DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico de Programación Lineal Entera

Integrante	LU	Correo electrónico
Alvariñas, Belén	344/21	belualvarinas@gmail.com
Klimkowski, Victoria	1390/21	02vicky02@@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Índice

1. Modelo PLE

Veamos las variables, restricciones y función objetivo de nuestro problema. Lo modelamos utilizando programación lineal entera.

1.1. Variables

Para empezar, veamos las variables que utilizaremos en nuestro modelo. En general, las variables van a estar ligadas a las tareas, los trabajadores, el turno, el día y la categoría del trabajador. Entendemos como categoría al salario asociado que tendrá un trabajador por alguna tarea resuelta dependiendo la cantidad de horas trabajadas.

1.1.1. Notación de nuestro modelo

- t: tarea (en las consignas está como O).
- w: trabajador (en las consignas está como T).
- h: turno (5 turnos).
- d: día (6 días).
- c: categoría del trabajador (4: 1000, 1200, 1400, 1500).

En general, cuando nos refiramos a esas cuestiones con minúscula (por ejemplo: la tarea t) nos referimos a alguna tarea cualquiera, mientras que si decimos T nos estamos refiriendo a la cantidad de tareas total.

1.1.2. $X_{twhdc} \in \{0, 1\}$

La variable X_{twhdc} tiene subíndices que representan, como antes, una tarea particular para un trabajador particular en un turno particular de un día particular con una categoría particular. Va a haber T*W*H*D*C = T*W*5*6*4 de esta variable y va a valer 1 si la tarea t es realizada por w de categoría c en el turno h en el día d y 0 en el caso contrario. Notar que esta variable va a existir para todas las combinaciones de t, w, h, d, c pero sólo va a valer 1 en los casos que consideremos pertinentes.

Poner la categoría genera muchas variables

1.1.3. $delta_t \in \{0, 1\}$

La variable $delta_t$ nos define si la tarea t fue realizada. En la sección 1.2. nos aseguraremos que valga 1 cuando la tarea es realizada en un día y horario determinado (único) con la cantidad apropiada de trabajadores y 0 en el caso contrario. Va a haber T variables.

1.1.4. $worker_{wc} \in \mathbb{N}_0$

Habrá W*C = W*4 de este tipo de variable. Nos dirá cuántas veces un trabajador es de cierta categoría (asociado a un sueldo de 1000, 1200, 1400 o 1500). Deberemos restringir que se vayan llenando "en orden" adecuado. Esto es: para un trabajador no permitir que se sea de una categoría superior a 1 si no es categoría 1 5 veces; no permitir que sea de una categoría superior a 2 si no es categoría 2 5 veces; no permitir que sea de una categoría superior a 3 si no es de categoría 3 5 veces.

$$worker_{w1} = \sum_{j=1}^{|T|} \sum_{k=1}^{|H|} \sum_{l=1}^{|D|} X_{j,w,k,l,1} \quad \forall w \tag{1}$$

$$worker_{w2} = \sum_{j=1}^{|T|} \sum_{k=1}^{|H|} \sum_{l=1}^{|D|} X_{j,w,k,l,2} \quad \forall w \tag{2}$$

$$worker_{w3} = \sum_{j=1}^{|T|} \sum_{k=1}^{|H|} \sum_{l=1}^{|D|} X_{j,w,k,l,3} \quad \forall w \tag{3}$$

$$worker_{w4} = \sum_{j=1}^{|T|} \sum_{k=1}^{|H|} \sum_{l=1}^{|D|} X_{j,w,k,l,4} \quad \forall w \tag{4}$$

1.1.5. $tareaEnEsteDiaYTurno_{thd} \in \{0,1\}$

Esta variable va a existir T*H*D veces = T*5*6. Vamos a modelar que valga 1 si la tarea t es realizada en el horario h en el día d y 0 en el caso contrario. Deberemos restringir que no haya más de un día u horario para la misma tarea.

1.1.6. $trabajaHoy_{wd} \in \{0,1\}$

Esta variable va a existir W*D=W*6 veces. Va a valer 1 si el trabajador w trabaja el día d y 0 en el caso contrario.

1.1.7. $deltaPago_{wc} \in \{0,1\}$

Usamos un abuso de notación porque en realidad no existirán las $deltaPago_{w4}$. Sólo las $deltaPago_{w1}$, $deltaPago_{w2}$, $deltaPago_{w3}$ Estas serán W*3 variables auxiliares para organizar los pagos. Las vamos a necesitar para cada trabajador de manera que: $deltaPago_{w1} = 1$ si $worker_{w1}$=5, 0 caso contrario, $deltaPago_{w2} = 1$ si $worker_{w2}$=5, 0 caso contrario, $deltaPago_{w3} = 1$ si $worker_{w3}$=5, 0 caso contrario, ✓

1.1.8. $conflicto_{tab} \in \{0,1\}$

Esta variable va a existir T*W*W veces. Vamos a modelar que valga 1 si los trabajadores a y b están conflictuados y además están asignados a la tarea t.

1.1.9. $repetitivas_{wab} \in \{0,1\}$

Esta variable va a existir W*T*T veces. Vamos a modelar que valga 1 si las tareas a y b son repetitivas y están asignadas al mismo trabajador w.

1.2. Función Objetivo

$$\begin{aligned}
\text{Maximizar} \quad & \sum_{j=1}^{|T|} (\text{beneficio}(\delta_j) \cdot \delta_j) \\
& - \sum_{i=1}^{|W|} (1000 \cdot \text{worker}_{i,1} + 1200 \cdot \text{worker}_{i,2} + 1400 \cdot \text{worker}_{i,3} + 1500 \cdot \text{worker}_{i,4}) && (5) \\
& - \sum_{j=1}^{|T|} \sum_{a=1}^{|W|} \sum_{b=1}^{|W|} conflicto_{jab} && (6) \\
& - \sum_{w=1}^{|W|} \sum_{a=1}^{|T|} \sum_{b=1}^{|T|} repetitivas_{wab}
\end{aligned}$$

Expliquemos el razonamiento paso a paso:

1.2.1. Maximizar beneficio

Quiero maximizar el beneficio económico que me produce realizar tareas sabiendo que tengo que pagarle a los trabajadores según la cantidad de horas que hayan trabajado

$$\text{Maximizar} \quad \sum_{j=1}^{|T|} (\text{beneficio}(\delta_j) \cdot \delta_j) \tag{7}$$

Cada δ_j está asociada a una tarea de forma biyectiva y cada tarea está asociada a un beneficio de forma biyectiva. Luego, cada δ_j está asociado al beneficio de la tarea j. Ese beneficio sólo será tenido en cuenta para la función objetivo en el caso que la tarea se realice, es decir, cuando δ_j=1 ✓

Notemos que acá la nombramos δ_j por estar dentro de una sumatoria y mostrar que suma cada una de las δ_j pero en las restricciones usaremos δ_t porque nos parece más claro para entender que cada variable δ_t está asociada a una tarea t.

1.2.2. Pago de haberes

Además, debemos tener en cuenta el sueldo de los trabajadores según la cantidad de horas que hayan trabajado. O sea, se lo restaremos a la función objetvo.

Cada empleado hace hasta 5 tareas por $1000 cada una, para las próximas 5 le pagan $1200 por cada una, las siguientes 5 le pagan $1400 por cada una y las demás $1500 cada una.

$$\begin{aligned}\text{Maximizar} \quad & \sum_{j=1}^{|T|} (\text{beneficio}(\delta_j) \cdot \delta_j) \\ & - \sum_{i=1}^{|W|} (1000 \cdot \text{worker}_{i,1} + 1200 \cdot \text{worker}_{i,2} + 1400 \cdot \text{worker}_{i,3} + 1500 \cdot \text{worker}_{i,4})\end{aligned}$$

La variable $worker_{ig}$ nos va a decir dado un trabajador i cuántas veces es de categoría g.

1.2.3. Minimizar conflictos y tareas repetitivas

Tenemos las variables $conflicto_{tab}$ y $repetitivas_{wab}$. $conflicto_{tab}$ me dice si la pareja conflictiva (a,b) fue asignada a la misma tarea t: si a y b están asignados a la tarea t vale 1. $repetitivas_{wab}$ me dice si el par de tareas repetitivas (a,b) fue asignada a la misma persona w: si a y b están asignadas al mismo trabajador w vale 1.

$$\begin{aligned}\text{Maximizar} \quad & \sum_{j=1}^{|T|} (\text{beneficio}(\delta_j) \cdot \delta_j) \\ & - \sum_{i=1}^{|W|} (1000 \cdot \text{worker}_{i,1} + 1200 \cdot \text{worker}_{i,2} + 1400 \cdot \text{worker}_{i,3} + 1500 \cdot \text{worker}_{i,4}) \qquad (8) \\ & - \sum_{j=1}^{|T|} \sum_{a=1}^{|W|} \sum_{b=1}^{|W|} conflicto_{jab} \qquad (9) \\ & - \sum_{w=1}^{|W|} \sum_{a=1}^{|T|} \sum_{b=1}^{|T|} repetitivas_{wab}\end{aligned}$$

En las secciones de Restricciones y Restricciones deseables habrá más desarrollo acerca de la construcción de estas variables y sus relaciones entre sí.

