Asignación de Cuadrillas

Trabajo Práctico para la asignatura Investigación Operativa

Nicolás Rozenberg, Santiago Eliges

1er Cuatrimestre 2024

Departamento de Computación,
FCEyN,
UBA

1: Modelado

1 Convenciones

- La letra **i** siempre indexará a un trabajador. Por lo tanto, salvo indicación, recorrerá todo el conjunto de trabajadores en sumatorias y cuantificadores.
- La letra **j** siempre indexará a una orden a realizar. Por lo tanto, salvo indicación, recorrerá todo el conjunto de órdenes en sumatorias y cuantificadores.
- La letra **d** siempre indexará a un día. Por lo tanto, salvo indicación, recorrerá todo el rango entero de 1 a 6 en sumatorias y cuantificadores.
- La letra **k** siempre indexará a un turno. Por lo tanto, salvo indicación, recorrerá todo el rango entero de 1 a 5 en sumatorias y cuantificadores.

2 Variables

$$A_{ijd} = \begin{cases} 1 & \text{si} \quad \text{El trabajador i fue asignado a la orden j en el dia d} \\ 0 & si \qquad\qquad\qquad\qquad\qquad\qquad cc \end{cases}$$

$$B_{jdk} = \begin{cases} 1 & \text{si} \quad \text{La j-esima orden fue asignada al turno k en el dia d} \\ 0 & si \qquad\qquad\qquad\qquad\qquad\qquad cc \end{cases}$$

$$Tr_{id} = \begin{cases} 1 & \text{si} \quad \text{El i-esimo trabajador trabajó en el dia d} \\ 0 & si \qquad\qquad\qquad\qquad\qquad\qquad cc \end{cases}$$

$$\delta_j = \begin{cases} 1 & \text{si} \quad \text{La orden j es realizada} \\ 0 & si \qquad\qquad\qquad\qquad cc \end{cases}$$

$X_{i1}, X_{i2}, X_{i3}, X_{i4}$ Usadas para hacer las cotas de lo que se le debe pagar a los trabajadores por las tareas
W_{i1}, W_{i2}, W_{i3} Usadas para hacer cotas

3 Función objetivo

$$\max \sum_{j,d,k} B_{jdk} b_j - \sum_{i} (1000X_{i1} + 1200X_{i2} + 1400X_{i3} + 1600X_{i4})$$

4 Restricciones

$A_{ijd}, B_{jdk}, Tr_{id}, \delta_j, W_{i1}, W_{i2}, W_{i3}, W_{i4} \in \{0,1\}\ \forall i,j,d,k$
$X_{i1}, X_{i2}, X_{i3}, X_{i4} \in \mathbb{N}_0$

$$\sum_{k,d} B_{jdk} = \delta_j \quad \forall j$$

$$\sum_{i,d} A_{i,j,d} = Tr_j \delta_j \quad \forall j$$

"Una orden fue asignada si y solo si cuenta con la cantidad de trabajadores Tr_i que necesita"

$$A_{ijd} \leq \sum_{k} B_{jdk} \quad \forall i,j,d$$

"Si el trabajador i trabaja en la orden j en el dia d, entonces la orden es asignada al día d"

$$\sum_{d} A_{i,j,d} <= 1 \quad \forall i,j$$

"Un trabajador es asignado a una orden en particular a lo sumo un dia"

$$\sum_{j} A_{i,j,d} \leq 4Tr_{id} \quad \forall i,d$$

$$Tr_{id} \leq \sum_{j} A_{i,j,d} \quad \forall i,d$$

"No hay ningun trabajador que ocupe todos los turnos de un dia "

$$\sum_{d} Tr_{i,d} \leq 5 \quad \forall i$$

"Ningun trabajador puede trabajar todos los días"

$A_{ij_1d} + B_{j_1dk} + A_{ij_2d} + B_{j_2dk+1} \leq 3 \quad \forall i,d,(j_1,j_2) \in OrdenesConflictivas\ 1 \leq k \leq 4$
"No existen turnos consecutivos en los que se hayan asignado al mismo trabajador en ordenes conflictivas"

