Documentación Datatón

Juan Pablo Gutierrez, Fabian Andrés Ruiz, Heldert Villegas

8 de noviembre de 2023

1. Introducción

La aproximación que escogimos para tratar con este problema se basa en problemas de programación lineal, especialmente aquellos de programación entera (integer programming) y optimización combinatoria. Estos son problemas de programación lineal en el cual las variables están restringidas a ser enteras y/o binarias [**Int**]. En la tesis "The staff scheduling problem: a general model and applications"[**Toaz**] y en el articulo .ªn integer programming approach for the physician rostering problem"[**Phy**], muestran de manera práctica como plantear los problemas de planificiación de manera lineal.

Para utilizar el método lo primero que se debe hacer es plantear el sistema lineal, esto es declarar las variables de decisión del problema, la o las funciones a optimizar y las restricciones del problema.

2. Planteamiento Matemático

Antes de plantear el problema vale la pena aclara que como el problema no requiere que administremos los empleados entre las sucursales, sino solo como planificarlos dentro de las mismas, se puede tratar el problema de planificación de cada una de las sucursales de manera independiente y análoga. De la misma manera el problema de los sábados es independiente al de toda la semana en cada una de las sucursales.

2.1. Variables

Primero que todo vamos a considerar la familia de índices que vamos a trabajar:

$d \in \{0, 1, 2, 3, 4, 5\}$, denota los días de la semana, donde 0 es el lunes y 5 el sábado.

$f \in \{0, ..., 49\}$, denota la f-ésima franja del día.

$e \in \{1, ..., c, c+1, .., c+m\}$, denota el e-ésimo empleado de la sucursal. Los primeros c son de tiempo completo y los otros m de medio tiempo.

Las variable de decisión principal del problema es

$$Pd_{d,f}, \text{ es la diferencia positiva entre la demanda y la cantidad trabajadores en el día } d \text{ en la franja } f.$$

Se toma solamente la diferencia positiva pues si en alguna franja la cantidad de trabajadores fuese mayor a la demanda no nos interesa que esto compense a las franjas en las cuales la demanda supere a la cantidad de trabajadores.

Las demás variables del problema son:

$d_{d,f}$, la demanda en el día d en la franja f.

$$w_{d,e,f} = \begin{cases} 1, & \text{si el trabajador } e \text{ trabaja en la franja } f \text{ del dia } d. \\ 0, & \text{en caso contrario.} \end{cases}$$

$$b_{d,e,f} = \begin{cases} 1, \text{ si el trabajador } e \text{ descansa en la franja } f \text{ del día } d. \\ 0, \text{ en caso contrario.} \end{cases}$$

$$l_{d,e,f} = \begin{cases} 1, \text{ si el trabajador } e \text{ almueza en la franja } f \text{ del dia } d. \\ 0, \text{ en caso contrario.} \end{cases}$$

$$a_{e,f} = \begin{cases} 1, \text{ si el trabajador } e \text{ está activo en la franja } f. \\ 0, \text{ en caso contrario.} \end{cases}$$

$$Ea_{e,f} = \begin{cases} 1, \text{ si el trabajador } e \text{ termina su turno en la franja } f. \\ 0, \text{ en caso contrario.} \end{cases}$$

$$El_{e,f} = \begin{cases} 1, \text{ si el trabajador } e \text{ termina su almuerzo en la franja } f. \\ 0, \text{ en caso contrario.} \end{cases}$$

Notese como las últimas 3 variables no dependen del día, ya que el periodo de actividad de los trabajadores son constantes durante toda la semana. Para referirnos a estas variables en los sábados simplemente usaremos una tilde ($\tilde{a}_{e,f}$, $\widetilde{Ea}_{e,f}$, $\widetilde{El}_{e,f}$).

2.2. Función Objetivo

Ya que queremos utilizar las técnicas de la programación lineal necesitamos que nuestra función objetivo sea una combinación lineal de nuestras variables. En nuestro caso esta será

$$\text{mín} \sum_{d,f} Pd_{d,f} \tag{1}$$

Utilizamos esta ya que deseamos que la cantidad de trabajadores sea lo más parecido a la demanda en la sucursal en cada una de las franjas.

