FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 3

[91.04] Modelos y Optimización I

Primer Cuatrimestre de 2024

Alumno:	Vilardo, Ezequiel
Padron:	104980
Mail:	evilardo@fi.uba.ar

Índice

Primer entrega:

Me enfoque más en solucionar el problema que en el resultado, me apoyo en POO para hacerlo sencillo y óptimo en cuanto a rendimiento. Al principio arranque recorriendo el array de prendas en orden e incorporándose al lavado si eran compatibles.
Después lo optimize ordenando el array de prendas de las que más tardaban en lavarse a las que menos, lo volví a iterar y agregue que en caso de empate agregue que vaya primero la que menos restricciones tenga.

Segunda entrega:

En esta entrega me trabe un poco al principio porque no se me ocurría qué camino podría tomar, así que arranque cambiando los criterios de ordenamiento del array, en vez de por tiempo por incompatibilidades pero no obtuve un resultado mejor. Viendo esto cambie el algoritmo por uno similar a uno de coloreo de grafos que en principio me dio una solución peor, volviendo a ordenar por tiempos descendente e incompatibilidades descendente en caso de empate conseguí mejorar mi solución inicial y terminar top 2.

```
PS C:\Users\ezequ\Documents\GitHub\Lavado-Prendas> python .\resolucion.py
Se hicieron: 11 lavados.
El tiempo que tardo en lavar todo es: 123
El tiempo que tardo en correr el script es: 0.0019991397857666016 segs.
PS C:\Users\ezequ\Documents\GitHub\Lavado-Prendas> python .\resolucion.py
Se hicieron: 11 lavados.
El tiempo que tardo en lavar todo es: 123
El tiempo que tardo en correr el script es: 0.003509521484375 segs.
PS C:\Users\ezequ\Documents\GitHub\Lavado-Prendas> python .\resolucion.py
Se hicieron: 11 lavados.
El tiempo que tardo en lavar todo es: 123
El tiempo que tardo en correr el script es: 0.0029981136322021484 segs.
PS C:\Users\ezequ\Documents\GitHub\Lavado-Prendas> python .\resolucion.py
Se hicieron: 11 lavados.
El tiempo que tardo en lavar todo es: 123
El tiempo que tardo en correr el script es: 0.0039980411529541016 segs.
PS C:\Users\ezequ\Documents\GitHub\Lavado-Prendas> python .\resolucion.py
Se hicieron: 11 lavados.
El tiempo que tardo en lavar todo es: 123
El tiempo que tardo en correr el script es: 0.002508401870727539 segs.
PS C:\Users\ezequ\Documents\GitHub\Lavado-Prendas> python .\resolucion.py
Se hicieron: 11 lavados.
El tiempo que tardo en lavar todo es: 123
El tiempo que tardo en correr el script es: 0.0030069639205932617 segs.
PS C:\Users\ezequ\Documents\GitHub\Lavado-Prendas> python .\resolucion.py
Se hicieron: 11 lavados.
El tiempo que tardo en lavar todo es: 123
El tiempo que tardo en correr el script es: 0.0030000020980834961 segs.
PS C:\Users\ezequ\Documents\GitHub\Lavado-Prendas> python .\resolucion.py
Se hicieron: 11 lavados.
El tiempo que tardo en lavar todo es: 123
El tiempo que tardo en correr el script es: 0.00299835205078125 segs.
```

```
126   99 6
127   83 6
128   20 6
129   45 6
130   57 6
131   95 7
132   81 7
133   78 7
134   72 8
135   135 8
136   36 9
137   116 10
138   74 11
139
```

Tercer entrega:

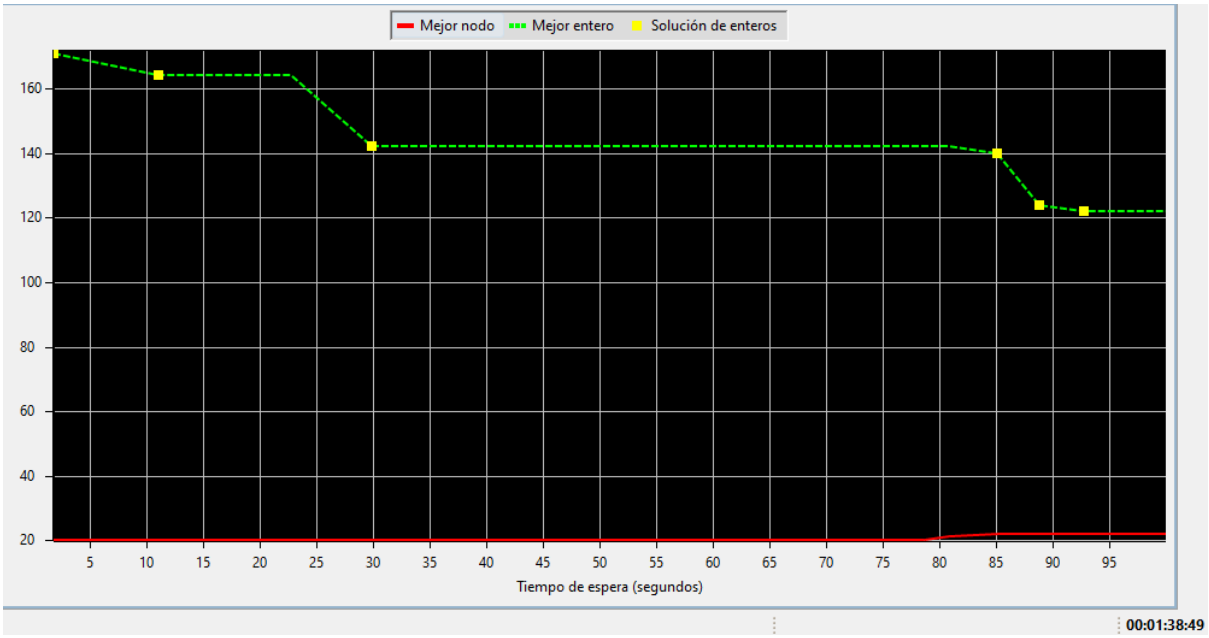
Primer paso:

