Maestría en Ciencia de Datos
Optimización y análisis prescriptivo
Trabajo práctico final
Octubre 2023

Profesor:
- Pousa, Federico

Alumnos:
- Aubone, Patricio
- Brosky, Brian
- Tentoni, Nicolás

Repositorio GitHub:
https://github.com/checkudesa/FieldServiceManagement

Introducción.

El problema de la asignación de trabajadores a órdenes específicas de trabajo dadas ciertas condiciones a cumplir es un problema común en la logística de diversas organizaciones. En el presente trabajo formulamos la resolución de la planificación semanal (lunes a sábado) con 5 turnos diarios de una cuadrilla de trabajadores, dado un listado de órdenes y ciertas restricciones, utilizando programación lineal entera como marco teórico con una implementación en IBM CPLEX ILOG Python API.

Datos.

Contamos con datos donde se encuentran la cantidad de trabajadores, cantidad de órdenes, ids y beneficio de cada orden, cantidad de trabajadores necesarios para llevarlas a cabo, cantidad de órdenes conflictivas (no pueden realizarse una después de la otra) con sus respectivas ids, cantidad de órdenes correlativas (deben realizarse una después de la otra) con sus respectivas ids, cantidad de conflictos entre trabajadores (idealmente no deberían ser asignados a la misma orden) con sus respectivas ids, órdenes repetitivas (idealmente no deberían ser asignadas al mismo trabajador) con sus respectivas ids. Considerando dicha estructura, generamos datos sintéticos para probar la capacidad resolutiva de nuestro modelo.

Objetivo del problema.

Se intentará maximizar el profit (beneficio - costo) siguiendo una serie de restricciones brindadas en la consigna. El beneficio se obtiene al completar una orden y el costo se compone de una función lineal a trozos en donde cada trabajador percibe 1000, 1200, 1400 o 1500 unidades por orden realizada dependiendo si realizó hasta 5, 10, 15 o más de 15 órdenes semanales.

Una aclaración importante es que se entiende por orden a un tipo de trabajo, es decir, una orden puede consistir en una instalación, un mantenimiento o una inspección que debe ser realizado por cuadrillas que son asignadas específicamente a los diferentes trabajos. Considerando esto, cada orden puede ser realizada más de una vez, no es un pedido particular que, una vez ejecutado, queda cumplido y descartado de la lista existente sino un tipo de tarea que tiene asociado un beneficio para la empresa y una cantidad de trabajadores necesarios para cumplirla en un turno laboral.

Modelo: Formular un modelo que cumpla todas las restricciones para realizar una asignación y que tenga en cuenta la función objetivo enunciada.

Definiciones.

Para poder formular el modelo de programación lineal entera consideramos las siguientes variables e índices:

$i \in \mathbb{N} = id\, del\, trabajador\, i \in \mathbb{N} = id\, del\, trabajador$

$n \in \mathbb{N} = cantidad\, de\, trabajadores\, en\, la\, cuadrilla\, n \in \mathbb{N} = cantidad\, de\, trabajadores\, en\, la\, cuadrilla$

$d \in \{1, 2, 3, 4, 5, 6\} = dia\, de\, la\, semana\, de\, lunes\, a\, sabado$

$t \in \{1, 2, 3, 4, 5\} = turno\, del\, dia$

$o \in \mathbb{N} = id\, de\, la\, orden$

$c \in \mathbb{N} = cantidad\, de\, ordenes\, solicitadas$

$beneficio_o \in \mathbb{Q}+ = beneficio\, de\, la\, orden\, o$

$trabajadores_o \in \mathbb{N} = trabajadores\, necesarios\, para\, la\, completar\, la\, orden\, o$

$X_{i,t,o,d} \in \{0, 1\} = trabajador\, i,\, en\, el\, turno\, t,\, realiza\, la\, orden\, o\, el\, dia\, d$

$Z_{t,o,d} \in \{0, 1\} = en\, el\, turno\, t\, se\, realiza\, la\, orden\, o\, el\, dia\, d$

$D_{i,d} \in \{0,1\} = trabajador\, i\, trabajo\, el\, dia\, d.$
$Y_{i,r} \in \{0,1,2,3,4,5\} = cantidad\, de\, ordenes\, que\, realizo\, el\, trabajador\, i\, en\, nivel\, de\, remuneracion\, r$
$r \in \{1,2,3\} = nivel\, de\, remuneracion$
$W_{i,r} \in \{0,1\} = indicadora\, para\, el\, trabajador\, i\, y\, nivel\, de\, remuneracion\, r$
$costo_r \in \{1000, 1200, 1400, 1500\} = costo\, por\, orden\, segun\, nivel\, de\, remuneracion$

Función objetivo

Construimos una función objetivo a **maximizar**. En la misma, definimos el beneficio - primer término - como la sumatoria de que los trabajadores i realicen la orden o, en el día d y turno t dividiendo el $beneficio_o$ por $trabajadores_o$. Dado que toda orden debe ser realizada por el total de trabajadores necesarios, esta definición no presenta casos de uso que no se encuentren en la misma. A su vez, definimos el costo como la sumatoria de la cantidad de órdenes que los trabajadores i realizaron en el nivel de compensación r multiplicado por el monto de dicho escalón.

$$\sum_{i=1}^{n}\sum_{t=1}^{5}\sum_{o=1}^{c}\sum_{d=1}^{6}\left(X_{i,t,o,d} * \frac{beneficio_o}{trabajadores_o}\right) - \sum_{i}^{n}\sum_{r}^{4}(Y_{i,r} * costo_r)$$

Restricciones del problema.

