedio por hora se mantienen en valores bajos, no superando los 12 autos por hora. A partir de las 06:00 comienza una marcada tendencia alcista en la demanda, que se intensifica rápidamente entre las 07:00 y las 12:00 (primer pico). Durante este intervalo, los arribos promedian entre 36 y 51 autos por hora, alcanzando su primer pico a las 12:00 con un promedio de 50.56 autos por hora.

Luego de este primer pico, la demanda se estabiliza levemente entre las 13:00 y las 15:00, manteniéndose en niveles elevados (en torno a 44-47 autos por hora). A partir de las 16:00, se observa un segundo ascenso que culmina en el segundo pico día a las 18:00, con un promedio de 59.56 autos por hora. Posteriormente, la demanda permanece relativamente alta a las 19:00 (55.60 autos por hora) y luego inicia un descenso sostenido a partir de las 20:00. A partir de las 21:00, la cantidad de arribos cae abruptamente, descendiendo hasta los 16.44 autos por hora en promedio a las 23:00.

Este comportamiento refleja un una distribución bimodal, con concentraciones importantes de demanda en franjas horarias vinculadas a inicios y finales de jornadas laborales de los habitantes de una ciudad.

En cuanto a la variabilidad, se observa una dispersión considerable entre los cuartiles y los extremos. Por ejemplo, a las 19:00, la diferencia entre el primer y el tercer cuartil es de aproximadamente 24 autos, lo que va a ser el principal desafío para el modelo. Además, la diferencia entre el valor mínimo y máximo en esa hora puede superar los 100 autos, lo que implicaría un riesgo operacional significativo si no se cuenta con suficiente personal en el lugar. Una observación en este punto es que va a haber momentos entre los dos picos con empleados desocupados, pero con el objetivo de cubrir la demanda de los dos picos aledaños, sin perder clientes.

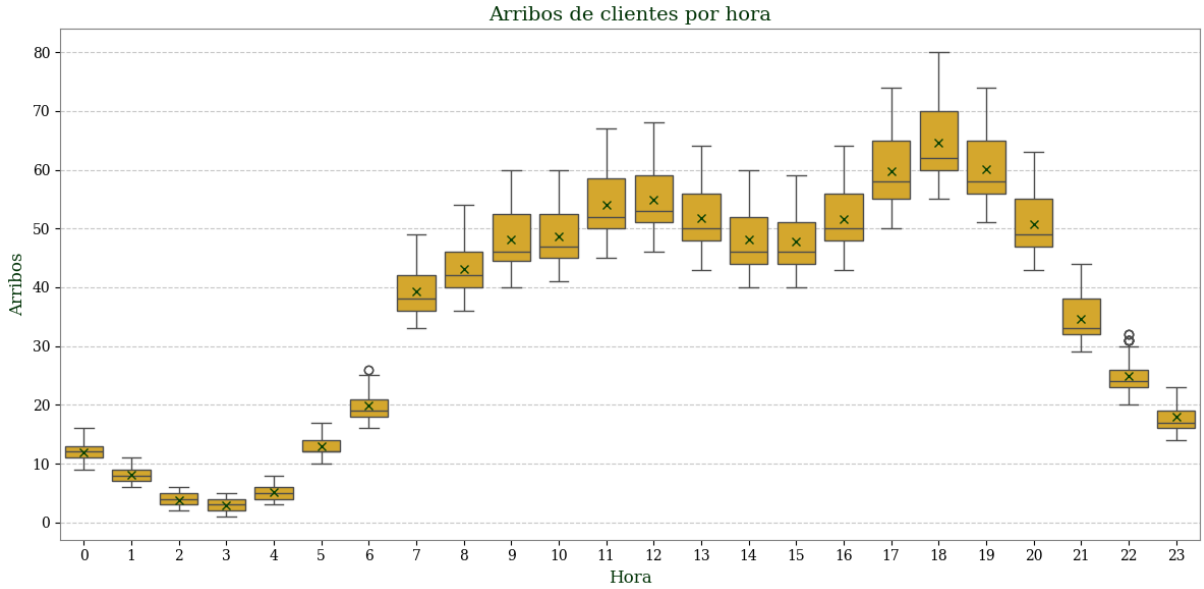

Fig. 2.1: Arribos de clientes por hora del día

2.3. Estimación de la tasa de atención

Definición 2. La **tasa de atención** se define como el número promedio de clientes que puede atender un empleado $e \in E$ en una hora laboral $h \in H$.

El uso de este promedio permite tener una formulación determinística, evitando las dificultades tanto teóricas como computacionales de un modelo estocástico. Si bien entre un empleado y otro puede haber cierta varianza no despreciable en su tiempo de servicio, en el trabajo previo [4] que motivó esta tesis se vio empíricamente que el uso de este promedio fue adecuado en la práctica, capturando razonablemente las necesidades de empleados en función de la demanda. Tanto en simulaciones como en la operatoria real, las soluciones generadas en [4] utilizando este promedio cubrieron adecuadamente la demanda.

Con el fin de obtener este número, se debe medir de alguna manera el tiempo de servicio promedio a un cliente. Dado que no se cuenta con información directa sobre los tiempos de servicio en la estación, éstos fueron inferidos a partir de los datos disponibles, calculando el intervalo entre despachos durante franjas horarias con flujo continuo de atención. Se puede asumir que si un empleado está desocupado y llega un cliente nuevo, este cliente será atendido automáticamente. Si suponemos está misma lógica en horarios donde se forma una cola de clientes, un flujo continuo de servicio es cuando cada vez que un empleado se desocupa, hay un cliente nuevo para atender.

Se consideraron únicamente aquellos despachos cuyo tiempo fue menor o igual a tres segundos por litro despachado, más un adicional de cinco minutos estimado como tiempo de cobro. Las observaciones que superaban este umbral fueron descartadas por presumirse afectadas por demoras ajenas al proceso de servicio. El análisis se realizó sobre un total de 556.067 registros correspondientes a 38 meses de operación y abarcando las seis bocas de despacho de la estación. Como resultado, se estimó un tiempo promedio de servicio de 3.24 minutos por cliente, con una desviación estándar de 1.50 minutos [4].

Adicionalmente, se puede asumir que cada empleado puede operar hasta dos caras de un surtidor sin comprometer la calidad del servicio. Esto es debido a que el despacho de combustible no requiere supervisión continua, por lo tanto se considera razonable asumir que el tiempo efectivo de atención por cliente por empleado es la mitad del estimado para una sola boca de despacho. Esta suposición, comúnmente aceptada en el sector, fue validada mediante la observación de registros de cámaras de seguridad. Bajo este criterio, se estima un tiempo de servicio de 1.62 minutos por cliente por empleado, lo que equivale a una tasa de servicio de aproximadamente 37.02 clientes por hora por empleado.

A partir de la distribución de datos de [4], se simularon los datos para este trabajo, que son muchos menos en cantidad, por lo que logicamente se asume la misma distribución de datos original. Por lo tanto, se utiliza la misma tasa de atención. Además de la tasa de atención estimada en 37.02 clientes por hora, se consideraron también valores alternativos (30, 25 y 20 clientes por hora) de forma exploratoria, como un recurso ad hoc para analizar el impacto de distintos niveles de exigencia en el servicio.

3. MODELO DE PLE PARA ASIGNACIÓN DE TURNOS

En la siguiente sección, se detalla la solución para el problema de Optimización de Turnos de Trabajo bajo el marco de un modelo de PLE. Se presentan las variables, restricciones y función objetivo de manera general y precisa, derivando en una formulación legible, correcta, bien estructurada y programable en cada caso. A su vez, es menester aclarar que no hay una sola forma de realizarla, pero es deseable que sea minimal en términos de restricciones por cuestiones numéricas en la experimentación.

Cada parte de la solución del problema se muestra en forma extensiva, explicando en un principio su propósito en palabras (o informalmente), para luego ser expresada matemáticamente. En primer lugar, se encuentra el Modelo General que cumple los requerimientos básicos para cubrir la demanda de la empresa. En segundo lugar, se agregan iterativamente las restricciones elásticas que corresponden con la estabilidad en el inicio de los turnos.

3.1. Modelo general

El Modelo General tiene como principal objetivo, a partir de los datos de entrada, dar una formulación matemática para el problema que permita **cubrir la demanda del negocio minimizando la cantidad de empleados contratados**. Dicha formulación es detallada a continuación.

3.2. Variables, Restricciones y Función Objetivo

Para definir el modelo, enumeramos previamente los iteradores y las constantes con las que va a contar. Como veremos en la siguiente sección, los mismos se corresponderán con los objetos del constructor de la clase `Instancia`. Sean:

- Empleados: $i \in \{1, ..., \#Emp\}$, donde $\#Emp$ se fijó en 20 empleados a partir de las condiciones planteadas por el rubro. De todas formas, esta constante podría variar según el problema.
- Horas: $h \in \{0, ..., \#Horas - 1\}$, donde $\#Horas$ es un múltiplo de 168, correspondiendo con la cantidad de horas que hay en una semana. Es determinada por el dataset.
- Días laborales: `#dias_laborales` = (#Horas/24) - `cant_francos` que son la cantidad de días en los que un trabajador puede comenzar un turno (ya que se arranca uno por día). `cant_francos` también es definido por el problema, y en este caso hay dos por semana.
- C_h: la cantidad de órdenes de clientes que llegaron para cada hora h. Éste es el número que contiene cada instancia del dataset.
- Se tiene la **tasa de atención**. Este número es determinista y tiene que ver con pedidos del negocio, como la cantidad de surtidores disponibles y el flujo de personas que se puede tener. Para una hora, si se tiene una cantidad de llamadas y una tasa, la **demanda por hora** se define como la cantidad de clientes esperados en una hora dividido por lo que requiere atenderlos:

$$\text{demanda\_por\_hora}_h = \frac{C_h}{\text{tasa\_atencion\_clientes}} \tag{3.1}$$

3.2.1. Variables

Se definen las variables binarias del modelo:

$$e_i = \begin{cases} 1 & \text{si el trabajador } i \text{ es contratado} \\ 0 & \text{caso contrario} \end{cases} \tag{3.2}$$

y

$$x_{i,h} = \begin{cases} 1 & \text{si el trabajador } i \text{ comienza un turno en la hora } h \\ 0 & \text{caso contrario} \end{cases} \tag{3.3}$$

3.2.2. Función Objetivo

Se quiere contratar la menor cantidad de trabajadores con el fin de mejorar la rentabilidad del negocio, satisfaciendo en todo momento la demanda de los clientes. Así, la función objetivo es

$$\textbf{min} \sum_{i=1}^{\#Emp} e_i. \tag{3.4}$$

3.2.3. Restricciones duras

- Sean O_h variables enteras ligadas a las variables $x_{i,h}$ que, para una hora h, indican la cantidad de trabajadores disponibles. Recordando que los turnos duran 8 horas, están disponibles todos los empleados que arrancaron a trabajar en esta hora y en hasta 7 anteriores:

$$O_h = \sum_{i=1}^{\#\text{Emp}} \sum_{j=max\{0,h-7\}}^{h} x_{i,j}. \quad \forall h \tag{3.5}$$

- Cada trabajador que fue contratado empieza 5 turnos por semana. Si el empleado es contratado, tiene que cumplir con la cantidad de días laborales que se imponen en la empresa, por lo tanto tiene que empezar esa cantidad de turnos según la cantidad de horas totales. Además, si el empleado no es contratado, tiene que pasar que nunca empiece ningún turno:

$$\sum_{h=1}^{\#Horas} x_{i,h} = \texttt{\#dias\_laborales} \cdot e_i. \quad \forall i \tag{3.6}$$

- Un trabajador no puede iniciar más de un turno en una ventana de 8 horas (todos los turnos duran lo mismo y no hay solapamiento de turnos para un trabajador):

$$\sum_{j=max\{0,h-7\}}^{h} x_{i,j} \leq 1 \ \ \forall i, \forall h. \tag{3.7}$$

- Cada día el empleado i-ésimo empieza a lo sumo un turno. Sea d un iterador sobre los días. Entonces:

$$\sum_{h=0}^{23} x_{i,(h+24\cdot d)} \leq 1 \ \ \forall i, \forall d \in \{0, ..., \#(Horas/24) - 1\}. \tag{3.8}$$

