Trabajo Práctico PLE - Asignación de cuadrillas

Fernández, M. (LU: 416/22); Ruz Veloso, L. (LU: 589/22)
Introducción a la Investigación Operativa y Optimización
Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires
(Dated: Fecha de presentación: 08/07/2024)

En el presente trabajo práctico se aborda un problema de asignación óptima de cuadrillas de trabajadores a órdenes de servicio. Para su solución se propone un modelo matemático de Programación Lineal Entera (PLE) implementado para ser usado en CPLEX, buscando maximizar las ganancias de la empresa al asignar los trabajadores. Más allá de que el modelo funcione correctamente, el objetivo de este trabajo fue poder testear diferentes parámetros de CPLEX y ver cómo estos podían afectar a la eficiencia del modelo, además de implementar restricciones deseables y medir el impacto que tenían estas en la función objetivo. Los resultados muestran que el ajuste de parámetros varía dependiendo del tipo de instancia y que las restricciones deseables pueden ser implementadas de distintas formas dependiendo de cómo se quiere medir la violación de una de estas restricciones. En conclusión, CPLEX parece adaptarse bastante bien a distintos contextos y necesidades operativas, pero hace falta un conocimiento detallado del problema para elegir bien el camino a tomar.

I. INTRODUCCIÓN

El presente trabajo tiene como objetivo desarrollar un modelo de programación lineal entera para abordar un problema específico de asignación de tareas. La programación lineal entera (PLE) es una técnica de optimización matemática en la que se busca maximizar o minimizar una función objetivo sujeta a un conjunto de restricciones, donde las variables de decisión son enteras. Esta técnica es ampliamente utilizada en diversos campos como la logística, la planificación de recursos y la gestión de proyectos. Para la implementación de nuestro modelo, utilizamos CPLEX, un software comercial avanzado que permite resolver problemas de PLE de gran tamaño y complejidad mediante algoritmos de optimización robustos y eficientes.

El problema a tratar en este trabajo es la asignación de cuadrillas de trabajo a órdenes de servicio, una tarea común en muchas empresas que requieren la ejecución de múltiples actividades en distintas ubicaciones. La correcta asignación de tareas es fundamental para maximizar la eficiencia operativa y minimizar costos, y es un tema ampliamente discutido y estudiado en ámbitos de investigación operativa y gestión empresarial. A modo general, las técnicas de asignación de tareas buscan optimizar el uso de los recursos disponibles, considerando restricciones como la disponibilidad de trabajadores, las habilidades requeridas para cada tarea y las limitaciones de tiempo.

En este contexto, nuestro modelo se centra en asignar cuadrillas de trabajo a órdenes de servicio de manera óptima. Este problema es relevante debido a la necesidad de maximizar la ganancia total mientras se cumplen todas las restricciones operativas y se mejoran ciertos aspectos deseables de la asignación. Para abordar este problema, diseñamos un modelo matemático detallado, el cual fue testeado y ajustado mediante diferentes parámetros de CPLEX. Además, exploramos distintos parámetros para evaluar sus tiempos de ejecución y su eficacia en la resolución del problema.

Una parte relevante del desarrollo de este trabajo fue la incorporación de restricciones deseables, que no son obligatorias pero buscan mejorar la calidad de la asignación, como evitar que dos trabajadores que tienen conflicto entre ellos sean asignados a una misma tarea o prevenir, dentro de lo posible, que tareas repetitivas sean asignadas a un mismo trabajador. Estas restricciones fueron implementadas de dos maneras distintas y ajustadas para observar su impacto en la función objetivo y analizar cómo influían en la solución final.

En resumen, este trabajo práctico presenta un modelo para la resolución de un problema de asignación de tareas utilizando programación lineal entera y CPLEX, experimentando con distintos parámetros y con restricciones deseables, tratando de generar una discusión tanto de los aspectos cuantitativos como cualitativos del problema.

II. MODELO

A. Sin restricciones deseables

Para la construcción del modelo nos basamos en lo solicitado en el documento que plantea este problema. En general, tomamos como importante poder diferenciar entre ordenes, días, turnos y trabajadores. Para ello creamos distintas variables que buscan distinguir entre estos cuatro aspectos, las cuales son las siguientes:

$$A_{ijhk} = \begin{cases} 1 & \text{si la orden } i \text{ se realiza el día } j \\ & \quad \text{en el turno } h \text{ por el trabajador } k \\ 0 & \text{c.c.} \end{cases} \quad (1)$$

Esta variable (1) fue importante para ver qué pasaba al fijar cada uno de los índices y para construir coherencia entre distintas variables. Las siguientes (desde la 2 hasta la 6) fueron útiles para poder obtener información acerca de algún momento puntual en el que sólo se quisiera saber algo específico de la situación. Por ejemplo, de (5)

se puede saber de manera más sencilla si una orden se realizó o no, o (6) es útil para saber si un trabajador fue asignado a alguna orden en algún día j. Además, esto produce una ventaja a la hora de escribir las restricciones, permitiendo que sean más fáciles de implementar y de leer.