$A_{ij_1d} + B_{j_1dk} + A_{ij_2d} + B_{j_2dk} \leq 3 \quad \forall i,d,k,j_1 \neq j_2 \in Ordenes$
"Un trabajador no puede trabajar en dos órdenes distintas en un mismo turno"

$B_{j_1,d,k} \leq B_{j_2,d,k+1} \quad \forall d, 1 \leq k \leq 4, (j_1,j_2) \in OrdenesConsecutivas$
"Si se realiza la orden j1 en el turno k del día d, entonces la orden j2 se realiza en el turno k+1 del d"

$B_{j_1,d,5} = 0 \quad \forall d, (j_1, j_2) \in OrdenesConsecutivas$

"Ninguna primera orden del conjunto de pares de OrdenesConsecutivas puede ser asignada al ultimo turno de un día"

$$\sum_{j,d} A_{i_1,j,d} - \sum_{j,d} A_{i_2,j,d} \leq 8 \quad \forall i_1, i_2$$

$$-8 \leq \sum_{j,d} A_{i_1,j,d} - \sum_{j,d} A_{i_2,j,d} \quad \forall i_1, i_2$$

"No existe una diferencia mayor a 8 entre la cantidad de turnos asignados a cada trabajador"

$$\sum_{d,k} B_{j,d,k} \leq 1 \quad \forall j$$

"La orden fue asignada una sola vez"

$$\sum_{j,d} Ai, j, d = X_{i1} + X_{i2} + X_{i3} + X_{i4} \quad \forall i$$

$$5W_{i1} \leq X_{i1} \leq 5 \quad \forall i$$

$$5W_{i2} \leq X_{i2} \leq 5W_{i1} \quad \forall i$$

$$5W_{i3} \leq X_{i3} \leq 5W_{i2} \quad \forall i$$

$$0 \leq X_{i4} \leq 5W_{i3} \quad \forall i$$

"Distinguimos a la cantidad de ordenes que le fueron asignadas a cada trabajador en intervalos"

5 Restricciones Deseables

$A_{i_1,j,d} + A_{i_2,j,d} \leq 1 \quad \forall j, d, (i_1, i_2) \in TrabajadoresEnConflicto$

"Ninguno de los trabajadores en conflicto están en una misma orden"

$$\sum_{d} A_{i,j_1,d} + \sum_{d} A_{i,j_2,d} \leq 1 \quad \forall i, (j_1, j_2) \in OrdenesRepetitivas$$

"Un trabajador no es asignado a dos ordenes repetitivas"

6 Modificación para análisis de perdida por los conflictos entre trabajadores

6.1 Variables Nuevas

$$Tc_{i_1,i_2,j} = \begin{cases} 1 & \text{Los trabajadores } i_1 \text{ e } i_2 \text{ fueron asignados a la orden j} \\ 0 & cc \quad \forall j, d, (i_1, i_2) \in TrabajadoresEnConflicto \end{cases}$$

6.2 Función objetivo

$$\max \sum_{j,d,k} B_{jdk} b_i - \sum_i 1000X_{i1} + 1200X_{i2} + 1400X_{i3} + 1500X_{i4}$$

$$- \sum_{\substack{j,(i_1,i_2)\in \\ TrabajadoresEnConflicto}} \alpha Tc_{i_1,i_2,j}$$

6.3 Restricciones Nuevas

$A_{i_1,j,d} + A_{i_2,j,d} \leq 1 + Tc_{i_1,i_2,j} \quad \forall d, j$

"Activación de las nuevas variables"

Cuando ambas variables se activan simultáneamente, $Tc_{i_1,i_2,j}$ se activa. No vale que si alguna de las dos no se activa, $Tc_{i_1,i_2,j}$ no se activa. Sin embargo, como las $Tc_{i_1,i_2,j}$ aparecen restando en la función objetivo, van a apuntar a ser 0.

2: Análisis del Algoritmo: Comparación de parámetros

En primer lugar, tanto para este capítulo como para el siguiente, las entradas referenciadas y los resultados se encuentran disponibles en la carpeta del proyecto. Se podrá ver el último snapshot de las ejecuciones en `notebook.ipynb`.

Se utilizó el módulo CPLEX, un módulo de código cerrado creado por IBM que entre otras cosas, resuelve Programación Lineal Entera, y da la posibilidad al usuario a configurar hiperparámetros del solver, de acuerdo con el problema que busca resolver.