- Cada trabajador que fue contratado tiene que gozar de exactamente 2 francos por semana, un nú-

mero que depende de la empresa y es pre-definido. Sea s un iterador sobre las semanas. Entonces:

$$\sum_{h=0}^{167} x_{i,(h+168\cdot s)} \leq \texttt{\#dias\_laborales} \;\; \forall i, \forall s = \{0, ..., (\#Horas/168) - 1\}. \tag{3.9}$$

- Para toda hora se quiere cubrir la demanda de clientes. Sabemos que la demanda por hora es una medida de empleados necesarios, y que O_h es una variable que mide cuántos empleados hay disponibles. Para cubrir la demanda, se quiere que los empleados disponibles para cada hora siempre superen a los necesarios. Entonces la resta de la demanda por O_h siempre debe ser negativa:

$$\frac{C_h}{\text{tasa\_atencion\_clientes}} - O_h \leq 0. \tag{3.10}$$

3.3. Restricciones deseables

Una vez que está planteado el modelo general, la idea es considerar restricciones elásticas (también conocidas como deseables) que correspondan con la búsqueda de estabilidad en el inicio de los turnos. En el contexto de programación lineal entera, una restricción deseable o elástica es una restricción que no es estrictamente obligatoria, pero que se prefiere satisfacer en la solución óptima en algún nivel junto con la función objetivo principal. Para modelarla formalmente, este *nivel* en el que se va a cumplirla restricción se modela mediante una penalización incorporada en la función objetivo, a la que llamaremos δ_k, donde k es el número que le adjudicamos en este trabajo a cada una de estas restricciones deseables planteadas por distintas necesidades del rubro.

Formulación matemática

Dada una restricción deseable del tipo:

$$a^\top x \leq b,$$

se introduce una variable de holgura $s \geq 0$ que representa la cantidad en que se permite violar la restricción, reformulándola como:

$$a^\top x \leq b + s.$$

Para penalizar esta violación, se modifica la función objetivo original mín$\{c^\top x\}$ agregando un término proporcional al exceso:

$$\text{mín} \;\; c^\top x + \delta_k \cdot s,$$

donde $\delta_k > 0$ es un parámetro de penalización.

Ahora bien, ¿cómo se agregan al modelo este tipo de restricciones? Dentro del modelo, se las agrega como cualquier otra restricción y luego en la función objetivo se indica el peso o importancia que va a tener la restricción a través de la penalización con las δ_k, de la siguiente manera.

Función Objetivo (modificada) Ahora a la función objetivo original se suman las funciones objetivo asociadas a su respeciva δ_k.

$$\textbf{min} \sum_{i=1}^{\#Emp} e_i + \delta_1 \cdot \text{obj}[1] + ... + \delta_n \cdot obj[n] \tag{3.11}$$

donde el objetivo k-ésimo es una función de la Restricción Deseable k (RD_k) que se agrega a las restric-

ciones del modelo:

$$\textbf{obj[k]} = f(RD_k).$$

Elección de las δ

El valor del parámetro δ_k se elige en este modelo en función del dominio de esta restricción deseable, comparándolo con el dominio de la función objetivo original de manera tal de que **siempre se priorice minimizar los empleados**. Para este cometido, la idea general es que todo valor que pueda tomar la restricción deseable debe estar acotado por el mínimo valor que pueda tomar el dominio (se nota DOM) de la función objetivo original, matemáticamente:

$$min\{\text{DOM}(\#\text{Emp})\} > max\{\text{DOM}(\text{RD}_k)\} \quad \forall k. \tag{3.12}$$

Se puede asumir que $min\{\text{DOM}(\#\text{Emp})\} = 1$ en todos los casos. Esto es porque por lo menos un empleado es contratado. Además $max\{\text{DOM}(\text{RD}_k)\}$ es distinto para cada k, pero por lo anterior sabemos que $\delta_k = \frac{1}{max\{\text{DOM}(\text{RD}_k)\}+1}$ es un valor general adecuado que siempre cumple la desigualdad. De todas formas, se detalla cuál es ese valor en cada caso.

A continuación, se presentan las cinco restricciones deseables (RD) que se decidió agregar al modelo. En cada caso, se da una descripción en castellano, para luego presentar la formulación. Por último, se concluye la sección con un breve comentario sobre detalles implementativos previo a la programación del solver y a las diferencias y similitudes entre ellas para intentar predecir cómo la flexibilidad impactará a la hora de resolver el problema computacionalmente.

3.3.1. Restricción Deseable 1

Maximizar la cantidad de veces que un empleado empieza su turno en el mismo horario que el día anterior.

La forma de pensar esto (fijando al i-ésimo empleado) es que cada vez que el horario elegido en el día actual coincide con el horario del turno del día anterior, "suma un punto". Si en el día anterior el empleado no trabajó, la variable representativa debe ser 0, y si no se trabaja el día actual también o no coincide el horario también.

Sean:

$$A_{i,h,d} = \begin{cases} 1 & \text{si el trabajador } i \text{ empieza su turno a la hora } h \text{ en los días } d-1 \text{ y } d \text{ simultáneamente} \\ 0 & \text{caso contrario} \end{cases}$$

donde $h \in \{0, \dots, 23\}$, $d \in \{1, \dots, (\texttt{\#Horas}/24) - 1\}$.

Esta restricción se formula a partir de una conjunción lógica. Se quiere que cuando las dos variables binarias asociadas estén activas simultáneamente (valgan 1), que consecuentemente $A_{i,h,d}$ también lo esté, y que en todos los demás casos valga 0. Esto se modela de forma clásica con las siguientes dos desigualdades:

$$\Rightarrow \begin{cases} 2 \cdot A_{i,h,d} \leq x_{i,h+24\cdot d} + x_{i,h+24\cdot(d-1)} \\ x_{i,h+24\cdot d} + x_{i,h+24\cdot(d-1)} \leq 1 + A_{i,h,d} \end{cases} \tag{3.13}$$

$\forall \quad i \in \{1, \dots, \#E\}, h \in \{0, \dots, 23\}, d \in \{1, \dots, (\texttt{\#Horas}/24) - 1\}.$

Función objetivo: maximizar la cantidad de veces que $A_{i,h,d}$ vale 1.

$$\text{máx} \sum_{i=1}^{\#\text{Emp}} \sum_{h=0}^{23} \sum_{d=2}^{\#\text{días}} A_{i,h,d} \quad = \quad \text{mín} - \sum_{i=1}^{\#\text{Emp}} \sum_{h=0}^{23} \sum_{d=2}^{\#\text{días}} A_{i,h,d} \tag{3.14}$$

Elección del parámetro de penalización δ_1

Como se mencionó anteriormente, se busca que el denominador de la δ sea el valor máximo que puede tomar el dominio de la restricción deseable. En este caso, si se tienen #Emp empleados y como máximo $A_{i,h,d} =$ #dias_laborales, entonces:

$$\delta_1 = \frac{1}{\#\text{Emp} \cdot \#\text{dias\_laborales} + 1} = \frac{1}{101} \tag{3.15}$$

3.3.2. Restricción Deseable 2

Maximizar la cantidad de veces que un empleado empieza su turno a la misma hora o una hora antes o después respecto al día anterior.

La formulación de esta restricción es similar a la anterior, salvo por el hecho de que se agregan dos términos (la hora anterior y la hora posterior del día actual) y se tienen que cumplir exactamente dos de ellas para que la nueva variable binaria esté prendida. Por ejemplo, si el día actual el empleado fue designado a las 8, "suma un punto" si el día anterior había comenzado entre las 7 y las 9, para esta longitud de intervalo.

$$C_{i,h,d} = \begin{cases} 1 & \text{si el trabajador } i \text{ empezó su turno a la hora } h \text{ del día } d-1 \text{ y en } \{h-1, h, h+1\} \text{ en el día } d, \\ 0 & \text{caso contrario} \end{cases}$$

donde $i \in \{1, \dots, \#\text{Emp}\}, h \in \{0, \dots, 23\}, d \in \{1, \dots, (\texttt{\#Horas}/24) - 1\}$.

La idea es modelar la siguiente condición lógica:

$$x_{i,h+24\cdot(d-1)} = 1 \Rightarrow x_{i,(h-1)+24\cdot d} = 1 \vee x_{i,h+24\cdot d} = 1 \vee x_{i,(h+1)+24\cdot d} = 1.$$

Queda una expresión de conjunción similar a la de la RD_1, solo que se agregaron las variables de la hora anterior y la hora siguiente (lo que flexibiliza el modelo en el caso de encender RD_2). Finalmente, el modelado en forma lineal (las variables no se multiplican) es:

$$\Rightarrow \begin{cases} 2 \cdot C_{i,h,d} \leq x_{i,h+24\cdot(d-1)} + x_{i,(h-1)+24\cdot d} + x_{i,h+24\cdot d} + x_{i,(h+1)+24\cdot d} \\ x_{i,h+24\cdot(d-1)} + x_{i,(h-1)+24\cdot d} + x_{i,h+24\cdot d} + x_{i,(h+1)+24\cdot d} \leq 1 + C_{i,h,d}. \end{cases} \tag{3.16}$$

Para simplificar, se define la suma de los tres horarios posibles para empezar en el día actual y así mejorar la legibilidad del modelo.

$$\texttt{intervalo\_válido} := x_{i,(h-1)+24\cdot d} + x_{i,h+24\cdot d} + x_{i,(h+1)+24\cdot d}.$$

Entonces la restricción puede escribirse como:

$$\Rightarrow \begin{cases} 2 \cdot C_{i,h,d} \leq x_{i,h+24\cdot(d-1)} + \texttt{intervalo\_válido} \\ x_{i,h+24\cdot(d-1)} + \texttt{intervalo\_válido} \leq 1 + C_{i,h,d} \end{cases} \tag{3.17}$$

para todo $i \in \{1, \ldots, \#E\}$, $h \in \{0, \ldots, 23\}$, $d \in \{1, \ldots, (\texttt{\#Horas}/24) - 1\}$.

Función objetivo: maximizar la cantidad de veces que se cumpla la condición deseada:

$$\text{máx} \sum_{i=1}^{\#\text{Emp}} \sum_{h=0}^{23} \sum_{d=2}^{\#\text{días}} C_{i,h,d} \quad = \quad \text{mín} - \sum_{i=1}^{\#\text{Emp}} \sum_{h=0}^{23} \sum_{d=2}^{\#\text{días}} C_{i,h,d} \tag{3.18}$$

Elección del parámetro de penalización δ_2: En el caso de RD2, se tienen #Emp empleados y como máximo $C_{i,h,d} = \#\text{dias\_laborales}$, entonces:

$$\delta_2 = \frac{1}{\#\text{Emp} \cdot \#\text{dias\_laborales} + 1} = \frac{1}{101} \tag{3.19}$$

Esta restricción se puede extender a otro intervalo centrado en la hora en la que se arrancó un turno en el día anterior. Sea $2 \cdot k$ la longitud del intervalo genérico centrado en $x_{i,h+24(d-1)}$, entonces la restricción para intervalos genéricos se puede formular de la misma forma que la anterior, re-definiendo `intervalo_válido` como:

$$\texttt{intervalo\_válido} := x_{i,max\{0,(h-k)+24\cdot d\}} + \ldots + x_{i,h+24\cdot d} + \ldots + x_{i,min\{\#\text{dias},(h+k)+24\cdot d\}}.$$

En este caso tanto la cota de empleados como la de días laborales queda igual, por lo que la elección de δ_2 es la misma que antes.

3.3.3. Restricción Deseable 3

Maximizar la cantidad de veces que un empleado empieza su turno en el mismo horario que el último turno trabajado (antes del día actual).