Está función objetivo deberá ser maximizada siguiendo las siguientes restricciones:

1) No toda orden de trabajo tiene que ser resuelta.

$$\sum_{o=1}^{c} X_{i,t,o,d} \geq 0$$

2) Ningún trabajador puede trabajar los 6 días de la planificación. Para cada trabajador, la suma del valor de su respectiva variable D para todos los días laborales debe ser menor o igual a 5.

$$\sum_{d=1}^{6} D_{i,d} \leq 5 \,\forall\, i$$

Para implementar esta restricción se debe también capturar la relación existente entre la variable D y la variable X. Para esto en una primera instancia definimos que, para cada trabajador en cada día, la suma de órdenes realizadas en los distintos turnos (X) menos la cantidad de días trabajados (D) debe ser mayor o igual a cero. De esta manera buscamos que la variable binaria D se active (valga 1) cuando el trabajador realiza alguna orden.

$$\sum_{t=1}^{5}\sum_{o=1}^{c} X_{i,t,o,d} - D_{i,d} \geq 0 \,\forall\, i, d$$

Sin embargo, esta restricción permite que habiendo realizado alguna orden, la variable D valga cero. Por eso, complementamos esa restricción con una segunda donde para cada trabajador en cada día, la suma de

órdenes realizadas en los distintos turnos (X) menos cuatro veces -cantidad de órdenes posibles en un día por trabajador- el valor de la variable D debe ser menor o igual a cero.
Si bien esta restricción permite que no se realicen órdenes pero D esté activada, entre ambas restricciones se cubren la totalidad de los escenarios.

$$\sum_{t=1}^{5}\sum_{o=1}^{c} X_{i,t,o,d} - 4D_{i,d} \leq 0 \; \forall \, i, d$$

3) Ningún trabajador puede trabajar los 5 turnos de un día.

$$\sum_{t=1}^{5}\sum_{0=1}^{c} X_{i,t,o,d} \leq 4 \, \forall \, i, d$$

4) Hay pares de órdenes de trabajo que no pueden ser satisfechas en turnos consecutivos de un trabajador.
Debemos aclarar que cuando hablamos de turnos consecutivos nos referimos a turnos de un mismo día.
A su vez, si A y B son órdenes conflictivas, no se puede realizar la orden A y luego la orden B, pero tampoco se puede realizar primero la orden B y luego la A. En otras palabras, la imposibilidad de realizar las órdenes conflictivas en turnos consecutivos es bidireccional.

$$X_{i,t,o_a,d} + X_{i,t+1,o_b,d} \leq 1 \, \forall \, i, t, d$$
$$X_{i,t,o_b,d} + X_{i,t+1,o_a,d} \leq 1 \, \forall \, i, t, d$$

5) Una órden de trabajo debe tener asignada sus To trabajadores en un mismo turno para poder ser resuelta.
Debemos aclarar que nos estamos refiriendo a contar con los trabajadores necesarios en un mismo turno y día.

$$\frac{\sum_{i=1}^{n} X_{i,t,o,d}}{trabajadores_o} - Z_{o,t,d} = 0 \, \forall \, o, t, d$$

6) Existen algunos pares de órdenes de trabajo correlativas. Un par ordenado de órdenes correlativas A y B, nos indica que si se satisface A, entonces debe satisfacerse B ese mismo día en el turno consecutivo.
Esto no implica que cada vez que se realiza la orden B tengamos que tener realizada la orden A en el turno precedente. La orden B puede realizarse independientemente de la orden A, pero si se realiza la orden A debemos realizar la orden B en el turno siguiente. Tampoco queda definido en la restricción que tengan que intervenir los mismos trabajadores en la realización de ambas órdenes.

$$\frac{\sum_{i=1}^{n} X_{i,t,o_a,d}}{trabajadores_{o_a}} - \frac{\sum_{i=1}^{n} X_{i,t+1,o_b,d}}{trabajadores_{o_b}} \leq 0 \, \forall \, t, d$$

7) Los trabajadores son remunerados según la cantidad de órdenes asignadas por lo que la diferencia entre el trabajador con más órdenes asignadas y el trabajador con menos órdenes no puede ser mayor a 10. Para esto se consideran todos los trabajadores, aún los que no tienen ninguna tarea asignada esta semana.

$$\sum_{d=1}^{6}\sum_{t=1}^{5}\sum_{o=1}^{c} X_{i,t,o,d} - \sum_{d=1}^{6}\sum_{t=1}^{5}\sum_{o=1}^{c} X_{i',t,o,d} \leq 10 \,\forall\, i \neq i'$$

Restricciones adicionales definidas en el código adjunto.