Como primer paso corremos la heurística sobre la que venía iterando desde el TP1 Se puede observar que para el input dado en este TP se están usando 11 lavados, estos tardan 123 unidades de tiempo y al script le está tomando aproximadamente entre 0,002 a 0,004 segundos correr.

Segundo paso:

Realizamos una primera corrida con el algoritmo provisto en CPLEX, de la cual analizaremos sus primeros 90 segundos.
Analizo pestaña por pestaña

Estadísticos



Se puede observar cómo a lo largo de los 90 segundos se va consiguiendo soluciones donde el funcional es cada vez menor, arranca en el orden de los 171 para terminar en 122, consiguiendo ya una solución mejor a mi heurística.

Registros del motor

Apenas arranca sale por pantalla:

```
Version identifier: 22.1.1.0 | 2022-11-27 | 9160aff4d
Legacy callback                                  pi
Tried aggregator 1 time.
MIP Presolve eliminated 120467 rows and 0 columns.
MIP Presolve modified 12013 coefficients.
Reduced MIP has 34783 rows, 19182 columns, and 121915 nonzeros.
Reduced MIP has 19044 binaries, 138 generals, 0 SOSs, and 0 indicators.
Presolve time = 0,22 sec. (253,66 ticks)
Found incumbent of value 2760,000000 after 0,31 sec. (403,01 ticks)
Probing time = 0,08 sec. (13,80 ticks)
Tried aggregator 1 time.
Detecting symmetries...
Reduced MIP has 34783 rows, 19182 columns, and 121915 nonzeros.
Reduced MIP has 19044 binaries, 138 generals, 0 SOSs, and 0 indicators.
Presolve time = 0,36 sec. (391,75 ticks)
Probing time = 0,08 sec. (13,12 ticks)
Clique table members: 15739.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 12 threads.
Root relaxation solution time = 0,50 sec. (488,70 ticks)
```

Que nos cuenta un poco qué versión estamos corriendo, fecha de esta y un identificador de versión, mas abajo nos dice que aplicó una estrategia de agregación con la que eliminó 120467 filas y 0 columnas simplificando el problema, además menciona que el problema quedó reducido a 34783 filas, 19182 columnas y 121915 elementos diferentes a 0.
También quiso identificar simetrías para eliminarlas pero no encontró por lo que el problema quedó dispuesto de la misma forma que antes.
Después aclara que métodos de búsqueda utilizada, en este caso búsqueda dinámica, además de la cantidad de threads que utilizara en una corrida determinista.
La salida tiene además otra tabla:

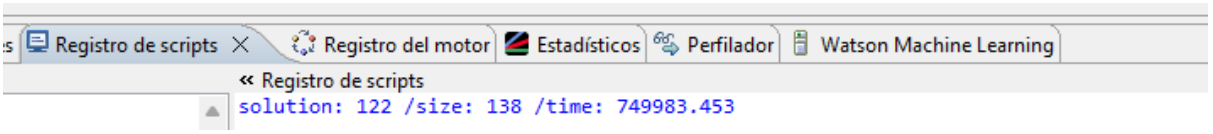
```
        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                          2760,0000        0,0000           100,00%
*     0+    0                          1467,0000        0,0000           100,00%
*     0+    0                           171,0000        0,0000           100,00%
      0     0       20,0000  4740       171,0000       20,0000       11   88,30%
*     0+    0                           164,0000       20,0000           87,80%
      0     0       20,0000  1852       164,0000     Cuts: 131     9640   87,80%
      0     0       20,0000  2161       164,0000    Cuts: 1676    17368   87,80%
      0     0       20,0000  2068       164,0000     Cuts: 147    29708   87,80%
*     0+    0                           154,0000       20,0000           87,01%
*     0+    0                           152,0000       20,0000           86,84%
*     0+    0                           147,0000       20,0000           86,39%
*     0+    0                           144,0000       20,0000           86,11%
*     0+    0                           142,0000       20,0000           85,92%
      0     0  -1,00000e+75     0       142,0000       20,0000    29708   85,92%
      0     0       20,0000  1919       142,0000    Cuts: 1709    36509   85,92%
      0     0       20,0000  1618       142,0000     Cuts: 156    42826   85,92%
      0     0       20,0000  2048       142,0000    Cuts: 1546    50451   85,92%
Heuristic still looking.
      0     2       20,0000   940       142,0000       20,0000    50451   85,92%
Elapsed time = 52,59 sec. (86093,18 ticks, tree = 0,02 MB, solutions = 9)
      1     3       37,0000  1017       142,0000       20,0000    56376   85,92%
```