Ahora lo que hay que guardar de los días anteriores para el día actual es el último turno, sin importar si fue el día anterior o no. La descripción semántica de la restricción es parecida a RD_1, salvo por el hecho de que tiene que poder ver más allá del alcance de un solo día. Si la restricción se cumple entonces implica RD_1.

Para explicar cómo se pensó el modelo de esta restricción, suponer que en el día d actual, el empleado i arranca algún turno (es un día laboral)$^{(*)}$. [1] En el modelo original, se planteó que cada trabajador arranca 5 turnos cada semana y que cada día hace un único turno. Esto implicaría que un total de dos días semanales son francos (en el MG se agregó aparte que sean *exactamente* dos por semana). Por lo tanto, lo que se busca para esta restricción deseable es tener alguna forma de guardarse para cada día laboral, cuándo fue el último turno que arrancó el trabajador, aunque hayan pasado días en el medio.

Ahora bien, puede que el día actual y el día anterior sean ambos laborales. En este caso la restricción se reduce a la RD1, y por lo tanto la cantidad mínima de francos que se pueden tener entre dos días es cero. Para la cantidad máxima, se puede observar por ejemplo que no hay ninguna restricción acerca de que el trabajador se tome franco un {sábado, domingo, lunes, martes}, ya que los primeros dos pertenecen

[1] $^{(*)}$ Ver la sección *Decisiones de modelado* al final del capítulo

a una semana y los segundos dos a la otra, por lo que la cantidad máxima de francos seguidos es cuatro.

Dicho esto, se querría tener una variable que dependa de la cantidad de francos anteriores, y a partir de eso se pueda identificar el último día en que se trabajó y chequear si el turno en ese día se arrancó a la misma hora que en el día actual.

Se quiere saber si un día es franco o no. Sea:

$$f_{i,d} = \begin{cases} 1 & \text{si el empleado } i \text{ tiene franco el día } d \\ 0 & \text{caso contrario} \end{cases} \quad \Rightarrow f_{i,d} = 1 - \sum_{h=0}^{23} x_{i,h+24d}. \tag{3.20}$$

A partir de saber si es franco o no, se quiere conocer el primer día hacia atrás a partir del día actual que no fue franco [3]. Se crea otra variable para eso:

$$\ell_{i,d,k} = \begin{cases} 1 & \text{si el último turno trabajado antes del día } d \text{ por el empleado } i \text{ fue en el día } d-k, \\ 0 & \text{caso contrario} \end{cases}$$

donde $k \in \{1, \dots, \text{mín}\{5, d\}\}$.

Ahora bien, se busca que este "último no franco" sea uno solo en esa ventana, entonces se puede utilizar una formulación estándar de la bibliografía [26], [19].

$$\sum_{k=1}^{5} \ell_{i,d,k} \leq 1$$

- $k = 1$**:**

$$\ell_{i,d,1} \leq 1 - f_{i,d-1}$$

- $k = 2$**:**

$$\begin{aligned} \ell_{i,d,2} &\leq f_{i,d-1} \\ \ell_{i,d,2} &\leq 1 - f_{i,d-2} \end{aligned}$$

- $k = 3$**:**

$$\begin{aligned} \ell_{i,d,3} &\leq f_{i,d-1} \\ \ell_{i,d,3} &\leq f_{i,d-2} \\ \ell_{i,d,3} &\leq 1 - f_{i,d-3} \\ &\vdots \end{aligned}$$

Esto en general quedaría:

$$\ell_{i,d,k} \leq f_{i,j} \quad \wedge \quad \ell_{i,d,k} \leq 1 - f_{i,d-k} \quad \forall k, j = \{1, ..., k-1\} \tag{3.21}$$

Ahora que se tienen los dos días que nos interesan, se requiere una variable para maximizar la cantidad de veces que coincide la hora elegida para empezar un turno en el día actual (laboral) con el último horario trabajado. Además, se quiere que funcione tanto para el caso en el que día actual sea o no laboral para el

empleado. Se define:

$$s_{i,d,h} = \begin{cases} 1 & \text{si el empleado } i \text{ comienza un turno el día } d \text{ y } d-j \text{ a la hora } h \\ 0 & \text{caso contrario} \end{cases}$$

donde j se corresponde con la cantidad de días de distancia entre el día actual y el último día en el que se trabajó. Esta variable está sujeta a las siguientes restricciones:

$$s_{i,h,d} \leq x_{i,h+24d} \text{ (solo va a estar prendida si } x_{i,h} \text{ lo está)}$$
$$s_{i,h,d} \leq \sum_{k=1}^{5} x_{i,h+max\{1,24(d-k)\}} \cdot \ell_{i,d,k} \text{ (acotada por el día que importa)}$$
$$s_{i,h,d} \leq 1 - f_{i,d} \text{ (solo prendida en un día d que se trabaje).}$$

Pero el segundo tipo de restricción no es lineal, por lo que se debe crear otra variable de conjunción como la de las restricciones anteriores. Básicamente, tiene que ser una variable que valga 1 si tanto $x_{i,h+max\{0,24(d-k)\}}$ como $\ell_{i,d,k}$ son iguales a 1 al mismo tiempo y 0 si no. Sea:

$$m_{i,h,d,k} = \begin{cases} 1 & \text{si el empleado } i \text{ comienza un turno el día } d \text{ a la hora } h \\ & \text{y a la vez el día } d-k \text{ es el último día en que se trabajó} \\ 0 & \text{caso contrario.} \end{cases}$$

Entonces se deben armar las restricciones para conjunción:

$$\Rightarrow \begin{cases} 2 \cdot m_{i,h,d,k} \leq x_{i,h+max\{1,24(d-k)\}} + \ell_{i,d,k} \\ x_{i,h+max\{1,24(d-k)\}} + \ell_{i,d,k} \leq 1 + m_{i,h,d,k} \end{cases} \tag{3.22}$$

$\forall i \in \{1, ..., \#E\}, h \in \{0, ..., 23\}, d \in \{2, ..., \#\text{Horas}/24\}, k \in \{1, .., 5\}$

Finalmente, lo anterior queda linealizado como:

$$s_{i,h,d} \leq x_{i,h+24d} \text{ (Solo va a estar prendida si } x_{i,h} \text{ lo está)} \tag{3.23}$$
$$s_{i,h,d} \leq \sum_{k=1}^{5} m_{i,h,d,k} \text{ (Acotada por el día que nos importa)} \tag{3.24}$$
$$s_{i,h,d} \leq 1 - f_{i,d} \text{ (sólo prendida en un día } d \text{ que se trabaje).} \tag{3.25}$$

Función objetivo: para definirla, se tiene que las s solo están acotadas por abajo porque son las que se van a querer maximizar. La semántica de s también se puede pensar como la cantidad de coincidencias (entre el horario de inicio del turno actual y el del último turno trabajado):

$$\text{máx} \sum_{i=1}^{\#\text{Emp}} \sum_{d=1}^{\#\text{días}} \sum_{h=0}^{23} s_{i,h,d}. \tag{3.26}$$

Elección del parámetro de penalización δ_3: Si hay D días y #Emp empleados, y como mucho se puede tener una coincidencia por día trabajado, el máximo valor posible de la suma es #Emp $\cdot$ D. Por lo tanto,

$$\delta_3 = \frac{1}{\#\text{Emp} \cdot \#\text{dias} + 1} = \frac{1}{141}. \tag{3.27}$$

3.3.4. Restricción Deseable 4

Maximizar la cantidad de empleados que empiezan siempre a la misma hora.

Semánticamente, esto significa que se necesita una variable por empleado que solo "esté prendida" cuando todos los inicios de turno para dicho trabajador sean iguales. Por ejemplo, que si empleado 1 empiece todos sus turnos a las 8 la totalidad de los días laborales, entonces la variable que se quiere maximizar "suma un punto".

Primero se elige de forma combinatoria un primer horario de entrada para cada empleado y ese horario queda fijo para todo el resto de los días siguientes.

Sea:

$$W_{i,h} = \begin{cases} 1 & \text{si el trabajador } i \text{ comienza todos sus turnos a la hora } h, \\ 0 & \text{caso contrario} \end{cases}$$

donde $i \in \{1, \dots, \#\text{Emp}\}$ y $h \in \{0, \dots, 23\}$.

Para que un empleado tenga una única hora fija de entrada, debe cumplirse:

$$\sum_{d=0}^{\#\text{días}} x_{i,h+24d} \geq \texttt{\#dias\_laborales} \cdot W_{i,h} \quad \forall i, h. \tag{3.28}$$

Esta igualdad fuerza a que la variable $W_{i,h}$ valga 1 únicamente si el empleado comienza todos sus turnos en esa hora. De otra forma, $W_{i,h} = 0$, implicando que el empleado no comienza todos sus turnos a la misma hora.

Función objetivo:

$$\text{máx} \sum_{i=1}^{\#\text{Emp}} \sum_{h=0}^{23} W_{i,h} \quad = \quad \text{mín} - \sum_{i=1}^{\#\text{Emp}} \sum_{h=0}^{23} W_{i,h} \tag{3.29}$$

Elección del parámetro de penalización δ_4:

Dado que por cada empleado como máximo puede haber un $W_{i,h} = 1$, se tiene:

$$\delta_4 = \frac{1}{\#\text{Emp} + 1} = \frac{1}{21}. \tag{3.30}$$

3.3.5. Restricción Deseable 5

Fijar un horario de entrada para cada empleado, y minimizar la diferencia total respecto a ese horario.

Esta restricción se construye a partir del resultado de la RD_4, donde se detecta a posteriori un horario z_i representativo de entrada para cada empleado. Se detalla el criterio de armado del vector z a continuación.

Criterio para determinar z:

1. Se computa, para cada hora $h \in \{0, ..., 23\}$, cuántas veces fue elegida por un empleado como horario de inicio fijo en RD_4. Esto expresa la cantidad de "uso"de cada hora, ya que los empleados son indistinguibles. Estas cantidades se guardan en un vector v de pares `(h, #vecesElegida)`. Se ordenan las horas de forma decreciente por segunda posición, es decir por `#vecesElegida`. Una

observación es que es indistinto ordenar `v` de forma creciente, ya que los empleados son indistinguibles. Se arma un vector z de la siguiente manera: se itera por los pares del vector `v`, y se agrega al vector z la hora `h, #vecesElegida` veces.

Por ejemplo, un posible vector z de horarios fijos podría ser:

z = [0,0,16,16,8,8,7,7,15,15,6,14,13,12,5,17,11,3,19,4]. En este caso se eligieron las horas del día 0,7,8,15 y 16 como las designadas para los primeros 10 empleados en índice (2 para cada uno en ese orden), y luego el resto de horarios del vector para un solo empleado cada vez.

Para formular la función objetivo, sean:

$$y_{i,h} = \sum_{d=0}^{\#\text{días}} x_{i,h+24d} \quad \text{con } h \in \{0, \dots, 23\} \tag{3.31}$$

Es decir, $y_{i,h}$ representa cuántos días el empleado i comenzó su jornada a la hora h.

Función objetivo: minimizar la suma ponderada por la diferencia horaria con respecto al horario fijo asignado:

$$\text{mín} \sum_{i=1}^{\#\text{Emp}} \sum_{h=0}^{23} y_{i,h} \cdot |z_i - h|. \tag{3.32}$$

Esta suma penaliza toda desviación respecto al horario fijo de entrada, siendo mayor cuanto más lejos esté el horario real de z_i.

Elección del parámetro de penalización δ_5:

Para cada empleado, la suma de las $y_{i,h}$ pueden ser a lo sumo `dias_laborales` y el valor del módulo es como mucho la diferencia horaria máxima entre dos horas, que es 12 (sino observar un reloj analógico).

