# Trabajo Práctico - Programación Lineal Entera

Intr. a la Investigación Operativa y Optimización

**Malena Sol Alamo (1620/21), Florencia Fontana Walser (1530/21)**

# Índice

# 1. Parte 1 - Modelo Original

## 1.1. Explicación

### 1.1.1. Variables

- $\delta_{ij} = \begin{cases} 1 \quad si \quad \textit{el trabajador } i \textit{ realiza la orden } j \\ 0 \qquad\qquad\qquad \textit{caso contrario} \end{cases}$
- $\alpha_{jkl} = \begin{cases} 1 \quad si \quad \textit{la orden } j \textit{ se realizo el dia } k \textit{ en el turno } l \\ 0 \qquad\qquad\qquad\qquad \textit{caso contrario} \end{cases}$
- $\beta_{ijkl} = \begin{cases} 1 \quad si \quad \delta_{ij}\alpha_{jkl} = 1 \textit{ i.e si el trabajador } i \textit{ realizo la orden } j \textit{ en el dia } k \textit{ en el turno } l \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{caso contrario} \end{cases}$
- $\omega_{ik} = \begin{cases} 1 \quad si \quad \textit{el trabajador } i \textit{ trabajo en al menos un turno en el día } k \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \textit{caso contrario} \end{cases}$
- $c_i^n = \textit{Cantidad de órdenes que hizo el trabajador } i \textit{ en el rango } n$
- $\Gamma_i^1 = \begin{cases} 1 \quad si \quad c_i^1 = 5 \\ 0 \qquad c_i^1 < 5 \end{cases}$
- $\Gamma_i^2 = \begin{cases} 1 \quad si \quad c_i^2 = 10 \\ 0 \qquad c_i^2 < 10 \end{cases}$
- $\Gamma_i^3 = \begin{cases} 1 \quad si \quad c_i^3 = 15 \\ 0 \qquad c_i^3 < 15 \end{cases}$
- $z = \textit{Cantidad máxima de órdenes que se realizaron por un trabajador}$
- $w = \textit{Cantidad mínima de órdenes que se realizaron por un trabajador}$

con:

- $i \in I = \{1, ..., T\}$
- $j \in J = \{1, ..., O\}$
- $k \in K = \{1, ..., 6\}$
- $l \in L = \{1, ..., 5\}$

### 1.1.2. Parámetros

- $b_j$ : *beneficio de la orden* $j$
- $t_j$ : *cantidad de trabajadores necesarios para la orden* $j$

### 1.1.3. Función objetivo

$$Max \sum_{j \in J} \sum_{k \in K} \sum_{l \in L} b_j \alpha_{jkl} - \sum_{i \in I} (1000\ c_i^1 + 1200\ c_i^2 + 1400\ c_i^3 + 1500\ c_i^4) - 0{,}00005\ z + 0{,}00005\ w$$

### 1.1.4. Restricciones obligatorias

1. Cada orden tiene los trabajadores que necesita. Como no necesariamente se deben realizar todas las órdenes, modelamos $\sum_{k\in K}\sum_{l\in L}\alpha_{jkl} > 0 \Rightarrow \sum_{i\in I}\delta_{ij} = t_j$:
   $\sum_{i\in I}\delta_{ij} = t_j \sum_{k\in K}\sum_{l\in L}\alpha_{jkl} \quad \forall j \in J$

2. Una orden se realiza en un solo turno, una sola a la vez:
   $\sum_{k\in K}\sum_{l\in L}\alpha_{jkl} \leq 1 \forall j \in J$

3. Para ver que ningún trabajador trabaje en dos órdenes al mismo tiempo, nos gustaría escribir lo siguiente, pero no es lineal:
   $\sum_{j\in J}\alpha_{jkl}\delta_{ij} \leq 1 \ \forall i \in I, k \in K, l \in L$
   Usando linealización, definimos $\beta_{ijkl}$ y agregamos las siguientes restricciones:

   - $\delta_{ij} + \alpha_{jkl} \leq \beta_{ijkl} + 1 \ \forall i \in I, j \in J, k \in K, l \in L$
   - $2\beta_{ijkl} \leq \delta_{ij} + \alpha_{jkl} \ \forall i \in I, j \in J, k \in K, l \in L$
   - $\sum_{j\in J}\beta_{ijkl} \leq 1 \ \forall i \in I, k \in K, l \in L$

4. Ningún trabajador puede trabajar los 6 días de la semana:

   - $\omega_{ik} \geq \frac{1}{5}\sum_{l\in L}\sum_{j\in J}\beta_{ijkl} \ \forall i \in I, k \in K$
   - $\sum_{i\in L}\sum_{j\in J}\beta_{ijkl} \geq \omega_{ik} \ \forall i \in I, k \in K$
   - $\sum_{k\in K}\omega_{ik} \leq 5 \ \forall i \in I$