Donde se hace mención del nodo actual de búsqueda y de los restantes, el valor objetivo que estamos buscando y IInf que es la cantidad de restricciones violadas por la solución.
Está presente también *Best Integer* que es el mejor valor posible que entramos para el funcional hasta ahora y *Best Bound* que es una estimación de una posible solución que “podemos” encontrar.
Y cerramos con *ItCnt* que es un contador de interacciones y *Gap* que es la diferencia entre el mejor valor del funcional que encontramos y el ideal estimado.
Podemos ver que la primera solución encontrada está en 2760 y rápidamente evolucionamos hasta un valor más cercano a la heurística nuestra, pasando a 171 y mejorando a lo largo que aumentan las iteraciones.

```
    601   377      121,0000   141      124,0000       21,9753  1055267   82,28%
*   617   396      integral     0      123,0000       21,9753  1097552   82,13%
*   652   400      integral     0      122,0000       21,9753  1106265   81,99%
    658   379       46,2868   904      122,0000       21,9753  1061607   81,99%
    679   392       46,2868   896      122,0000       21,9753  1168565   81,99%
    702   395       88,8299   786      122,0000       21,9753  1161078   81,99%
    736   417       56,0000   901      122,0000       21,9753  1253466   81,99%
Elapsed time = 96,23 sec. (116015,76 ticks, tree = 4,27 MB, solutions = 16)
    759   426       82,6004   925      122,0000       21,9753  1287965   81,99%

Implied bound cuts applied:  5861
Zero-half cuts applied:  300
Gomory fractional cuts applied:  4

Root node processing (before b&c):
  Real time             =   52,36 sec. (85899,48 ticks)
Parallel b&c, 12 threads:
  Real time             =   45,77 sec. (31473,85 ticks)
  Sync time (average)   =    9,73 sec.
  Wait time (average)   =    0,00 sec.
                          ------------
Total (root+branch&cut) =   98,12 sec. (117373,33 ticks)
```

Para terminar finalmente con 122 a los 98 segundos, podemos ver que la encontró en el nodo 652, tiene 0 restricciones incumplidas y le costo 1106265 iteraciones y todavía considera posible hallar un óptimo de valor 21,9753.
Sobre el final menciona cuántos cortes se hicieron y de qué tipo son además del tiempo total de ejecución.

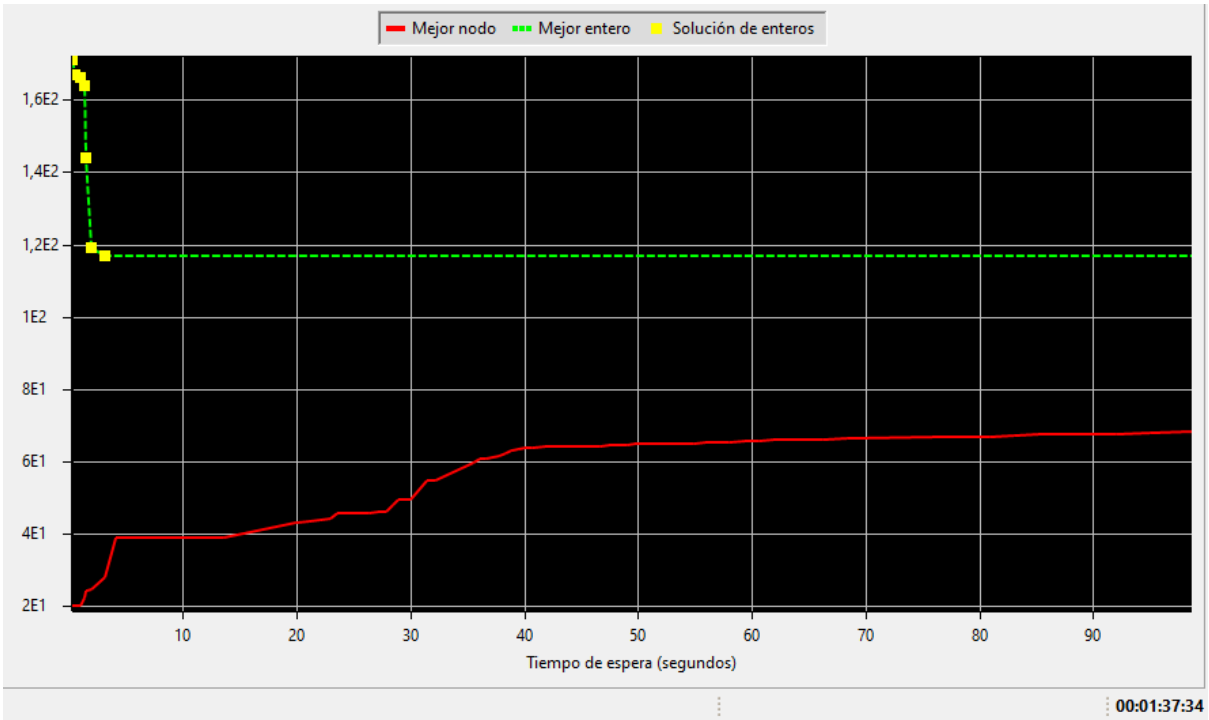
Registro de scripts



Aca termina haciendo un resumen de la mejor solución válida encontrada, además del tamaño de la búsqueda y el tiempo consumido.