Salvo indicación, los casos de prueba para el análisis algorítmico se realizaron sobre la *entrada_mediana* y *entrada_grande* en el problema original (sin las restricciones adicionales). La *entrada_mediana* consiste de 30 trabajadores y 60 órdenes, y la *entrada_grande* 100 trabajadores y 50 órdenes.

Para las mediciones temporales se tomaron los valores promedios de 5 mediciones en cada situación para el caso de la entrada mediana. Para el caso de la entrada grande, realizamos una sóla medición en cada caso.

También es importante destacar que salvo para el caso donde se estudió el impacto del preprocesamiento, se desactivó el preprocesamiento, puesto que los tiempos totales han sido menores.
La cantidad de variables creadas serán:

$$\#A_{ijd} = \#Ordenes * \#Trabajadores * 6$$
$$\#B_{jdk} = \#Ordenes * \#Trabajadores * 5$$
$$\#Tr_{id} = \#Trabajadores * 6$$
$$\#\delta_j = \#Ordenes$$
$$\#X_{i1}, X_{i2}, X_{i3}, X_{i4} = \#Trabajadores * 4$$
$$\#W_{i1}, W_{i2}, W_{i3} = \#Trabajadores * 3$$
$$\#total = \#A_{ijd} + \#B_{jdk} + \#Tr_{id} + \#\delta_j + \#X_{i1}, X_{i2}, X_{i3}, X_{i4} + \#W_{i1}, W_{i2}, W_{i3}$$

El aumento de trabajadores y ordenes generará una gran cantidad de variables y restricciones nuevas (a razón #Ordenes*#Trabajadores) muchas de las cuales serán pueden llegar redundantes para casos específicos. Al realizar un **preprocesamiento**, el mismo podrá realizar varias operaciones para achicar el espacio factible en la primera relajación lineal, y que se asemeje lo más posible a la cápsula convexa factible.

Preprocesamiento	Tiempo Total (seg)	Tiempo Primera Relajación Lineal (seg)
No	29.12	12.50
Si	109.53	10.61

Table 1: Preprocesamiento con entrada mediana

Según nuestras observaciones, el preprocesamiento del problema aumenta el tiempo de ejecución de forma significativa, mas los tiempos para resolver la

Preprocesamiento	Tiempo Total (seg)	Tiempo Primera Relajación Lineal (seg)
No	30.48	19.89
Si	226.89	15.64

Table 2: Preprocesamiento con entrada grande

primera relajación son menores. Lo que puede explicarse por la gran cantidad de variables y restricciones que CPLEX buscará preprocesar, sin llegar a una región factible que resulte significativamente más favorable.

En segundo lugar, al tomar una gran cantidad de variables también pueden generarse diferencias en los tiempos de ejecución con los distintos **criterios de ramificación**. Por ejemplo, puede ocurrir que al tomar un criterio de *Strong Branching* el tiempo de ejecución sea mayor que otros criterios -como el de pseudocosto- ya que se debe simular un branching partiendo de cada una de las variable. En contraste con el criterio anterior, el pseudocosto promedia los puntajes obtenidos a medida que avanza en las ramas de búsqueda sin depender tanto de la cantidad de variables, como ocurre en *Strong Branching*.

Entrada	Default	Pseudocosto	StrongBranching
Tiempo Mediana	29.12	29.13	29.45
Tiempo Grande	45	42.17	49.5

Table 3: Criterio de branching

En la entrada mediana de ejemplo obtuvimos resultados coherentes con estas observaciones, donde el criterio de pseudocosto muestra ser más rápido que el de *StrongBranching* y al mismo tiempo, el criterio dado por default por CPLEX muestra un redimiento similar al de pseudocosto (ver Tabla 4).

Debido a que la cantidad de subproblemas en el árbol de B&B aumenta proporcionalmente con la cantidad de variables no enteras en cada relajación lineal, se pueden implementar **heurísticas primales** que, para cada relajación lineal, intentan con una determinada estrategia llevar la solución óptima a una solución entera, y de esta forma intentar generar más podas en el árbol.