Entonces:

$$\delta_5 = \frac{1}{\#\text{Emp} \cdot \texttt{\#dias\_laborales} \cdot 12 + 1} = \frac{1}{1201}. \tag{3.33}$$

3.3.6. Comentarios Finales del Modelo

Flexibilidad

En primer lugar, es interesante observar cuáles restricciones son más parecidas. Quizás la relación más obvia está entre RD_1 con RD_2, cuya formulación es prácticamente igual, salvo por el hecho de que en RD2 se agregan dos términos (el de la hora anterior y la hora posterior), y para una extensión general se agregarían los términos del `intervalo_valido`. En el primer caso, para que las $A_{i,h,d}$ estén prendidas, del día anterior sólo se puede elegir una $x_{i,h}$ mientras que en el otro caso hay tres opciones (o más, depende el `intervalo_valido` elegido) para que $C_{i,h,d}$ esté prendida. Entonces se podría alegar que RD2 es más laxa que RD_1.

En segundo lugar, se podría decir que RD_4 es la menos flexible de todas, debido a que todos los días el empleado tiene que comenzar a la misma hora para que $W_{i,h} = 1$. Por otro lado, la RD_5 más allá de depender de RD_4, permite iniciar el turno en un día no franco en cualquier momento (más allá de que aporte menos en un caso que h esté lejos de z_i para algunos casos, es perceptible).

Por último, RD3 es una extensión de RD1, convirtiéndose la segunda en un caso particular de la primera. Entonces, RD3 es más flexible que RD1 pero bastante más difícil de enunciar y modelar, debido a que se necesitaron muchas más variables para controlarla.

Decisiones de modelado

En todas las restricciones, para explicar cada formulación, se analiza la semántica a partir de asumir que o bien $x_{i,h} = 1$, o bien $x_{i,h+24\cdot d} = 1$, y que si vale cero no importa. Alguien podría preguntarse por qué esto es correcto. Para explicarlo se podría tomar el siguiente ejemplo para RD2. Supongamos que se tiene $x_{1,26} = 0$, $x_{1,27} = 1$, $x_{1,28} = 0$, $x_{1,50} = 1$, $x_{1,51} = 0$, $x_{1,52} = 0$. Estas variables corresponden al día 2 y 3. Si la restricción en vez de ser la que se tiene fuera:[2]

$$\begin{cases} 2 \cdot C_{i,h,d} \leq x_{i,h+24\cdot(d-1)} + x_{i,(h-1)+24\cdot d} + x_{i,h+24\cdot d} + x_{i,(h+1)+24\cdot d} \\ x_{i,h+24\cdot(d-1)} + x_{i,(h-1)+24\cdot d} + x_{i,h+24\cdot d} + x_{i,(h+1)+24\cdot d} \leq 1 + C_{i,h,d} \end{cases} \vee$$

$$\Rightarrow \begin{cases} 2 \cdot C_{i,h,d} \leq x_{i,h+24\cdot d} + x_{i,(h-1)+24\cdot(d-1)} + x_{i,h+24\cdot(d-1)} + x_{i,(h+1)+24\cdot(d-1)} \\ x_{i,h+24\cdot d} + x_{i,(h-1)+24\cdot(d-1)} + x_{i,h+24\cdot(d-1)} + x_{i,(h+1)+24\cdot(d-1)} \leq 1 + C_{i,h,d} \end{cases}$$

Si se agrega: $x_{1,74} = 0$, $x_{1,75} = 1$, $x_{1,76} = 0$ del día 4, ahora $C_{1,3,3} = 2$ porque se cumple la restricción tanto para un lado como para el otro, pero entonces C deja de ser binaria, por lo que se pierde el poder semántico de la variable. Esto se evita con el modelo planteado, que contempla que esto puede pasar y cuenta todo una sola vez. Esto se puede extender a un `intervalo_valido` más grande, y puede notarse como este ejemplo se extiende a las demás restricciones.

Restricciones de rompimiento de simetría

Existen estrategias en la literatura que pueden agregar al modelo restricciones que podrían mejorar los tiempos de cómputo. En modelos como este, se puede notar que existen muchísimas soluciones (factorial de empleados contratados) equivalentes ya que los empleados son indistinguibles. Eso puede generar que el solver explore todas esas soluciones simétricas y las opciones que revisa sean muchas más de las que realmente interesan. Se decidió en este trabajo investigar con el modelo general si el agregado de restricciones de rompimiento de simetría tiene algún impacto positivo en los tiempos de cómputo, sin afectar la factibilidad u optimalidad del problema.

Una forma de hacerlo sería ordenar a los empleados según algún criterio. En este caso se eligió ordenarlos por tiempos de inicio en la semana por empleado. Lo que se hace es sumar todas las horas de inicio de turnos de cada empleado, y la restricción semanticamente busca que esa suma sea menor en el empleado i que en el empleado j cuando $i < j$. Por ejemplo, el empleado 1 tiene esa suma más chica de horas que el empleado 3.

$$\sum_{h=0}^{\#\text{Horas}} h \cdot x_{i,h} \leq \sum_{h=0}^{\#\text{Horas}} h \cdot x_{i+1,h} \quad \forall i \in \{1, ..., \#Emp - 1\} \tag{3.34}$$

Una aclaración con respecto a esto. Puede pasar que la suma de los inicios de turno sea 0 si el empleado no es contratado. En tal caso, asumimos que en las desigualdades de la formulación van a quedar los empleados de menor índice "desempleados" (es decir, sin turnos asignados ya que la suma es cero), y los de mayor índice son los contratados con el criterio que impone la restricción.

[2] Básicamente, si se observa tanto la ventana de horas ± 1 desde el día d como desde el día d-1, se puede notar que con esta formulación, $C_{1,4,2} = 1$ y $C_{1,3,3} = 1$, por lo que se estaría contando la relación entre los días 2 y 3 dos veces.

Complejidad

Los mejores algoritmos conocidos para la solución exacta de problemas generales de PLE requieren un tiempo exponencial en la cantidad de datos de entrada. No se conocen algoritmos polinomiales para este problema, y esto es consecuencia de que el problema general de PLE es NP-hard. No obstante, existen solvers que, aunque implementan algoritmos que en el peor caso requieren un tiempo exponencial, suelen tener buena performance para muchas instancias (dado que incluyen heurísticas, planos de corte y técnicas de descomposición de los problemas). Por estos motivos, en el siguiente capítulo se utilizará un solver de PLE para intentar la resolución computacional del problema planteado. El objetivo en nuestro caso es analizar la performance del solver, interpretar cómo se comporta ante las distintas RDk, y evaluar su potencial aplicación en la práctica.

Para el análisis del modelo en el siguiente capítulo, es pertinente presentar las siguientes cuestiones para intentar responder una vez que se tengan los resultados. Es interesante observar cuáles son las RD que más demoran en terminar su ejecución, si la cantidad de variables y restricciones en el modelo juega un rol en el tiempo de ejecución del solver, y cuáles son las restricciones que pueden resultar más interesantes para armar el calendario final de empleados.

4. EXPERIMENTACIÓN Y RESULTADOS

En este capítulo se explora todo el proceso experimental llevado a cabo con el objetivo de probar empíricamente distintas alternativas de modelos para cumplir con los requerimientos del negocio. Se analizan formalmente los resultados que obtuvo el solver para cada parte del modelo, teniendo en cuenta las características de flexibilidad y penalización de cada restricción deseable. Al final, se concluye el capítulo intentando responder las preguntas que surgieron al formular el modelo en el Capítulo 3, y realizando observaciones finales sobre la performance computacional del solver.

4.1. Resultados

A continuación, se presenta el análisis detallado de los resultados obtenidos de la experimentación computacional, con el objetivo de abordar las preguntas de investigación planteadas previamente. Este análisis se basa en la evaluación del rendimiento del solver ante las distintas variaciones del modelo general para la asignación de turnos, haciendo especial énfasis en el impacto de las restricciones deseables (RD) en métricas como el tiempo de ejecución, el gap de optimalidad y la cantidad de empleados contratados en comparación con el modelo general (RD_0).

Las herramientas computacionales utilizadas en este trabajo sufren de problemas de performance con instancias de más de una semana. Por este motivo, en este capítulo se utilizan instancias de una sola semana, pero el modelo se puede utilizar con instancias de varias semanas. Al intentar con varias semanas, el solver interrumpe su ejecución por falta de memoria o no llega a un gap de optimalidad que le permita realizar alguna asignación de turnos a empleados razonable. En el Anexo B se dan detalles sobre estas cuestiones.

Para el análisis de resultados, se utilizarán todas las 25 instancias. La cantidad de configuraciones posibles de los tres parámetros de entrada (instancia x RD_k x tasa) es 25 x 6 x 4 = 600. Cada uno de ellos se encuentra en un archivo txt que contiene la información que genera `PySCIPOpt` (Fig 4.1, 4.3). Esta información incluye (1) Scip Status: {timelimit, optimal}. Se corresponde con el criterio de parada del solver; si es timelimit significa que alcanzó los 1800s, y si es optimal alcanzó una solución cuya diferencia relativa entre la cota primal y dual es menor a `TOLERANCE`. (2) Solving Time $\in \{0.0, 1800.0\}$. Cantidad de tiempo en que se demora el solver en llegar a algún criterio de parada. (3) Solving Nodes ($\in \mathbb{Z}^+$): cantidad de nodos que potencialmente expanden el árbol de ramificación. (4) Primal Bound $\in \mathbb{R}$: cota primal. Como es un problema de minimización, se corresponde con la cota superior que utiliza el árbol de búsqueda del *Branch and Cut*. En caso de `timelimit`, se utiliza la solución combinatoria correspondiente con este valor óptimo. (5) Dual Bound: $\in \mathbb{R}$: cota dual. Como es un problema de minimización, se corresponde con la cota inferior que utiliza el árbol de búsqueda del *Branch and Bound*. (6) Gap: $\frac{\text{Primal Bound}}{|\text{Primal Bound} + \text{Dual Bound}|}$.

En las Figuras 4.2 y 4.4 se tiene la asignación de turnos a empleados dados distintos valores que puede tomar la tupla (instancia $\times$ RD_k $\times$ tasa). En ambas tablas se refleja qué empleados de entre los 20 posibles son contratados (primera columna), y cuáles son sus horarios de entrada en esa semana (medidos en horas de la semana). Las variables y restricciones que figuran en cada caso se corresponden con el tamaño del modelo luego del `presolving`, una etapa de pre-procesamiento que se encarga de extraer del modelo las partes redundantes. Gracias a esta heurística, se pueden analizar los resultados bajo el supuesto de que todos los modelos se resolvieron utilizando la mínima cantidad de variables y restricciones según el caso. Una aclaración importante que también se menciona en el Anexo C es que los Id de la columna empleados

Parámetro	Valor
SCIP Status	optimal
Solving Time	4.0 s
Solving Nodes	1
Primal Bound	11.0
Dual Bound	11.0
Gap	0.0

Fig. 4.1: Descripción del resultado

Empleado	Turno 1	Turno 2	Turno 3	Turno 4	Turno 5
6	16	31	111	136	157
8	71	85	113	126	162
10	35	75	98	127	149
11	13	44	60	124	154
13	33	63	90	103	159
14	21	40	52	88	101
15	6	83	108	135	152
16	7	39	56	106	132
17	15	39	66	108	146
18	0	61	78	116	138
19	8	28	54	80	131

Fig. 4.2: Programación final de inicio de turnos por empleado (en horas) para la instancia 1 con el modelo RD_0 y tasa de atención 25

Parámetro	Valor
Scip Status	timelimit
Solving Time	1800.0
Solving Nodes	2455
Primal Bound	10.57
Dual Bound	10.48
Gap	0.009

Fig. 4.3: Descripción del resultado

Empleado	Turno 1	Turno 2	Turno 3	Turno 4	Turno 5
1	0	62	110	134	151
2	20	44	68	92	116
3	36	60	84	108	132
4	30	54	78	102	126
7	6	54	102	126	150
8	8	32	80	128	152
9	8	42	89	142	157
10	12	36	60	84	108
12	28	52	76	100	124
13	15	87	111	135	159
18	16	40	64	136	160

Fig. 4.4: Programación final de inicio de turnos por empleado (en horas) para la instancia 1 con el modelo RD_4 y tasa de atención 20

Comparación de dos soluciones del modelo. Arriba: criterio de parada por `TOLERANCE`. Abajo: criterio de parada por `timelimit`.

elegidos podrían haber sido cualquier otra combinación entre *n* números del 1 al 20, con *n* entre 1 y 20. La razón por la que se eligen las combinaciones de las tablas es arbitraria y depende de decisiones internas del solver, dado que los empleados son indistinguibles.