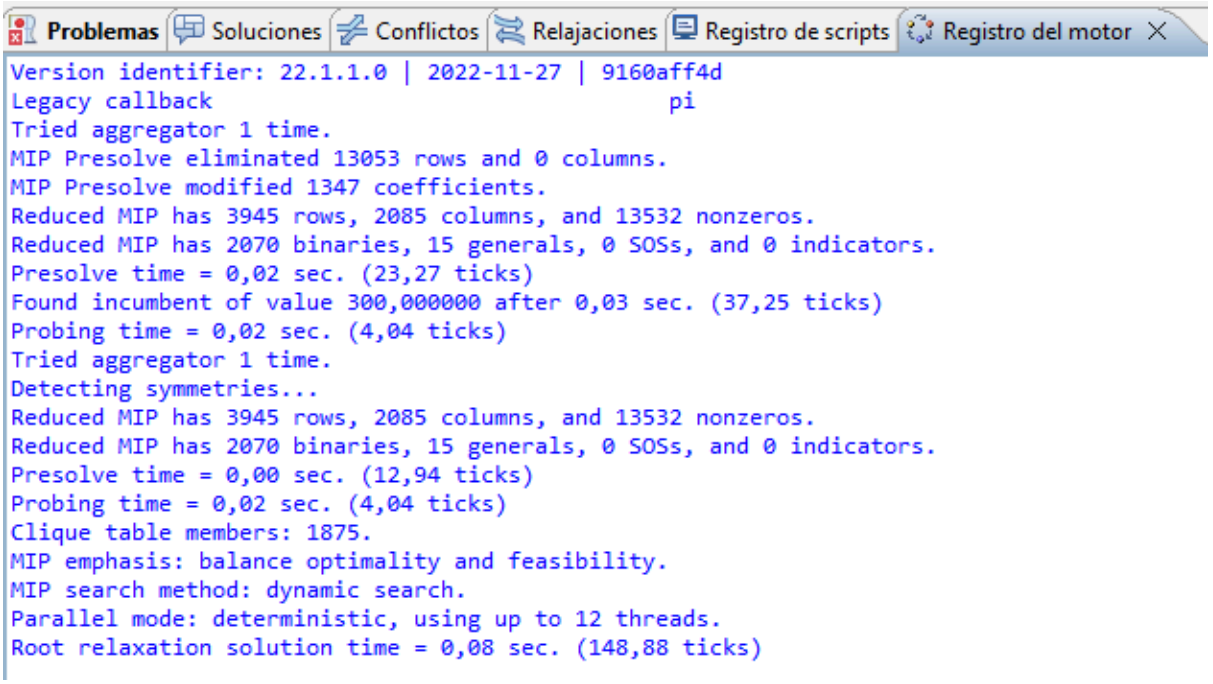
Tercer paso:

Estadísticos



Acotando el problema que solo pueda usar 15 colores se observa que la mejora en el funcional es mucho más explosivo que antes, lograran llegar al valor óptimo de 117, a partir del cual queda estancado.

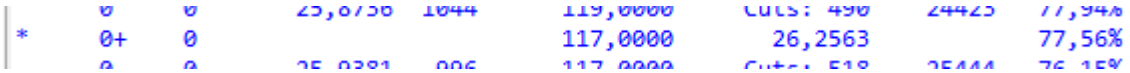
Registros del motor



La principal diferencia que se puede encontrar es que el modelo a solucionar ya es mucho más chico, teniendo solo 3945 filas, 2085 columnas y 13532 valores diferentes a 0 comparados con las anteriores 34783 filas, 19182 columnas y 121915 elementos diferentes a 0.
Esto permitió no solo reducir la fase de armado del problema sino la cantidad de cortes a realizar.

```
Clique cuts applied:  18
Implied bound cuts applied:  382
Flow cuts applied:  175
Mixed integer rounding cuts applied:  542
Zero-half cuts applied:  97
Gomory fractional cuts applied:  77

Root node processing (before b&c):
  Real time             =    4,03 sec. (6430,36 ticks)
Parallel b&c, 12 threads:
  Real time             =   93,22 sec. (85365,78 ticks)
  Sync time (average)   =   16,39 sec.
  Wait time (average)   =    0,02 sec.
                          ------------
Total (root+branch&cut) =   97,25 sec. (91796,13 ticks)
```