8) Cada trabajador solo puede realizar una orden en cada turno de cada día.

$$\sum_{o=1}^{c} X_{i,t,o,d} \leq 1 \,\forall\, i, d, t$$

9) Función lineal a trozos para definir rangos salariales.
9.1) Definimos para el rango inicial de cada trabajador que la variable $Y_{i,1}$ no puede tomar valores mayores a 5. Esta restricción ya se había realizado en forma implícita al establecer los límites de la variable $Y_{i,r}$.

$$Y_{i,1} \leq 5 \,\forall\, i$$

9.2) Si la W correspondiente al primer rango ($W_{i,1}$) se activa (vale 1), debemos tener completo ese rango salarial ($Y_{i,1}$).

$$Y_{i,1} - (5W_{i_1}) \geq 0 \,\forall\, i$$

9.3) Únicamente podemos acceder al Y de segundo rango ($Y_{i,2}$) si se activó el W que nos indica que el rango precedente se completó ($W_{i,1}$).

$$Y_{i,2} - (5W_{i_1}) \leq 0 \,\forall\, i$$

9.4) Nuevamente, si la W correspondiente al segundo rango ($W_{i,2}$) se activa (vale 1), debemos tener completo ese rango salarial ($Y_{i,2}$).

$$Y_{i,2} - (5W_{i_2}) \geq 0 \,\forall\, i$$

9.5) Únicamente podemos acceder al Y de tercer rango ($Y_{i,3}$) si se activó el W que nos indica que el rango precedente se completó ($W_{i,2}$).

$$Y_{i,3} - (5W_{i_2}) \leq 0 \,\forall\, i$$

9.6) Si la W correspondiente al tercer rango ($W_{i,3}$) se activa (vale 1), debemos tener completo ese rango salarial ($Y_{i,3}$).

$$Y_{i,3} - (5W_{i_3}) \geq 0 \, \forall \, i$$

9.7) Accedemos al Y de cuarto rango ($Y_{i,4}$) si se activó el W que nos indica que el rango precedente se completó ($W_{i,3}$).

$$Y_{i,4} - (5W_{i_3}) \leq 0 \, \forall \, i$$

Debemos tener presente que en el último rango no controlamos el nivel de saturación, con lo cual no hace falta una restricción adicional.

9.8) Si se activa un rango salarial, se requiere que se haya activado el rango precedente. Esto nos permite no saltear rangos y completarlos en forma ordenada.

9.8.1) Si indicamos prendido el rango 2 ($W_{i,2}$) es porque antes habíamos indicado prendido el rango uno ($W_{i,1}$)

$$W_{i,1} - W_{i_2} \geq 0 \, \forall \, i$$

9.8.2) Si indicamos prendido el rango 3 ($W_{i,3}$) es porque antes habíamos indicado prendido el rango dos ($W_{i,2}$).

$$W_{i,2} - W_{i_3} \geq 0 \, \forall \, i$$

9.9) La variable $Y_{i,r}$ captura la cantidad de órdenes realizadas por el trabajador i en el nivel de compensación r. La suma de las órdenes realizadas por cada trabajador a lo largo de todos los turnos y días disponibles debe ser igual a la suma de la cantidad de órdenes registrada en los distintos rangos salariales de dicho trabajador.

$$\sum_{d=1}^{6} \sum_{t=1}^{5} \sum_{o=1}^{c} X_{i,t,d,o} - \sum_{r=1}^{4} Y_{i,r} = 0 \, \forall \, i$$

Implementación: Realizar la implementación de las alternativas propuestas utilizando CPLEX.

Si bien se adjunta el archivo .py con el modelo de optimización correspondiente, a partir del archivo .lp generado con un escenario de 5 trabajadores y 10 tipos de órdenes distintas podemos ver reflejadas en CPLEX las definiciones comentadas anteriormente.

Función objetivo.
Se refleja la búsqueda de maximizar nuestra función objetivo, la cual se compone de una primera instancia con los costos/pagos correspondientes a cada trabajador para cada rango (coeficientes negativos) y una segunda parte donde se suma el beneficio de haber realizado cada orden a partir de un coeficiente que

pondera dicho ingreso prorrateado por la cantidad de trabajadores que son necesarios para realizarla. En nuestro escenario, la orden 0 aporta un ingreso de 5.000 pero requiere de 2 trabajadores, por eso el coeficiente en nuestra función objetivo es 2.500.