A pesar de que CPLEX no informa el criterio usado como heuristica primal, el parámetro "Heuristic-effort" pondera la importancia dada por el modulo CPLEX a las heuristicas primales. Para cada valor de esfuerzo, Se tomaron 5 valores temporales (promediados) del algortimo usando la entrada mediana. En nuestros resultados notamos una tendencia a mejorar el rendimiento a mayor esfuerzo en la heurística primal para nuestro caso (figura 1).

Por último, podemos analizar cómo varían los tiempos con el **criterio de selección de nodo**. Es decir, el criterio para determinar qué nodo abierto será el próximo a explorar. Por ejemplo, si se utiliza búsqueda por profundidad es

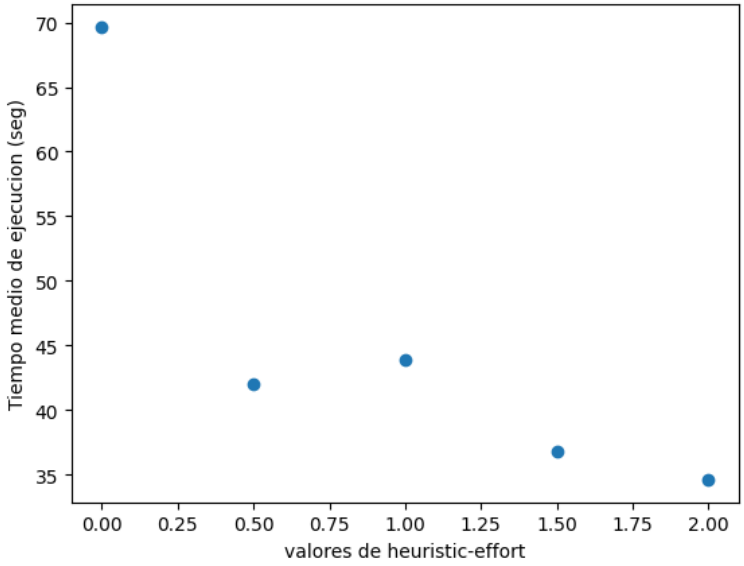

Figure 1: En el gráfico puede observarse un scatter plot dado por los tiempos de ejecución del algoritmo en segundos en funcion del heuristic-effort.

esperable que aumente la probabilidad de realizar la búsqueda por ramas que no llegan al óptimo del problema. Por este motivo, es razonable pensar que el criterio de seleccion por profundidad a veces tarde más tiempo en encontrar el óptimo del problema lineal que otros criterios como el de BestBound donde se elige algún nodo abierto con mejor cota dual en la relajación lineal como próximo nodo a explorar, tienden a ser más rápidos (aunque menos eficientes en memoria) Nuestro análisis será únicamente en el tiempo final de ejecución.

Entrada	BestBound(Default)	BestEstimate	Profundidad
Tiempo Mediana	29.12	30.10	29.71
Tiempo Grande	32.22	36	34.64

Table 4: Criterio de Selección de Nodo

Al testear con la *entrada_mediana* se observa que el tiempo de ejecución usando el criterio de profundidad es mayor que el tiempo de ejecución obtenido por el criterio de BestBound (ver tabla 5) lo que puede explicarse por la observación previamente hecha.

Al usar el criterio de BestEstimate se obtiene un tiempo de ejecución similar -aun que menor- al criterio de profundidad. Esto puede deberse a que el criterio de BestEstimate opta por los nodos que aumentan la integrabilidad de

las soluciones. Esto podría acelerar en un principio la búsqueda de la rama de optimalidad más cercana a la solución del problema entero, pero al seguir la búsqueda con varios nodos es esperable que los valores de los óptimos lineales de los subproblemas sean similares entre si por lo que el criterio de búsqueda terminará asemejandose al criterio de profundidad.

3: Análisis de Datos

7 Testeo de correctitud

7.1 Entrada Pequeña

Para esta sección, utilizamos una entrada pequeña (10 trabajadores, 11 órdenes)

Cantidad de trabajadores: 10
Conflictos entre trabajadores:[(0,1);(2,3)]
Órdenes correlativas: [(0,1)]
Órdenes conflictivas: [(1,2),(3,4)]
Órdenes repetitivas: [(5,6),(9,10)]

Orden	Beneficio	Cant.Trab.
0	8000	3
1	3000	2
2	4000	2
3	6000	4
4	2000	1
5	8000	5
6	100	1
7	7000	3
8	100000	10
9	100000	9
10	20000	9

7.2 Resultados

Puede verse que los resultados obtenidos en la entrada pequeña respetan todas las restricciones necesarias del problema (ver tabla 1 y tabla 2).