Se pudo observar que cuando dos instancias corresponden a una misma tasa de atención y a un mismo modelo de restricción deseable, al correr el solver éstas tienen resultados parecidos. Por lo tanto, para responder a los cuestionamientos planteados, se resumen las instancias en el promedio de los resultados, donde en cada fila están agrupadas las 25 instancias que tienen la misma tasa y pertenecen a la misma RD.

El modelo general (RD_0), que no considera la estabilidad de turnos, sirve como línea base para la comparación (a lo que también se suele llamar "control" en experimentación). Los resultados promedio para este modelo muestran que se resuelve rápidamente, con tiempos de ejecución bajos (entre 4.40 y 6.16 segundos) y un gap de optimalidad menor a `TOLERANCE` ($<10^{-6}$). Esto indica que el solver SCIP logra encontrar la solución óptima de forma eficiente para el problema principal de minimización de personal. La cantidad de empleados contratados en el modelo general varía según la tasa de atención, siendo 8 empleados con una tasa de 37.02 clientes/hora a 11 empleados con una tasa de 20 clientes/hora. Esto corrobora que la capacidad de atención por empleado es un factor determinante en la cantidad de

personal contratado.

Al incorporar las restricciones deseables de estabilidad en el inicio de los turnos (RD_1 a RD_5), el comportamiento computacional del solver cambia significativamente. En general, se observa un incremento sustancial en el tiempo de resolución y, en muchos casos, el solver alcanza el límite de tiempo de 1800 segundos establecido por instancia, lo que resulta en un gap de optimalidad mayor a cero. La Tabla 4.1 ilustra claramente esta diferencia: mientras que RD0 siempre termina con un gap de 0.0 y bajos tiempos, las RDs 1, 2, 3 y la 4 y 5 (para tasas bajas) frecuentemente llegan al límite de tiempo. La Figura 4.6 visualiza estos gaps no nulos para las instancias que terminaron por `timelimit` y no por `TOLERANCE`.

Observando la tabla de resultados (Tabla 4.1), se pueden notar que parece haber una relación entre la cantidad de variables/restricciones promedio y el gap de optimalidad. En general, mientras mayor es el número de variables/restricciones, mayor diferencia relativa se presenta entre las cotas. Además, prácticamente todas las instancias con más de 3800 variables tuvieron como criterio de parada `timelimit`.

Instancias x	#Emp	#Variables	#Restricciones	Solving Nodes	Solving Time	Gap
RD_0 x 37.02	8			1	6.16	0.000000
RD_0 x 30.0	8	3380	7022	1	4.40	0.000000
RD_0 x 25.0	10			1	4.60	0.000000
RD_0 x 20.0	11			1	5.00	0.000000
RD_1 x 37.02	8			3873	1800.00	0.117556
RD_1 x 30.0	8	6260	10044	2381	1800.00	0.080906
RD_1 x 25.0	10			1936	1800.00	0.073528
RD_1 x 20.0	11			727	1800.00	0.023226
RD_2 x 37.02	8			602	1800.00	0.118950
RD_2 x 30.0	8	5780	14161	798	1800.00	0.081931
RD_2 x 25.0	10			1006	1800.00	0.069919
RD_2 x 20.0	11			474	1800.00	0.023748
RD_3 x 37.02	9			139	1800.00	0.333077
RD_3 x 30.0	9	13400	22342	119	1800.00	0.184932
RD_3 x 25.0	10			128	1800.00	0.080487
RD_3 x 20.0	11			99	1800.00	0.085916
RD_4 x 37.02	8			3845	1542.34	0.070373
RD_4 x 30.0	8	3860	7502	2066	1598.40	0.061256
RD_4 x 25.0	10			10320	1800.00	0.055828
RD_4 x 20.0	11			2594	1800.00	0.016152
RD_5 x 37.02	8			183885	1795.36	0.067860
RD_5 x 30.0	8	3380	7022	61005	1800.00	0.072479
RD_5 x 25.0	10			151007	1800.00	0.103763
RD_5 x 20.0	11			54130	1800.00	0.150720

Tab. 4.1: Resultados promedio de 25 instancias correspondientes con cada RD y tasas de atención

La restricción deseable 3 (RD_3), que maximiza la coincidencia del horario de inicio con el último turno trabajado (considerando días francos), introduce la mayor complejidad en la formulación en términos de cantidad de variables y restricciones. En promedio, RD_3 presenta el mayor número de variables (en promedio 13400) y restricciones (en promedio 22342) en comparación con otras RDs. Sin embargo, en términos de programación de horarios, es la única que pudo mantener lógicamente el inicio a la misma hora con francos en el medio, que en la práctica es muy útil para el negocio. Si bien estos gaps son grandes en comparación con el resto de RDs, para los estándares de este tipo de problemas [4], son gaps de optimalidad aceptables.

Consistentemente, RD_3 es la que muestra el peor rendimiento computacional, alcanzando el límite de tiempo en todos los casos y reportando los gaps de optimalidad más altos (hasta 0.333077). Esto sugiere

que la cantidad de variables y restricciones introducidas por una formulación compleja tiene un fuerte impacto negativo en el tiempo de cómputo y la capacidad del solver para encontrar una solución óptima dentro del límite de tiempo. Los resultados señalan explícitamente que la mayoría de las instancias con más de 3800 variables o restricciones no alcanzaron el criterio de parada por `TOLERANCE` dentro de la media hora.

Al comparar RD_1 (mismo horario que el día anterior) y RD_2 (mismo horario o una hora antes/después que el día anterior), se esperaba que RD_2, al ser semánticamente más flexible, pudiera ser computacionalmente más fácil de resolver. Sin embargo, los resultados muestran que ambas RDs exhiben gaps de optimalidad y tiempos de resolución promedio similares, con RD_2 incluso presentando ligeramente más restricciones (Fig. 4.6). Esto apoya la observación de que la cantidad de variables y restricciones pesa más que la flexibilidad semántica a la hora de determinar la dificultad empírica de resolver el problema.

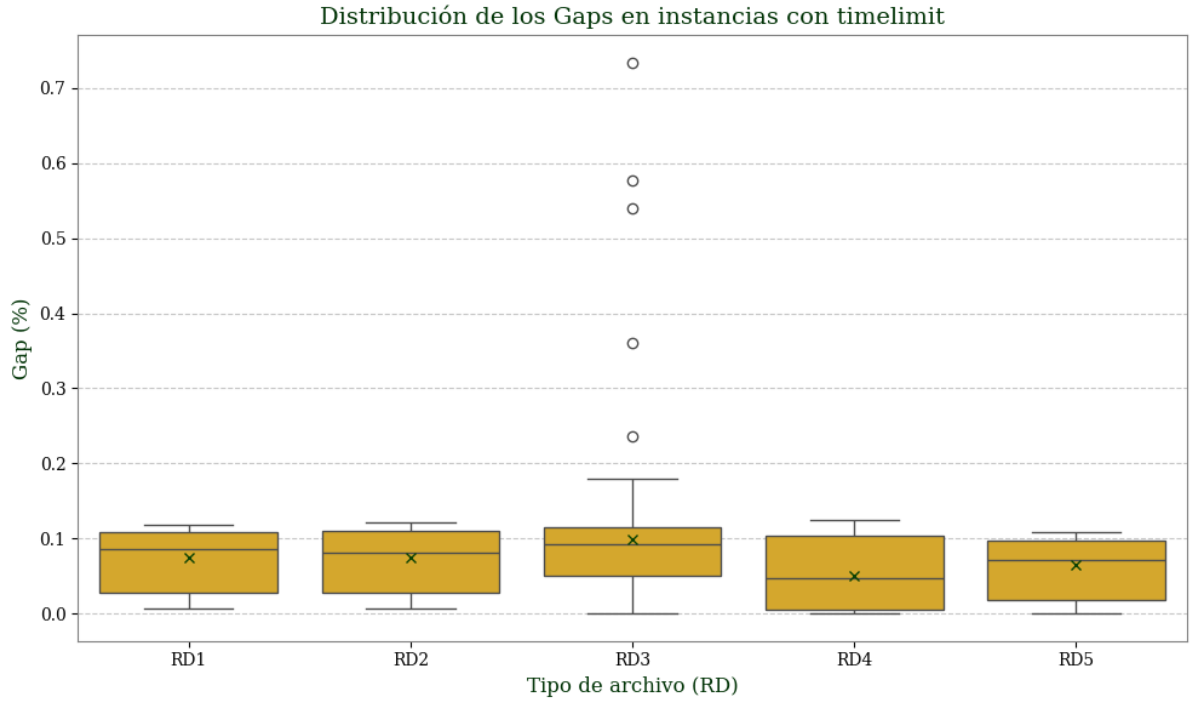

Fig. 4.5: Gaps relativos entre la cota primal y dual para las restricciones que no terminaron (`time-limit`).

La restricción deseable 4 (RD_4), que busca que los empleados empiecen siempre a la misma hora, fue identificada conceptualmente como la menos flexible. Sus resultados computacionales son mixtos: con tasas de atención más altas (37.02 y 30), se resuelve en algo más de 20 minutos con un gap cercano a cero. Sin embargo, con tasas más bajas (25 y 20), donde posiblemente se requiere una mayor adaptación de horarios para cubrir la demanda, RD_4 alcanza el límite de tiempo y presenta gaps elevados en comparación con las tasas altas. Esto podría indicar que, aunque menos flexible, parece depender mucho de la tasa de atención, y por lo tanto se encuentra una especie de dependencia con esa restricción estática. A pesar de ser menos flexible que RD_1 o RD_3, RD_4 generalmente presenta gaps más pequeños que ambas restricciones, lo que refuerza la idea de que el tamaño de la formulación es crucial.

La restricción deseable 5 (RD_5), que minimiza la diferencia total respecto a un horario fijo predeterminado, se basa en un análisis posterior de los resultados de RD_4. Aunque introduce variables auxiliares en la función objetivo, la estructura base de variables y restricciones es similar a la de RD_0, debido a que solo se agrega una variable dependiente de las $x_{i,h}$ que no agranda el número de variables luego del `presolving` de `SCIP`. Computacionalmente, RD_5 a menudo alcanza el límite de tiempo, pero en algunos casos muestra tiempos de resolución promedio menores que RD_1, RD_2 o RD_3, y gaps moderados. Esto

sugiere que, aunque añade complejidad a la función objetivo, la estructura de las restricciones principales heredada de RD_0 la hace algo más manejable que las formulaciones más extensas de RD_1, RD_2 o RD_3. RD_5 sería un buen ejemplo de una situación semejante a la realidad, en donde el empleado tiene horarios más rígidos y previamente fijados; o donde por cuestiones de dinámica laboral o colaborativas, se requiere que varios empleados empiecen a la vez. Fijar los horarios en estos contextos tiene bastante sentido.

Análisis cualitativo de las soluciones

En este apartado se analiza la calidad cualitativa de las soluciones del solver en términos de requerimientos de negocio a partir del modelado propuesto. Para este apartado, con el fin de lograr una mejor comprensión del lector, los horarios expresados en horas corridas ahora están expresados según la hora que es de ese día (módulo 24), por ejemplo la hora 30 se convierte en la hora 8 del día martes. Las horas van de 0 a 23. Las columnas van de lunes a domingo. Los días laborales tienen su respectivo horario de inicio y los francos se denotan con una letra F.

El foco del análisis es revisar si cada restricción brinda estabilidad en los inicios de turnos, y de qué forma cumplió las restricciones estáticas observables. Sin embargo, algunos cumplimientos son más difíciles de observar. Por ejemplo, sabemos que los turnos estrictamente deberían duran 8 horas, pero no hay forma de leerlo de las tablas, puesto que como se cumple la restricción de un turno por día, consecuentemente no se puede discernir la duración de los turnos. Lo que sí se ve claramente en las Tablas 4.2 y 4.4 por su formato de solución es que todos los turnos tienen una diferencia de 8 horas o más. Podemos suponer que debido a ser restricción estática es seguro que los turnos duran 8 horas.