2.3. Restricciones

2.3.1. Restricciones generales

Primero presentamos las restricciones que se mantienen de lunes a sábado.

$$Pd_{d,f} \geq d_{d,f} - \sum_{e} w_{d,e,f} \tag{2}$$

$$\sum_{f=0}^{4} b_{d,e,f} = 0, \text{ para todo } d \text{ y } e. \tag{3}$$

$$4b_{d,e,f} \leq \sum_{i=1}^{4} (w_{d,e,f-i} + l_{d,e,f-i}), \text{ para cada } d, e \text{ y } f. \tag{4}$$

$$4Ea_{e,f} \leq \sum_{i=1}^{4} (w_{d,e,f-i} + l_{d,e,f-i}), \text{ para cada } d, e \text{ y } f. \tag{5}$$

$$\sum_{i=0}^{8} w_{d,e,f+i} \leq 8, \text{ para cada } d, e \text{ y } f. \tag{6}$$

$$a_{e,f} \geq w_{d,e,f}, \; a_{e,f} \geq b_{d,e,f}, \; a_{e,f} \geq l_{d,e,f}, \text{ para cada } d \text{ , } e \text{ y } f. \tag{7}$$

$$a_{e,f} \leq w_{d,e,f} + b_{d,e,f} + l_{d,e,f}, \text{ para cada } d \text{ , } e \text{ y } f. \tag{8}$$

$$a_{d,e,f} \leq a_{d,e,f+1} + Ea_{e,f}, \text{ para cada } d, e \text{ y } f. \tag{9}$$

$$\sum_f Ea_{e,f} = 1, \text{ para todo } e. \tag{10}$$

$$w_{d,e,f} \geq Ea_{e,f}, \text{para cada } d, e \text{ y } f. \tag{11}$$

$$w_{d,e,f} + b_{d,e,f} + l_{d,e,f} \leq 1, \text{ para cada } d, e \text{ y } f. \tag{12}$$

$$d_{d,f} \cdot \sum_e w_{d,e,f} \geq d_{d,f}, \text{ para cada } d \text{ y } f. \tag{13}$$

$$\sum_f w_{d,e,f} + b_{d,e,f} = 16, \text{ para todo } d \text{ cuando } c+1 \leq e \leq c+m \tag{14}$$

En la siguiente tabla se explica brevemente cada una de las restricciones.

Ecuación	**Significado**
(2)	La diferencia positiva es siempre mayor a la diferencia entre la demanda y la cantidad de empleados trabajando.
(3)	Las primeras 4 franjas deben ser de trabajo.
(4) y (5)	Antes de un descanso o de el fin de la jornada deben haber 4 franjas de trabajo.
(6)	Solo pueden haber 8 bloques de trabajo consecutivos.
(7) y (8)	Estar activo significa estar trabajando, descansando o almorzando.
(9)	Esta condición asegura la continuidad del bloque de actividad.
(10)	Solo se termina la actividad en un bloque del día.
(11)	El último bloque se está trabajando.
(12)	Solo hay un estado activo simultáneamente.
(13)	Cuando hay demanda debe haber al menos un empleado trabajando.
(14)	Los trabajadores de medio tiempo solo están activos 16 franjas.

Cuadro 1: Significado de las ecuaciones generales

2.3.2. Restricciones exclusivas entre semana

En el caso de la semana existen las restricciones adicionales correspondientes a la longitud de las jornadas de los trabajadores de tiempo completo y a las restricciones relacionadas con el tiempo de almuerzo.