5. Ningún trabajador puede trabajar los 5 turnos en un día:
   $\sum_{l\in L}\sum_{j\in J}\beta_{ijkl} \leq 4 \ \forall i \in I, k \in K$

6. Algunas órdenes no pueden resolverse por el mismo trabajador consecutivamente. Por ejemplo, sean $j$ y $j^*$ dos tareas conflictivas: si el empleado $i$ hace la tarea $j$ en el turno $l$, no puede hacer j* en el turno $l+1$. Asimismo, si el empleado $i$ hace la tarea j* en el turno $l$, no puede hacer la $j$ en el turno $l+1$. Podemos pensarlo como una implicacion X⇒¬ Y, que se modela de la siguiente manera:

   - $\beta_{ijkl} \leq 1 - \beta_{ij^*kl+1} \forall i \in I; (j, j^*) \in \{tareas\_conflictivas\}; k \in K; l \in L - \{5\}$
   - $\beta_{ij^*kl} \leq 1 - \beta_{ijkl+1} \forall i \in I; (j, j^*) \in \{tareas\_conflictivas\}; k \in K; l \in L - \{5\}$

7. Una orden debe tener asignada sus $t_i$ trabajadores en el mismo turno:

   - $\sum_{i\in I}\beta_{ijkl} = t_j \ \alpha_{jkl} \ \forall j \in J, k \in K, l \in L$

8. Algunas órdenes son correlativas. Sean $j$, $j^*$ correlativas, si satisfago la orden $j$ en el dia $k$ en el turno $l$ entonces la orden $j^*$ debera realizarse en el turno siguiente. Es una implicacion X ⇒ Y, que se modela de la siguiente manera:
   $\alpha_{jkl} \leq \alpha_{j^*kl+1} \quad \forall (j, j^*) \in \{ordenes\_correlativas\}, k \in K, l \in L - \{5\}$

9. La diferencia entre el trabajador que más tareas realizó y el que menos realizó no debe ser mayor a 8. Para modelar esta restricción definimos dos variables: $z$ representará el máximo y $w$ el mínimo. Agregamos las restricciones:

   - $z \geq \sum_{j \in J} \delta_{ij} \ \forall i \in I$
   - $w \leq \sum_{j \in J} \delta_{ij} \ \forall i \in I$
   - $z - w \leq 8$

   Ahora bien, nos gustaría además que $z$ y $w$ se ajusten lo mejor posible a la cantidad máxima y mínima de órdenes que realizó un trabajador respectivamente. Para ello agregamos a la función objetivo a $z$ restando y $w$ sumando, acompañados de un coeficiente despreciable de 0,00005.

10. Para modelar la remuneración, agregamos las siguientes restricciones:

    - $5\ \Gamma_i^1 \leq c_i^1 \leq 5 \ \forall i \in I$
    - $5\ \Gamma_i^2 \leq c_i^2 \leq 5\ \Gamma_i^1 \ \forall i \in I$
    - $5\ \Gamma_i^3 \leq c_i^3 \leq 5\ \Gamma_i^2 \ \forall i \in I$
    - $c_i^4 \leq M\ \Gamma_i^3 \ \forall i \in I$

    con $M$ una cota superior para la cantidad de órdenes que realiza un trabajador. En nuestra implementación M = O (cantidad de órdenes).

    Para definir las $c_i$ podemos escribir:
    $\sum_{k=1}^{4} c_i^k = \sum_{j \in J} \delta_{ij} \ \forall i \in I$

## 1.2. Modelo Completo

### 1.2.1. Función objetivo

$$Max \sum_{j\in J}\sum_{k\in K}\sum_{l\in L} b_j \alpha_{jkl} - \sum_{i\in I}(1000\ c_i^1 + 1200\ c_i^2 + 1400\ c_i^3 + 1500\ c_i^4) - 0{,}00005\ z + 0{,}00005\ w$$