La solución pasó de 122 a 117



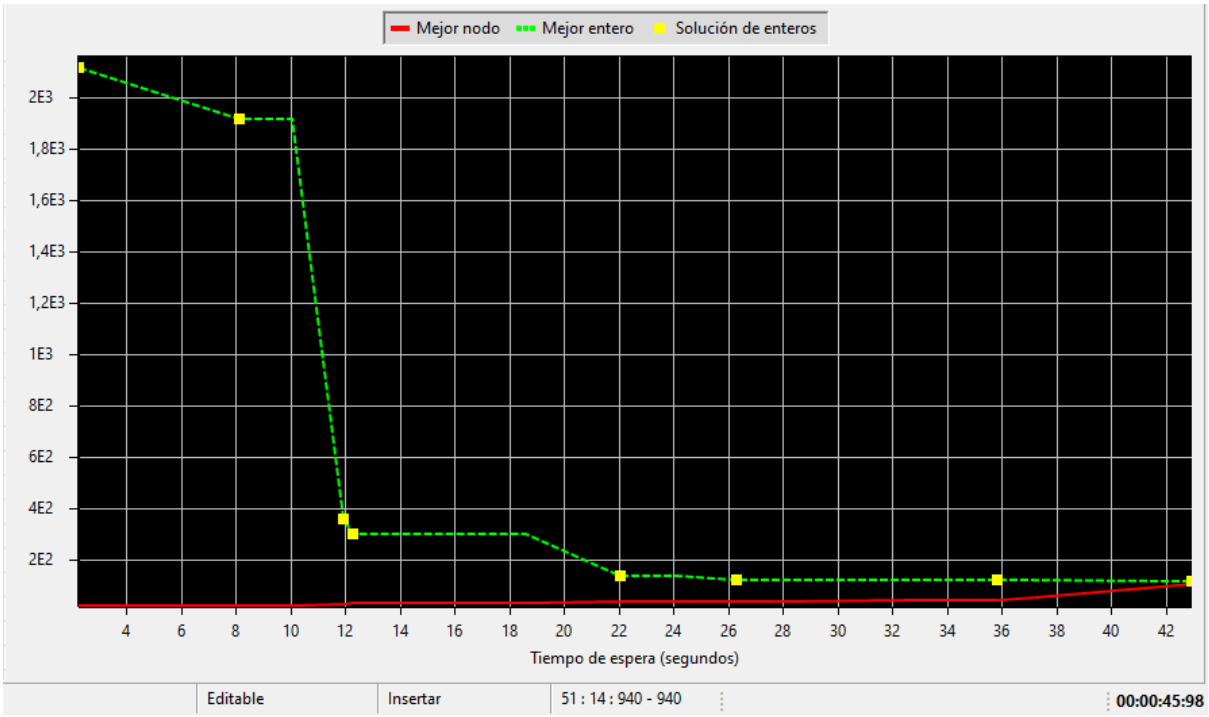
además de reacomodar la estimación de la posible mejor solución de los aprox 22 a 68,0795.

```
54280 44438       94,5813   341      117,0000       68,0795  5829463   41,81%
```

Cuarto paso:

Estadísticos

Volviendo atrás la restricción de solo usar 15 colores pero ahora con la restricción de simetría activada.



Si bien ya no se ve esa explosion al principio ya que el armado es igual de complejo al algoritmo original pero se resuelve mucho más rápido, ya es un problema que termina de ejecutarse en 45,98 segundos, alcanzando finalmente una solución óptima de 117.

Registros del motor

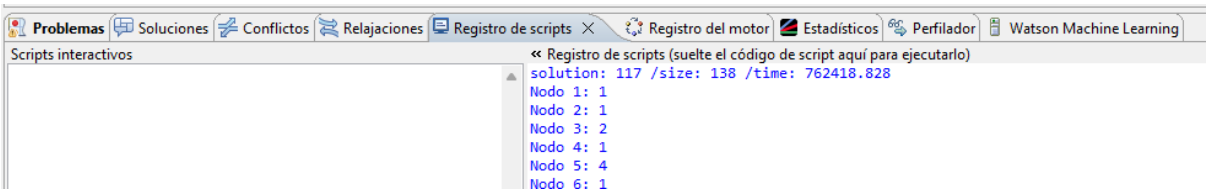
```
Version identifier: 22.1.1.0 | 2022-11-27 | 9160aff4d
Legacy callback                                  pi
Tried aggregator 1 time.
MIP Presolve eliminated 120489 rows and 0 columns.
MIP Presolve modified 11991 coefficients.
Reduced MIP has 34898 rows, 19182 columns, and 122062 nonzeros.
Reduced MIP has 19044 binaries, 138 generals, 0 SOSs, and 0 indicators.
Presolve time = 0,22 sec. (253,34 ticks)
Found incumbent of value 2760,000000 after 0,36 sec. (465,00 ticks)
Probing time = 0,08 sec. (13,81 ticks)
Tried aggregator 1 time.
Detecting symmetries...
Reduced MIP has 34898 rows, 19182 columns, and 122062 nonzeros.
Reduced MIP has 19044 binaries, 138 generals, 0 SOSs, and 0 indicators.
Presolve time = 0,09 sec. (144,06 ticks)
Probing time = 0,08 sec. (13,14 ticks)
Clique table members: 15717.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 12 threads.
Root relaxation solution time = 1,16 sec. (1059,19 ticks)
```