$$x_{ijh} = \begin{cases} 1 & \text{si la orden } i \text{ se realiza el día } j \\ & \quad \text{en el turno } h \\ 0 & \text{c.c.} \end{cases} \tag{2}$$

$$w_{jhk} = \begin{cases} 1 & \text{si en el día } j \text{ y turno } h \\ & \quad \text{está asignado el trabajador } k \\ 0 & \text{c.c.} \end{cases} \tag{3}$$

$$y_{ik} = \begin{cases} 1 & \text{si a la orden } i \text{ se le asigna el trabajador } k \\ 0 & \text{c.c.} \end{cases} \tag{4}$$

$$z_i = \begin{cases} 1 & \text{si la orden } i \text{ se realiza} \\ 0 & \text{c.c.} \end{cases} \tag{5}$$

$$d_{jk} = \begin{cases} 1 & \text{si el trabajador } k \text{ trabaja el día } j \\ 0 & \text{c.c.} \end{cases} \tag{6}$$

Finalmente, las siguientes tres tipos de variables sirvieron para implementar el sistema de pagos solicitado, donde a partir de cierta cantidad de órdenes cumplidas el pago de las mismas iba creciendo.

$$P_k\text{: pago total recibido por el trabajador } k \tag{7}$$

$$p_{k(num)}\text{: tareas realizadas por k en el nivel } num \tag{8}$$

$$f_{k(num)} = \begin{cases} 1 & \text{si el trabjador } k \text{ completó el nivel } num \\ 0 & \text{c.c.} \end{cases} \tag{9}$$

En base a estas variables construimos la siguiente función objetivo:

$$\text{Maximizar} \quad \sum_i z_i b_i - \sum_k p_k \tag{10}$$

Donde b_i es el beneficio que se obtiene de la tarea i y a la suma de todo eso se le resta el pago total de cada trabajador.

A modo de que se entienda mejor el por qué de nuestras restricciones, explicaremos cada una de ellas y diremos por qué cumplen lo que cumplen. De ahora en adelante los valores de los índices serán los siguientes:

$$\begin{aligned} i &= 0, \ldots, N-1 \quad \text{para } N \text{ órdenes} \\ j &= 0, \ldots, 5 \quad \text{para 6 días} \\ h &= 0, \ldots, 4 \quad \text{para 5 turnos} \\ k &= 0, \ldots, T-1 \quad \text{para } T \text{ cantidad de trabajadores} \\ num &= 0, \ldots, 3 \quad \text{para 4 níveles del sistema de pagos} \end{aligned}$$

Además, tenemos los siguientes vectores:

$$\begin{aligned} OC &:= \text{ pares de índices de órdenes conflictivas} \\ OC' &:= \text{pares de índices de órdenes correlativas} \end{aligned}$$

Como primer grupo de restricciones, tenemos aquellas que sirven para que las variables tengan coherencia entre sí.

$$3A_{ijhk} \leq x_{ijh} + w_{jhk} + y_{ik} \quad \forall i, j, h, k \tag{1}$$

De (1) se tiene que el A_{ijhk} sólo puede dar 1 si las otras tres que están relacionadas a esa tienen valor 1. Si es cero no restringe nada.

$$x_{ijh} + w_{jhk} + y_{ik} \leq 2 + A_{ijhk} \quad \forall i, j, h, k \tag{2}$$

Esta restricción (2) sirve para que si A_{ijhk} da 0, las otras tres no puedan tomar valor 1 al mismo tiempo. Si da 1 no restringe nada.

$$z_i = \sum_j \sum_h x_{ijh} \quad \forall i \tag{3}$$

Aquí (3) se busca que los z_i sean coherentes con los x_{ijh}, es decir, que si una tarea se hizo (o no) entonces esa tarea sólo pudo ser asignada a un turno (o ninguno si $z_i = 0$)

$$d_{jk} \leq \sum_h w_{jhk} \quad \forall j, k \tag{4}$$

$$5 \cdot d_{jk} \geq \sum_h w_{jhk} \quad \forall j, k \tag{5}$$

De (4) y (5) se consigue que, por un lado, si un trabajador realiza una orden en algún día j, entonces la sumatoria de todas las tareas que hizo ese día sea al menos una, mientras que por otro lado, esa sumatoria de 0 si no trabaja ese día.

$$x_{ijh} + y_{ik} \leq w_{jhk} + 1 \quad \forall i, j, h, k \tag{6}$$

$$w_{jhk} \leq \sum_i A_{ijhk} \quad \forall j, h, k \tag{7}$$

La restricción (6) busca coherencia entre las variables w, x e y. De tal manera que si, por ejemplo, se tiene que una orden se hace en un turno y tiene trabajadores asociados a ella, entonces la variable que asocia ese turno con trabajadores también debe estar en 1. Además, en la (7) se busca que si un trabajador está asociado a un turno en algún día, entonces debe haber una tarea asociada a ese trabajador y ese momento.