$$\sum_f w_{d,e,f} + b_{d,e,f} = 28, \text{ para todo } d \text{ cuando } 1 \leq e \leq c \tag{15}$$

$$4l_{d,e,f} \leq \sum_{i=1}^{4} (w_{d,e,f-i} + l_{d,e,f-i}), \text{ para cada } d, e \text{ y } f. \tag{16}$$

$$l_{d,e,f} \leq l_{d,e,f+1} + El_{e,f}, \text{para cada } d \text{ y } f \text{ si } 1 \leq e \leq c. \tag{17}$$

$$\sum_f l_{d,e,f} = 6, \text{ para cada } d \text{ si } 1 \leq e \leq c. \tag{18}$$

$$\sum_{f} El_{e,f} = 1, \text{ para todo } 1 \leq e \leq c. \tag{19}$$

$$\sum_{i=0}^{15} l_{d,e,i} = 0, \text{ para todo } d \text{ y } 1 \leq e \leq c. \tag{20}$$

$$\sum_{i=31}^{48} l_{d,e,i} = 0, \text{ para todo } d \text{ y } 1 \leq e \leq c. \tag{21}$$

Ahora la tabla con el significado de las mismas

Ecuación	**Significado**
(15)	Todo empleado de tiempo completo trabaja exactamente 28 franjas diarias.
(16)	El almuerzo solo puede iniciar después de 4 franjas de trabajo.
(17)	Esta condición garantiza la continuidad del bloque de almuerzo.
(18)	Los empleados de tiempo completo solo tienen 6 franjas de almuerzo.
(19)	Solo se termina el almuerzo en un bloque del día.
(20) y (21)	No hay almuerzo fuera de las franjas 16 - 30.

Cuadro 2: Significado de las ecuaciones especificas entre semana

2.3.3. Restricciones exclusivas sabado

Lo único que está pendiente del sábado con respecto a las generales es la duración de la jornada para los trabajadores de tiempo completo, es decir

$$\sum_{f} w_{d,e,f} + b_{d,e,f} = 20, \text{ para todo } d \text{ cuando } 1 \leq e \leq c \tag{22}$$

3. Implementación

Dado que todo el planteamiento se basa en programación entera usamos la librería por excelencia para programación entera en `Python`, `pulp`. Además se usó `pandas` para manejar el flujo de datos desde y hacia los archivos `csv`.

Para recorrer todos los códigos se hará desde los más independientes a los más robustos.

3.1. Recursos y miscelanea:

En esta sección cubrimos los scripts que se son necesarios para ejecutar y visualizar las funciones optimizadores principales.

1. `import_file.import_file_etapa2()` :
   Función que devuelve un diccionario cuyas claves son los códigos de las sucursales y los valores son los diccionarios que se indican en el cuadro 3.

Clave	Tipo	Descripción
TM	Lista	Códigos de los trabajadores a tiempo completo.
MT	Lista	Códigos de los trabajadores a medio tiempo.
dias	Conjunto	Dias de la semana.
demandas	Lista	Lista con las demandas por hora de la sucursal.

Cuadro 3: Diccionario que retorna la función para cada una de las sucursales.

2. `print_schedules.print_3d_schedule_channels(work_matrix, lunch_matrix, break_matrix)` :
   Método que toma las matrices `work_matrix`, `lunch_matrix` y `break_matrix`, de dimensiones `dias` × `numero_empleados` × `numero_franjas` y las imprime en consola en un formato legible. (verde para trabajo, azul para descanso y amarillo para almuerzo).

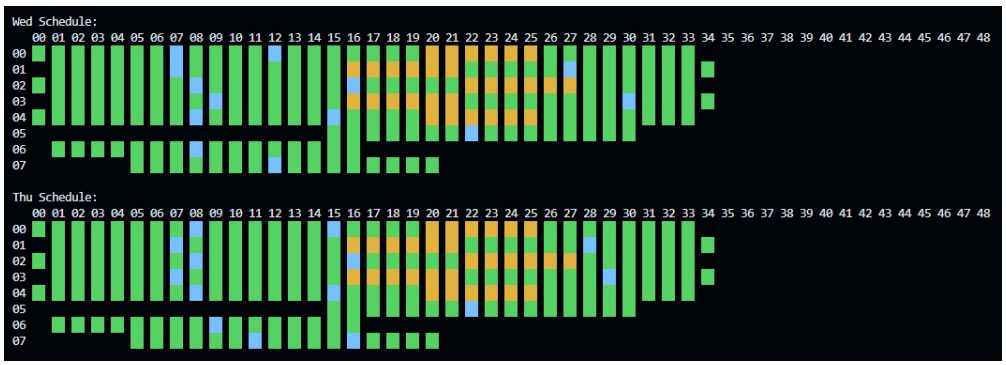

Figura 1: Impresión de los horarios para miércoles y jueves con el método.