Notemos que el beneficio por realizar la orden 8 es muy alto pero requiere de todos los trabajadores para poder ser realizada. Por lo que es esperable que si no hay conflictos entre trabajadores, se priorice la asignacion de la orden 8 como puede observarse en los resultados del problema sin restricciones adicionales. Al agregar conflictos entre trabajadores, la orden 8 no podrá ser realizada ya que existe un par de trabajadores que no puede estar en la misma orden lo que causa la disminución del valor objetivo de 210000 al resolver el problema sin las restricciones adicionales a 109000 cuando se resueve con las restricciones adicionales.

La segunda restricción deseable que condiciona a los trabajadores a no realizar tareas repetitivas también juega un rol en la disminución del valor objetivo. Observemos que la orden 9 tiene un beneficio muy alto en comparación al resto (100000) por lo que es esperable que el problema de maximización priorice esta orden asignandole 9 trabajadores sin conflictos entre ellos.

Observemos también que la orden 10 también cuenta con un beneficio alto pero menor que el beneficio de la orden 9. Como estas dos ordenes (9 y 10) son repetitivas, no podrán ser realizadas ambas por lo que el óptimo no podrá sumar el beneficio de ambas y tendrá que optar por una. En nuestro ejemplo, vemos que en el óptimo se asignaron a los trabajadores a la orden 9 por que aporta mayor beneficio que la orden 10.

Table 5: Sin restricciones deseables

Orden	Trabajadores
0	[0, 4, 8]
1	[1, 5]
2	[2, 9]
3	[0,4,7,9]
4	[0]
5	[2,3,5,8,9]
6	no asignado
7	[1, 3, 7]
8	[0,1,2,3,4,5,6,7,8,9]
9	[0,1,2,3,4,5,6,7,8]
10	[1,2,4,5,6,7,8,9]

Funcion Objetivo: 210000

Table 6: Con restricciones deseables

Orden	Trabajadores
0	[0, 4, 7]
1	[7, 9]
2	[4, 6]
3	[0,4,6,9]
4	[2]
5	[1, 2, 4, 8, 9]
6	no asignado
7	[2, 6, 7]
8	no asignado
9	[1,2,3,4,5,6,7,8, 9]
10	no asignado

Funcion Objetivo: 109000

Table 7: Sólo eliminando conflictos entre trabajadores

Orden	Trabajadores
0	[2, 4, 5]
1	[3, 6]
2	[0, 3]
3	[2, 3, 5, 6]
4	[7]
5	[1, 4, 6, 7, 9]
6	no asignado
7	[0, 2, 5]
8	no asignado
9	[0, 2, 3, 4, 5, 6, 7, 8, 9]
10	[1, 2, 3, 4, 5, 6, 7, 8, 9]

Funcion Objetivo: 120000

Table 8: Sólo eliminando órdenes repetitivas

Orden	Trabajadores
0	[0, 3, 5]
1	[2, 3]
2	[0, 2]
3	[4, 6, 7, 8]
4	[5]
5	[0, 5, 6, 7, 8]
6	no asignado
7	[2, 3, 8]
8	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
9	[0, 1, 2, 4, 5, 6, 7, 8, 9]
10	no asignado

Funcion Objetivo: 199000

También se puede concluir que la restricción de eliminar completamente los conflictos entre trabajadores en este caso disminuye considerablemente la función objetivo, en comparación con la otra restricción deseable.

7.3 Análisis penalizando por trabajadores en conflicto

Ya analizamos la diferencia en los valores óptimos obtenidos al agregar las restricciones adicionales. En particular, observamos en nuestro ejemplo que al restringir que los trabajadores conflictuados no pueden estar en una misma orden, obtenemos una disminución en el valor objetivo causada por disminuir la región factible.