Una vez asociadas las variables entre sí, las siguientes restricciones son en base a lo que se propone desde la consigna del trabajo práctico

$$t_i \cdot z_i = \sum_k y_{ik} \quad \forall i \tag{8}$$

Donde t_i es la cantidad de trabajadores que requiere la tarea i. Entonces, lo que hace esta restricción (8) es pedir que si una tarea se realiza, la cantidad de trabajadores asignados sean justamente los que se necesitan.

$$\sum_j \sum_h x_{ijh} \leq 1 \quad \forall i \tag{9}$$

Luego, la restricción 9 sirve para que cada orden esté asignada a lo sumo a un turno de cualquier día.

$$\sum_i A_{ijhk} \leq 1 \quad \forall j, h, k \tag{10}$$

Aquí (10) se busca que dado un trabajador este no puede estar asociado a más de una orden al mismo tiempo.

$$\sum_j d_{jk} \leq 5 \quad \forall k \tag{11}$$

Esta restricción (11) busca que los trabajadores tengan por lo menos un día de descanso.

$$t_i \cdot x_{ijh} = \sum_k A_{ijhk} \quad \forall i, j, h \tag{12}$$

Luego, en (12) se busca que para cada orden, si esta se hace en algún momento dado, entonces los trabajadores asignados deben estar asociados a ese mismo momento y orden.

$$\sum_h w_{jhk} \leq 4 \quad \forall j, k \tag{13}$$

En esta restricción (13) se busca que nadie trabaje los 5 turnos de un mismo día

$$A_{i'jhk} + A_{ij(h+1)k} \leq 1 \quad \forall j, k \ \forall (i, i') \in OC, \ h = 0, \ldots, 3 \tag{14}$$

$$A_{ijhk} + A_{i'j(h+1)k} \leq 1 \quad \forall j, k \ \forall (i, i') \in OC, \ h = 0, \ldots, 3 \tag{15}$$

Tanto en la restricción 14 como en la 15 se busca que si un trabajador hace una de las tareas conflictivas, entonces en el turno siguiente del mismo día, el mismo trabajador no puede ser asignado a la otra tarea. Es por esto que la suma de ambas puede dar a lo sumo 1.

$$x_{ijh} \leq x_{i'j(h+1)} \quad \forall j \ \forall (i, i') \in OC' \ h = 0, \ldots, 3 \tag{16}$$

$$\sum_j x_{ij4} = 0 \quad \forall i \tag{17}$$

Luego se busca cumplir lo de las órdenes correlativas. De esta manera la restricción 16 se ocupa de que si el valor de hacer la orden i en algún momento es 1, entonces en el turno siguiente del mismo día tiene que hacerse la orden i'. La restricción 17 se ocupa de que la orden i no se haga en el último turno de ningún día, ya que esto impediría que se haga i' y además crearía lío con los índices h.

$$\sum_i y_{ik} - \sum_i y_{ik'} \leq 8 \quad \forall i, k, k' \text{ con } k \neq k' \tag{18}$$

Posteriormente, esta restricción (18) se preocupa de que para cada par de trabajadores, al comparar cuántas tareas hizo cada uno la diferencia no puede ser mayor a 8.

Lo siguiente a tener en cuenta era el sistema de pagos. Dependiendo de cuántas órdenes haya realizado cada trabajador, este podía recibir una mayor compensación por cada tarea completada. Había cuatro niveles: el primero iba de 0 a 5 órdenes, el segundo de 6 a 10, el tercero de 11 a 15 y, finalmente, el cuarto consideraba más de 15 órdenes.

$$\sum_{num} p_{k(num)} = \sum_i y_{ik} \quad \forall k \tag{19}$$

Las variables $p_{k(num)}$ son las que señalan, por cada trabajador, cuántas órdenes hizo de cada nivel. Entonces, lo que busca esta restricción (19) es que eso sea coherente con lo que dicen las variables y_{ik} acerca de ese trabajador.

$$P_k = 1000 \cdot p_{k0} + 1200 \cdot p_{k1} + 1400 \cdot p_{k2} + 1500 \cdot p_{k3} \quad \forall k \tag{20}$$

Después, las variables P_k son enteros que tienen el pago total que recibe cada trabajador sumando todas las tareas de todos los niveles que realizó, tal como se observa en la restricción 20.

$$\begin{aligned} 5 \cdot f_{k0} &\leq p_{k0} && \forall k, \\ p_{k0} &\leq 5 && \forall k, \\ 5 \cdot f_{k1} &\leq p_{k1} && \forall k, \\ p_{k1} &\leq 5 \cdot f_{k0} && \forall k, \\ 5 \cdot f_{k2} &\leq p_{k2} && \forall k, \\ p_{k2} &\leq 5 \cdot f_{k1} && \forall k, \\ 0 &\leq p_{k3} && \forall k, \\ p_{k3} &\leq 15 \cdot f_{k2} && \forall k. \end{aligned} \tag{21}$$

Para que el esquema de pagos funcione bien, fue necesario agregar estas restricciones (21) que permiten que un trabajador realice tareas de algún nivel sólo si es que completó todas las del nivel anterior.

$$\begin{aligned} 0 \leq P_k \leq 40500 \quad \forall k \\ 0 \leq p_{k0} \leq 5 \\ 0 \leq p_{k1} \leq 5 \\ 0 \leq p_{k2} \leq 5 \\ 0 \leq p_{k3} \leq 15 \end{aligned} \tag{22}$$

Finalmente, las variables que no son binarias tienen esta restricción (22) en cuanto a sus valores. La cota superior de P_k se debe a que nunca un trabajador podrá superar esa remuneración ya que aunque trabajara en todos los turnos no superaría ese valor. Las otras son las cotas que tiene cada nivel de remuneración, son 5 para cada uno de los primeros y 15 para el último ya que si un trabajador llega a hacer 15 tareas, por la manera en la que construimos el modelo, este no tiene cómo hacer 15 tareas más.