1.3. Restricciones

como necesite

Utilizamos la letra M para denotar al big M: una constante lo suficientemente grande, tanto como yo quiera. Aclaramos que algunas de las restricciones pueden ser redundantes (por ejemplo, la 1.3.8 implica la 1.3.9) o bien podríamos probar si es más óptimo formularlas de otra manera (por ejemplo en la 1.3.2 podría hacerse sin la desigualdad con la big M y dejando fijos los trabajadores y categorías en lugar de tenerlos en una sumatoria) . Sin embargo, decidimos realizarlo así porque nos parece que ayuda a la claridad del modelo y no necesitamos buscar la representación más óptima a fines de este trabajo

más eficiente

Por otra parte, tengamos en cuenta que al escribir las restricciones en el código debemos hacerlas de otra manera. Esto sería:

- Si hay una cadena de desigualdades, separarlas de a 2. Por ejemplo, si $x_1 \leq x_2 \leq x_3$, por un lado vamos a tener la restricción $x_1 \leq x_2$ y por otro lado la restricción $x_2 \leq x_3$.
- No podemos tener variables del lado derecho de la restricción. Esto es, si por ejemplo tenemos $x_1 \leq x_2$ la deberíamos escribir como $x_1 - x_2 \leq 0$ o bien $-x_1 + x_2 \geq 0$
- No podemos tener términos independientes del lado izquierdo de la restricción. Esto es, si por ejemplo tenemos $x_1 - 1 \leq 0$ la deberíamos escribir como $x_1 \leq 1$ o bien $-x_1 \geq -1$

En la sección 1.1.4 al crear la variable $worker_{wc}$ dijimos cómo se definía porque nos pareció más claro en la lectura. En el LP, son restricciones. Las recordamos.

$$worker_{w1} = \sum_{j=1}^{|T|}\sum_{k=1}^{|H|}\sum_{l=1}^{|D|} X_{j,w,k,l,1} \quad \forall w \tag{10}$$

$$worker_{w2} = \sum_{j=1}^{|T|}\sum_{k=1}^{|H|}\sum_{l=1}^{|D|} X_{j,w,k,l,2} \quad \forall w \tag{11}$$

$$worker_{w3} = \sum_{j=1}^{|T|}\sum_{k=1}^{|H|}\sum_{l=1}^{|D|} X_{j,w,k,l,3} \quad \forall w \tag{12}$$

$$worker_{w4} = \sum_{j=1}^{|T|}\sum_{k=1}^{|H|}\sum_{l=1}^{|D|} X_{j,w,k,l,4} \quad \forall w \tag{13}$$

1.3.1. No toda orden es resuelta

Modelamos que no toda orden de trabajo tiene que ser resuelta. Si existe algún X_{twhdc}=1 para una tarea determinada entonces δ_t vale 1. En cambio, si todas valían 0 el δ_t también lo hace. El δ_t lo utilizamos en la Función objetivo para contar o no el beneficio correspondiente a la tarea:

No son necesarias

$$\delta_t \leq \sum_{i=1}^{|W|}\sum_{k=1}^{|H|}\sum_{l=1}^{|D|}\sum_{g=1}^{|C|} X_{t,i,k,l,g} \leq M \cdot \delta_t \quad \forall t \tag{14}$$

Más adelante pediremos que si alguna X_{twhdc}=1 es porque tiene a los trabajadores necesarios trabajando al mismo tiempo.

Notemos que en este caso es suficiente tomar M=TrabajadoresRequeridos(δ_t). Es decir, la cantidad de trabajadores que se necesitan para la tarea determinada que dejamos fija.

1.3.2. Se realiza (o no) una tarea en un día y turno determinados

Aseguramos que si no hay trabajadores asignados a una tarea en un turno y día específicos, la variable $tareaEnEsteDiaYTurno_{t,h,d}$ será 0. Si la tarea t está asignada a algún trabajador en el horario h del día d valdrá 1:

No son necesarias

$$tareaEnEsteDiaYTurno_{t,h,d} \leq \sum_{i=1}^{|W|}\sum_{g=1}^{|C|} X_{t,i,h,d,g} \leq M \cdot tareaEnEsteDiaYTurno_{t,h,d} \quad \forall t,h,d \tag{15}$$

Notemos que en este caso es suficiente con tomar M=TrabajadoresRequeridos(t)

1.3.3. La tarea se realiza en un sólo día y turno o ninguno

La variable δ_t se prende si hay sólo un día y turno donde se realice la tarea (sólo vale 0 o 1):

$$\sum_{h=1}^{|H|}\sum_{d=1}^{|D|} tareaEnEsteDiaYTurno_{t,h,d} = \delta_t \quad \forall t \tag{16}$$

Notemos que esto lo hacemos para todos los δ_t.

1.3.4. Trabajadores necesarios para cada tarea

Para cada tarea en un día y turno específicos, si la variable $tareaEnEsteDiaYTurno_{t,h,d}$ toma el valor 1 (es decir, hay al menos una tarea asignada), entonces la suma de asignaciones de trabajadores debe ser igual a los trabajadores requeridos multiplicados por $tareaEnEsteDiaYTurno_{t,h,d}$, es decir, cada órden de trabajo o_i debe tener asignados sus t_i trabajadores en un mismo turno

$$\sum_{i=1}^{|W|}\sum_{g=1}^{|C|} X_{t,i,h,d,g} = \text{TrabajadoresRequeridos}(\delta_t) \cdot tareaEnEsteDiaYTurno_{t,h,d} \quad \forall t,h,d \tag{17}$$

Notemos que TrabajadoresRequeridos(δ_t) está bien definido porque cada δ_t está asociada a una tarea en particular y cada tarea tiene una cantidad de trabajadores requeridos asociada.

1.3.5. Una "categoría" de pago por tarea

Para cada tarea, día, horario y trabajador específicos, se establece que la asignación de la tarea debe ser únicamente a una categoría de trabajador (por ejemplo, no puede ganar 1000 y 1200 por la misma tarea)

$$\sum_{g=1}^{|C|} X_{t,w,h,d,g} \leq 1 \quad \forall t, w, h, d \tag{18}$$

1.3.6. Máximo una tarea por trabajador a la vez

Para cada trabajador, día y horario específicos, se establece que el trabajador puede realizar como máximo una tarea:

$$\sum_{j=1}^{|T|} X_{j,w,h,d,c} \leq 1 \quad \forall w, h, d, c \tag{19}$$

1.3.7. Una tarea se realiza máximo una vez

Para cada tarea y trabajador específicos, se establece que la tarea puede asignarse en como máximo un día y horario:

No es necesaria

$$\sum_{k=1}^{|H|} \sum_{l=1}^{|D|} X_{t,w,k,l,c} \leq 1 \quad \forall t, w, c \tag{20}$$

No iteramos por la categoría porque la restricción 6 nos asegura que no hay múltiples categorías para un mismo día y horario (es indistinto)

1.3.8. Trabajan menos de 6 días

Para cada trabajador, pedimos que no puede trabajar más de 6 días en la semana. $trabajaHoy_{w,d}$ será 1 si dado un trabajador y un día este realiza alguna tarea en algún horario con alguna categoría y 0 en el caso contrario :

$$trabajaHoy_{w,d} \leq \sum_{j=1}^{|T|} \sum_{k=1}^{|H|} \sum_{g=1}^{|C|} X_{j,w,k,d,g} \leq M \cdot trabajaHoy_{w,d} \quad \forall w, d \tag{21}$$

$$\sum_{l=1}^{|D|} trabajaHoy_{w,l} \leq 5 \quad \forall w \tag{22}$$

Notemos que en este caso es suficiente tomar M=4 porque cada trabajador, cada día puede trabajar máximo 4 turnos.

1.3.9. Menos de 5 turnos por día

Para cada trabajador y día específicos, establecemos que el trabajador no puede trabajar en los 5 turnos:

No es necesaria

$$\sum_{j=1}^{|T|} \sum_{k=1}^{|H|} \sum_{g=1}^{|C|} X_{j,w,k,d,g} \leq 4 \quad \forall w, d \tag{23}$$

1.3.10. Órdenes no consecutivas

Suponemos que B no puede ser consecutiva a A y que A no puede ser consecutiva a B. Pedimos que las órdenes de trabajo A y B no se asignen correlativamente en turnos consecutivos para un mismo trabajador. Si A se asigna en el horario h de un día d , entonces la tarea B no puede ser asignada en el horario h+1 del mismo día d (o viceversa). Análogamente para que A no sea consecutiva a B.

✓
$$\sum_{g_1=1}^{|C|}\sum_{g_2=1}^{|C|}(X_{A,w,h,d,g_1}+X_{B,w,h+1,d,g_2})\leq 1 \quad \forall w,h\in[1,|H|-1],d,(A,B) \tag{24}$$

$$\sum_{g_1=1}^{|C|}\sum_{g_2=1}^{|C|}(X_{B,w,h,d,g_1}+X_{A,w,h+1,d,g_2})\leq 1 \quad \forall w,h\in[1,|H|-1],d,(A,B) \tag{25}$$

1.3.11. Órdenes consecutivas

Para permitir que las órdenes A y B sean correlativas, pero no necesariamente asignadas al mismo trabajador, es decir, si se satisface A, entonces debe satisfacerse B, (y si B se satisface es porque primero A fue satisfecha). O ambas se satisfacen o no se satisface ninguna:

Esto funciona sólo si ambas órdenes tienen igual cantidad de trabajadores!!

$$\sum_{i_1=1}^{|W|}\sum_{i_2=1}^{|W|}\sum_{g_1=1}^{|C|}\sum_{g_2=1}^{|C|}(X_{A,i_1,h,d,g_1}-X_{B,i_2,h+1,d,g_2})=0 \quad \forall h\in[1,|H|-1],d,(A,B) \tag{26}$$

Lo deberían modelar con las variables tareaEnEsteDiaYTurno

1.3.12. Diferencia entre tareas realizadas menores o iguales a 8

Pedimos que la diferencia de órdenes asignadas para 2 trabajadores distintos sea menor o igual a 8. En particular esto significa que el módulo de la diferencia entre tareas sea menor o igual a 8 o, equivalentemente, abrir el módulo.