### 1.2.2. Sujeto a

- $\sum_{i\in I} \delta_{ij} = t_j \sum_{k\in K}\sum_{l\in L} \alpha_{jkl} \quad \forall j \in J$
- $\sum_{k\in K}\sum_{l\in L} \alpha_{jkl} \leq 1\ \forall j \in J$
- $\delta_{ij} + \alpha_{jkl} \leq \beta_{ijkl} + 1 \quad \forall i \in I, j \in J, k \in K, l \in L$
- $2\beta_{ijkl} \leq \delta_{ij} + \alpha_{jkl} \quad \forall i \in I, j \in J, k \in K, l \in L$
- $\sum_{j\in J} \beta_{ijkl} \leq 1 \quad \forall i \in I, k \in K, l \in L$
- $\omega_{ik} \geq \frac{1}{5} \sum_{l\in L}\sum_{j\in J} \beta_{ijkl}\ \forall i \in I, k \in K$
- $\sum_{i\in L}\sum_{j\in J} \beta_{ijkl} \geq \omega_{ik}\ \forall i \in I, k \in K$
- $\sum_{k\in K} \omega_{ik} \leq 5\ \forall i \in I$
- $\sum_{l\in L}\sum_{j\in J} \beta_{ijkl} \leq 4\ \forall i \in I, k \in K$
- $\beta_{ijkl} \leq 1 - \beta_{ij^*kl+1} \quad \forall i \in I; (j, j^*) \in \{tareas\_conflictivas\}; k \in K; l \in L - \{5\}$
- $\beta_{ij^*kl} \leq 1 - \beta_{ijkl+1} \quad \forall i \in I; (j, j^*) \in \{tareas\_conflictivas\}; k \in K; l \in L - \{5\}$
- $\sum_{i\in I} \beta_{ijkl} = t_j\ \alpha_{jkl}\ \forall j \in J, k \in K, l \in L$
- $\alpha_{jkl} \leq \alpha_{j^*kl+1} \quad \forall (j, j^*) \in \{ordenes\_correlativas\}, k \in K, l \in L - \{5\}$
- $z \geq \sum_{j\in J} \delta_{ij}\ \forall i \in I$
- $w \leq \sum_{j\in J} \delta_{ij}\ \forall i \in I$
- $z - w \leq 8$
- $5\ \Gamma_i^1 \leq c_i^1 \leq 5\ \forall i \in I$
- $5\ \Gamma_i^2 \leq c_i^2 \leq 5\ \Gamma_i^1\ \forall i \in I$
- $5\ \Gamma_i^3 \leq c_i^3 \leq 5\ \Gamma_i^2\ \forall i \in I$

- $c_i^4 \leq M\ \Gamma_i^3\ \forall i \in I$

  con $M$ una cota superior para la cantidad de órdenes que realiza un trabajador. En nuestra implementación M = O (cantidad de órdenes).

- $$\sum_{k=1}^{4} c_i^k = \sum_{j \in J} \delta_{ij}\ \forall i \in I$$

  donde:

- $\delta_{ij},\ \alpha_{jkl},\ \beta_{ijkl},\ \omega_{ik},\ \Gamma_i^n \in \{0,1\}\ \ \forall i \in I, j \in J, k \in K, l \in L, n \in \{1,2,3\}$
- $c_i^n,\ z,\ w \in Z_{\geq 0}\ \ \forall i \in I, n \in \{1,2,3,4\}$

## 1.3. Experimentación Algorítmica

Para esta parte generamos 5 entradas con una cantidad fija de 100 órdenes para correr con nuestra implementación, variando la cantidad de trabajadores de 5 a 30.

A partir de estas 5 entradas corrimos nuestro algoritmo variando los siguientes parámetros:

- `mip.strategy.nodeselect`: determina la estrategia que el solver utiliza para seleccionar el siguiente nodo a explorar en el árbol de búsqueda. Los valores posibles son 0 (Depth-first search), 1 (mejor cota, default), 2 (mejor estimación) y 3 (mejor estimación alternativa).
- `mip.strategy.variableselect`: determina cuál es la siguiente variable a la que se aplica la ramificación en el árbol. Los valores posibles son -1 (mínima infactibilidad), 0 (automático), 1 (máxima infactibilidad), 2 (pseudo costos), 3 (ramificación fuerte) y 4 (pseudo costos reducidos).
- `preprocessing.presolve`: determina si se usan técnicas de preprocesamiento para simplificar el modelo antes de resolverlo. Los valores posibles son 0 (no se aplica preprocesamiento) y 1 (preprocesamiento completo).
- `mip.strategy.heuristiceffort`: controla la cantidad de esfuerzo dedicado a la búsqueda heurística de soluciones factibles. Los valores posibles son 0 (ningún esfuerzo heurístico), 0.5 (esfuerzo heurístico reducido), 1 (esfuerzo heurístico moderado) y 2 (esfuerzo heurístico alto).