Vemos que el tamaño del problema vuelve a ser similar al comienzo en cuanto a filas,columnas y valores diferentes a 0.

```
Elapsed time = 41,05 sec. (57518,28 ticks, tree = 9,49 MB, solutions = 22)
   3996  1942      117,0000    99      118,0000      103,9624   236801   11,90%
*  4003  1935      integral     0      117,0000      103,9624   243270   11,14%
   5387  1615    infeasible            117,0000      108,6405   297345    7,14%
   7204   843      115,0000   108      117,0000      113,1297   336454    3,31%

Clique cuts applied:  10
Implied bound cuts applied:  1436
Mixed integer rounding cuts applied:  5
Zero-half cuts applied:  35
Gomory fractional cuts applied:  24

Root node processing (before b&c):
  Real time             =   36,73 sec. (54096,14 ticks)
Parallel b&c, 12 threads:
  Real time             =    8,81 sec. (6593,63 ticks)
  Sync time (average)   =    2,50 sec.
  Wait time (average)   =    0,00 sec.
                          ------------
Total (root+branch&cut) =   45,55 sec. (60689,78 ticks)
```

Pero al eliminar simetrías nos permite bajar exponencialmente la cantidad de interacciones necesarias ya que no debíamos pasar por la búsqueda de soluciones con un resultado similar a sus simétricos. Es por esto que nos basto con 336454 iteraciones vs 1106265 del problema original.

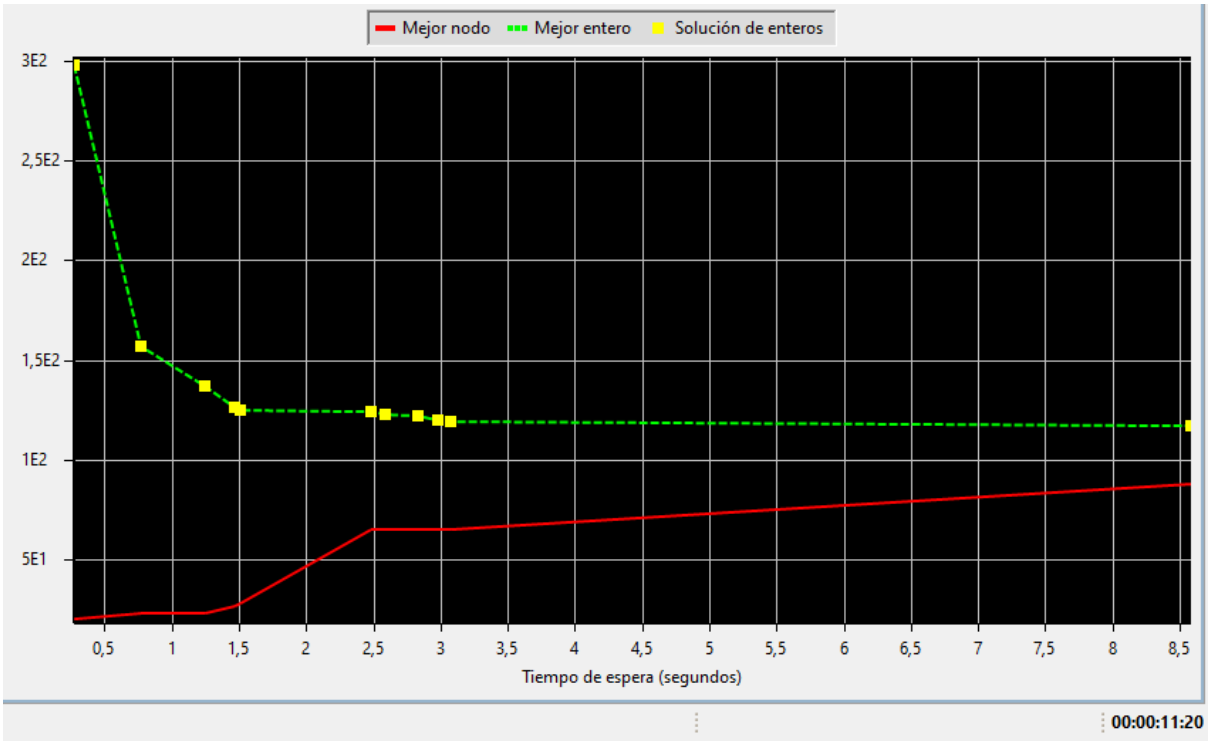
Registros de scripts



Ya aparece una distribución válida para las prendas en los diferentes lavados.

Quinto paso:

Estadísticos



Al volver a incorporar el límite de 15 colores sumado a la simetría tenemos las 2 bondades descritas en los pasos anteriores juntas, el armado es rápido y al ahorrarnos el verificar las simetrías la búsqueda también es rápida, permitiéndonos alcanzar la solución en solo 11,2 segundos.

Registros del motor

```
Version identifier: 22.1.1.0 | 2022-11-27 | 9160aff4d
Legacy callback                                 pi
Tried aggregator 1 time.
MIP Presolve eliminated 13050 rows and 0 columns.
MIP Presolve modified 1350 coefficients.
Reduced MIP has 3962 rows, 2085 columns, and 13578 nonzeros.
Reduced MIP has 2070 binaries, 15 generals, 0 SOSs, and 0 indicators.
Presolve time = 0,03 sec. (23,33 ticks)
Found incumbent of value 300,000000 after 0,05 sec. (53,08 ticks)
Probing time = 0,02 sec. (4,05 ticks)
Tried aggregator 1 time.
Detecting symmetries...
Reduced MIP has 3962 rows, 2085 columns, and 13578 nonzeros.
Reduced MIP has 2070 binaries, 15 generals, 0 SOSs, and 0 indicators.
Presolve time = 0,00 sec. (12,66 ticks)
Probing time = 0,02 sec. (4,28 ticks)
Clique table members: 1878.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 12 threads.
Root relaxation solution time = 0,08 sec. (136,45 ticks)
```

Nuevamente el problema vuelve a ser de menor complejidad en cuanto a su tamaño por la restricción de colores.