B. Con restricciones deseables

En la siguiente sección proponemos dos formas de abordar las restricciones deseables. Antes que nada, presentamos las variables que expresan estas caracteristicas deseables.

$$C_{ikk'} = \begin{cases} 1 & \text{si } k \text{ y } k' \text{ tienen conflicto y} \\ & \quad \text{son asignados a la orden } i \\ 0 & \text{c.c.} \end{cases} \tag{23}$$

$$R_{ii'k} = \begin{cases} 1 & \text{si la orden } i, i' \text{ son realizadas por} \\ & \quad \text{el mismo trabajador } k \\ 0 & \text{c.c.} \end{cases} \tag{24}$$

Además, se tienen dos vectores nuevos que nos dan información:

$CT :=$ pares de índices de conflictos entre trabajadores
$OR :=$ pares de índices de órdenes repetitivas

Para que estas variables expresen lo que deben agregamos las siguientes restricciones:

$$y_{ik} + y_{ik'} \leq C_{ikk'} \quad \forall i \ \forall (k, k') \in CT \tag{25}$$

$$y_{ik} + y_{i'k} \leq R_{ii'k} \quad \forall k \ \forall (i, i') \in OR \tag{26}$$

Dadas estas variables, proponemos dos formas de ajustar el modelo para que las contemple. En primer lugar, le asignamos un costo (valor negativo) en la función objetivo a cada una de estas variables. Esto implica que para maximizar el modelo se favorece reducir repeticiones o conflictos en la solución. Luego, para conocer el valor real de la función objetivo basta con solo calcular el valor de la funcion objetivo previa (sin estos costos) para la solución obtenida. Así quedaría la función objetivo del modelo:

$$\begin{aligned} \text{Max} \quad & \sum_i z_i b_i - \sum_k p_k \\ & - c_1 \cdot \sum_i \sum_k \sum_{k'} C_{ikk'} - c_2 \cdot \sum_k \sum_i \sum_{i'} R_{ii'k} \end{aligned} \tag{27}$$

Como otra forma de abordar las deseables, proponemos agregar una resticcion que depende de un factor porcentual que permita cierta cantidad de repeticiones o conflictos. Veremos que este factor tiene sentido si se elije con valores entre 0 y 1, donde 1 significa que el modelo permite todas las repeticiones o conflictos posibles y 0 significa que no permite ninguno. Lo positivo de esta segunda manera es que el valor de la funcion objetivo del modelo no debe ser corregido. Mostramos las restricciones para esta segunda propuesta:

$$\frac{\sum_i \sum_k \sum_{k'} m_{k'} C_{ikk'}}{|CT| \cdot N} \leq \text{factor deseable conflicto} \tag{28}$$

$$\frac{\sum_k \sum_i \sum_{i'} R_{ii'k}}{|OR| \cdot T} \leq \text{factor deseable repetición} \tag{29}$$

III. RESULTADOS Y DISCUSIÓN

Dividimos esta sección en dos apartados. Por un lado una discusión cuantitativa acera del tiempo de ejecución del modelo ajustando diferentes parámetros, mientras por otro lado una discusión cualitativa sobre cómo implementar restricciones deseables y cómo medirlas. En ambos casos, usaremos de base 3 tests a partir de ahora llamados A, B y C. Cada uno expresa una instancia de 10 trabajadores, 25 órdenes, 5 pares de ordenes correlativas y 5 pares de órdenes conflictivas. Los tres fueron generados al azar con necesidad de trabajadores entre 1 y 8

y beneficio de entre 4000 y 15000. Si bien el tamaño de las instancias es pequeño, fue lo suficientemente complejo para que se puedan notar diferencias a la hora de correr el modelo ajustando parámetros o agregando las restricciones deseables, pero también con el tamaño adecuado para que los tiempos de ejecución no sean demasiado altos. De ahora en más siempre mostraremos valores para cada uno de los tests y el promedio.

A. Sin restricciones deseables

Para esta parte se ejecutó el modelo sin considerar las restricciones deseables y lo que se buscó fue analizar el tiempo de ejecución para diferentes parametros de CPLEX. Cubriremos NodeSelect, VariableSelect, Presolve y HeuristicEffort.

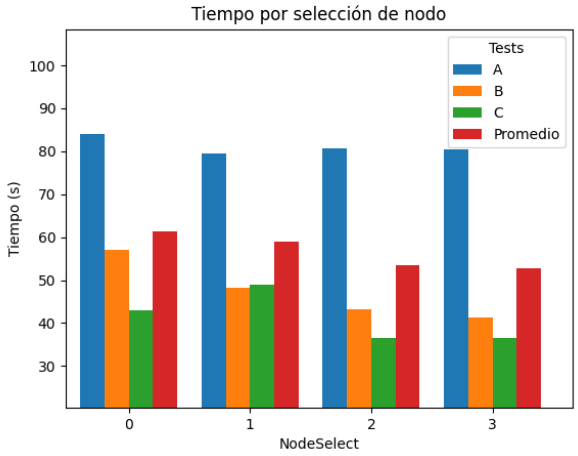

Figura 1: Gráfico del tiempo en función de ajustar los parámetros de NodeSelect para cada uno de los tests y el promedio.