Para esto consideramos todos los trabajadores y, en particular, vale para el que sume la cantidad máxima de tareas realizadas (suma de todas las tareas de todos los horarios de todos los días de todas las categorías) y la cantidad mínima:

✓
$$-8\leq\sum_{j_1=1}^{|T|}\sum_{j_2=1}^{|T|}\sum_{k_1=1}^{|H|}\sum_{k_2=1}^{|H|}\sum_{l_1=1}^{|D|}\sum_{l_2=1}^{|D|}\sum_{g_1=1}^{|C|}\sum_{g_2=1}^{|C|}(X_{j_1,w_1,k_1,l_1,g_1}-X_{j_2,w_2,k_2,l_2,g_2})\leq 8 \quad \forall w_1\neq w_2 \tag{27}$$

1.3.13. Pago

Necesitamos modelar los diferentes niveles de salario fijo, según la categoría, asegurando que cada tipo de trabajador reciba el salario correcto según su categoría siguiendo el esquema de remuneración pedido:

✓
$$\begin{aligned}
5\cdot\text{deltaPago}_{w_1}\leq\text{worker}_{w_1}\leq 5 \quad &\forall w\\
5\cdot\text{deltaPago}_{w_2}\leq\text{worker}_{w_2}\leq 5\cdot\text{deltaPago}_{w_1} \quad &\forall w\\
5\cdot\text{deltaPago}_{w_3}\leq\text{worker}_{w_3}\leq 5\cdot\text{deltaPago}_{w_2} \quad &\forall w\\
0\leq\text{worker}_{w_4}\leq M\cdot\text{deltaPago}_{w_3} \quad &\forall w
\end{aligned} \tag{28}$$

Notemos que en este caso es suficiente con tomar M=max(0,|T|-15) cantidad total de tareas menos la cantidad que debería haber trabajado por otro rango salarial.

1.4. Restricciones Deseables

1.4.1. Conflictos entre trabajadores

Para evitar conflictos entre trabajadores A y B, al asignar la misma tarea en el mismo día y horario, pedimos que ninguno o uno solo la tenga asignada.

Si el objetivo fuera no permitir conflictos, deberíamos agregar esta restricción:

$$\sum_{k=1}^{|H|}\sum_{l=1}^{|D|}\sum_{g_1=1}^{|C|}\sum_{g_2=1}^{|C|}(X_{t,A,k,l,g_1}+X_{t,B,k,l,g_2})\leq 1 \quad \forall t,(A,B) \tag{29}$$

Sin embargo, queremos minimizar el conflicto pero permitirlo. Lo tenemos en cuenta en la función objetivo. Estamos asumiendo que por cada tarea donde (A,B) estén juntos (para cada par existente de (A,B)) voy a tener $1 menos.

¿Por qué decidieron este valor? ¿Experimenntaron con otros valores?

Cómo impacta en las restricciones Podríamos tener un $conflicto_{tAB}$ una variable que es 1 si A y B están juntos en la tarea t. Utilizamos linealización.

$$\sum_{k=1}^{|H|}\sum_{l=1}^{|D|}\sum_{g_1=1}^{|C|}\sum_{g_2=1}^{|C|}(X_{t,A,k,l,g_1}+X_{t,B,k,l,g_2}) \leq conflicto_{tAB}+1 \quad \forall t,(A,B) \tag{30}$$

$$2 * conflicto_{tAB} \leq \sum_{k=1}^{|H|}\sum_{l=1}^{|D|}\sum_{g_1=1}^{|C|}\sum_{g_2=1}^{|C|}(X_{t,A,k,l,g_1}+X_{t,B,k,l,g_2}) \quad \forall t,(A,B) \tag{31}$$

Cómo impacta en la función objetivo Agregamos

$$-\sum_{j=1}^{|T|}\sum_{a=1}^{|W|}\sum_{b=1}^{|W|} conflicto_{jab} \tag{32}$$

Notemos que la sumatoria recorre todas las parejas conflictivas que hay o no en cada tarea.

Es decir, la función objetivo quedaría

$$\begin{aligned}\text{Maximizar} \quad & \sum_{j=1}^{|T|}(\text{beneficio}(\delta_j)\cdot\delta_j) \\ & -\sum_{i=1}^{|W|}(1000\cdot \text{worker}_{i,1}+1200\cdot \text{worker}_{i,2}+1400\cdot \text{worker}_{i,3}+1500\cdot \text{worker}_{i,4} \\ & -\sum_{j=1}^{|T|}\sum_{a=1}^{|W|}\sum_{b=1}^{|W|} conflicto_{jab}\end{aligned} \tag{33}$$

1.4.2. Tareas repetitivas

Para evitar que un mismo trabajador realice tareas repetitivas A y B deberíamos pedir que o bien realice A o bien realice B.

Si el objetivo fuera no permitir tareas repetitivas, deberíamos agregar esta restricción:

$$\sum_{k_1=1}^{|H|}\sum_{k_2=1}^{|H|}\sum_{l_1=1}^{|D|}\sum_{l_2=1}^{|D|}\sum_{g_1=1}^{|C|}\sum_{g_2=1}^{|C|}(X_{A,w,k_1,l_1,g_1}+X_{B,w,k_2,l_2,g_2}) \leq 1 \quad \forall w,(A,B) \tag{34}$$

Sin embargo, queremos minimizar la repetición pero permitirla. Lo tenemos en cuenta en la función objetivo. Estamos asumiendo que por cada trabajador que realice las tareas repetitivas A y B (para todo A y B par repetivo) vamos a tener $1 menos.

Cómo impacta en las restricciones Podríamos tener un $repetitivas_{wAB}$ una variable que es 1 si el trabajador w hace A y B pero es 0 si no. Utilizamos linealización.

$$\sum_{k=1}^{|H|}\sum_{l=1}^{|D|}\sum_{g_1=1}^{|C|}\sum_{g_2=1}^{|C|}(X_{A,w,k,l,g_1}+X_{B,w,k,l,g_2}) \leq repetitivas_{tAB}+1 \quad \forall w,(A,B) \tag{35}$$

$$2 * repetitivas_{wAB} \leq \sum_{k=1}^{|H|}\sum_{l=1}^{|D|}\sum_{g_1=1}^{|C|}\sum_{g_2=1}^{|C|}(X_{A,w,k,l,g_1}+X_{B,w,k,l,g_2}) \quad \forall t,(A,B) \tag{36}$$

Cómo impacta en la función objetivo Agregamos

$$-\sum_{w=1}^{|W|}\sum_{a=1}^{|T|}\sum_{b=1}^{|T|} repetitivas_{wab} \tag{37}$$

Notemos que la sumatoria recorre todos los pares de tareas repetitivas que tiene o no un trabajador.

Es decir, la función objetivo quedaría

$$\text{Maximizar} \quad \sum_{j=1}^{|T|} (\text{beneficio}(\delta_j) \cdot \delta_j)$$

$$- \sum_{i=1}^{|W|} (1000 \cdot \text{worker}_{i,1} + 1200 \cdot \text{worker}_{i,2} + 1400 \cdot \text{worker}_{i,3} + 1500 \cdot \text{worker}_{i,4}) \tag{38}$$

$$- \sum_{j=1}^{|T|} \sum_{a=1}^{|W|} \sum_{b=1}^{|W|} conflicto_{jab} \tag{39}$$

$$- \sum_{w=1}^{|W|} \sum_{a=1}^{|T|} \sum_{b=1}^{|T|} repetitivas_{wab}$$

2. Experimentación

2.1. Creación de la instancia

Para empezar, vamos a hablar un poco de instancias "random" que creamos y testeamos para darnos una idea de qué podríamos variar para nuestro análisis.

Hicimos un script de Python para generar instancias grandes al azar. Los beneficios son enteros (elegidos al azar) entre 11000 y 32000 y la cantidad de trabajadores requeridos para una tarea es un entero al azar entre 3 y 12.

En todos los casos usamos una tolerancia de 1e-10 como parámetro de CPLEX.

Primero, intentamos generar una instancia del orden de 850 tareas con 250 trabajadores (y otros parámetros razonables para ese tamaño de la entrada).

Para ese tamaño, luego de 20 minutos nos dio CPLEX Error 1001: Out of memory por lo que decidimos achicar el tamaño de la instancia.

Luego, con 100 trabajadores y 300 tareas nos dio el mismo error de terminación.

Pensamos en instancias más chicas. Tienen muchas restricciones no necesarias.

¿Probaron cambiando el recorrido del árbol?

2.2. Variación de tareas dejando fija la cantidad de trabajadores

¿Por qué 1?

En esta sección, decidimos que el coeficiente en la función objetivo de los conflictos y de las tareas repetitivas sea = 1. Es decir, se perdía $1 por cada conflicto o tarea repetitiva que había. Por otro lado, cuando hablamos de la ejecución con las deseables pusimos que haya 40 conflictos y 40 tareas repetitivas.

2.2.1. 10 trabajadores, 100 tareas

Sin deseables tardó 0 segundos en crearse y en ejecutarse, mientras que con las deseables tardó 7 segundos en crearse y 3 minutos en resolverse. Hubo 3 trabajadores asignados a tareas repetitivas.