Capturamos una parte de nuestra función objetivo.

```
Maximize
 obj1: - 1000 Y_0_0 - 1200 Y_0_1 - 1400 Y_0_2 - 1500 Y_0_3 - 1000 Y_1_0
       - 1200 Y_1_1 - 1400 Y_1_2 - 1500 Y_1_3 - 1000 Y_2_0 - 1200 Y_2_1
       - 1400 Y_2_2 - 1500 Y_2_3 - 1000 Y_3_0 - 1200 Y_3_1 - 1400 Y_3_2
       - 1500 Y_3_3 - 1000 Y_4_0 - 1200 Y_4_1 - 1400 Y_4_2 - 1500 Y_4_3
       + 2500 X_0_0_0_0 + 2500 X_0_0_0_1 + 2500 X_0_0_0_2 + 2500 X_0_0_0_3
       + 2500 X_0_0_0_4 + 2500 X_0_0_0_5 + 4250 X_0_0_1_0 + 4250 X_0_0_1_1
       + 4250 X_0_0_1_2 + 4250 X_0_0_1_3 + 4250 X_0_0_1_4 + 4250 X_0_0_1_5
       + 3250 X_0_0_2_0 + 3250 X_0_0_2_1 + 3250 X_0_0_2_2 + 3250 X_0_0_2_3
       + 3250 X_0_0_2_4 + 3250 X_0_0_2_5 + 3500 X_0_0_3_0 + 3500 X_0_0_3_1
```

Restricciones.

1) No toda orden de trabajo tiene que ser resuelta. Esta restricción no consideramos que requiriera ser definida explícitamente en nuestro código.

2) Ningún trabajador puede trabajar los 6 días de la planificación. Para cada trabajador, la suma del valor de su respectiva variable D para todos los días laborales debe ser menor o igual a 5.

```
D_0_0 + D_0_1 + D_0_2 + D_0_3 + D_0_4 + D_0_5 <= 5
D_1_0 + D_1_1 + D_1_2 + D_1_3 + D_1_4 + D_1_5 <= 5
D_2_0 + D_2_1 + D_2_2 + D_2_3 + D_2_4 + D_2_5 <= 5
D_3_0 + D_3_1 + D_3_2 + D_3_3 + D_3_4 + D_3_5 <= 5
D_4_0 + D_4_1 + D_4_2 + D_4_3 + D_4_4 + D_4_5 <= 5
```

Para implementar esta restricción se debe también capturar la relación existente entre la variable D y la variable X. Para esto en una primera instancia definimos que, para cada trabajador en cada día, la suma de órdenes realizadas en los distintos turnos (X) menos la cantidad de días trabajados (D) debe ser mayor o igual a cero. De esta manera buscamos que la variable binaria D se active (valga 1) cuando el trabajador realiza alguna orden.

Ejemplo para el trabajador 0 en el día 0.

```
- D_0_0 + X_0_0_0_0 + X_0_0_1_0 + X_0_0_2_0 + X_0_0_3_0 + X_0_0_4_0
+ X_0_0_5_0 + X_0_0_6_0 + X_0_0_7_0 + X_0_0_8_0 + X_0_0_9_0 + X_0_1_0_0
+ X_0_1_1_0 + X_0_1_2_0 + X_0_1_3_0 + X_0_1_4_0 + X_0_1_5_0 + X_0_1_6_0
+ X_0_1_7_0 + X_0_1_8_0 + X_0_1_9_0 + X_0_2_0_0 + X_0_2_1_0 + X_0_2_2_0
+ X_0_2_3_0 + X_0_2_4_0 + X_0_2_5_0 + X_0_2_6_0 + X_0_2_7_0 + X_0_2_8_0
+ X_0_2_9_0 + X_0_3_0_0 + X_0_3_1_0 + X_0_3_2_0 + X_0_3_3_0 + X_0_3_4_0
+ X_0_3_5_0 + X_0_3_6_0 + X_0_3_7_0 + X_0_3_8_0 + X_0_3_9_0 + X_0_4_0_0
+ X_0_4_1_0 + X_0_4_2_0 + X_0_4_3_0 + X_0_4_4_0 + X_0_4_5_0 + X_0_4_6_0
+ X_0_4_7_0 + X_0_4_8_0 + X_0_4_9_0 >= 0
```

Sin embargo, esta restricción permite que habiendo realizado alguna orden, la variable D valga cero. Por eso, complementamos esa restricción con una segunda donde para cada trabajador en cada día, la suma de órdenes realizadas en los distintos turnos (X) menos cuatro veces el valor de la variable D debe ser menor o igual a cero. Si bien esta restricción permite que no se realicen órdenes pero D esté activada, entre ambas restricciones se cubren la totalidad de los escenarios.