Los parámetros de NodeSelect significan selección por profundidad (0), mejor cota (1), y por mejor estimación de CPLEX (2) y (3). De la figura 1 se observa que al explorar por profundidad se obtuvieron los mayores tiempos de ejecución, mientras que este fue cambiando hasta obtener mínimos con el parámetro de mejor estimación 2 y 3, siendo este último el que tiene mejores tiempos. Si bien la cantidad de tests no son los suficientes para concluir qué parámetro es mejor, si consideramos que para instancias similares a las testeadas el parámetro 2 y 3 pueden ser más útiles que el que usa CPLEX por defecto (1). También es importante aclarar que si bien se observan diferencias considerables en los tiempos de ejecución si comparamos entre distintos tests (por ejemplo los tiempos de A son muy superiores a los de C), no sabemos realmente a qué se deba esta diferencia, ya que como se dijo más arriba, ambos tienen la misma cantidad de órdenes y trabajadores.

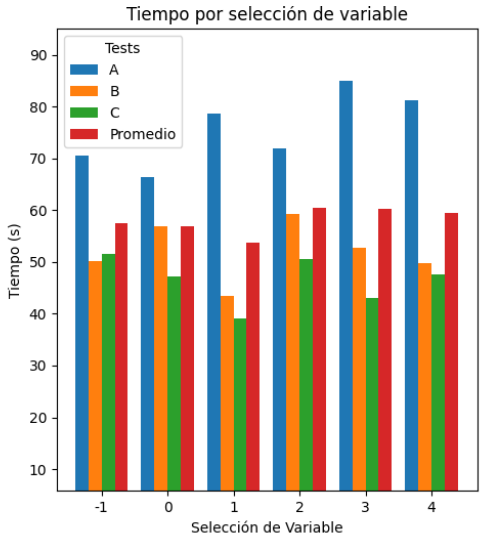

Figura 2: Gráfico del tiempo (en segundos) en función de ir modificando el cómo se van seleccionando las variables del Branch & Bound para cada uno de los tests y el promedio.

Los parámetros de VariableSelect indican: variable no entera más cercana a un entero (-1), elección de CPLEX (0), variable más lejana a un entero (1), por pseudocosto (2), strong branching (3), pseudocosto reducido (4). Si atendemos al promedio del tiempo de ejecución entre los test vemos que la diferencia entre parámetros es muy poca, no es significativa teniendo en cuenta que es el promedio entre solo 3 tests. Elegimos recomendar mantener el parámetro en su valor default (0) y permitir que se implemente la versión optimizada de elección de variables de Cplex. De todas maneras sería interesante medir este parámetro con una cantidad considerable de tests y analizar su desempeño en investigaciones futuras.

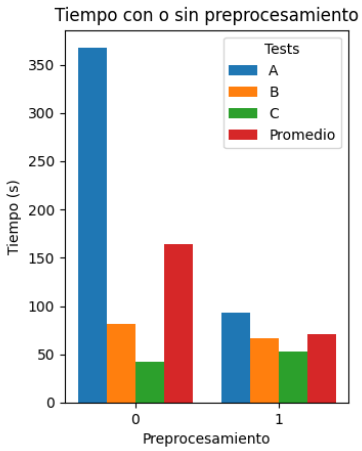

Figura 3: Gráfico del tiempo (en segundos) en función de si se desactiva o no el preprocesamiento para cada uno de los tests y el promedio.

Aquí vemos la diferencia en el tiempo de ejecución al apagar el preprocesamiento (0) comparado cuando se activa (1). Si bien para el test C parece disminuir levemente el tiempo al apagarlo, para los otros tests ocurre justamente lo contrario, observándose aún más en el A. Nuevamente, para instancias similares a las propuestas aquí, parece ser mejor mantener activado el preprocesamiento. Igualmente, hay que destacar que para instancias que se resuelven rapido como el test C, el preprocesamiento puede implicar más costo que beneficio y alargar el tiempo de ejecución.

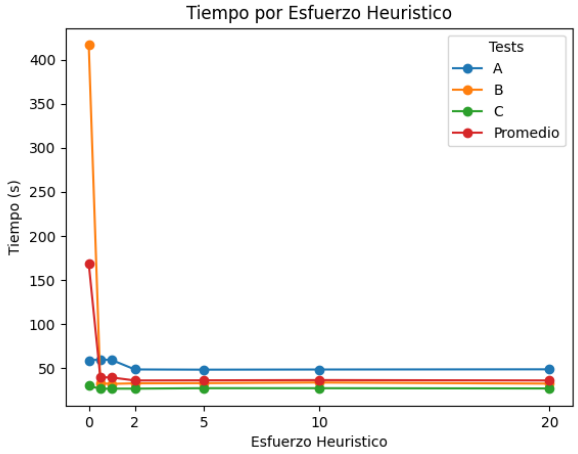

Figura 4: Gráfico del tiempo en función del esfuerzo heurístico para cada uno de los tests y el promedio.