2.2.2. 10 trabajadores, 400 tareas

Para esta instancia además comparamos con variación de parámetros.

Por un lado, sin las deseables pero variando algunos parámetros obtuvimos los siguientes resultados:

- **Tiempo de creación del LP:** 0:0:16
- **Tiempo de resolución del LP sin variar parámetros:** 0:1:21
- **Tiempo variando Depth-first search:** 0:1:18
- **Tiempo utilizando pseudocosto:** 0:0:19
- **Tiempo eligiendo strong branching:** 0:0:20
- **Tiempo quitándole el preprocesamiento:** 0:0:32
- **Tiempo eligiendo "más heurísticas":** 0:2:01
- **Tiempo sin los cortes en la raíz:** 0:1:55
- **Tiempo sin los cortes en los nodos:** 0:0:59
- **Tiempo total:** 0:11:01

Vemos que lo que tarda más tiempo es haber elegido las heurísticas y lo que más rápido hace es lo elegido por default.

La primer solución entera que se encontró en todas esas ejecuciones dio una función objetivo de valor 941740 y no se mejoró en iteracion posteriores. ¿La primera soluciónn que encontró fue la óptima?

Al correrla con deseables, tardó 29 segundos en crearse pero 1:5:34 en resolverse la default. Claramente tardó muchísimo más (al rededor de 65 veces más). Nos dio una función objetivo de 941706 con 34 conflictos que es exactamente el valor de la función objetivo sin las deseables - 34.

2.2.3. 10 trabajadores, 500 tareas

Para esta instancia además comparamos con variación de parámetros.

Por un lado, sin las deseables pero variando algunos parámetros obtuvimos los siguientes resultados:

- **Tiempo de creación del LP:** 0:0:21
- **Tiempo de resolución del LP sin variar parámetros:** 0:7:59
- **Tiempo variando Depth-first search:** 0:15:45
- **Tiempo utilizando pseudocosto:** 0:2:12
- **Tiempo eligiendo strong branching:** 0:3:51
- **Tiempo quitándole el preprocesamiento:** 0:10:03
- **Tiempo eligiendo "más heurísticas":** 0:4:40
- **Tiempo sin los cortes en la raíz:** 0:2:11
- **Tiempo sin los cortes en los nodos:** 0:5:53
- **Tiempo total:** 0:56:14

Vemos que esta vez lo que tarda más tiempo es hacer DFS seguido por quitar el preprocesamiento. A diferencia del ítem anterior donde agregar heurísticas era lo que más tiempo tomaba, en esta instancia achica bastante la cantidad de tiempo en comparación al default.

En ambos casos fue más óptimo quitar los cortes.

Nuevamente encontró la misma función objetivo con todas las variaciones de parámetros de valor 1008005. Sin embargo, esta vez no fue la función objetivo asociada a la primer solución entera en la ejecución por default sino que fue la 6ta.

Al correrla con deseables, tardó 37 segundos en crearse pero 12:21:03 (sí, más de 12 horas, lo dejamos correr toda la noche + mientras estábamos en clase de la materia) en resolverse la default. Claramente tardó muchísimo más (al rededor de 92 veces más). Nos dio una función objetivo de 1007973 con 32 conflictos que es exactamente el valor de la función objetivo sin las deseables - 32.

Esta instancia fue interesante porque esta vez hubo variables que no tomaron valores enteros. Por ejemplo, tomaron valores como 1.0000000000110272 o 0.9999999999889729. Esto sucedió para 14 variables.

2.2.4. Conclusiones

Claramente, estos tiempos nos dieron la pauta de que agregar las deseables hace que el tiempo de ejecución sea demasiado grande en comparación a no agregarlas así que decidimos hacer la variación de parámetros solamente para instancias sin las restricciones deseables.

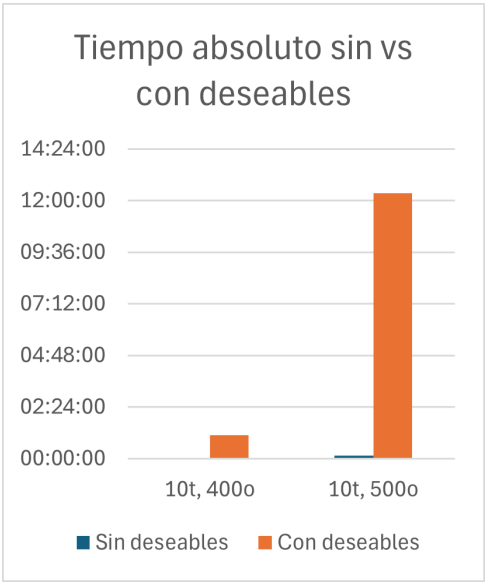

Figura 1: Tiempo absoluto instancias iniciales.

Tomamos logaritmo para verlo mejor

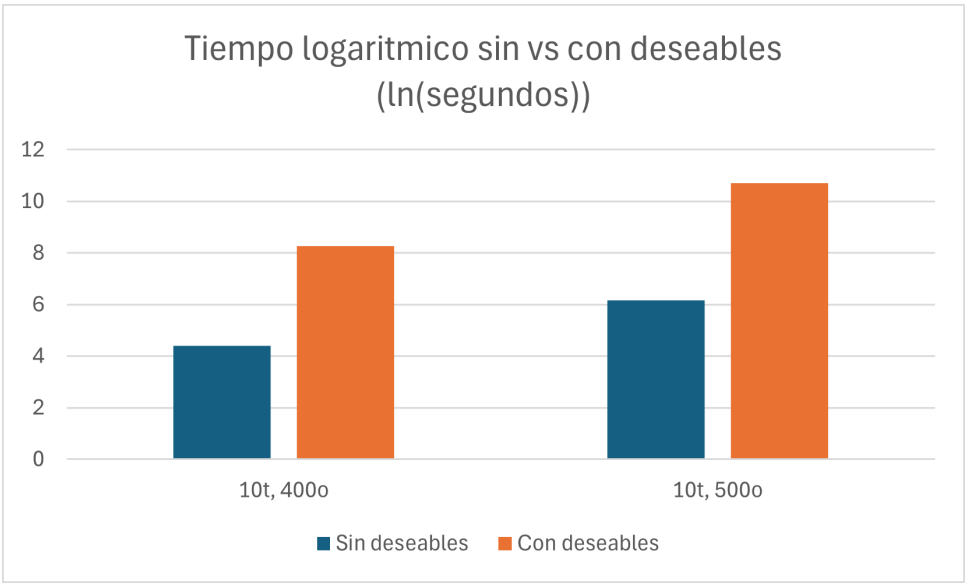

Figura 2: Tiempo logaritmico instancias iniciales.

Podemos notar que el tiempo que lleva realizar las instancias con las restricciones deseables es exponencial en relación a no haberlas incluido.

Ahora que tenemos una idea de qué tamaños podemos armarnos, haremos un análisis más exhaustivo de tamaños de la instancia y de variación de parámetros.

Además, notamos que como el peso de las deseables es muy bajo siempre elige que se violen. Tratamos de buscar algo más equitativo.

2.3. Variación de trabajadores dejando fija la cantidad de tareas

Encontramos "a ojo" una tendencia en la variación de tiempo con proporciones distintas de trabajadores-tareas en instancias "de juguete" que fuimos creando. Esto es, el tiempo que se tardaba no dependía solo del tamaño de la entrada (trabajadores+tareas): no era lo mismo si una instancia tenía tamaño 200 con 10 trabajadores y 190 tareas o si era con 180 trabajadores y 20 tareas. Esto creemos que se debe a los distintos salarios que tenían los trabajadores dependiendo de la cantidad de tareas realizables.

Veamos distintos ejemplos y los tiempos que tomaron con deseables vs sin deseables.

Nuevamente, dejamos fija la cantidad de conflictos y de tareas repetitivas en 40.

Sin embargo, esta vez pesamos de otra manera a las deseables. Sea el beneficio (como lo explicamos anteriormente) una variable aleatoria uniforme discreta entre 11000 y 32000, por la ley de los grandes números sabemos que el promedio de todos los beneficios de todas las ordenes de una instancia (se generan independientemente y son variables aleatorias idénticamente distribuídas) convergerá a la esperanza, en este caso particular sería 21500 (pues los beneficios son variables aleatorias uniformes). Ésta (en negativo) es el valor que le dimos a las restricciones deseables de conflictos y de tareas repetitivas en la función objetivo.

En criollo: tomamos el promedio de los beneficios como coeficientes de las deseables en la función objetivo en lugar de un 1.

2.3.1. 20 trabajadores, 200 tareas

Sin deseables

- Función objetivo: 1 375 351
- Tiempo de creación del LP: 0:00:24
- Tiempo de resolución del LP: 0:00:44
- Solución entera encontrada: Luego de 4 encuentros de soluciones enteras

Con deseables

- Función objetivo: 1 375 351
- Tiempo de creación del LP: 0:00:29
- Tiempo de resolución del LP: 0:13:14
- Solución entera encontrada: Luego de 9 encuentros de soluciones enteras

Notamos que con las deseables apenas tarda más tiempo en crearse pero en resolverse tarda ~~un~~ muchísimo más (18 veces más). Además de que encontró la solución entera óptima luego de más del doble de soluciones enteras encontradas con las deseables en comparación a sin ellas.

Esta vez, a diferencia de la sección anterior, la función objetivo no varía.