Ejemplo para el trabajador 0 en el día 0.

```
- 4 D_0_0 + X_0_0_0_0 + X_0_0_1_0 + X_0_0_2_0 + X_0_0_3_0 + X_0_0_4_0
+ X_0_0_5_0 + X_0_0_6_0 + X_0_0_7_0 + X_0_0_8_0 + X_0_0_9_0 + X_0_1_0_0
+ X_0_1_1_0 + X_0_1_2_0 + X_0_1_3_0 + X_0_1_4_0 + X_0_1_5_0 + X_0_1_6_0
+ X_0_1_7_0 + X_0_1_8_0 + X_0_1_9_0 + X_0_2_0_0 + X_0_2_1_0 + X_0_2_2_0
+ X_0_2_3_0 + X_0_2_4_0 + X_0_2_5_0 + X_0_2_6_0 + X_0_2_7_0 + X_0_2_8_0
+ X_0_2_9_0 + X_0_3_0_0 + X_0_3_1_0 + X_0_3_2_0 + X_0_3_3_0 + X_0_3_4_0
+ X_0_3_5_0 + X_0_3_6_0 + X_0_3_7_0 + X_0_3_8_0 + X_0_3_9_0 + X_0_4_0_0
+ X_0_4_1_0 + X_0_4_2_0 + X_0_4_3_0 + X_0_4_4_0 + X_0_4_5_0 + X_0_4_6_0
+ X_0_4_7_0 + X_0_4_8_0 + X_0_4_9_0 <= 0
```

3) Ningún trabajador puede trabajar los 5 turnos de un día. Cada trabajador en cada día puede realizar órdenes en un máximo de 4 turnos.

Ejemplo para el trabajador 0 en el día 0 evaluando todos los turnos y órdenes posibles.

```
X_0_0_0_0 + X_0_0_1_0 + X_0_0_2_0 + X_0_0_3_0 + X_0_0_4_0 + X_0_0_5_0
+ X_0_0_6_0 + X_0_0_7_0 + X_0_0_8_0 + X_0_0_9_0 + X_0_1_0_0 + X_0_1_1_0
+ X_0_1_2_0 + X_0_1_3_0 + X_0_1_4_0 + X_0_1_5_0 + X_0_1_6_0 + X_0_1_7_0
+ X_0_1_8_0 + X_0_1_9_0 + X_0_2_0_0 + X_0_2_1_0 + X_0_2_2_0 + X_0_2_3_0
+ X_0_2_4_0 + X_0_2_5_0 + X_0_2_6_0 + X_0_2_7_0 + X_0_2_8_0 + X_0_2_9_0
+ X_0_3_0_0 + X_0_3_1_0 + X_0_3_2_0 + X_0_3_3_0 + X_0_3_4_0 + X_0_3_5_0
+ X_0_3_6_0 + X_0_3_7_0 + X_0_3_8_0 + X_0_3_9_0 + X_0_4_0_0 + X_0_4_1_0
+ X_0_4_2_0 + X_0_4_3_0 + X_0_4_4_0 + X_0_4_5_0 + X_0_4_6_0 + X_0_4_7_0
+ X_0_4_8_0 + X_0_4_9_0 <= 4
```

4) Hay pares de órdenes de trabajo que no pueden ser satisfechas en turnos consecutivos de un trabajador. Definiendo que la orden 1 y 3 son conflictivas, si el trabajador 0 en el día 0 realiza la orden 1 en el turno 0, no puede realizar la orden 3 en el turno 1. Mismo caso si realiza primero la orden 3 y luego intenta realizar la 1 en el turno consecutivo.

Ejemplo para el trabajador 0 en el día 0 y turno 0.

```
X_0_0_1_0 + X_0_1_3_0 <= 1
X_0_0_3_0 + X_0_1_1_0 <= 1
```

5) Una orden de trabajo debe tener asignada sus To trabajadores en un mismo turno para poder ser resuelta. En esta restricción interviene la variable Z que toma el valor de 1 en caso de realizarse la orden en cuestión en el turno y día definido. Por otro lado, evaluamos la variable X para todos los trabajadores en el mismo turno, día y orden. En este caso, el coeficiente es igual a la inversa de la cantidad de trabajadores necesarios para dicha orden, de forma tal que únicamente obtengamos cero en la resta si se activa esa misma cantidad de Xs, es decir, si intervienen en la realización de esa orden en ese turno y día una cantidad de trabajadores igual a los definidos en To.

Ejemplo para el trabajador 0, en el turno 0 del día 0 evaluando la orden 0 que requiere 2 trabajadores.

```
- Z_0_0_0 + 0.5 X_0_0_0_0 + 0.5 X_1_0_0_0 + 0.5 X_2_0_0_0
+ 0.5 X_3_0_0_0 + 0.5 X_4_0_0_0  = 0
```

6) Existen algunos pares de órdenes de trabajo correlativas. Un par ordenado de órdenes correlativas A y B, nos indica que si se satisface A, entonces debe satisfacerse B ese mismo día en el turno consecutivo. En este caso, los coeficientes de la desigualdad son la inversa de la cantidad de trabajadores necesarios de cada orden que interviene. Los coeficientes de la orden B son negativos para permitirnos realizar la orden B sin necesidad de realizar la orden A previamente. Por otro lado, si la orden A se realiza, al tener coeficientes positivos, nos obliga a hacer la orden B para poder cumplir con la desigualdad planteada.