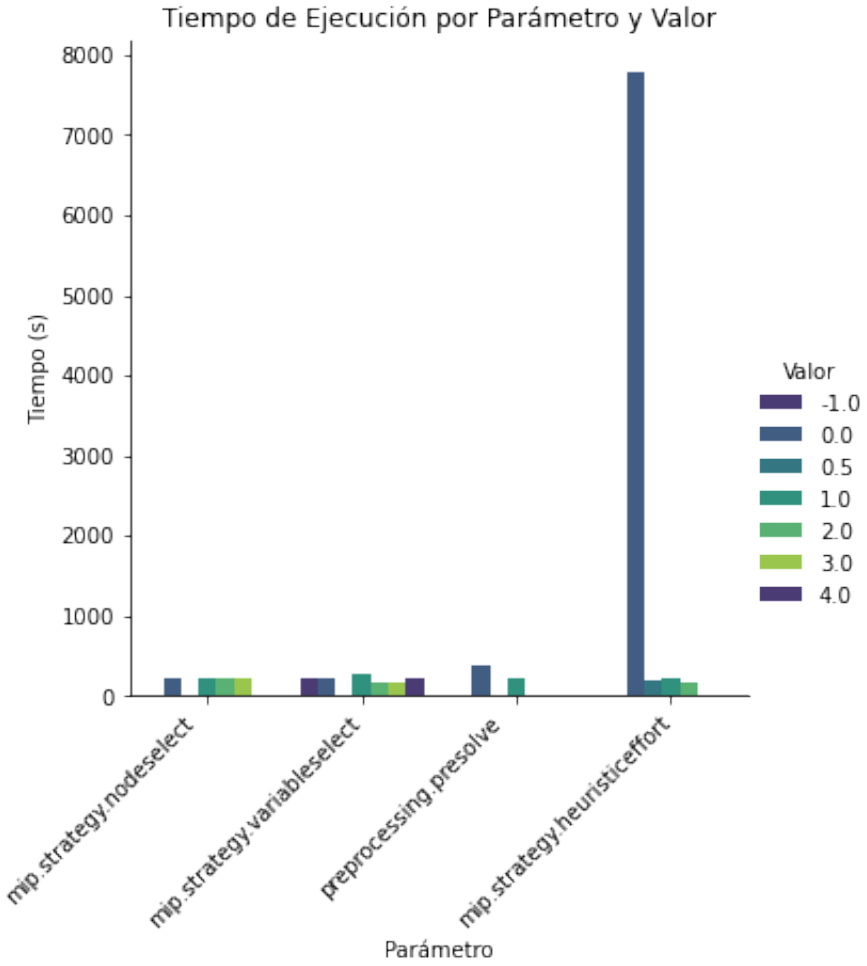
A continuacioń se muestran los resultados del tiempo promedio de ejecución para cada una de estas configuraciones.

| **Parámetro** | **Valor** | **Tiempo (s)** | **Objetivo** |
|---|---|---|---|
| mip.strategy.nodeselect | 0.0 | 218.310875 | 246149.99996 |
| mip.strategy.nodeselect | 1.0 | 212.359879 | 246149.99996 |
| mip.strategy.nodeselect | 2.0 | 220.355264 | 246149.99996 |
| mip.strategy.nodeselect | 3.0 | 216.540686 | 246149.99996 |
| mip.strategy.variableselect | -1.0 | 212.037502 | 246149.99996 |
| mip.strategy.variableselect | 0.0 | 214.373362 | 246149.99996 |
| mip.strategy.variableselect | 1.0 | 263.930900 | 246149.99996 |
| mip.strategy.variableselect | 2.0 | 175.138922 | 246149.99996 |
| mip.strategy.variableselect | 3.0 | 166.015699 | 246149.99996 |
| mip.strategy.variableselect | 4.0 | 217.198892 | 246149.99996 |
| preprocessing.presolve | 0.0 | 376.246160 | 246149.99996 |
| preprocessing.presolve | 1.0 | 221.553343 | 246149.99996 |
| mip.strategy.heuristiceffort | 0.0 | 7792.805994 | 246149.99996 |
| mip.strategy.heuristiceffort | 0.5 | 208.776560 | 246149.99996 |
| mip.strategy.heuristiceffort | 1.0 | 218.751865 | 246149.99996 |
| mip.strategy.heuristiceffort | 2.0 | 167.251504 | 246149.99996 |

Cuadro 1: Resultados promediados de los 5 ejemplos.

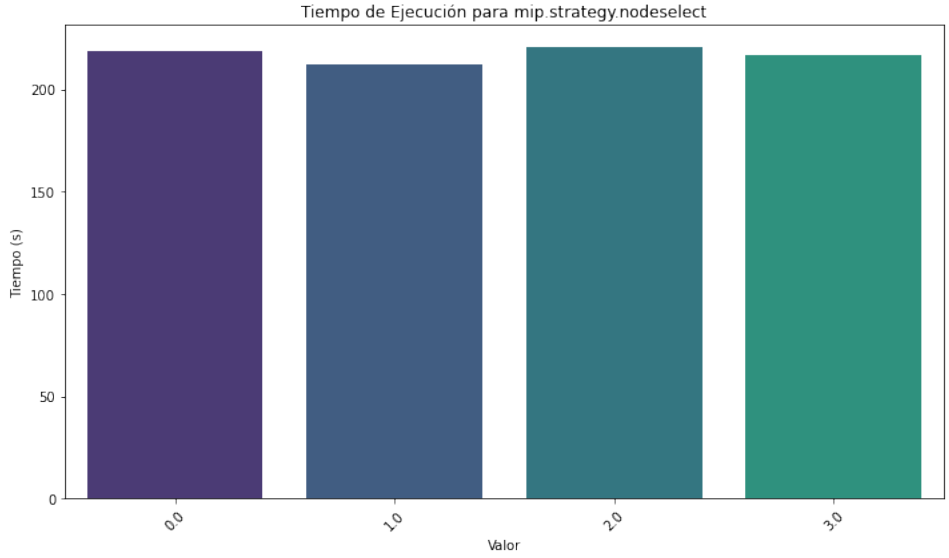
En principio podemos notar que el valor de la Función Objetivo llega al mismo valor para todas las configuraciones.