¿En la solución se cumplen las deseables?

2.3.2. 50 trabajadores, 200 tareas

Sin deseables

- Función objetivo: 1 920 622
- Tiempo de creación del LP: 0:01:43
- Tiempo de resolución del LP: 2:0:55
- Solución entera encontrada: Luego de 5 encuentros de soluciones enteras

Con deseables

- Función objetivo: 1 920 622
- Tiempo de creación del LP: 0:01:55
- Tiempo de resolución del LP: 7:04:19
- Solución entera encontrada: Luego de 3 encuentros de soluciones enteras

Notamos que con las deseables apenas tarda más tiempo en crearse pero en resolverse tarda bastante más (3.5 veces más) proporcionalmente, esto es menos que en la anterior. A demás, a diferencia del ítem anterior encontró ~~soluciones~~ menos soluciones enteras factible al utilizar las restricciones deseables.

Una vez más, la función objetivo no varía.

2.3.3. 100 trabajadores, 200 tareas

Sin deseables

- Función objetivo: 2 302 095
- Tiempo de creación del LP: 0:06:13
- Tiempo de resolución del LP: 0:02:53
- Solución entera encontrada: Luego de 2 encuentros de soluciones enteras

Con deseables

- Función objetivo: 2 302 095
- Tiempo de creación del LP: 0:06:25
- Tiempo de resolución del LP: 0:04:18
- Solución entera encontrada: Luego de 2 encuentros de soluciones enteras

Notamos que con las deseables apenas tarda más tiempo en crearse pero en resolverse tarda más (casi el doble). Una vez más vemos que en proporción el tiempo añadido es menor a los ítems anteriores. Esta vez encontró la misma cantidad de soluciones enteras factibles en ambos casos. La función objetivo no varía.

2.3.4. 150 trabajadores, 200 tareas

Sin deseables

- Función objetivo: 2 690 419
- Tiempo de creación del LP: 0:13:00
- Tiempo de resolución del LP: 0:10:31
- Solución entera encontrada: Luego de 2 encuentros de soluciones enteras

Con deseables

- Función objetivo: 2 690 419
- Tiempo de creación del LP: 0:13:20
- Tiempo de resolución del LP: 0:11:56
- Solución entera encontrada: Luego de 2 encuentros de soluciones enteras

Notamos que con las deseables apenas tarda más tiempo en crearse y en resolverse tarda ligeramente más. La cantidad de soluciones factibles enteras encontradas es la misma y la función objetivo no varía.

2.3.5. 200 trabajadores, 200 tareas

Sin deseables

- Función objetivo: 2 571 176
- Tiempo de creación del LP: 0:22:33
- Tiempo de resolución del LP: 0:13:37
- Solución entera encontrada: Luego de 2 encuentros de soluciones enteras

Con deseables

- Función objetivo: error termination (CPLEX Error 1001: Out of memory)
- Tiempo de creación del LP: 0:22:50
- Tiempo de resolución del LP: 0:06:04
- Solución entera encontrada: No llegó a encontrar ninguna

Vemos que el tiempo de creación del LP es casi el mismo pero la vamos a excluir de las conclusiones por quedarse sin memoria.

2.3.6. Conclusiones

A diferencia del ítem anterior, nunca se prefirió violar las deseables al hacer que pesen más en la función objetivo y se consiguió la misma función objetivo con que sin ellas.

Es raro que la función objetivo no varíe con las deseables. ¿Tienen alguna explciación? ¿Puede ser porque en las soluciones óptimas ya se cumplen las deseables?

En las instancias donde cambiaba al ponerle peso 1, ¿qué sucede con peso mayor? Tiene que cambiar la fo

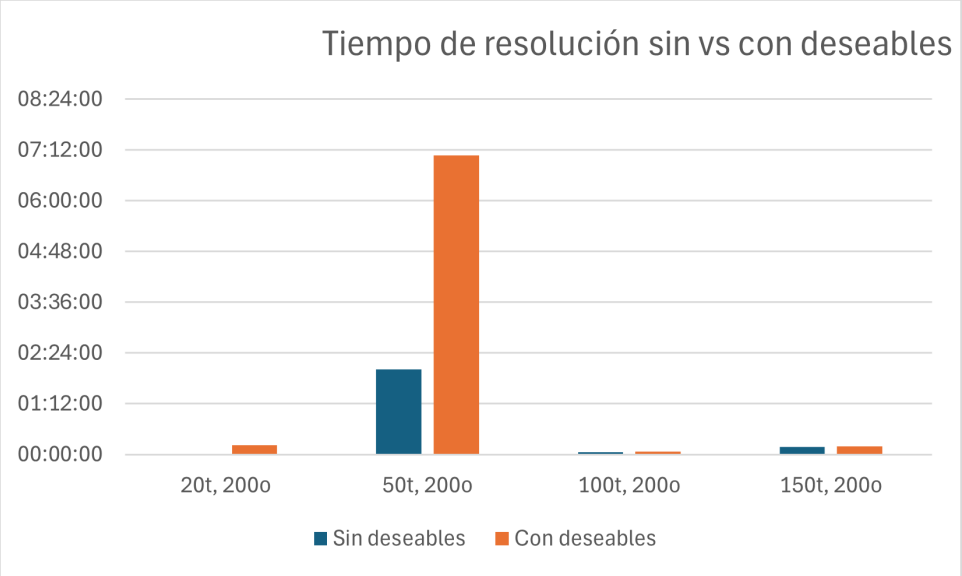

Figura 3: Tiempo de realización.

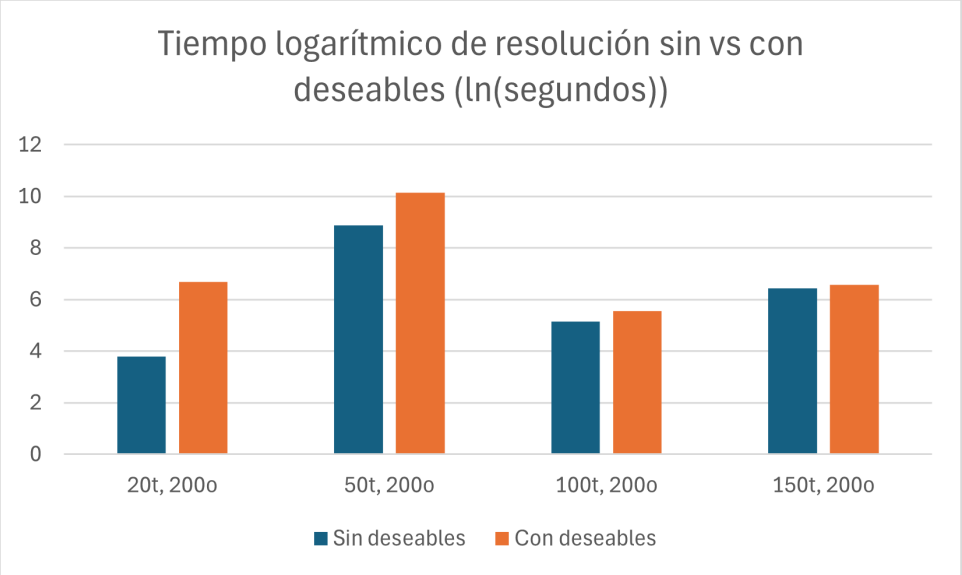

Figura 4: Logaritmo del tiempo de realización.

Notamos que siempre tarda más en resolverse cuando se agregan las restricciones deseables.

Vemos que en la instancia de 50 trabajadores hay un pico (tardó 7 horas) en la duración. Creemos que esto tiene que ver con la combinación de trabajadores/tareas que hay con los rangos salariales disponibles. ✓

Por un lado, creemos que tarda más la de 50 trabajadores que la de 20 trabajadores porque se resuelve mayor cantidad de tareas (en la de 20 trabajadores me quedo sin personas disponibles más rápido). Por otro lado, en la de 50 trabajadores hay personas a las que se les paga lo máximo disponible mientras que cuando hay 100 o 150 trabajadores se les puede pagar siempre $1000 o $1200 y aún así realizar todas las tareas. Esto probablemente impacte fuertemente en el tiempo de ejecución porque siempre se va a intentar pagar lo menos posible (más conseguible con las instancias con mayor disponibilidad de trabajadores) y de ahí ir aumentado. ✓

Además, notamos que la diferencia de tiempo que conlleva agregar las deseables (en proporción) disminuye a medida que se aumenta la cantidad de trabajadores disponibles. Creemos que esto tiene sentido porque estamos dejando fijas la cantidad de deseables en 4 conflictos y 40 tareas repetitivas ya que tal vez habrá menos "densidad"de deseables (más restricciones deseables sobre el total de restricciones). ✓

Tal vez, en las instancias más grandes, sea más fácil no violar las deseables porque hay menor densidad.

Veamos si el tiempo varía a mayor densidad de deseables.

2.4. Variación de cantidad de deseables dejando fija la cantidad de trabajadores y de órdenes

Tomamos como fijos 20 trabajadores y 200 tareas.