Ejemplo en el día 0 para las órdenes correlativas 1-2 en el turno 0-1 evaluando todos los empleados.

```
0.5 X_0_0_1_0 - 0.5 X_0_1_2_0 + 0.5 X_1_0_1_0 - 0.5 X_1_1_2_0
+ 0.5 X_2_0_1_0 - 0.5 X_2_1_2_0 + 0.5 X_3_0_1_0 - 0.5 X_3_1_2_0
+ 0.5 X_4_0_1_0 - 0.5 X_4_1_2_0 <= 0
```

7) Los trabajadores son remunerados según la cantidad de órdenes asignadas por lo que la diferencia entre el trabajador con más órdenes asignadas y el trabajador con menos órdenes no puede ser mayor a 10. Para esto se consideran todos los trabajadores, aún los que no tienen ninguna tarea asignada esta semana. La restricción se plantea comparando, de a pares, todos los trabajadores entre sí, buscando que en ningún caso la diferencia entre la cantidad de órdenes realizada por cada uno sea mayor a 10.

Fragmento donde se compara la cantidad de órdenes que realiza el trabajador 0 y el trabajador 1 durante todos los días y turnos disponibles.

```
+ X_0_4_0_0 + X_0_4_0_1 + X_0_4_0_2 + X_0_4_0_3 + X_0_4_0_4 + X_0_4_0_5
+ X_0_4_1_0 + X_0_4_1_1 + X_0_4_1_2 + X_0_4_1_3 + X_0_4_1_4 + X_0_4_1_5
+ X_0_4_2_0 + X_0_4_2_1 + X_0_4_2_2 + X_0_4_2_3 + X_0_4_2_4 + X_0_4_2_5
+ X_0_4_3_0 + X_0_4_3_1 + X_0_4_3_2 + X_0_4_3_3 + X_0_4_3_4 + X_0_4_3_5
+ X_0_4_4_0 + X_0_4_4_1 + X_0_4_4_2 + X_0_4_4_3 + X_0_4_4_4 + X_0_4_4_5
+ X_0_4_5_0 + X_0_4_5_1 + X_0_4_5_2 + X_0_4_5_3 + X_0_4_5_4 + X_0_4_5_5
+ X_0_4_6_0 + X_0_4_6_1 + X_0_4_6_2 + X_0_4_6_3 + X_0_4_6_4 + X_0_4_6_5
+ X_0_4_7_0 + X_0_4_7_1 + X_0_4_7_2 + X_0_4_7_3 + X_0_4_7_4 + X_0_4_7_5
+ X_0_4_8_0 + X_0_4_8_1 + X_0_4_8_2 + X_0_4_8_3 + X_0_4_8_4 + X_0_4_8_5
+ X_0_4_9_0 + X_0_4_9_1 + X_0_4_9_2 + X_0_4_9_3 + X_0_4_9_4 + X_0_4_9_5
- X_1_0_0_0 - X_1_0_0_1 - X_1_0_0_2 - X_1_0_0_3 - X_1_0_0_4 - X_1_0_0_5
- X_1_0_1_0 - X_1_0_1_1 - X_1_0_1_2 - X_1_0_1_3 - X_1_0_1_4 - X_1_0_1_5
- X_1_0_2_0 - X_1_0_2_1 - X_1_0_2_2 - X_1_0_2_3 - X_1_0_2_4 - X_1_0_2_5
- X_1_0_3_0 - X_1_0_3_1 - X_1_0_3_2 - X_1_0_3_3 - X_1_0_3_4 - X_1_0_3_5
- X_1_0_4_0 - X_1_0_4_1 - X_1_0_4_2 - X_1_0_4_3 - X_1_0_4_4 - X_1_0_4_5
- X_1_0_5_0 - X_1_0_5_1 - X_1_0_5_2 - X_1_0_5_3 - X_1_0_5_4 - X_1_0_5_5
- X_1_0_6_0 - X_1_0_6_1 - X_1_0_6_2 - X_1_0_6_3 - X_1_0_6_4 - X_1_0_6_5
- X_1_0_7_0 - X_1_0_7_1 - X_1_0_7_2 - X_1_0_7_3 - X_1_0_7_4 - X_1_0_7_5
- X_1_0_8_0 - X_1_0_8_1 - X_1_0_8_2 - X_1_0_8_3 - X_1_0_8_4 - X_1_0_8_5
- X_1_0_9_0 - X_1_0_9_1 - X_1_0_9_2 - X_1_0_9_3 - X_1_0_9_4 - X_1_0_9_5
```

```
- X_1_4_0_0 - X_1_4_0_1 - X_1_4_0_2 - X_1_4_0_3 - X_1_4_0_4 - X_1_4_0_5
- X_1_4_1_0 - X_1_4_1_1 - X_1_4_1_2 - X_1_4_1_3 - X_1_4_1_4 - X_1_4_1_5
- X_1_4_2_0 - X_1_4_2_1 - X_1_4_2_2 - X_1_4_2_3 - X_1_4_2_4 - X_1_4_2_5
- X_1_4_3_0 - X_1_4_3_1 - X_1_4_3_2 - X_1_4_3_3 - X_1_4_3_4 - X_1_4_3_5
- X_1_4_4_0 - X_1_4_4_1 - X_1_4_4_2 - X_1_4_4_3 - X_1_4_4_4 - X_1_4_4_5
- X_1_4_5_0 - X_1_4_5_1 - X_1_4_5_2 - X_1_4_5_3 - X_1_4_5_4 - X_1_4_5_5
- X_1_4_6_0 - X_1_4_6_1 - X_1_4_6_2 - X_1_4_6_3 - X_1_4_6_4 - X_1_4_6_5
- X_1_4_7_0 - X_1_4_7_1 - X_1_4_7_2 - X_1_4_7_3 - X_1_4_7_4 - X_1_4_7_5
- X_1_4_8_0 - X_1_4_8_1 - X_1_4_8_2 - X_1_4_8_3 - X_1_4_8_4 - X_1_4_8_5
- X_1_4_9_0 - X_1_4_9_1 - X_1_4_9_2 - X_1_4_9_3 - X_1_4_9_4 - X_1_4_9_5
<= 10
```