Veamos qué sucede con el tiempo de ejecución.




En el gráfico se ve claramente que el tiempo de ejecución promedio aumenta drásticamente cuando `mip.strategy.heuristiceffort = 0`, es decir, cuando no se aplican heurísticas.

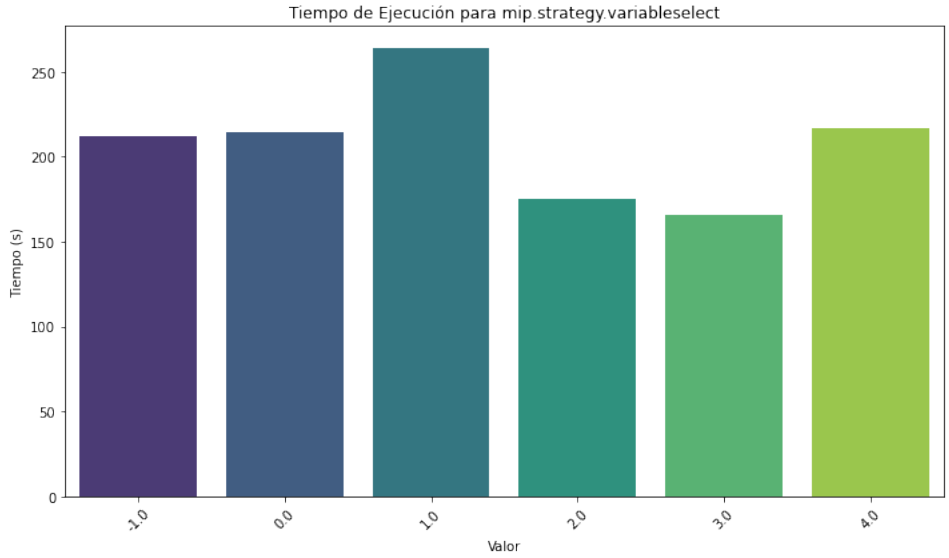
Miremos ahora cada parámetro en particular, empezando por `mip.strategy.nodeselect`.




Vemos que cuando `mip.strategy.nodeselect = 1`, es decir, cuando utilizamos la regla de mejor cota,

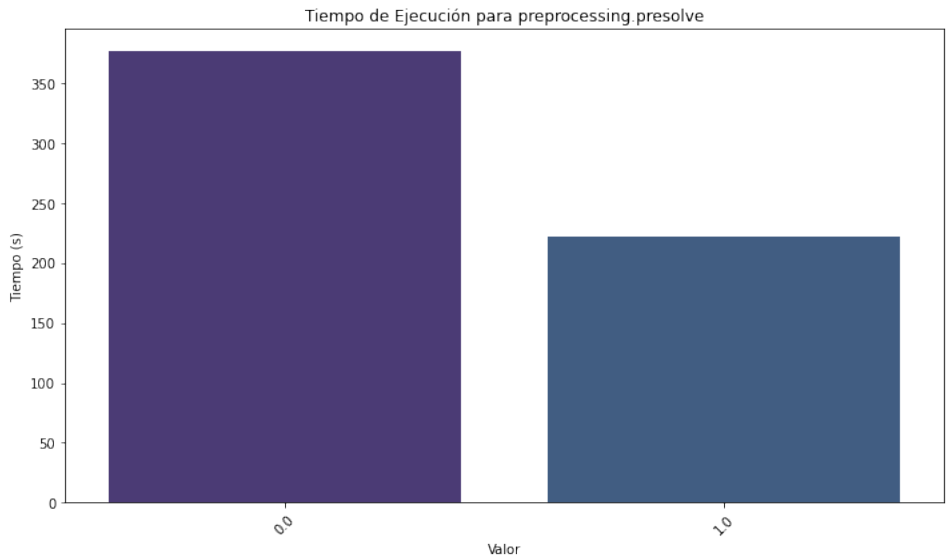
el tiempo de ejecución es el mínimo. El valor con el que más tarda el solver es con `mip.strategy.nodeselect = 2`, es decir, con la regla de mejor estimación, que selecciona el nodo que tiene la mejor estimación de la solución óptima basándose en una heurística. Como a veces las heurísticas no son precisas y la calidad de la estimación varía según el problema, tiene sentido que esta sea la alternativa que más tarda en correr.

Veamos ahora el gráfico para `mip.strategy.variableselect`.