2.4.1. 40 conflictos, 40 tareas repetitivas

Igual que antes

- Tiempo de creación del LP: 0:00:29
- Tiempo de resolución del LP: 0:13:14

2.4.2. 80 conflictos, 80 tareas repetitivas

- Tiempo de creación del LP: 0:00:30
- Tiempo de resolución del LP: 0:21:21

2.4.3. 130 conflictos, 130 tareas repetitivas

- Tiempo de creación del LP: 0:00:34
- Tiempo de resolución del LP: 0:25:17

2.4.4. 185 conflictos, 185 tareas repetitivas

- Tiempo de creación del LP: 0:00:34
- Tiempo de resolución del LP: 0:42:48

2.4.5. Conclusiones

Vemos que no hay mucha diferencia en el tiempo de creación de las instancias. Notamos que el tiempo que tarda con 80 conflictos y 80 tareas repetitivas es ligeramente menos que el doble que cuando había 40 conflictos y 40 tareas repetitivas. Sin embargo, el salto es marginalmente más grande al pasar a 130 conflictos y 130 tareas repetitivas. Al pasar a 185 conflictos y 185 tareas repetitivas el salto es ligeramente menos que el doble que en la anterior pero esta vez no se estaba duplicando a cantidad absoluta como sí pasaba en el salto de 40 a 80.

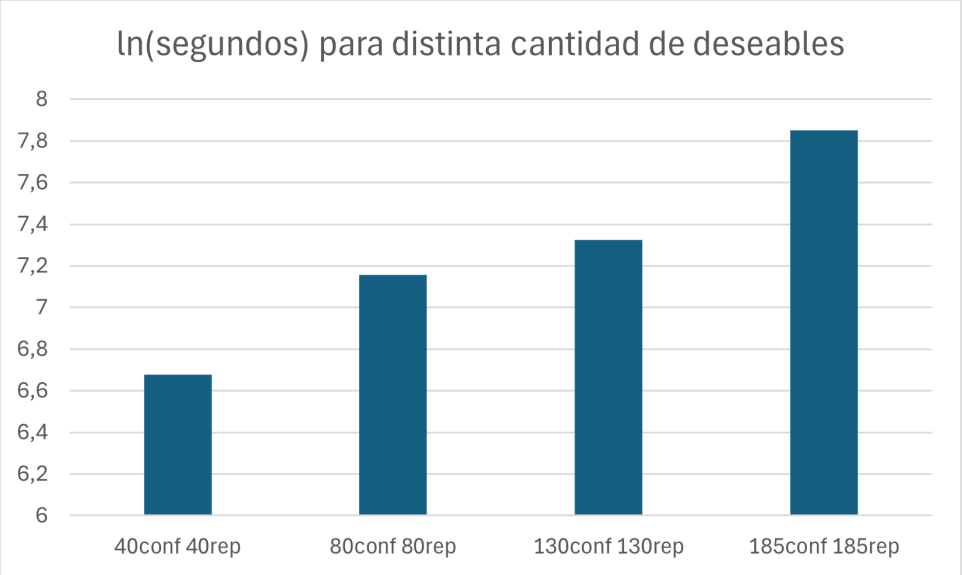

Figura 5: Logaritmo del tiempo de realización para distinta cantidad de deseables.

Evidentemente, a mayor cantidad de restricciones deseables para cierta cantidad fija de tareas y trabajadores, mayor es el tiempo que tarda en realizarse. ✓

2.5. Variación de parámetros

Para realizar la experimentación decidimos variar los siguientes parámetros y tomar sus tiempos para obtener distintos resultados: En primer lugar corrimos las instancias con los parámetros por default que utiliza cplex, tomando también el tiempo que le toma crear el modelo. Y luego para la variación comenzamos con Depth-first search (DFS) luego con pseudo-costo, strong branching, preprocesamiento, agregando más heurísticas, ejecutando sin cortes en la raíz y por último sin cortes en los nodos.

Los resultados obtenidos fueron los siguientes (las unidades que utilizamos son horas,minutos y segundos):

2.5.1. 20 trabajadores y 200 órdenes

- Tiempo de creación: 0:0:20
- Tiempo Ejecución Default sin variar parámetros: 0:10:39
- Tiempo Depth-first search : 0:2:54
- Tiempo con Pseudocosto : 0:1:07
- Tiempo Strong branching : 0:1:00
- Tiempo Preprosesamiento : 0:2:54
- Tiempo con mas heuristicas:: 0:1:13
- Tiempo termina de ejecutar sin los cortes en la raiz: 0:7:58
- Tiempo termina de ejecutar sin los cortes en los nodos: 0:3:57
- Tiempo total: 0:34:54

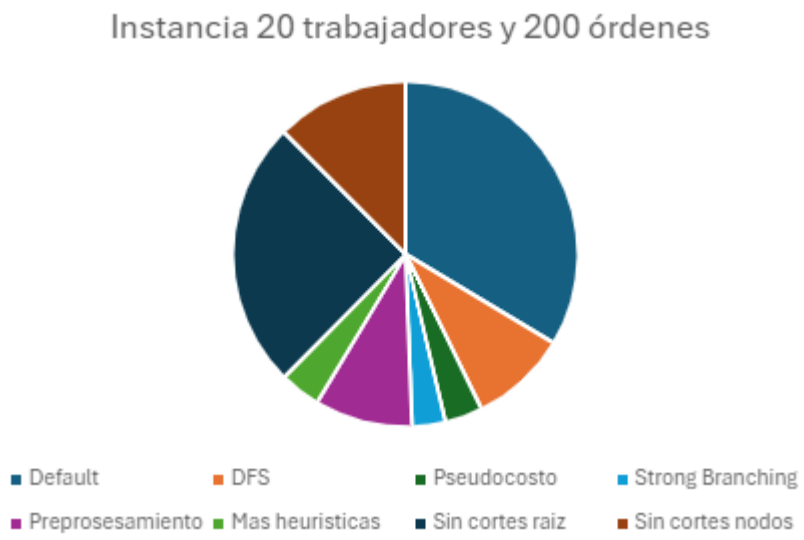

Figura 6: Comparación tiempos variando parámetros para la instancia 20-200

En este caso podemos observar, a partir del gráfico y los tiempos obtenidos, que a diferencia de lo que habiamos pensado que iba a ocurrir, los parámetros por default no resultaron tan buenos. Con otros parámetros como DFS, pseudocosto, Strong Branching y el caso con más heurísticas obtuvimos mejores tiempos, siendo el mejor, Strong Branching, con 1 minuto de ejecución (sin contar el tiempo de creación del LP). El que más tiempo requirió, además del default fue el caso sin cortes en la raíz.

2.5.2. 50 trabajadores y 200 órdenes

- Tiempo de creación: 0:1:27
- Tiempo Ejecución Default sin variar parámetros: 0:45:36
- Tiempo Depth-first search: 16:55:42
- Tiempo con Pseudocosto : 0:3:22
- Tiempo Strong branching : 0:3:11
- Tiempo Preprosesamiento : 0:7:05
- Tiempo con mas heuristicas: 0:44:08
- Tiempo termina de ejecutar sin los cortes en la raiz 0:34:06
- Tiempo termina de ejecutar sin los cortes en los nodos: 0:56:07

- Tiempo total: 20:24:58

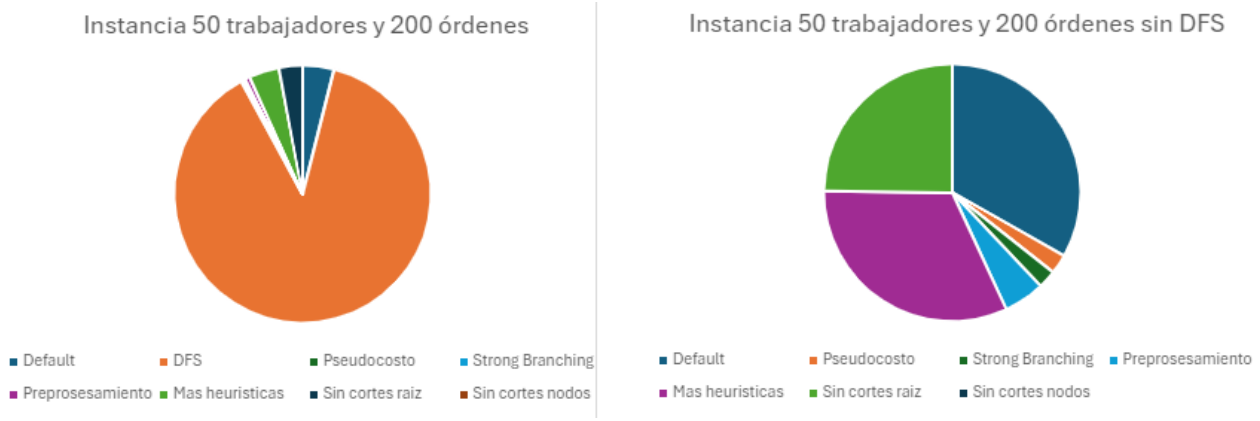

Figura 7: Comparación tiempos variando parámetros para la instancia 50-200

Para esta instancia, tuvimos que dejar muchas horas corriendo el programa ya que a diferencia de la instancia anterior, al agregarle un poco mas del doble de trabajadores obtuvimos una performace mucho peor en el tiempo de ejecución. Como se observa en el gráfico de la izquierda la gran mayoria del tiempo estuvo corriendo con la modificación del parámetro con DFS, con un tiempo de 16 horas y 56 minutos aproximadamente. Más aún, el que más tiempo requirió, sin contar DFS, es el caso sin cortes en los nodos, con un tiempo de 56 minutos. Pero si comparamos los tiempos de estos, DFS tardó 16 veces más en ejecutarse.

