MODELOS Y OPTIMIZACIÓN I [71.14]

ENTREGA FINAL

$2^{do} Cuatrimestre$ 2022

Alumna: Mazza Reta, Tizziana

Padrón: 101715

Mail: tmazzar@fi.uba.ar

Git: github.com/tizziana/Modelos-y-Optimizacion-I

Índice

Presentación heurística

Modelo

A continuación, tenemos el modelo que se utilizó para la tercer entrega de la materia.

Hipótesis:

- No se descompone ninguna maquinaria mientras se lava
- El tiempo de lavado tiene en cuenta sacar y poner la ropa
- No se tiene en cuenta el tiempo entre un lavado y otro

Variables

- Cantidad total de prendas que hay → n

$$i = 0, \dots, n$$

$$j = 0, \dots, n$$

$$T_i = Tiempo\ de\ lavado\ de\ la\ prenda\ i$$

$$X_j = Tiempo\ del\ lavado\ j$$

$$X_{ij} = Tiempo\ de\ lavado\ de\ la\ prenda\ i\ en\ el\ lavado\ j$$

- Para saber en qué lavado va cada prenda:

$$Y_{ij} = \begin{cases} 1 & Si\ la\ prenda\ i\ entra\ en\ el\ lavado\ j \\ 0 & Si\ no \end{cases}$$

- Por ahora contamos con que hay n lavados, y eso sería cierto en el peor de los casos, o sea cuando hay una prenda por cada lavado. Pero si tenemos más de una prenda por lavado hay, entonces, menos lavados que numero de prendas. Por lo tanto, tenemos la siguiente variable bivalente:

$$W_j = \begin{cases} 1 & Si\ el\ lavado\ j\ se\ realiza \\ 0 & Si\ no \end{cases}$$

- Ahora tenemos que agregar el tema de las incompatibilidades entre prendas, es decir, que las prendas que son incompatibles no pueden ir en el mismo lavado, por lo que necesitamos otra variable para asegurarnos de que esto no suceda:

$$E_{uv} = \begin{cases} 1 & Si\ la\ prendas\ u\ y\ v\ son\ incompatibles \\ 0 & Si\ no \end{cases}$$

Modelo

$$Z(min) = \sum_{j=1}^{n} X_j$$

- Si W_j está en cero (o sea, que ese lavado no se va a realizar) => Y_{ij} debe ser cero ∀i:

  $Y_{ij} \leq W_j$ => ninguna prenda puede ir a ese lavado.

- Una prenda sólo puede ir a un lavado $\sum_{j=1}^{n} X_{ij} = 1 \quad \forall i$
- Las prendas u y v incompatibles no pueden encontrarse en el mismo lavado:

  $$Y_{uj} + Y_{uj} \leq E_{uv} + 1,$$

  y las prendas compatibles pueden, o no, ir en el mismo lavado.

- Necesitamos que cada lavado tenga el valor de la prenda que lleva más tiempo de lavado

  $$X_{ij} = T_i * Y_{ij}$$

  $$X_{ij} \leq X_j \leq X_{ij} + M(1 - Max_{ij}),$$

  Siendo

  $Max_{ij} = \begin{cases} 1 & Si\ la\ prenda\ i\ del\ lavado\ j\ es\ la\ que\ tiene\ maximo\ tiempo\ de\ lavado \\ 0 & Si\ no \end{cases}$

- Entonces $\sum_{i=1}^{n} Max_{ij} = 1$

Soluciones

Heurística utilizada en la entrega anterior

Con el método utilizado en las entregas anteriores a esta, lo que se quiso hacer fue encontrar una solución que se acerque lo más posible al óptimo, consiguiendo así, con el problema número cuatro, una cantidad de 11 lavados, en 123 tiempo de lavado.

```
23  def main():
24      lavanderia = create_grafo()
25      generate_solution(solucion_lavados(lavanderia))
26      tiempo_total, lavados_totales = tiempo_total_lavados(lavanderia, CREATE_FILE)
27      print('El tiempo que tarda en lavar todas las prendas es de ' + str(tiempo_total) +
28      ', y se hace en '+ str(lavados_totales) + ' lavados')
29
30
31  main()
```