La configuración con menor tiempo de ejecución es `mip.strategy.variableselect = 3`, es decir, cuando se selecciona la variable por ramificación fuerte (*Strong Branching*).
Es curioso que cuando `mip.strategy.variableselect = 0`, es decir, cuando CPLEX elige automáticamente la regla de selección de variable, el tiempo no es mínimo. Es posible que para este problema en particular la regla de *Strong Branching* sea la más eficiente, y CPLEX no lo haya detectado.
La opción que más tarda en ejecutar es `mip.strategy.variableselect = 1`, que es la regla de máxima infactibilidad. En esta opción se elige la variable cuyo valor fraccional está más alejado de un valor entero. La desventaja de esta opción es que a veces explora ramas no tan prometedoras antes de encontrar una buena solución, a esto se podría deber el tiempo de ejecución.

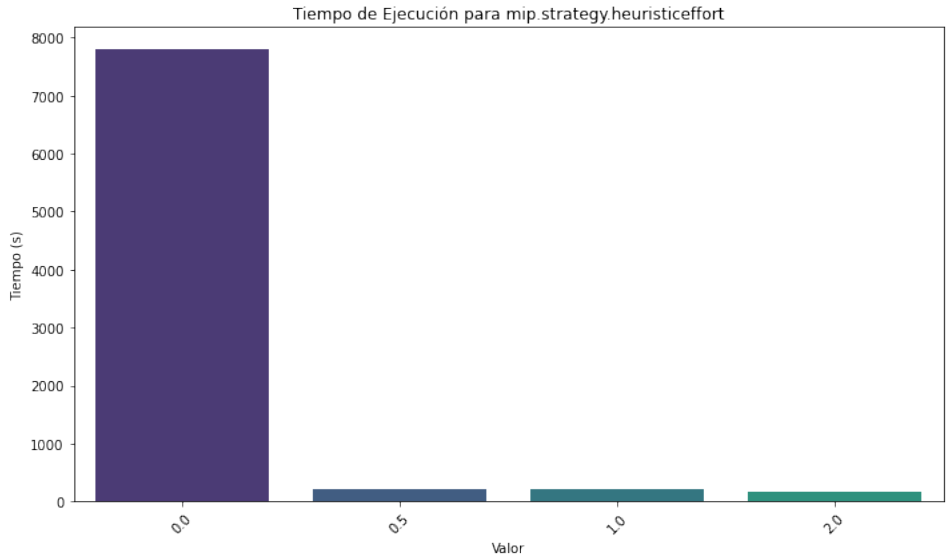
Veamos qué sucede cuando aplicamos o no preprocesamiento.




Como era de esperar, cuando `preprocessing.presolve = 0`, es decir, cuando no aplicamos preprocesamiento, el tiempo de ejecución aumenta. Esto se debe a que el modelo sin preprocesar es

generalmente más grande y complejo, ya que no se reduce su tamaño con las técnicas varias que aplicaría CPLEX cuando `preprocessing.presolve = 1`.

Veamos por último los tiempos promediados según el nivel de esfuerzo heurístico.




Como mencionamos anteriormente, el tiempo de ejecución aumenta considerablemente al quitar del todo las heurísticas, es decir, `mip.strategy.heuristiceffort = 0`. Esto demuestra que para nuestro problema, las heurísticas mejoran las posibilidades de encontrar soluciones factibles rápidamente.

# 2. Parte 2 - Modelo con Restricciones Deseables

## 2.1. Explicación

### 2.1.1. Variables agregadas

Además de las variables del modelo original se agregan las siguientes:

- $\epsilon_{ii^*j} = \begin{cases} 1 \quad si \quad \textit{si asigné al trabajador i y al i* en la misma orden j} \\ 0 \qquad\qquad\qquad\qquad \textit{caso contrario} \end{cases}$

  $\forall i \in I = \{1, ..., T\}, \forall (j, j^*) \in conflictos\_trabajadores$

- $\mu_{ijj^*} = \begin{cases} 1 \quad si \quad \textit{si asigné al trabajador i en la orden j y en la j*} \\ 0 \qquad\qquad\qquad\qquad \textit{caso contrario} \end{cases}$

  $\forall j \in J = \{1, ..., O\}, \forall (i, i^*) \in ordenes\_repetitivas$

### 2.1.2. Función Objetivo Modificada

A la hora de reescribir nuestra función objetivo, nos gustaría asignarle un "peso" a las restricciones deseables. Esto es, designar un coeficiente adecuado que acompañe a las variables $\epsilon_{ii^*j}$ y $\mu_{ijj^*}$, para modelar de manera correcta el costo de que suceda algo no deseable. Para ello, multiplicamos por \$1000 cada variable, así cada vez que dos trabajadores con conflicto sean asignados a la misma tarea, o asigne a un trabajador en dos órdenes repetitivas, le costará a la empresa \$1000. Este valor fue pensado teniendo en cuenta la escala del pago por hora por cada tarea, para que sea un costo razonable.