Para poder observar mejor los tiempos de ejecución decimos realizar el gráfico de la derecha sin DFS, donde se puede ver lo anteriormente mencionado, que el que más tiempo estuvo ejecutandose es el caso sin cortes en los nodos, junto con el caso sin cortes en la raíz y con más heurísticas. Mientras que los que menos tiempo estuvieron fueron Strong Branching y Pseudocosto con 3 minutos, 11 segundos y 3 minutos, 22 segundos respectivamente. Nuevamente estos casos resultan más rapidos que el tiempo de ejecución con los parámetros por default de cplex.

2.5.3. 100 trabajadores y 200 órdenes

- Tiempo de creación :0:4:54
- Tiempo Ejecución Default sin variar parámetros: 0:3:12
- Tiempo Depth-first search: 0:4:40
- Tiempo con Pseudocosto : 0:2:26
- Tiempo Strong branching : 0:3:18
- Tiempo Preprosesamiento : 0:6:31
- Tiempo con mas heuristicas: 0:2:36
- Tiempo termina de ejecutar sin los cortes en la raiz 0:2:37
- Tiempo termina de ejecutar sin los cortes en los nodos: 0:2:55
- Tiempo total: 1:27:44

Figura 8: Comparación tiempos variando parámetros para la instancia 100-200

Para esta instancia, obtuvimos resultados bastante diferentes a la instancia anterior ya que el parámetro que menos tiempo tardó fue con Pseudocosto con 2 minutos 26 segundos y le siguieron el caso con más heuristicas y sin cortes en la raíz. Por otro lado, el que más tiempo estuvo corriendo fue el que tiene Preprocesamiento con 6 minutos 31 segundos, el doble del tiempo que le llevó para el caso más rápido. En este caso, si bien no estba muy alejado, el default tampoco fue el mejor caso.

Si bien para este caso triplicamos la cantidad de trabajadores respecto a la instancia anterior, obtuvimos resultados mejores en el tiempo de ejecución. Con mejores nos referimos a que, si bien los que menos y más tiempo tardaron fueron diferentes a los casos anteriores, si miramos el tiempo total es significativamente menor y a su vez la distribución de los tiempos es más equitativa y razonable para los diferentes parámetros.

2.5.4. 150 trabajadores y 200 órdenes

- Tiempo de creación: 0:10:20
- Tiempo Ejecución Default sin variar parámetros: 0:10:51
- Tiempo Depth-first search: 0:11:13
- Tiempo con Pseudocosto : 0:08:58
- Tiempo Strong branching : 0:10:58
- Tiempo Preprosesamiento : 0:10:24
- Tiempo con mas heuristicas: 0:10:18
- Tiempo termina de ejecutar sin los cortes en la raiz 0:12:58
- Tiempo termina de ejecutar sin los cortes en los nodos: 0:10:18
- Tiempo total: 2:46:59

Figura 9: Comparación tiempos variando parámetros para la instancia 150-200

Notamos que en esta instancia los tiempos de ejecución se encuentran distribuidos casi equitativamente respecto al tiempo total.

La variación de parámetros que más tiempo requirió fue con el caso sin cortes en la raiz con 12 minutos 8 segundos y le siguió DFS. Mientras que, el que menos tiempo tardó fue el caso con Pseudocosto que tardó 8 minutos 58 segundos. Notemos que no hay mucha diferencia entre estos, aproximadamente 3 minutos.

Al igual que en los casos anteriores obtuvimos menor que tiempo que el default de cplex, sin variación de parámetros (por aproximadamente 1 minuto).

2.5.5. 200 trabajadores y 200 órdenes

- Tiempo de creación: 0:17:30
- Tiempo Ejecución Default sin variar parámetros: 1:58:19
- Tiempo Depth-first search: 0:23:20
- Tiempo con Pseudocosto : 0:18:46
- Tiempo Strong branching : 9:12:53
- Tiempo Preprosesamiento : 0:7:21
- Tiempo con mas heuristicas: 0:14:51
- Tiempo termina de ejecutar sin los cortes en la raiz 1:59:37
- Tiempo termina de ejecutar sin los cortes en los nodos: 0:14:58
- Tiempo total: 16:47:24

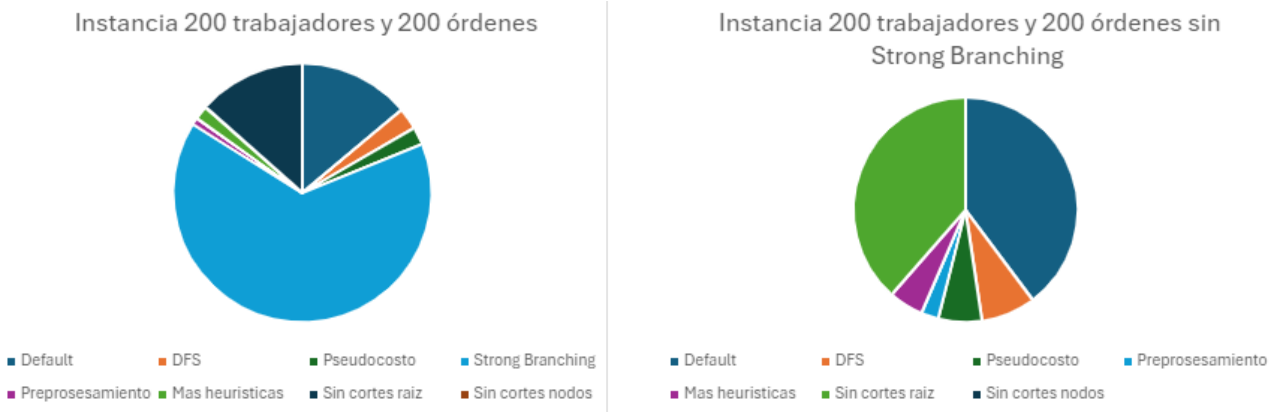

Figura 10: Comparación tiempos variando parámetros para la instancia 200-200

Para esta instancia notamos algo parecido a lo que ocurrió para la instancia 50 trabajadores, 200 órdenes.

Nuevamente obtuvimos una distribución de los tiempos muy desigual. Si bien unicamente aumentamos en un 25 % la cantidad de trabajadores, tuvimos un gran incremento en el tiempo total, respecto a la instancia anterior.

Como se puede observar en el gráfico de la izquierda la mayor cantidad de tiempo fue ocupada por la ejecución de Strong Branching con 16 horas 47 minutos y le siguió, como se puede observar en el gráfico de la derecha, el caso sin cortes en la raíz, con casi 2 horas. Notemos que nuevamente hay un gran diferencia entre estos tiempos, Strong Branching le llevó aproximadamente 8 veces más, que sin cortes en la raiz.

Para poder analizar el resto de parámetros que variamos, observamos el gráfico de la derecha. Notamos que el que menos tiempo tardó fue Preprocesamiento con 7 minutos aproximadamente y le siguió el caso con más heuristicas y sin cortes en los nodos con 15 minutos aproximadamente.

Nuevamente a diferencia de lo que esperabamos que ocurriera, el caso por default, sin variar parámetros, no fue mejor que variándolos. Más aún, si miramos el gráfico de la derecha se puede observar que, sin contar Strong Branching, fue de los que más tiempo tardó, junto con el caso sin cortes en la raíz, con un tiempo de casi 2 horas. Que si lo comparamos con el que mejor desempeño tuvo (preprocesamiento) hay una diferencia muy importante.

2.5.6. Conclusiones

En primer lugar notamos que el tiempo de creación del modelo va creciendo a medida que aumentamos la cantidad de trabajadores, como esperabamos.

Sobre los tiempos de ejecución por default también observamos que si bien la instancia de 50 trabajadores tardó mucho más que las siguientes, que contaban con mayor cantidad de trabajadores, en el resto de instancias, a medida que aumentamos la cantidad de trabajadores, aumenta el tiempo de ejecución por default de cplex.

Luego, notamos que un parámetro que nos dió buenos resultados es Pseudocosto ya que para las intancias con 20, 100 y 150 trabajadores fue el mejor, para la instancia con 50 trabajadores fue el 2do mejor (por 11 segundos de diferencia) y para el caso con 200 trabajadores fue el 4to mejor, pero sin mucha diferencia con el mejor resultado obtenido.

Por otro lado, un parámetro del que no obtuvimos buenos resultados fue el caso sin cortes en los nodos. Para la instancia 150 trabajadores fue el que más tiempo tardó, para 20 y 200 trabajadores fue el segundo peor parámetro y para 50 y 100 fue el tercer peor. Algo muy similar ocurre para el caso sin cortes en los nodos, obtuvimos resultados un poco mejores que el caso sin cortes en la raíz, pero fue de los peores resultados que obtuvimos (teniendo en cuenta las 5 instancias).

Otro parámetro con el que, en casi todas las instancias, no tuvimos buenos resultados es DFS, especialmente para el caso de 50 trabajadores, que le llevó 16 veces más tiempo que al segundo peor. Para el caso de 200 trabajadores obtuvo una performance bastante buena. Sin embargo, para los casos con menos trabajadores fue de los más lentos.

Si vemos que ocurrió con Strong Branching obtuvimos una gran diferencia a medida que aumentamos la cantidad de trabajadores. Cuando tenemos 20 y 50 trabajadores, fue de los más eficientes, a medida que aumentamos a 100 y 150 trabajadores, no se mantiene ni entre los mejores, ni peores. Pero al llegar a los 200 trabajadores fue el peor tiempo obtenido, superando 8 veces al siguiente parámetro más lento.

Con los parámetros preprocesamiento y el caso con más heurísticas no encontramos ninguna conlusión certera ya que para algunas instancias obtuvimos buenos resultados, pero para otras malos, sin ninguna relación con el aumento de los trabajadores.