Por este motivo, se propone agregar una penalización dada por un parámetro

α a la función objetivo permitiendo soluciones factibles donde los trabajadores en conflicto sean asignados a una misma orden, pagando un costo como penalización ponderada (ver capítulo 1, sección 6).

Table 9: Sin trabajadores en conflicto

Orden	Trabajadores
0	[4,8,9]
1	[0,9]
2	[1,9]
3	[0,3,5,7]
4	[4]
5	[1,3,5,6,8]
6	no asignado
7	[0,2,6]
8	no asignado
9	[0,2,3,4,5,6,7,8,9]
10	[0,2,3,4,5,6,7,8,9]

Funcion Objetivo: 120000

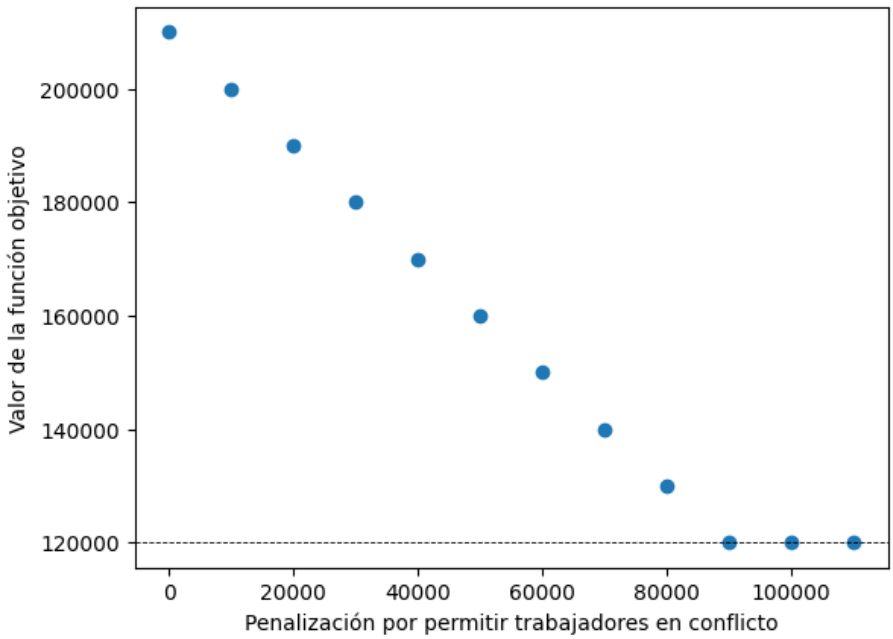

Figure 2: En el gráfico puede observarse un gráfico de dispersión dado los valores de la función objetivo al penalizar por tener trabajadores en conflicto trabajando en una misma orden. Notese que se marco una linea horizontal cuando el valor de la función objetivo vale 120000 (ver tabla 6) demarcando una situación limite cuando la perdida por penalización supera al beneficio de la orden.

Lógicamente, cuando α valga 0, se espera obtener el valor óptimo del problema sin las restricciones deseables. Análogamente, cuando α tome valores mayores al beneficio acumulado por las ordenes con trabajadores en conflicto, es esperable obtener valores óptimos igualen al problema con la restricción deseable.

Justamente, nuestros resultados reflejan esto. Como cota inferior del óptimo con penalización se obtuvo al valor 120000 (ver figura 2) que coincide con el óptimo del problema con la restriccion adicional que no permite que dos trabajadores en conflicto sean asignados a una misma orden (tabla 7).

Si agregamos una penalización a la cantidad de asignaciones de trabajadores a órdenes repetitivas, esperaremos observar un fenómeno similar.

4: Conclusiones

Dado que Programación Lineal Entera es NP-Completo, los algoritmos que busquen resolver problemas de PLE no poseen complejidad polinomial, por lo que sin mejoras prácticas, se vuelven imprácticos para resolver. En este trabajo, para una situación específica como es la de modelar la asignación de cuadrillas, se ha realizado un análisis teniendo como métrica al tiempo de diversos factores que uno puede considerar para conseguir soluciones en tiempos prácticos, que provee el solver CPLEX.

En la segunda parte, dada una instancia específica, se ha realizado un análisis acerca de cómo varía la función objetivo de acuerdo a los cambios deseables.