$$Max \sum_{j \in J} \sum_{k \in K} \sum_{l \in L} b_j \alpha_{jkl} - \sum_{i \in I} (1000\ c_i^1 + 1200\ c_i^2 + 1400\ c_i^3 + 1500\ c_i^4) - 0{,}00005\ z + 0{,}00005\ w$$
$$- \sum_{j \in J} \sum_{(i,i^*) \in C} 1000\ \epsilon_{ii^*j} - \sum_{i \in I} \sum_{(j,j^*) \in R} 1000\ \mu_{ijj^*}$$

donde $C = \{conflictos\_trabajadores\},\ R = \{ordenes\_repetitivas\}$

### 2.1.3. Restricciones deseables

Además de las restricciones del modelo original se agregan las siguientes:

1. Definimos $\epsilon_{ii^*j}$. Queremos que $\delta_{ij} \wedge \delta_{i^*j} \iff \epsilon_{ii^*j}$. Agregamos las siguientes restricciones:
   - $2\ \epsilon_{ii^*j} \leq \delta_{ij} + \delta_{i^*j} \quad \forall j \in J, (i, i^*) \in C$
   - $\delta_{ij} + \delta_{i^*j} \leq \epsilon_{ii^*j} + 1 \quad \forall j \in J, (i, i^*) \in C$
2. Definimos $\mu_{ijj^*}$. Queremos que $\delta_{ij} \wedge \delta ij^* \iff \mu_{ijj^*}$ Agregamos las siguientes restricciones:
   - $2\ \mu_{ijj^*} \leq \delta_{ij} + \delta_{i^*j}\ \forall i \in I, \quad \forall (j, j^*) \in R$
   - $\delta_{ij} + \delta_{ij^*} \leq \mu_{ijj^*} + 1\ \forall i \in I, \quad \forall (j, j^*) \in R$

## 2.2. Modelo Completo con Restricciones Deseables

### 2.2.1. Función objetivo

$$Max \sum_{j\in J}\sum_{k\in K}\sum_{l\in L} b_j \alpha_{jkl} - \sum_{i\in I}(1000\ c_i^1 + 1200\ c_i^2 + 1400\ c_i^3 + 1500\ c_i^4) - 0{,}00005\ z + 0{,}00005\ w$$
$$- \sum_{j\in J}\sum_{(i,i^*)\in C} 1000\ \epsilon_{ii^*j} - \sum_{i\in I}\sum_{(j,j^*)\in R} 1000\ \mu_{ijj^*}$$

### 2.2.2. Sujeto a

- $\sum_{i\in I}\delta_{ij} = t_j \sum_{k\in K}\sum_{l\in L}\alpha_{jkl} \quad \forall j \in J$
- $\sum_{k\in K}\sum_{l\in L}\alpha_{jkl} \leq 1 \forall j \in J$
- $\delta_{ij} + \alpha_{jkl} \leq \beta_{ijkl} + 1\ \forall i \in I, j \in J, k \in K, l \in L$
- $2\beta_{ijkl} \leq \delta_{ij} + \alpha_{jkl}\ \forall i \in I, j \in J, k \in K, l \in L$
- $\sum_{j\in J}\beta_{ijkl} \leq 1\ \forall i \in I, k \in K, l \in L$
- $\omega_{ik} \geq \frac{1}{5}\sum_{l\in L}\sum_{j\in J}\beta_{ijkl}\ \forall i \in I, k \in K$
- $\sum_{i\in L}\sum_{j\in J}\beta_{ijkl} \geq \omega_{ik}\ \forall i \in I, k \in K$
- $\sum_{k\in K}\omega_{ik} \leq 5\ \forall i \in I$
- $\sum_{l\in L}\sum_{j\in J}\beta_{ijkl} \leq 4\ \forall i \in I, k \in K$
- $\beta_{ijkl} \leq 1 - \beta_{ij^*kl+1} \forall i \in I; (j, j^*) \in \{tareas\_conflictivas\}; k \in K; l \in L - \{5\}$
- $\beta_{ij^*kl} \leq 1 - \beta_{ijkl+1} \forall i \in I; (j, j^*) \in \{tareas\_conflictivas\}; k \in K; l \in L - \{5\}$
- $\sum_{i\in I}\beta_{ijkl} = t_j\ \alpha_{jkl}\ \forall j \in J, k \in K, l \in L$
- $\alpha_{jkl} \leq \alpha_{j^*kl+1} \quad \forall (j, j^*) \in \{ordenes\_correlativas\}, k \in K, l \in L - \{5\}$
- $z \geq \sum_{j\in J}\delta_{ij}\ \forall i \in I$
- $w \leq \sum_{j\in J}\delta_{ij}\ \forall i \in I$
- $z - w \leq 8$
- $5\ \Gamma_i^1 \leq c_i^1 \leq 5\ \forall i \in I$
- $5\ \Gamma_i^2 \leq c_i^2 \leq 5\ \Gamma_i^1\ \forall i \in I$

- $5\ \Gamma_i^3 \leq c_i^3 \leq 5\ \Gamma_i^2\ \forall i \in I$

- $c_i^4 \leq M\ \Gamma_i^3\ \forall i \in I$

  con $M$ una cota superior para la cantidad de órdenes que realiza un trabajador. En nuestra implementación M = O (cantidad de órdenes).