Por último, como antes mencionamos, si bien creíamos que los mejores tiempos que ibamos a obtener era con los parámetros por default de cplex, sin variar los parámetros, no obtuvimos buenos resultados. En todos los casos resulta mejor variar algún parametro que usar el default. También en algunos casos resultó casi tan lento, como los parámetros que tomamos como los "peores".

Además, dada la disparidad de tiempos, creemos que no es es el mejor análisis mirar los promedios porque cada instancia tiene sus matices.

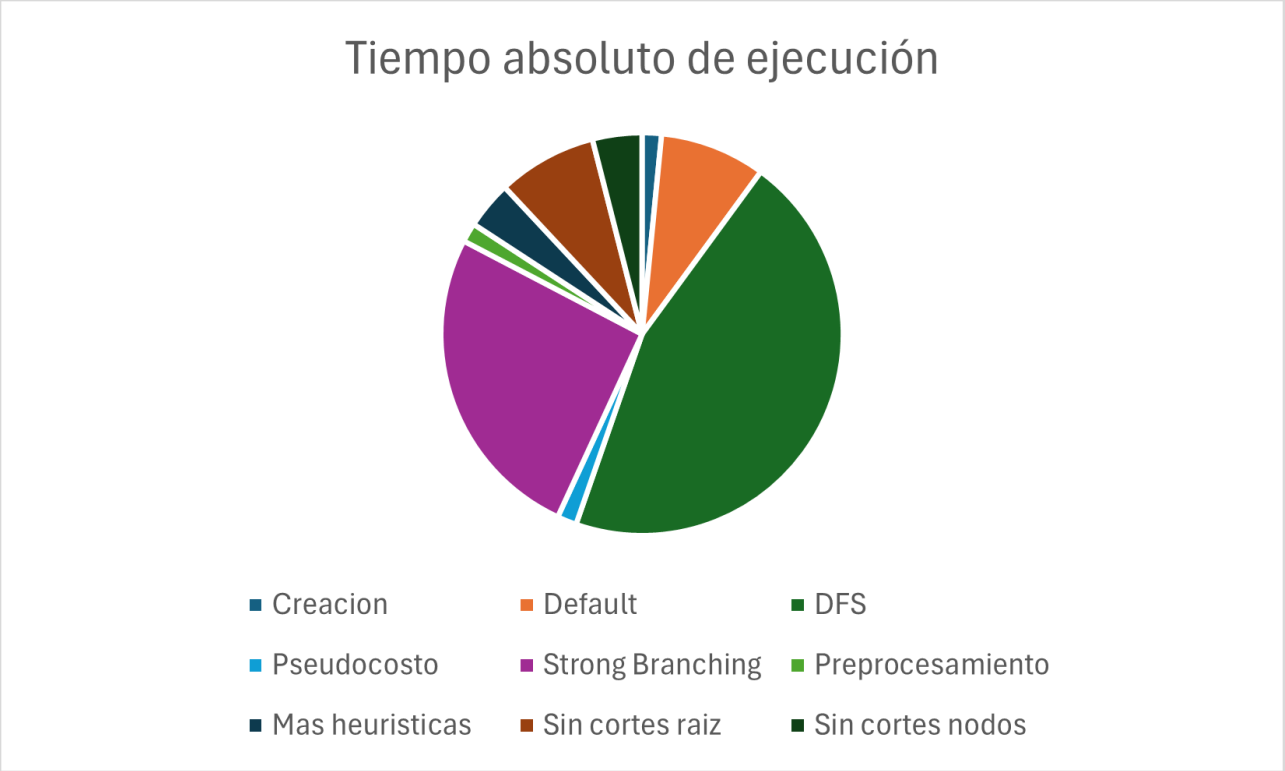

Figura 11: Tiempo absoluto de ejecución para cada parámetro

Mostramos los tiempos absolutos de ejecución para cada parámetro. Vemos que definitivamente DFS y Strong Branching acapararon la mayor parte de tiempo de ejecución y las que menos fueron las de más heurísticas, preprocesamiento y pseudocosto como fuimos viendo antes. ✓

2.6. Performance en distintas computadoras

Las comparaciones entre deseables-no deseables fueron corridas en la computadora de Victoria y las comparaciones de variaciones de parámetros en la computadora de Belén.

Las instancias sin deseables, con 40 conflictos y 40 tareas repetitivas, con la resolución por default de CPLEX fueron corridas en ambas computadoras.

Comparemos los tiempos que tardó la misma instancia en distintas computadoras para tener una idea de cómo podrían llegar a cambiar los resultados

	20t, 200o	50t, 200o	100t, 200o	150t, 200o	200t, 200o
Victoria	00:00:44	02:00:55	00:02:53	00:10:31	00:13:37
Belén	00:10:39	00:45:36	00:03:12	00:10:51	01:58:19

Cuadro 1: Tiempos de ejecución para diferentes instancias

No pensábamos que iba a variar tanto entre ambas computadoras. No encontramos criterio de cuál será más lenta o más rápida. Claramente ambas obtuvimos la misma función objetivo en todos los casos.

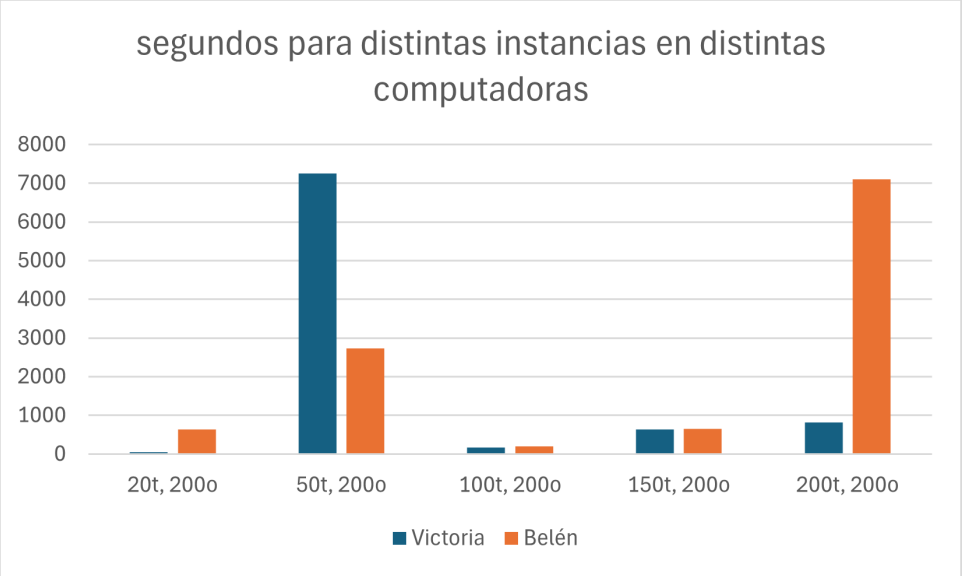

Figura 12: Tiempo absoluto en segundos de las instancias sin variación de parámetros y sin deseables.

Para cuando hay 20 o 200 trabajadores la computadora de Belén tarda (al rededor de) 10 veces más que la de Victoria. Para cuando hay 50 trabajadores la computadora de Belén tarda menos de la mitad que la de Victoria. Para cuando hay 100 o 150 trabajadores ambas computadoras tardan parecido.

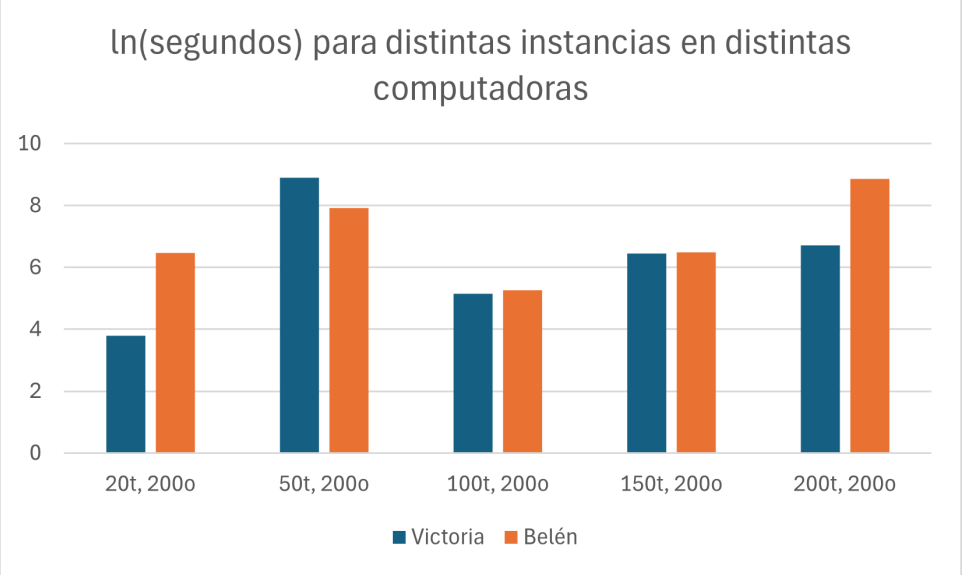

Figura 13: ln(segundos) de las instancias sin variación de parámetros y sin deseables.

No encontramos criterio. Por lo tanto es sumamente importante que las comparaciones (y por lo tanto conclusiones) de cada sección siempre se realicen entre instancias corridas por la misma computadora. Así lo hicimos.

¡Qué raro! Habría que ver los logs de los árboles para analizar mejor este comportamiento...-