8) Cada trabajador solo puede realizar una orden en cada turno de cada día. Si observamos un turno de un día de un trabajador, solo debemos encontrar que realizó una orden como máximo, pudiendo no haber hecho alguna.

Ejemplo del trabajador 0 en el turno 0 del día 0, evaluando las 10 órdenes listadas.

```
X_0_0_0_0 + X_0_0_1_0 + X_0_0_2_0 + X_0_0_3_0 + X_0_0_4_0 + X_0_0_5_0
+ X_0_0_6_0 + X_0_0_7_0 + X_0_0_8_0 + X_0_0_9_0 <= 1
```

9) Función lineal a trozos para definir rangos salariales.

9.1) Definimos para el rango cero de cada trabajador que la variable $Y_{i,0}$ no puede tomar valores mayores a 5. Esta restricción ya se había realizado en forma implícita al definir los límites de la variable $Y_{i,r}$.

```
Y_0_0 <= 5
Y_1_0 <= 5
Y_2_0 <= 5
Y_3_0 <= 5
Y_4_0 <= 5
```

9.1) Si la W correspondiente al primer rango ($W_{i,0}$) se activa (vale 1), debemos tener completo ese rango salarial ($Y_{i,0}$).

```
- 5 W_0_0 + Y_0_0 >= 0
- 5 W_1_0 + Y_1_0 >= 0
- 5 W_2_0 + Y_2_0 >= 0
- 5 W_3_0 + Y_3_0 >= 0
- 5 W_4_0 + Y_4_0 >= 0
```

9.3) Únicamente podemos acceder al Y de segundo rango ($Y_{i,1}$) si se activó el W que nos indica que el rango precedente se completó ($W_{i,0}$).

```
- 5 W_0_0 + Y_0_1 <= 0
- 5 W_1_0 + Y_1_1 <= 0
- 5 W_2_0 + Y_2_1 <= 0
- 5 W_3_0 + Y_3_1 <= 0
- 5 W_4_0 + Y_4_1 <= 0
```

9.4) Nuevamente, si la W correspondiente al segundo rango ($W_{i,1}$) se activa (vale 1), debemos tener completo ese rango salarial ($Y_{i,1}$).

```
- 5 W_0_1 + Y_0_1 >= 0
- 5 W_1_1 + Y_1_1 >= 0
- 5 W_2_1 + Y_2_1 >= 0
- 5 W_3_1 + Y_3_1 >= 0
- 5 W_4_1 + Y_4_1 >= 0
```

9.5) Únicamente podemos acceder al Y de tercer rango ($Y_{i,2}$) si se activó el W que nos indica que el rango precedente se completó ($W_{i,1}$).

```
- 5 W_0_1 + Y_0_2 <= 0
- 5 W_1_1 + Y_1_2 <= 0
- 5 W_2_1 + Y_2_2 <= 0
- 5 W_3_1 + Y_3_2 <= 0
- 5 W_4_1 + Y_4_2 <= 0
```

9.6) Si la W correspondiente al tercer rango ($W_{i,2}$) se activa (vale 1), debemos tener completo ese rango salarial ($Y_{i,2}$).

```
- 5 W_0_2 + Y_0_2 >= 0
- 5 W_1_2 + Y_1_2 >= 0
- 5 W_2_2 + Y_2_2 >= 0
- 5 W_3_2 + Y_3_2 >= 0
- 5 W_4_2 + Y_4_2 >= 0
```

9.7) Accedemos al Y de cuarto rango ($Y_{i,3}$) si se activó el W que nos indica que el rango precedente se completó ($W_{i,2}$).

Debemos tener presente que en el último rango no controlamos el nivel de saturación, con lo cual no hace falta una restricción adicional.

```
- 5 W_0_2 + Y_0_3 <= 0
- 5 W_1_2 + Y_1_3 <= 0
- 5 W_2_2 + Y_2_3 <= 0
- 5 W_3_2 + Y_3_3 <= 0
- 5 W_4_2 + Y_4_3 <= 0
```

9.8) Si se activa un rango salarial, se requiere que se haya activado el rango precedente. Esto nos permite no saltear rangos y completarlos en forma ordenada.