- $\sum_{k=1}^{4} c_i^k = \sum_{j \in J} \delta_{ij}\ \forall i \in I$
- $2\ \epsilon_{ii^*j} \leq \delta_{ij} + \delta_{i^*j}\ \ \forall j \in J, (i, i^*) \in C$
- $\delta_{ij} + \delta_{i^*j} \leq \epsilon_{ii^*j} + 1\ \ \forall j \in J, (i, i^*) \in C$
- $2\ \mu_{ijj^*} \leq \delta_{ij} + \delta_{i^*j}\ \ \forall i \in I, \forall (j, j^*) \in R$
- $\delta_{ij} + \delta_{ij^*} \leq \mu_{ijj^*} + 1\ \ \forall i \in I, \forall (j, j^*) \in R$

  donde:

- $\delta_{ij},\ \alpha_{jkl},\ \beta_{ijkl},\ \omega_{ik},\ \Gamma_i^n,\ \epsilon_{ii^*j},\ \mu_{ijj^*}\ \in \{0, 1\}\ \ \forall i \in I, j \in J, k \in K, l \in L, n \in \{1, 2, 3\}$, $(i, i^*) \in C, (j, j^*) \in R$
- $c_i^n,\ z,\ w \in Z_{\geq 0}\ \ \forall i \in I, n \in \{1, 2, 3, 4\}$

### 2.3. Experimentación Algorítmica

Para la experimentación del modelo con restricciones deseables usamos algunos ejemplos y obtuvimos los siguientes resultados.

En una de las instancias (6 trabajadores y 100 órdenes) se realizan órdenes repetitivas únicamente cuando los mismas tienen costo <1 en la función objetivo. En otro caso, se observa que en la solución no se asignan trabajadores conflictivos ni se realizan órdenes repetitivas.

| **Ejemplo** | **Solución sin restricciones deseables** | **Solución con restricciones deseables y costo 1000** | **Solución con restricciones deseables y costo 500** | **Solución con restricciones deseables y costo 1** |
|---|---|---|---|---|
| a | 210371 | 209629 | 209629 | 210369 |

Cuadro 2: Ejemplo función objetivo según costo de restricciones deseables

En otros ejemplos, en cambio, se puede observar que el valor de la función objetivo es el mismo incluyendo las restricciones deseables y sin incluirlas. Esto podría deberse a que las órdenes que dan más beneficio no se ven afectadas por los conflictos entre trabajadores o por repeticiones.

| **Ejemplo** | **Sin restricciones deseables** | **Con restricciones deseables y costo 1000** | **Con restricciones deseables y costo 500** | **Con restricciones deseables y costo 1** |
|---|---|---|---|---|
| b | 248479 | 248479 | 248479 | 248479 |
| c | 299970 | 299970 | 299970 | 299970 |
| d | 204934 | 204934 | 204934 | 204934 |

Cuadro 3: Ejemplos función objetivo según costo de restricciones deseables

## 3. Anexo - Implementación

En esta sección se describen los archivos incluidos en el archivo comprimido `.zip` adjunto al presente informe. Estos archivos contienen los códigos y datos utilizados para el desarrollo del modelo y la experimentación de parámetros con CPLEX. A continuación, se detalla el contenido y la finalidad de cada archivo:

- `asignacion_cuadrillas.py`: este archivo contiene el código principal para la definición y resolución del modelo original de asignación de cuadrillas. Toma como parámetro un archivo `.txt` formateado como en el enunciado, y resuelve el problema lineal.
- `asignacion_cuadrillas_deseables.py`: contiene las mismas definiciones de variables y restricciones que el archivo anterior, y agregamos las variables y restricciones deseables.
- `asignacion_cuadrillas_experimentacion.py`: contiene mismas definiciones de variables y restricciones que el primer archivo pero en vez de resolver el problema una sola vez, lo resuelve probando los distintos valores de los parámetros vistos. Devuelve un archivo `.csv` con el tiempo de ejecución y el valor de la función objetivo para el problema cuando se cambia cada valor en cada parámetro.
- `ej1.txt, ej2.txt, ej3.txt, ej4.txt y ej5.txt`: son las 5 entradas que generamos.
- `generador_tests.py`: script para generar ejemplos de entrada. Es el que usamos para generar los ejemplos mencionados arriba.
- `graficos_experimentacion.ipynb`: es el Notebook de Python utilizado para generar los gráficos a partir de los resultados que generó `asignacion_cuadrillas_experimentacion.py`.

- `resultados_parametros_ej1.csv`, `resultados_parametros_ej2.csv`, `resultados_parametros_ej3.csv`, `resultados_parametros_ej4.csv` y `resultados_parametros_ej5.csv`: son las 5 tablas que generamos para promediar los tiempos de ejecución.