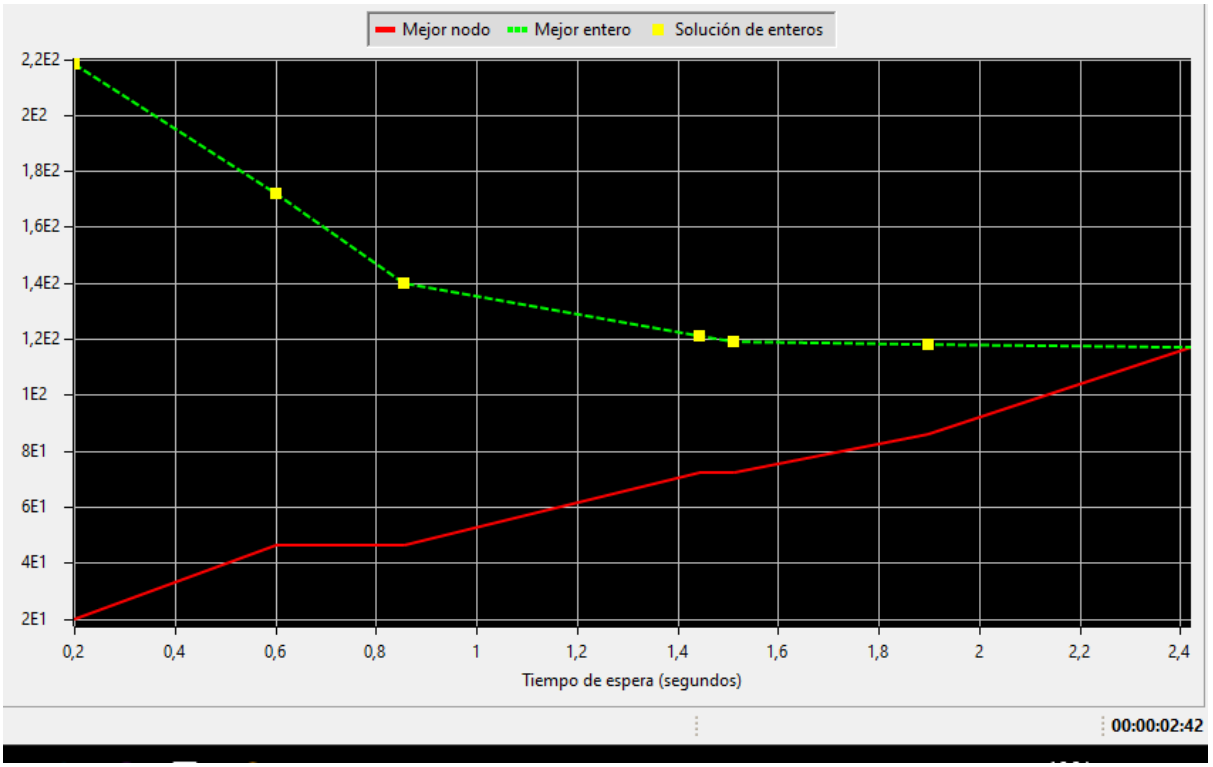
```
   6011     2       87,8279   391      117,0000       87,8279   224914   24,93%
Elapsed time = 8,52 sec. (9787,65 ticks, tree = 0,02 MB, solutions = 19)
   6031     7       88,6023   333      117,0000       88,2773   230887   24,55%
   6864   335       99,6157   314      117,0000       89,1219   296625   23,83%
   8280   527      114,0000   109      117,0000      109,8111   381184    6,14%

Clique cuts applied:  7
Implied bound cuts applied:  6
Flow cuts applied:  59
Mixed integer rounding cuts applied:  152
Zero-half cuts applied:  41
Lift and project cuts applied:  1
Gomory fractional cuts applied:  21

Root node processing (before b&c):
  Real time             =    2,17 sec. (3100,18 ticks)
Parallel b&c, 12 threads:
  Real time             =    8,95 sec. (9804,24 ticks)
  Sync time (average)   =    1,53 sec.
  Wait time (average)   =    0,01 sec.
                          ------------
Total (root+branch&cut) =   11,12 sec. (12904,41 ticks)
```

Además de llegar a la solución final con muy pocos cortes comparado a las ejecuciones anteriores.

Sexto paso:

Estadísticos

Ya con la limitación a 11 colores la solución llega a su solución final de forma muy rápida



Bastando con solo 2,42 segundos para alcanzar el óptimo y completa la búsqueda. Permitiendo encontrar la solución en la etapa de armado previo al branch & cut.