Previo a analizar cada caso, hay cuestiones generales en todas las soluciones. A primera vista es claro que los empleados en todas las tablas trabajan exactamente 5 días y tienen 2 francos. Para revisar en detalle el cumplimiento de la demanda, se comparan las asignaciones con la distribución de los datos de demanda operativa de la Figura 2.1. Los horarios de mayor caudal de clientes son cerca del mediodía y al horario de salida del trabajo (aproximadamente 18hs), habiendo un primer salto a las 7 de la mañana y comenzando a descender abruptamente a partir de las 19hs. Aunque la demanda nocturna es menor, en todas las tablas hay turnos que cubren la demanda en la noche. La mayoría de los turnos se espera que comiencen entre las 7 y las 18hs. Aparte, se conoce que la demanda cercana al comienzo del fin de semana es mayor que en el resto de la semana, por lo que se esperan menos francos en esos días.

Modelo General

El modelo general tenía como único objetivo minimizar la cantidad de empleados satisfaciendo la demanda de clientes. No hay ninguna restricción de estabilidad en los turnos por empleado, por lo que los horarios varian a lo largo de toda la semana para cada uno. La cantidad de empleados necesarios para cubrir los requerimientos de esta instancia es 9, aunque en los resultados promedio esta cantidad es 8, y algunas instancias tienen menos de eso. Se deriva del análisis anterior que la tasa de atención de clientes influye mucho en la cantidad de empleados en pos de cumplir la demanda, por lo que cuando baja la tasa, se pueden observar tablas con más empleados como veremos en casos posteriores.

En cuanto al cubrimiento de la demanda, en esta instancia particular, se ve que hubo mayor demanda de clientes el martes y el sábado y menor el resto de los días. Esto se puede observar a través de la cantidad de francos. Además, solo 1 o 2 empleados por día cubren horarios nocturnos, mientras que el resto tiene horarios cercanos a los picos de la distribución, lo que permite inferir a grandes rasgos que los turnos coinciden con la demanda operativa.

Empleado	lunes	martes	miércoles	jueves	viernes	sábado	domingo
0	7	F	7	18	17	F	16
1	F	13	F	10	2	7	9
7	F	15	2	15	17	9	F
11	F	2	15	2	F	11	1
13	16	7	9	F	7	4	F
14	8	0	F	F	12	17	14
17	15	6	18	5	F	16	F
18	F	6	F	13	9	1	6
19	0	18	17	F	F	17	1

Tab. 4.2: instancia 25 x RD_0 x tasa30

Restricción deseable 1

En el caso de esta restricción, lo que se esperaba visualizar es el mismo horario de inicio de turno para la mayoría de los empleados **solo para días contiguos**. Como la restricción no se extiende a más de un día, cuando hay un franco en el medio, se espera que en la asignación el trabajador retome la jornada laboral en cualquier horario.

Empleado	lunes	martes	miércoles	jueves	viernes	sábado	domingo
1	12	F	F	7	7	7	7
2	F	F	16	16	16	16	16
3	14	14	F	11	11	11	F
6	F	19	19	19	19	19	F
9	5	5	5	5	5	F	F
12	8	8	8	F	F	5	5
14	F	11	11	11	11	11	F
15	0	0	F	F	13	13	13
18	F	F	3	3	3	3	3
19	17	17	17	17	F	F	14

Tab. 4.3: instancia 20 x RD_1 x tasa25

Los resultados de la tabla son los esperados. Si tomamos por ejemplo al empleado 12 el lunes, martes y miércoles inicia los turnos a las 8 de la mañana, tiene franco jueves y viernes, y luego retoma en otro horario según la demanda, a las 5 de la mañana. Esto podría ser porque, de suponer que no estuviera el empleado 12, los horarios de la mañana solo los cubriría el empleado 1 y como vimos en la distribución, probablemente para esa instancia no alcanza. Esto sumado a que la tasa de atención es 25, y se necesitaron 10 empleados para asignar los turnos, algo predecible ya que el promedio para esta restricción es de 10 empleados para esta tasa.

Restricción deseable 2

En este caso, se esperaba observar es el mismo horario de inicio de turno para la mayoría de los empleados **para días contiguos con flexibilidad de una hora**. Además, esta restricción no se extiende a más de un día, por lo que ante un franco, se espera que en la asignación el trabajador retome la jornada laboral en cualquier horario.

Empleado	lunes	martes	miércoles	jueves	viernes	sábado	domingo
2	F	17	18	17	17	16	F
7	14	13	F	F	5	6	5
8	16	16	15	F	F	7	7
10	F	F	7	6	5	6	13
11	6	7	8	8	9	F	F
12	8	9	F	F	22	22	9
13	0	0	0	1	1	F	F
14	16	F	F	13	12	13	17
16	F	F	12	12	13	14	15
17	F	16	15	15	16	15	F
19	8	8	7	6	F	11	F

Tab. 4.4: instancia15 x RD_2 x tasa20

La asignación coincide con lo requerido. Si centramos la atención en el empleado 7, el lunes empieza a las 14hs, así que lo esperado es que el martes comience entre las 13 y las 15. Efectivamente tiene asignado arrancar a las 13hs. Luego miércoles y jueves tiene franco, y luego retoma al final de la semana, también respetando la flexibilidad de una hora de diferencia entre viernes (5hs), sábado (6hs) y domingo (5hs). Por otro lado, la tasa de atención es 25, lo que explica la necesidad de 11 empleados para asignar los turnos, que coincide con el promedio para la combinación RD_2 x tasa20.

Restricción deseable 3

Para esta restricción **sí se tienen en cuenta los francos para mantener el mismo horario**. En este caso, sí se observan algunas discrepancias con la formulación original, pero hay que recordar que al ser una restricción elástica, no siempre todos los empleados tendrán asignaciones al mismo horario todos los días de la semana (como el empleado 17). De todas formas, la performance no es tan buena como en RD_1 y RD_2 cualitativamente. Más allá de eso, sí es capaz de propagar el horario a través de los francos en algunos casos (10, 11, 14, 17, 18).

Empleado	lunes	martes	miércoles	jueves	viernes	sábado	domingo
10	F	7	7	7	7	7	F
11	16	8	8	8	F	F	8
12	14	F	F	23	23	23	23
13	0	F	F	14	14	14	14
14	F	16	16	16	F	16	16
16	8	0	0	0	6	F	F
17	F	7	7	F	7	7	7
18	8	15	15	F	15	15	F

Tab. 4.5: instancia10 x RD_3 x tasa37.02

Esta fluctuación horaria podría deberse a cuestiones de demanda, pero sobre todo parece tener que ver con que el gap de optimalidad no es tan ajustado como el resto de restricciones. Una opción sería dejar correr RD_3 más tiempo para obtener asignaciones mejores. La cantidad de empleados es 8, que coincide con los promedios para esta tasa de atención.

Restricción deseable 4

Empleado	lunes	martes	miércoles	jueves	viernes	sábado	domingo
0	19	19	19	F	F	19	19
1	3	3	3	3	F	F	3
2	16	F	F	16	16	16	16
3	6	F	6	F	6	6	6
4	13	13	13	F	13	F	13
6	0	0	F	0	0	0	F
7	F	16	16	16	16	16	F
8	F	8	F	8	8	8	8
10	11	11	11	11	F	11	F

Tab. 4.6: instancia5 x RD_4 x $tasa30$

Para la cuarta restricción, la solución fijaba un horario para cada empleado que **se mantiene a lo largo de toda la semana.** Observando la tabla, la semántica del modelo está reflejada en la asignación. Añadiendo a esto, la asignación de horarios en general tiene sentido con la demanda, contratando 9 empleados a tasa 30. Esto no solo beneficia a la RD_4 como candidata para la elección del modelo elegido para la empresa, sino que además logró terminar en 25 minutos.

Restricción deseable 5

Por último, en la restricción deseable 5 se fijaba el horario utilizando RD_4 (o en un caso más realista, horarios sugeridos por la empresa) para los inicios de turnos, para luego **penalizar el alejamiento del horario elegido en un día contra el horario fijado en el vector** z. La semántica de la restricción y la asignación están bien relacionadas.

Empleado	lunes	martes	miércoles	jueves	viernes	sábado	domingo
0	0	0	0	0	0	F	F
2	F	16	16	16	F	16	16
3	16	16	F	F	16	16	16
4	8	8	F	8	8	8	F
6	7	7	7	F	F	7	7
8	15	F	F	15	15	15	15
13	12	F	12	12	12	12	F
16	F	11	11	11	9	F	11
18	F	19	19	19	20	20	F
19	F	F	4	4	4	4	4

Tab. 4.7: instancia1 x RD_5 x tasa25

En primer lugar es importante realizar la observación acerca de la *robustez a francos* que tiene RD_5, algo que podría no suceder a priori. Solo en dos casos se desvía un tanto del horario fijado, para el empleado 16 y 18. En cuanto a la demanda operativa, se parece a asignaciones anteriores y tiene los horarios distribuidos consistentemente. Otro punto a favor es el tiempo de cómputo. A tasa 30 el solver con el modelo de la restricción 5 termina por optimalidad, por lo que se infiere que dejando correr esta instancia a tasa 25 un poco más de tiempo, se llegaría a una asignación cercana al óptimo. Los 10 empleados coinciden con el promedio para la tripla (instancia1 $\times$ RD_5 $\times$ tasa25).

Experimentación con restricciones de rompimiento de simetría

Para analizar los cambios en el tiempo computacional bajo esta restricción, se realizó un experimento detallado en el Anexo C. En el mismo se exploró la performance de agregar esta restricción en el solver y los resultados mostraron que tanto los gaps de optimalidad como el resto de parámetros son prácticamente los mismos que con la formulación original. En algunos casos empeoran los tiempos de ejecución, aunque este cambio no es significativo.

Conclusiones

Para concluir este análisis, la experimentación computacional demuestra que agregar objetivos de estabilidad en el inicio de los turnos aumenta significativamente la complejidad computacional del problema de asignación de turnos en comparación con el modelo general que solo minimiza empleados. Este incremento es observable directamente de los resultados, y más explícitamente en la comparativa entre la Figura 4.2 y la Figura 4.4, que ilustran cómo RD_0 termina en solo segundos pero lógicamente no respetando ninguna estabilidad en los turnos, mientras que RD_4 tiene un tiempo de cómputo significativamente mayor pero manteniendo una estabilidad elástica en el inicio de turnos.

Esto pone de manifiesto un trade-off: lograr una mayor estabilidad horaria (objetivo de las RDs) puede requerir un esfuerzo computacional considerable, potencialmente resultando en soluciones no óptimas si se cuenta con menos tiempo de resolución. La elección de qué condición de estabilidad llevar a la práctica dependerá de la importancia relativa que la empresa le otorgue a la estabilidad del horario bajo algún criterio, en parte porque aunque haya diferencias entre las restricciones elásticas en los tiempos computacionales, los gaps de optimalidad alcanzados son razonables para fines prácticos en todos los casos. A la par de este hecho, las asignaciones realizadas por el solver mayoritariamente cumplen la estabilidad de inicio en los turnos impuesta por cada restricción deseable, aunque tengan sus diferencias semánticas son funcionales para el objetivo original de cada una. Ahora bien, estos gaps son razonables para una sola semana (ver el Anexo B). Quedaría como experimentación futura ver cómo bajar los tiempos computacionales de instancias más grandes para que la empresa tenga la libertad de elegir la restricción elástica que más le sirva según la semántica de las mismas.