9.8.1) Si indicamos prendido el rango 1 ($W_{i,1}$) es porque antes habíamos indicado prendido el rango cero ($W_{i,0}$).

Ejemplo para el trabajador 0.

```
W_0_0 - W_0_1 >= 0
```

9.8.2) Si indicamos prendido el rango 2 ($W_{i,2}$) es porque antes habíamos indicado prendido el rango precedente ($W_{i,1}$).

Ejemplo para el trabajador 0.

```
W_0_1 - W_0_2 >= 0
```

9.9) La suma de las órdenes realizadas por cada trabajador a lo largo de todos los turnos y días disponibles debe ser igual a la suma de la cantidad de órdenes registrada en los distintos rangos salariales de dicho trabajador. En este caso, para cada trabajador evaluamos cuántas órdenes realizó en la semana (X) y le restamos la suma de la cantidad de órdenes que se registraron en los distintos rangos salariales (Y).

Ejemplo para el trabajador 0.

```
- Y_0_0 - Y_0_1 - Y_0_2 - Y_0_3 + X_0_0_0_0 + X_0_0_0_1 + X_0_0_0_2
+ X_0_0_0_3 + X_0_0_0_4 + X_0_0_0_5 + X_0_0_1_0 + X_0_0_1_1 + X_0_0_1_2
+ X_0_0_1_3 + X_0_0_1_4 + X_0_0_1_5 + X_0_0_2_0 + X_0_0_2_1 + X_0_0_2_2
+ X_0_0_2_3 + X_0_0_2_4 + X_0_0_2_5 + X_0_0_3_0 + X_0_0_3_1 + X_0_0_3_2
+ X_0_0_3_3 + X_0_0_3_4 + X_0_0_3_5 + X_0_0_4_0 + X_0_0_4_1 + X_0_0_4_2
+ X_0_0_4_3 + X_0_0_4_4 + X_0_0_4_5 + X_0_0_5_0 + X_0_0_5_1 + X_0_0_5_2
+ X_0_0_5_3 + X_0_0_5_4 + X_0_0_5_5 + X_0_0_6_0 + X_0_0_6_1 + X_0_0_6_2
```

```
+ X_0_4_5_3 + X_0_4_5_4 + X_0_4_5_5 + X_0_4_6_0 + X_0_4_6_1 + X_0_4_6_2
+ X_0_4_6_3 + X_0_4_6_4 + X_0_4_6_5 + X_0_4_7_0 + X_0_4_7_1 + X_0_4_7_2
+ X_0_4_7_3 + X_0_4_7_4 + X_0_4_7_5 + X_0_4_8_0 + X_0_4_8_1 + X_0_4_8_2
+ X_0_4_8_3 + X_0_4_8_4 + X_0_4_8_5 + X_0_4_9_0 + X_0_4_9_1 + X_0_4_9_2
+ X_0_4_9_3 + X_0_4_9_4 + X_0_4_9_5  = 0
```

Experimentación: Realizar experimentación sobre las diferentes alternativas para lograr una discusión tanto cuantitativa como cualitativa de los resultados obtenidos.

Una vez finalizado el código, nos propusimos crear diferentes instancias para realizar varios experimentos. Para eso, creamos una función en Python cuyo input son los distintos parámetros que forman los datos y el output es el .txt con el formato esperable.
A medida que íbamos experimentando con mayor cantidad de órdenes y trabajadores, nos dimos cuenta que cada vez se demoraba más en llegar a una solución óptima. Es por eso que decidimos sacrificar un poco de precisión en el valor final de la función objetivo en pos de una eficiencia en tiempo de cómputo a partir de aumentar la tolerancia de la Mip Gap.
Estos son los resultados obtenidos para cada una de las instancias:

Cantidad de trabajadores	Cantidad de órdenes	Órdenes conflictivas	Órdenes correlativas	Tiempo de cómputo (segs)
40	10	3	1	339
20	15	4	3	165
20	20	5	4	58
30	30	6	5	1008
50	40	12	10	665

Conclusiones.

Mediante la formulación bajo un modelo de programación lineal entera con implementación en IBM CPLEX ILOG Python API se logró resolver el problema planteado con distinto grado de complejidad.
Más allá del control que realizamos en cuanto a definición de variables, cumplimiento de restricciones y cálculo de la función objetivo, lo que sobresale al revisar los resultados en cada optimización es la búsqueda

que hace el modelo por no alcanzar los rangos salariales más altos, tratando siempre de distribuir lo máximo posible la ejecución de las órdenes entre todos los trabajadores. Este punto tiene lógica si consideramos que se busca maximizar una ganancia neta y concentrar órdenes en pocos trabajadores implica alcanzar rangos salariales superiores obteniendo un costo por orden mayor.
Por último, también resalta la tendencia a priorizar las órdenes que aportan un mayor beneficio por trabajador necesario, en línea con la función objetivo definida.