Registros del motor

Al bajar en 1 color, nuevamente la solución es menor complejidad al paso anterior.

```
MIP Presolve eliminated 9577 rows and 0 columns.
MIP Presolve modified 983 coefficients.
Reduced MIP has 2935 rows, 1529 columns, and 9909 nonzeros.
Reduced MIP has 1518 binaries, 11 generals, 0 SOSs, and 0 indicators.
```

```
        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                          220,0000        0,0000           100,00%
*     0+    0                          218,0000        0,0000           100,00%
      0     0       20,0000   914      218,0000       20,0000     1394   90,83%
*     0+    0                          172,0000       20,0000            88,37%
      0     0       29,5206   794      172,0000     Cuts: 833     3326   73,09%
*     0+    0                          140,0000       46,2868            66,94%
      0     0       37,0000   727      140,0000     Cuts: 564     4408   66,94%
*     0+    0                          121,0000       46,2868            61,75%
      0     0       41,7808   657      121,0000     Cuts: 833     8851   40,20%
*     0+    0                          119,0000       72,3584            39,19%
      0     0  -1,00000e+75     0      119,0000       72,3584     8851   39,19%
*     0+    0                          118,0000       72,3584            38,68%
      0     0       49,9861   624      118,0000     Cuts: 833    11556   27,15%
*     0+    0                          117,0000       85,9632            26,53%
      0     0        cutoff           117,0000      117,0000    15356    0,00%
Elapsed time = 2,34 sec. (2556,18 ticks, tree = 0,01 MB, solutions = 8)
```

Puede observarse cómo llega a la solución óptima en la fase de armado, además de por fin llegar al 0% en el gap que era la diferencia entre el mejor valor encontrado y un estimado del mejor valor posible. Brecha que no habíamos podido llegar a cerrar en las ejecuciones anteriores.
En este caso nos alcanzó con sólo 15356 iteraciones para alcanzar el final.

```
Clique cuts applied:  9
Implied bound cuts applied:  380
Mixed integer rounding cuts applied:  497
Zero-half cuts applied:  288
Multi commodity flow cuts applied:  24
Gomory fractional cuts applied:  1

Root node processing (before b&c):
  Real time             =    2,34 sec. (2556,36 ticks)
Parallel b&c, 12 threads:
  Real time             =    0,00 sec. (0,00 ticks)
  Sync time (average)   =    0,00 sec.
  Wait time (average)   =    0,00 sec.
                          ------------
Total (root+branch&cut) =    2,34 sec. (2556,36 ticks)
```

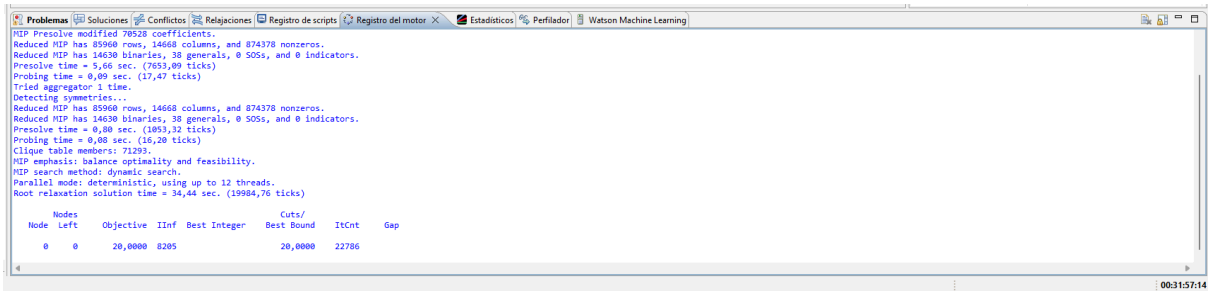
Séptimo paso:

La heurística armada para los tps 1 y 2 no queda tan mal parada en contra de la versión en CPLEX ya que consumiendo solo el 0,17% del tiempo de esta logra un resultado un solo un 5,12% peor, faltaría más experimentación para tomar una decisión final, por ejemplo qué sucede si aumentamos la cantidad de prendas y de restricciones, ¿se mantendrá la diferencia del tiempo de ejecución en forma de la heurística%? ¿La solución de la heurística empeora al aumentar la complejidad del problema?

Es por eso que le quise dar una vuelta más de rosca a esta comparación y tome el input del TP2 que en mi solución obtiene un resultado de 38 lavados en los que tardaba 476 unidades de tiempo, con un tiempo de corrida de alrededor de 0,1 segundos.

```
Se hicieron: 38 lavados.
El tiempo que tardo en lavar todo es: 476
El tiempo que tardo en correr el script es: 0.09351038932800293 segs.
```

Y lo metí en el .dat de CPLEX acomodando con un script para poder correrlo y lo deje corriendo con las optimizaciones de la simetría y de 38 lavados que era lo hallado por mi heurística a ver que tanto podía mejorar mi solucion y cuanto tiempo le costaría.



Luego de 30 minutos de no ver solución, trabajé un poco más buscando un valor para el límite de colores con el cual en algún momento me mostrase una solución óptima pero no tuve suerte, los tiempos parecían ser demasiado largos, lo cual tenía sentido al tratarse un problema NP pero quería obtener más información para poder comparar contra mi heurística. Explore también si existía la posibilidad de correr CPLEX usando la GPU, conocidas por ser mejores a la hora del cálculo al menos en criptografía, pero no encontré implementaciones.
Por lo que concluí que una buena heurística tiene la ventaja para este tipo de problemas.