Por otra parte, se quería responder a la cuestión de si la flexibilidad de una restricción deseable mejora de alguna manera los tiempos del solver. Esto se refiere a permitir cierto intervalo de diferencia horaria entre los turnos contiguos. Como era esperable dados los coeficientes de la función objetivo, las soluciones priorizan minimizar la cantidad de empleados contratados, y en segundo lugar se optimiza el cumplimiento de las restricciones deseables. La mínima cantidad de empleados necesaria está determinada por la tasa de atención y por la duración de la jornada laboral, mientras que las restricciones relativas a la estabilidad en el inicio de turnos están supeditadas a la cantidad mínima de empleados. En términos operativos, si RD_1 y RD_2 presentan tiempos de resolución similares debido a su tamaño, resulta preferible optar por RD_1, ya que impone una mayor rigidez en los horarios de inicio, contribuyendo más eficazmente al objetivo de estabilidad.

En resumen, las RDs que incrementan significativamente el tamaño del modelo, como RD_3, son las que exhiben el peor rendimiento computacional. La flexibilidad conceptual de una restricción deseable tiene un efecto secundario frente al peso que ejerce el tamaño del modelo en la resolubilidad del problema. Sin embargo, a fines prácticos, los resultados son correctos y, aunque en muchos casos subóptimos, cumplen con los requerimientos prácticos de la empresa.

5. CONCLUSIONES Y TRABAJO FUTURO

Esta Tesis de Licenciatura tuvo como objetivo formular y evaluar varios modelos (modelo general + RD_k, con $1 \leq k \leq 5$) para el problema de asignación de turnos en una estación de servicio, incorporando condiciones de estabilidad en el inicio de los turnos. A partir de la experimentación computacional realizada, se obtuvieron varias conclusiones importantes.

Primero, se confirmó que la incorporación de restricciones deseables (RDs) para promover la estabilidad en los turnos incrementa significativamente la complejidad computacional del problema de asignación, en comparación con el modelo general (RD_0) que solo busca minimizar la cantidad de empleados. Este aumento se manifiesta en mayores tiempos de resolución y, frecuentemente, en que el solver SCIP alcanza el límite de tiempo establecido, resultando en gaps de optimalidad mayores a `TOLERANCE`.

Esto subraya un trade-off fundamental: si bien incorporar la estabilidad horaria es un objetivo deseable para la empresa y el personal, lograr una mayor estabilidad puede requerir un esfuerzo computacional considerable. Este esfuerzo podría impedir que el solver encuentre la solución óptima global en un tiempo razonable, llevando a soluciones subóptimas.

La experimentación sugiere que hay una relación entre el tamaño de la formulación del modelo (cantidad de variables y restricciones) y la performance del solver. Aunque la estructura del modelo puede tener un impacto mayor que su tamaño, en general las RDs que introducen un mayor número de variables y restricciones fueron las que presentaron el peor rendimiento. Específicamente, la RD_3, que buscaba maximizar la coincidencia con el último turno trabajado (considerando días francos), generó la formulación de mayor tamaño y, por consiguiente, mostró el peor rendimiento computacional, alcanzando el límite de tiempo en todas las instancias y reportando los gaps de optimalidad más altos. Los resultados sugieren que la mayoría de las instancias con más de 3800 variables o restricciones no lograron alcanzar la tolerancia de optimalidad dentro del límite de tiempo.

En comparación, la flexibilidad semántica de una restricción deseable (es decir, cuánto “margen"permite para ser cumplida) parece tener un impacto secundario en la computabilidad si se compara con el peso del tamaño de la formulación de cada modelo. Por ejemplo, aunque la RD_2 (que permite inicios de turno con una hora de diferencia) es semánticamente más flexible que la RD_1 (que exige el mismo horario exacto), ambas presentaron gaps y tiempos de resolución promedio similares. Esto refuerza la conclusión de que el factor dominante en la dificultad empírica de resolución es la cantidad de elementos agregados al modelo de optimización, y no las diferencias en la flexibilidad conceptual de la restricción.

Si bien las soluciones encontradas para las RDs suelen ser subóptimas en términos del gap de optimalidad, en la práctica, mayoritariamente cumplen la condición de estabilidad impuesta para una o dos semanas. La cantidad de empleados contratados, por otro lado, parece depender más fuertemente de la tasa de atención de clientes y del tamaño del modelo, que de la flexibilidad de la restricción deseable.

Luego de toda la experimentación y los resultados a la vista, solo queda decidir cuál opción de restricción deseable es la mejor para la empresa en cuanto a mantener la estabilidad en el inicio de turnos de sus empleados. Como se dijo anteriormente, depende de los objetivos y la forma de trabajo de la Estación de Servicio, pero hay algunas cuestiones que ya desde los resultados se pueden vislumbrar. La primera es que RD_2, la prueba sobre flexibilidad, no tuvo mejor performance que su versión inflexible. De todas formas, ninguna de las dos puede mantener la estabilidad luego de un franco, por lo que para instancias de más semanas, sin dicha estabilidad, los horarios de inicio tendrán demasiadas variaciones (recordando que RD_3 es la más costosa en general, y empíricamente la que peor escala). Por otro lado, RD_4 de forma elástica fija el mismo horario para los días que trabaje un empledo contratado. Si bien es más rápida y

es robusta a los francos, podría ser poco predecible por su elasticidad. En último lugar, se alegó anteriormente que la semántica de RD_5 es más atinada en términos reales pero tiene la desventaja de que si no existen todavía horarios fijos, hay que encontrar buenos horarios de alguna manera externa.

Es por eso que se proponen dos opciones preferidas. La primera es elegir el modelo de RD_3 con un alto costo computacional (y por momentos colgando por completo al solver), que a la larga podría asegurar estabilidad. Esta opción tiene otra desventaja analizada en toda la experimentación que es que no escala bien y con corridas del mismo tiempo que las demás restricciones, no llega a asignaciones semánticamente precisas en términos prácticos.

La segunda es algo más alentadora, y puede que sea más útil a fines prácticos. La idea es utilizar las RD_4 y RD_5, primero correr algunas instancias de las totales (no hace falta todas) con RD_4, y con eso fijar los horarios del vector z para correr el modelo RD_5. De esta manera, la empresa puede guiarse de qué horarios fijos poner (sugerencia) o usar los propios, y finalmente correr RD_5 con ellos. Por lo tanto, si es necesario tomar una decisión, la segunda propuesta de combinar $RD_4 + RD_5$ resulta superadora por sobre las demás.

Como trabajo futuro, se podría continuar investigando la forma de disminuir los tiempos de cómputo en el caso de querer estabilidad estricta, pero sobre todo en instancias de más semanas, que experimentalmente no lograban llegar a un gap de optimalidad razonable. Existen técnicas de división de problemas PLE que prometen mejorar estos tiempos de cómputo [27]. Estas se asemejan a la técnica de programación *dividir & conquistar*, cuya complejidad depende del tamaño del modelo. Por otro lado, también quedaría entender mejor el comportamiento de la relación - tamaño del modelo vs gap de optimalidad - tomando más datos y analizando comportamientos de la demanda en otros meses del año. Por último se podrían realizar experimentos con datos reales y más semanas, cambiando el `timelimit` para llegar a asignaciones razonables. En resumen, el trabajo futuro se centra particularmente en cómo el modelo puede escalar para adaptarse mejor al mundo real.

ANEXOS

Anexo A: Código fuente

En esta sección se detalla el proceso de implementación del modelo planteado en el Capítulo 3, partiendo de los datos descriptos en el Capítulo 2. Se explicará paso a paso cómo se construyó el código en `Python` utilizando el solver `SCIP` a través de su wrapper `PySCIPOpt` [5]. Las razones por las que se eligió `Python` es por su legibilidad y su traduccion inmediata de modelo a código con su IDE en colaboración con `SCIP`; definir cada elemento del modelo se vuelve muy práctico y prolijo. Si bien `Python` es un lenguaje de tipado dinámico y lento en comparación con `C++`, por decir alguno, no tiene particular impacto para este uso. Por otro lado, `SCIP` es gratuito y de código abierto, por lo que cuenta con muchísima documentación que fue muy útil para la construcción del solver (característica que comparte con `Python`).

Para manipular las instancias de entrada se define la clase `InstanciaEstacionServicio`. Esta clase tiene como objetivo encapsular la lectura y preparación de los datos desde archivos Excel generados sintéticamente. A partir de esta clase, se obtiene la cantidad de horas, los días laborales y la serie temporal de llamados por hora.

```
class InstanciaEstacionServicio():
    def __init__(self):
        self.cantidad_max_empleados = 20
        self.cantidad_horas = None
        self.dias_laborales = None
        self.llamados_por_hora = []

    def leer_datos(self, nombre_archivo, numero_instancia):
        datos = pd.read_excel(nombre_archivo)
        self.cantidad_horas = datos.shape[0]
        self.dias_laborales = int((self.cantidad_horas/24) * 5/7)
        self.llamados_por_hora = datos.iloc[:, numero_instancia].tolist()
```

El main del código recibe como parámetros `numero_instancia`, que va del 1 al 25; `RD_elegida`, que o bien vale 0 (modelo general, no elegir ninguna) o bien del 1 al 5 (alguna de las RD que se tienen); y `tasa_atencion_clientes`, que se elige entre las que se setearon en el Capítulo 2 (37.02, 30, 25, 20).

Una vez definida la clase `Instancia` y la clase `Model` que se importa de la librería `PySCIPOpt`, se puede comenzar a transcribir el modelo. El programa está compuesto por las siguientes funciones:

```
    def main(numero_instancia, RD_elegida, tasa_atencion_clientes):
        # Lectura de datos desde el archivo de entrada
        instancia = cargar_instancia(numero_instancia)

        # Definicion del problema
        prob = Model()
```

```
# Definicion del modelo
x_dict, e_dict = agregar_elementos(prob, instancia, RD_elegida,
                                   tasa_atencion_clientes)

# Resolucion del modelo
resolver_lp(prob)

# Mostrar los resultados
mostrar_resultados(prob, instancia, x_dict, e_dict)
```

El modelo matemático se implementa en la función `agregar_elementos_modif`, que recibe como argumentos el modelo `SCIP`, una instancia del problema, la restricción deseable activada (`RD_elegida`), y la tasa de atención de clientes. Esta función devuelve dos diccionarios x_{dict} y e_{dict} que contienen las variables correspondientes a las x_{ih} y e_i respectivamente. Durante la función, el PLE se sobreescribe dentro de prob: se crean las variables de decisión, se agregan las restricciones duras del modelo, se implementan las restricciones deseables de forma condicional según la elección de `RD_elegida`.

El código permite activar dinámicamente una de las restricciones deseables. Esta elección se realiza a través del argumento `RD_elegida` (entero del 1 al 5). Cada restricción deseable se modela con variables auxiliares y restricciones adicionales que se incorporan al modelo solo si la penalización asociada (delta) es no nula. El valor de cada δ_k se calcula como:

$$\delta_k = \frac{1}{\text{máx}(\texttt{DOM}(\texttt{RD}_k)) + 1}.$$

Recordar que esto garantiza que la penalización de las ***RD*** nunca sea más importante que el costo de contratar un empleado adicional (1 es el mínimo valor que puede tomar la función objetivo original). La función define internamente un vector `deltas`, del que sólo uno de los elementos puede ser distinto de cero. Este mecanismo simula "prender y apagar" restricciones deseables, manteniendo una única activada por corrida.

La función objetivo principal es minimizar la cantidad total de empleados contratados. A esta se le resta (o suma, en el caso de penalización positiva) un término asociado a la restricción deseable seleccionada:

$$\text{mín} \sum_i e_i - \delta_1 \cdot \texttt{obj}_a - \delta_2 \cdot \texttt{obj}_c - \delta_3 \cdot \texttt{obj}_s - \delta_4 \cdot \texttt{obj}_w + \delta_5 \cdot \texttt{obj}_y.$$

Una vez definido el modelo, se resuelve llamando a la función `resolver_lp`, que establece un límite de tiempo llamado de resolución y una tolerancia de optimalidad (`gap`). Esto tiene sentido en parte porque se tienen que correr 500 instancias y algunas de ellas tardarían muchas horas sin un límite razonable para poder correrlas todas. Además, se fija un gap entre la cota primal y la dual debido principalmente a asumntos numéricos y de redondeo, por lo que el gap nunca "llega a cero"; entonces desde cierto *threshold* se decide qué es suficientemente cerca.

```
def resolver_lp(prob):
    # Definir los parametros del solver
    prob.setParam("limits/gap", TOLERANCE) #10e-6
    prob.setParam("limits/time", 1800) #Media hora por instancia
    # Resolver el lp
```

```
prob.optimize()
```

Posteriormente, la función `mostrar_resultados` extrae los valores óptimos de las variables $x_{i,h}$ y e_i, identifica qué empleados fueron contratados y sus horarios de inicio de turno, y guarda esta información en archivos de texto.