CPLEX permite variar un parámetro de esfuerzo heurístico. Como entrada recibe un float que por default es 1. Un valor menor a uno implica menos esfuerzo en búsqueda heurística, y mayor en caso contrario. Observamos que el tiempo de ejecución decrece a la par que el esfuerzo heurístico hasta 2. Luego se estabiliza y aunque no se aprecie en el gráfico, a partir de 2, los tiempos de ejecución crecen levemente. Teniendo en cuenta esto, para instancias similares a las realizadas, un esfuerzo heurístico en 2 parece ser suficiente para optimizar los tiempos de ejecución. También, cabe señalar que si hasta el momento el test A era el que mayor tiempo tomaba, al experimentar con este parámetro fue el test C el que tomó más tiempo cuando el valor del esfuerzo era 0, mientras el A desde un principio fue el que se demoró menos.

Los siguientes graficos muestran exploraciones para observar cómo cambian los tiempos al variar la cantidad de trabajadores, cantidad de correlatividades, cantidad de órdenes conflictivas. Todo lo siguiente se hace variando un parámetro por vez y manteniendo los demás fijos.

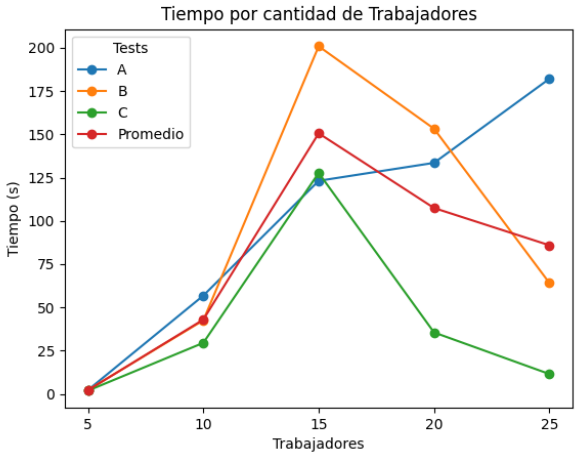

Figura 5: Gráfico del tiempo en función de ir variando la cantidad de trabajadores en cada uno de los tests y el promedio.

Es interesante ver que no hay una relación lineal entre tiempo de ejecución y trabajadores, como podría creerse de manera intuitiva. Pocos trabajadores implican menos restricciones para la instancia lo que puede reducir el tiempo de ejecución. A su vez, muchos trabajadores pueden facilitar su asignación a las tareas y menor cálculo de las funciones de costo partidas. Atribuimos a estas relaciones el hecho de que algunas de las curvas muestren un pico en el tiempo de ejecución.

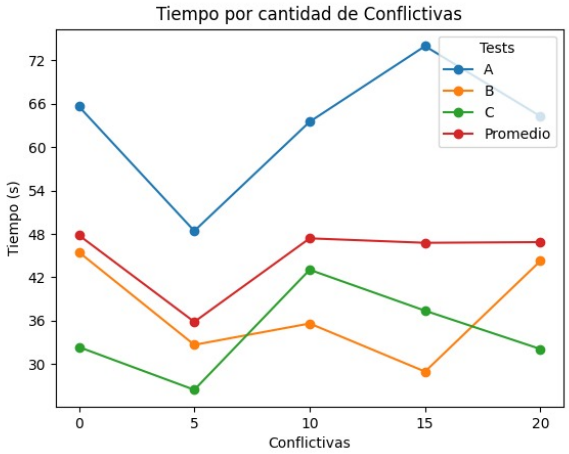

Figura 6: Gráfico del tiempo en función de ir variando la cantidad de órdenes conflictivas para cada uno de los tests y el promedio.

Esta figura (6) nos muestra que no hay relación clara entre la cantidad de ordenes conflictivas y el tiempo de ejecución. Si bien en todos se obtienen descensos cuando pasan de 0 a 5, luego a medida que aumentan las órdenes conflictivas toman comportamientos distintos. El A parece ir aumentando, el B aumenta un poco y después se

mantiene más o menos igual, el C aumenta justo después pero luego comienza a disminuir.

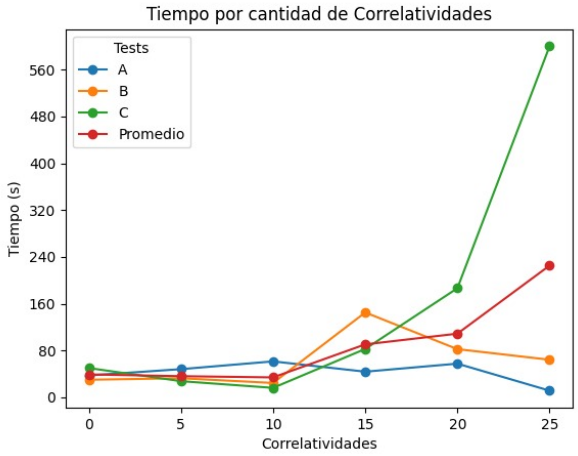

Figura 7: Gráfico del tiempo en función de ir variando la cantidad de órdenes correlativas para cada uno de los tests y el promedio.

Luego, al ir variando las correlatividades en esta figura (7), se observan desempeños similares al principio, pero desde las 10 en adelante las curvas parecen tomar caminos distintos para cada test. El test C había disminuido su tiempo hasta los 10 pero luego fue subiendo de manera considerable hasta los 25, los otros dos en un momento alcanzan un máximo y luego tienen una tendencia a disminuir sus tiempos.