3. `print_schedules.print_2d_schedule_channels(work_matrix, lunch_matrix)` :
   Método análogo a `print_3d_schedule_channels`, con solo dos entradas para imprimir el horario de los sábados.

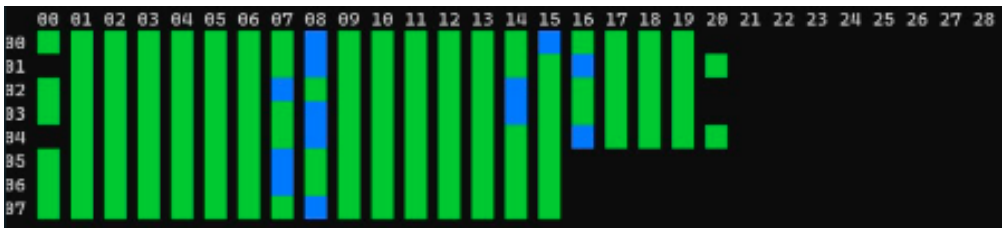

Figura 2: Impresión de un horario de sábado con la función.

4. `print_schedules.generate_week_schedule_csv(work_matrix, break_matrix, lunch_matrix, employee_ids, dates, suc_code,filename="schedule.csv"):`
   Función que retorna un `csv` cuyas columnas son `suc_cod`, `documento`, `fecha`, `hora`, `estado` y `hora_franja`; cuyas filas corresponden a el estado de un trabajador en la franja y dia correspondiente.

5. `print_schedules.generate_saturday_schedule_csv(work_matrix, break_matrix, employee_ids, dates, suc_code, filename="schedule.csv"):`
   Función análoga a `generate_week_schedule_csv`, con solo dos entradas de matrices para exportar el horario de los sábados.

6. `merge_csv.merge_csv_files():`
   Método para combinar los `csv` de los días de la con el de los sábados.

3.2. Implementación principal:

1. `etapa2_week.solve_week_optimization(demand_workers, sucursal_id)`: Función que declara el problema lineal para los días de la semana y lo soluciona tomando en cuenta el un vector de demanda y el identificador de la sucursal. Es posible cambiar el solver que es utilizado al solucionar el problema de optimización, en las pruebas fueron usados `CPLEX`, `PULP_CBC_CMD` y `COIN_CMD`. El solver por defecto lo dejamos como `CPLEX` ya que es el que mejor se desempeña tanto en windows como en linux.

```
pd = pulp.LpVariable.dicts(
    "Positive Difference",
    [(x, z) for x in range(DAYS) for z in range(SCHEDULE)],
    lowBound=0,
    cat=pulp.LpInteger,
)
```

Figura 3: Declaración de variable objetivo dentro de la función.

```
# Objective function: Minimize pd.
prob += pulp.lpSum(pd[(x, i)] for x in range(DAYS) for i in range(SCHEDULE))
```

Figura 4: Declaración de función objetivo dentro de la función.

```
# (No breaks in the first WORK_BLOCKS_BEFORE_BREAK blocks.)
prob += (
    pulp.lpSum(b[(d, k, i)] for i in range(WORK_BLOCKS_BEFORE_BREAK)) == 0
)
```

Figura 5: Declaración de restricción (4) dentro de la función.

2. `etapa2_sat.solve_saturday_optimization(demand_workers, sucursal_id)` :
Función análoga a `solve_week_optimization` que resuelve el problema de optimización dada la demanda del sábado en la sucursal indicada.

3.3. Ejecución:

1. `main.solve_optimization()` :
Método que llama las demás funciones, exporta el `csv` unificado y el valor de la función objetivo final.