El código principal se encuentra dentro de la función `main(numero_instancia, RD_elegida, tasa_atencion_clientes)`, que define una corrida completa del modelo con una configuración dada de parámetros.

Anexo B: Experimentación en instancias de varias semanas

Durante todo el trabajo de tesis, se utilizaron datos correspondientes con una sola semana. Para poder explorar instancias de más de una semana, se decidió realizar pruebas con algunas instancias del modelo general (una instancia de 2 semanas y una de 3), y probar si era factible correr dichas instancias con los recursos computacionales disponibles.

Las instancias siguen la misma distribución original, solo que duran el doble o el triple de tiempo respectivamente. El modelo es procesado por el solver sin ninguna modificación, ya que el mismo fue escrito de manera genérica según la cantidad de semanas de las que se trate. Por el comportamiento observado para una sola semana, se decidió que alcanzaba analizar la performance para una instancia de RD_0, una de RD_1 y una de RD_3 para ver si con los recursos disponibles se podría realizar una comparativa entre las restricciones. El modelo general siempre corre en pocos segundos, la restricción elástica 1 se pasa del límite de media hora pero se puede visualizar que con un poco más de tiempo llegaría al gap de optimalidad requerido, y luego la RD_3, que es la de peor performance.

A continuación se pueden visualizar los resultados:

#Semanas	RD	#Emp	#Variables	#Restricciones	Solving Nodes	Solving Time	Gap
2	0	8	6420	2487	26	9.0	0.000
	1	8	12500	13105	9	1800.00	0.133
	3	?	35240	48697	1	1800.00	inf
3	0	8	9780	3795	27	19.00	0.000
	1	?	19220	20583	1	1800.00	inf
	3	?	59600	81645	1	1800.00	inf

Tab. 5.1: Resultados de instancias de 2 y 3 semanas en los modelos general, RD_1 y RD_3.

Lo primero que se puede visualizar es que solo tres de seis experimentos logran un gap de optimalidad suficientemente cercano para dar una asignación de turnos. El solver tuvo como tiempo máximo 1800, lo mismo que para una semana. Este experimento enfatiza aún más en el hecho de que el tamaño del modelo (en las cuales la cantidad de variables y restricciones crece con las semanas) influye significativamente en los tiempos de cómputo.

Se podría pensar que por estos resultados, el modelo no escala bien. En caso de ser necesario resolver instancias de más de una semana, sería importante recurrir a técnicas computacionales para acelerar la resolución o bien para generar soluciones en forma heurística.

En el caso de RD_1 para dos semanas, la asignación de los turnos viola en varias oportunidades esta restricción elástica. Las restricciones de dos francos por semana y las condiciones estáticas sobre los turnos parecen cumplirse. Además, podemos observar algunas rachas dentro de la tabla en días laborales

contiguos. Si se deja correr la instancia más tiempo, probablemente la cantidad de veces en las que no se cumple la estabilidad de los turnos va a ser menor.

Empleado	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
11	32	56	92	116	140	223	241	265	309	328
12	16	84	108	132	162	186	210	234	280	304
13	0	69	93	124	154	227	251	268	292	320
14	40	64	89	113	137	178	202	229	253	295
16	21	45	53	77	101	175	199	260	284	325
17	13	37	102	126	146	170	194	218	247	302
18	8	79	109	133	157	181	205	252	276	300
19	5	29	61	85	150	174	225	249	273	317

Tab. 5.2: Asignación de turnos para una instancia de dos semanas en RD_1

Empleado	Lun	Mar	Mié	Jue	Vie	Sáb	Dom	Lun	Mar	Mié	Jue	Vie	Sáb	Dom
11	F	8	8	20	20	20	F	F	F	7	1	1	21	16
12	16	F	F	12	12	12	18	18	18	18	F	16	16	F
13	0	F	F	21	21	F	4	10	11	11	4	4	F	8
14	F	16	16	17	17	17	F	10	10	13	13	F	7	F
16	21	21	5	5	5	F	F	7	7	F	20	20	F	13
17	13	13	F	F	6	6	2	2	2	2	7	F	14	F
18	8	F	F	7	13	13	13	13	13	F	12	12	12	F
19	5	5	21	21	F	F	6	6	F	9	9	9	F	5

Tab. 5.3: Asignación horaria por día para una instancia de dos semanas en RD_1

Para concluir con este experimento, se puede afirmar que el modelo es generalizable para dos semanas, pero que necesita más tiempo para dar soluciones cualitativa y cuantitativamente correctas. Si bien hay muchos casos en la asignación de la tabla 5.3 en que no mantiene el turno en días contiguos, tiene rachas del mismo horario para cada empleado, lo cual es útil e interesante para los objetivos postulados desde un principio.

Anexo C: Experimentación en rompimiento de simetría para el modelo general

Si bien los modelos con la menor cantidad de variables y restricciones no solo mejora la legibilidad e intuitivamente generaría un árbol de búsqueda más acotado, no siempre están relacionado el tamaño del modelo con el tiempo de cómputo de esta manera. Existen formulaciones que, al agregarles tipos particulares de restricciones, restringen significativamente el espacio de búsqueda. Una que se decidió explorar aquí son las de rompimiento de simetría.

En esta experimentación, se agregó la restricción (3.34) a cada versión del modelo (general y las 5 restricciones deseables junto con sus funciones objetivo). Como se fundamentó en esta tesis, las 25 instancias sintéticas son muy parecidas, y las formulaciones tuvieron efectos similares en todas ellas. Como prueba de concepto del efecto que tiene esta nueva restricción en cada modelo, se corrió la primera instancia sintética en cada uno de los seis modelos disponibles añadiendo (3.34) en el solver, con la tasa de atención original (37.02). Los resultados se ven reflejados en la Tabla 5.1

Una pequeña observación es que se tomó el inverso del vector z para la restricción 5 debido a que originalmente los empleados con menor índice recibían los comienzos de horario más requeridos, y este dato podía empeorar la factibilidad.

Se puede observar que la mayor discrepancia entre el modelo con rompimiento de simetría y sin, es que en 1800 segundos, el nuevo modelo no pudo llegar a un gap suficientemente aceptable (reportado como “inf” en las tablas). En cambio su contraparte sin esa restricción lo hace muchísimo mejor. En el resto de comparaciones entre modelos análogos, no se observan diferencias significativas.

Instancia	#Emp	#Variables	#Restricciones	Solving Nodes	Solving Time	Gap
SIM_0	8	3363	1200	99	6.16	0.000
RD_0	8	3380	7022	1	9.00	0.000
SIM_1	8	5780	6052	1695	1800.00	0.120
RD_1	8	6260	10044	3873	1800.00	0.117
SIM_2	8	5780	10790	1664	1800.00	0.000
RD_2	8	5780	14161	602	1800.00	0.119
SIM_3	8	10880	15732	-	1800.00	inf
RD_3	9	13400	22342	139	1800.00	0.333
SIM_4	8	3860	4132	1257	1800.00	0.146
RD_4	8	3860	7502	3845	1542.34	0.0704
SIM_5	8	3370	1200	768933	1800.00	0.084
RD_5	8	3380	7022	183885	1795.36	0.103

Tab. 5.4: Resultados promedio de 25 instancias correspondientes con cada RD y tasas de atención

Bibliografía

[1] John Thornton and Husain Sattar. Nurse rostering and integer programming revisited. In *International Conference on Computational Intelligence and Multimedia Applications*, pages 49–58, 1997.

[2] Tony Hürlimann. *Mathematical Modeling Basics*. Department of Informatics, University of Fribourg, 2024.

[3] Juan P. Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57(1):3–57, 2015.

[4] Octavio Eneas Biasoli and Javier Marenco. Predicción de arribos y programación de turnos en una estación de servicio. *Revista Ingeniería de Sistemas*, XXXV, December 2021.

[5] Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Christoph Graczyk, Katrin Halbig, Ivo Hedtke, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Julian Manns, Gioni Mexi, Erik Mühmer, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Dieter Weninger, and Lixing Xu. The SCIP optimization suite 9.0. Technical report, Optimization Online, February 2024.

[6] Vasek Chvátal. *Linear Programming*. W.H. Freeman and Company, New York, 1946.

[7] Andreas T. Ernst. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127(1):21–144, 2004.

[8] Eric Soubeiga. *Development and application of hyperheuristics to personnel scheduling*. PhD thesis, University of Nottingham, 2003.

[9] Eric Soubeiga. *Development and application of hyperheuristics to personnel scheduling*. PhD thesis, University of Nottingham, 2003.

[10] Osman T. Aydas. Short-term nurse schedule adjustments under dynamic patient demand. *Journal of the Operational Research Society*, 74(1):310–329, 2023.

[11] Eman Ouda, Andrei Sleptchenko, and Mecit Can Emre Simsekler. Nurse rostering via mixed-integer programming. In *Industrial Engineering and Applications*, pages 815–823. IOS Press, 2023.

[12] Victoria Bihler. Improving scheduling efficiency in the setting of staff turnover and increased surgical volume in the post-anesthesia care unit using census data trends. *Nurse Leader*, 22(5):632–636, 2024.

[13] Mohammed Chowdhury, Arfat Raihan, Md Al Amin, and Md Rakibul Hasan. Optimization of patient and nurse management in health care: A case study. *Journal of Engineering Science*, 15(1):83–94, 2024.

[14] Janaina Marchesi, Silvio Hamacher, and Julia Fleck. A stochastic programming approach to the physician staffing and scheduling problem. *Computers & Industrial Engineering*, 142:106281, 2020.

[15] Nagaraj V. Dharwadkar. Enhanced parallel-particle swarm optimization (ep-pso) approach for solving nurse rostering problem: Enhanced parallel-particle swarm optimization (ep-pso) algorithm. *International Journal of Swarm Intelligence Research (IJSIR)*, 13(1):1–17, 2022.

[16] Yijing Chen. A comparison of approaches to the nurse scheduling problem. *Operations Research II Group 4*, pages 1–10, 2016.

[17] Edmund Burke, Jingpeng Li, and Rong Qu. A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. *European Journal of Operational Research*, 203(2):484–493, 2010.

[18] Zhipeng Lü and Jin-Kao Hao. Adaptive neighborhood search for nurse rostering. *European Journal of Operational Research*, 218(3):865–876, 2012.

[19] Albert H. Schrotenboer, Rob Wenneker, Evrim Ursavas, and Stuart X. Zhu. *Models and algorithms for airline crew rostering*. Department of Operations, Faculty of Economics and Business, University of Groningen, 2016.

[20] A. Nearchou, I. C. Giannikos, and A. G. Lagodimos. Multisite and multishift personnel planning with set-up costs. *IMA Journal of Management Mathematics*, 31(1):5–31, 2020.

[21] Jonathan Bard, Canan Binici, and Anura H. Desilva. Staff scheduling at the united states postal service. *Computers & Operations Research*, 30(5):745–771, 2003.

[22] Andreas Nearchou, I. C. Giannikos, and A. G. Lagodimos. A genetic algorithm for the economic manpower shift planning problem. *Cybernetics and Systems*, 45(5):439–464, 2014.

[23] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009.

[24] Karen I. Aardal and C. P. M. Van Hoesel. Polyhedral techniques in combinatorial optimization. *Memorandum COSOR*, 9515, 1995.

[25] P. C. Gilmore and Ralph E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[26] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review. *Computers & Operations Research*, 31(3), 2004.

[27] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming Series B*, 98:23–47, 2003.