En resumen, se observa que el ajuste de parámetros parece responder distinto dependiendo de la instancia, más allá de que el modelo sea el mismo. En algunos casos puede ayudar a disminuir considerablemente los tiempos, como al mantener activado el preprocesamiento en el test A (Figura 3), o el esfuerzo heurístico que ayudó bastante al desempeño del test C (Figura 4). Es por esto, que creemos que el ajuste de parámetros dependerá mucho de la instancia que se le pase al modelo y deberá estudiarse cada caso en su contexto. Es importante remarcar que si bien la cantidad de test empleados no es concluyente para ser categóricos en el ajuste de un parámetro, sí fue útil para visualizar cómo puede afectar cuantitavemente el uso de estos.

B. Con restricciones deseables

En este apartado nos interesaba observar cómo las restricciones deseables podían afectar al valor de la función objetivo. Para ello implementamos estas restricciones de dos maneras distintas. Como se dijo más arriba al describir el modelo, una manera fue con las variables $C_{ikk'}$ (23) y las $R_{ii'k}$ (24) y las restricciones 25 y 26, junto a la función objetivo que se observa en la ecuación 27. La otra manera fue conservando la primera función objetivo y agregando las dos restricciones 28 y 29, donde fuimos ajustando los valores del factor deseable por conflicto y el de repetición.

Para la primera implementación, donde modificamos la función objetivo, es importante aclarar que al graficar el valor de esta se le sumaron nuevamente los valores restados relacionados al coste de conflictos y órdenes repetitivas, ya que el valor que buscábamos era el de la solución pedida (beneficio de la empresa menos el pago de trabajadores). También, entendemos que establecer el costo en 0 es lo mismo que no contemplar las deseables y establecerlo en un valor suficientemente grande es igual a pedir que las restricciones deseables se cumplan en su totalidad.

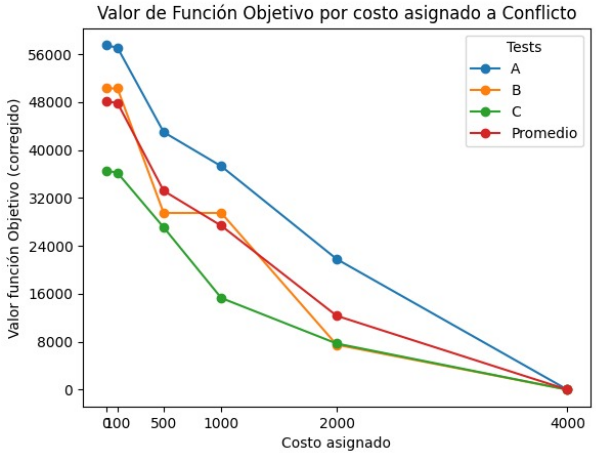

Figura 8: Gráfico del valor de la función objetivo en relación con el costo de asignar órdenes a trabajadores en conflicto.

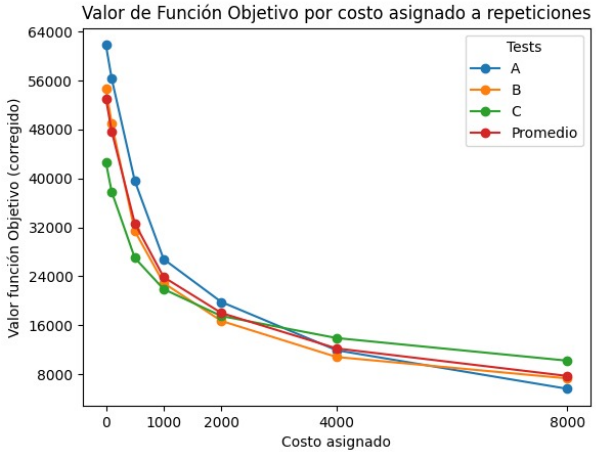

Figura 9: Gráfico del valor de la función objetivo en relación con el costo de asignar órdenes repetitivas a un mismo trabajador.

En el primer gráfico (Figura 8) se observa un comportamiento similar en los distintos tests, donde a medida que aumenta el costo asignado a cada conflicto el valor de la función objetivo tiende a bajar de manera, más o menos, lineal (con excepciones, como por ejemplo en el test B que en un momento incrementa levemente su valor para después seguir bajando).

Luego, al observar lo que pasa con las órdenes repetitivas (Figura 9) se tiene, al igual que antes, que a medida que el costo asignado aumenta, el valor de la función objetivo desciende, aunque esta vez la curva obtenida en los tres tests es similar, bajando considerablemente entre 0 y 1000, para luego ir descendiendo cada vez de manera más lenta.

Por otro lado, tenemos la otra manera de implementar las restricciones deseables, ligadas a lo que llamamos factor deseable. En este caso, recordemos que establecer el valor en 0 implica pedir que se cumplan todas las restricciones deseables mientras que el valor 1 es hacer que el modelo no dependa de ellas.

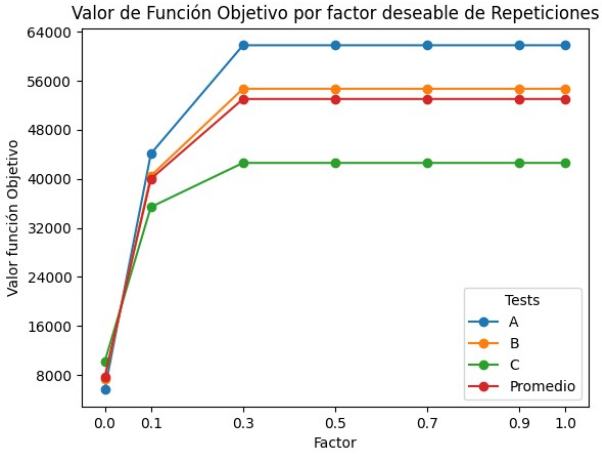

Figura 10: Gráfico del valor de la función objetivo en relación con el factor deseable elegido.

En la Figura 10 se observa que cuando el factor es 0 la función objetivo está en sus niveles más bajos, mientras que al 0.1 ya aumenta bastante en cada uno de los tests, para luego aumentar en menor medida y más o menos establecerse en el 0.3.

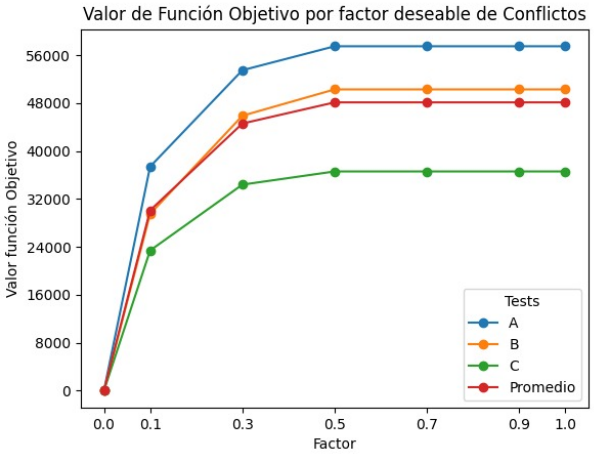

Figura 11: Gráfico del valor de la función objetivo en relación con el factor deseable elegido.

En este gráfico (Figura 11) se observa un comportamiento similar al obtenido con las órdenes repetitivas, aunque si se tienen un par de diferencias. En este caso cuando el factor es 0, es decir queremos que las deseables sean obligatorias, el modelo no encuentra manera de asignar las órdenes sin violar esta restricción. Luego, cuando el factor es 0.1 pasa lo mismo que en la Figura 10, sólo que esta vez se estabiliza en el 0.5.

Finalmente, al observar los cuatro gráficos descritos, se puede ver que los valores máximos de la función objetivo se alcanzan, en promedio, cuando usamos el factor deseable. Además, el uso de este método es fácil de ser ajustado y no es necesario poner variables extras en la función objetivo, lo que podría ser una ventaja en ciertos casos. El hecho de que se estabilice en cierto momento también es una señal de que es un buen método, lo que no sería fácil de conseguir con la primera manera, asignando un costo. De todas maneras, al igual que en el análisis cuantitativo, la cantidad de test empleados no es suficiente para tener un visión acabada del tema pero si nos permitió observar cómo se pueden comportar los modelos a la hora de implementar restricciones deseables.

IV. CONCLUSIONES

En este trabajo, hemos desarrollado y evaluado un modelo de programación lineal entera (PLE) para la asignación de cuadrillas de trabajadores a órdenes de servicio utilizando CPLEX. El objetivo era poder testear el modelo explorando diversos parámetros que nos proporciona este software midiendo tiempo de ejecución y abordar tanto restricciones obligatorias como deseables, viendo cómo el ajuste de estas últimas podía afectar el valor de la función objetivo.

El análisis realizado mostró que el ajuste de parámetros en CPLEX tiene un impacto significativo en los tiempos de ejecución y eficiencia del modelo, pero que este

varía según cada instancia específica del problema. Las pruebas realizadas evidencian que la activación de ciertas características, como el preprocesamiento y el esfuerzo heurístico, pueden mejorar el rendimiento en diferentes contextos. Sin embargo, es importante destacar que los resultados obtenidos no son concluyentes debido a la cantidad limitada de tests empleados, aunque proporcionan una visión inicial sobre el comportamiento del modelo bajo diferentes configuraciones.

Además, al incorporar restricciones deseables, se observó que los valores máximos de la función objetivo se alcanzan más consistentemente cuando se utiliza el factor deseable. Este enfoque demostró ser eficaz y relativamente sencillo de ajustar, evitando la necesidad de agregar variables adicionales a la función objetivo. Aunque los resultados mostrados nos pueden dar un panorama de cómo abordar restricciones deseables, la manera de implementarlas dependerá de cómo se valoren cualitativamente estas en cada contexto (para una empresa puede ser mejor atribuirle un costo que un factor, o viceversa).

En conclusión, nuestro modelo y los experimentos realizados contribuyen a un mayor entendimiento de la programación lineal entera, de cómo la modificación de distintos parámetros puede cambiar el tiempo de ejecución de distintos casos de tests, o de cómo distintas maneras de implementar restricciones deseables pueden afectar al valor de la función objetivo del modelo. Además, las estrategias y heurísticas testeadas, así como las observaciones derivadas de los resultados, proporcionan una buena base para futuras investigaciones y aplicaciones prácticas en problemas de optimización.

Finalmente, CPLEX muestra una notable capacidad de adaptación a diversos contextos y requerimientos operativos, aunque se necesita un profundo entendimiento del problema y la instancia para tomar decisiones acertadas en su implementación.