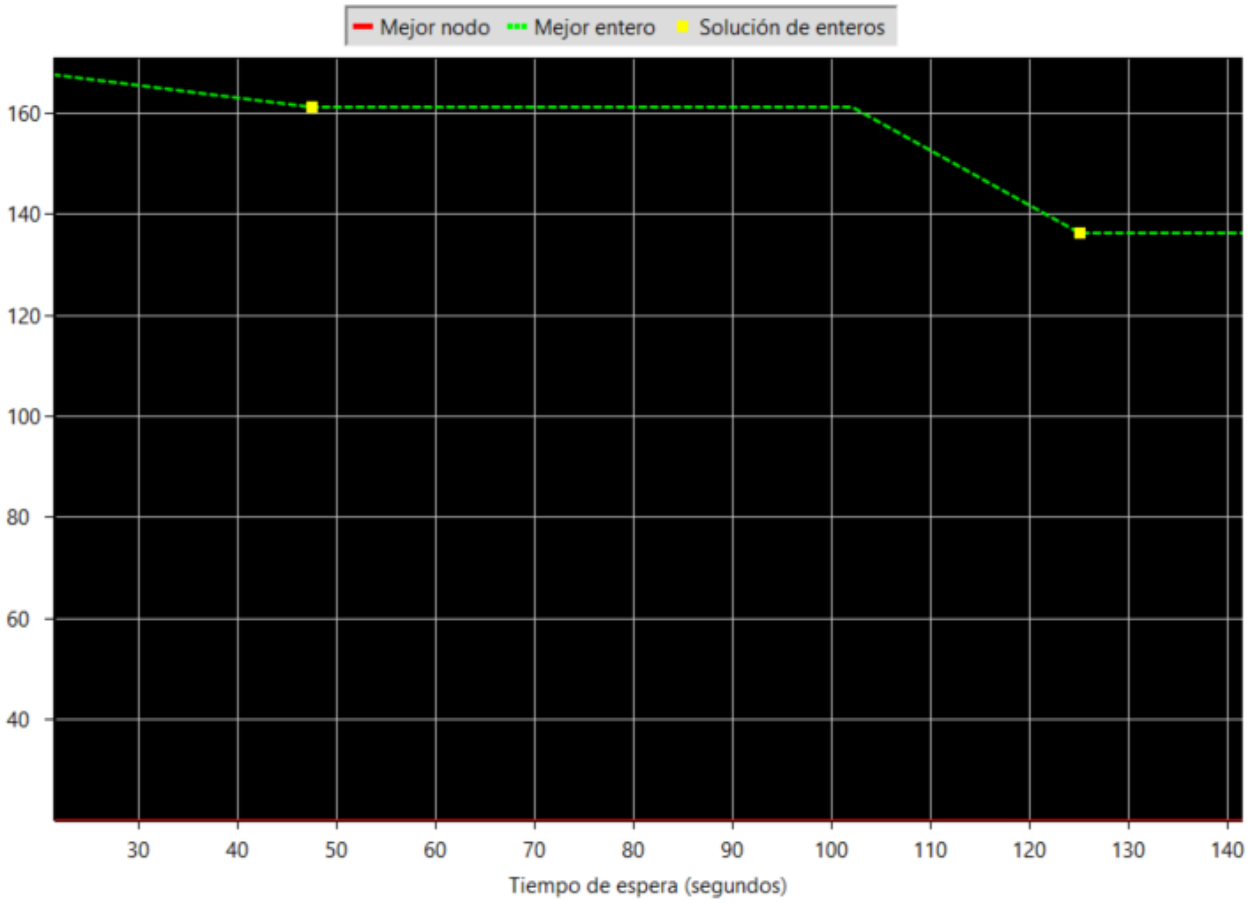
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
El tiempo que tarda en lavar todas las prendas es de 123, y se hace en 11 lavados
PS C:\Users\004559613\Desktop\Titi\FIUBA\Modelos y Optimizacion\Trabajos Practicos\Modelos-y-Optimizacion-I>
```

Este método tardó, como mucho, un segundo en correr.

CPLEX: Programación lineal

Modelo simple

Observamos la estadística de la solución en los primeros segundos:



Al llegar a los 90 segundos se mantiene un tiempo con el mismo resultado que venía teniendo poco antes de los 50 segundos. No mejora muy rápidamente.

La solución que nos da casi al instante de correr el script:

```
Root relaxation solution time = 2,06 sec. (475,39 ticks)

        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  4286      171,0000       20,0000       11   88,30%
*     0+    0                        161,0000       20,0000           87,58%
      0     0       20,0000  1567      161,0000     Cuts: 181     6970   87,58%
      0     0       20,0000  2225      161,0000    Cuts: 1498    17262   87,58%
      0     0       20,0000  1187      161,0000      Cuts: 58    19066   87,58%
*     0+    0                        143,0000       20,0000           86,01%
*     0+    0                        136,0000       20,0000           85,29%
      0     0  -1,00000e+75     0      136,0000       20,0000    19066   85,29%
      0     0       20,0000  1803      136,0000    Cuts: 1091    29019   85,29%
      0     2       20,0000   944      136,0000       20,0000    29019   85,29%
Elapsed time = 244,13 sec. (69300,89 ticks, tree = 0,02 MB, solutions = 6)
      1     3       37,0000  1063      136,0000       20,0000    35579   85,29%
      2     3       20,0000  2187      136,0000       20,0000    33692   85,29%
      3     4       37,0000   928      136,0000       20,0000    38871   85,29%
      4     4       46,2868   803      136,0000       20,0000    46037   85,29%
      5     6       37,0000   936      136,0000       20,0000    46824   85,29%
      6     7       46,2868  1119      136,0000       20,0000    56851   85,29%
      8     6       46,2868  1055      136,0000       20,0000    49372   85,29%
     13     9       49,5795  1055      136,0000       20,0000    62465   85,29%
     15    14       56,0000   700      136,0000       20,0000    86403   85,29%
     20    17       59,0000   547      136,0000       20,0000    95587   85,29%
Elapsed time = 304,55 sec. (76657,74 ticks, tree = 0,05 MB, solutions = 6)
     27     8       46,2868  1205      136,0000       20,0000    66480   85,29%
     34    22       40,9697   928      136,0000       20,0000   138271   85,29%
     43    36       65,0000   612      136,0000       20,0000   206302   85,29%
     57    45       59,4062   536      136,0000       20,0000   234015   85,29%
     75    49       53,0000   990      136,0000       20,0000   250456   85,29%
     86    50       53,0000   813      136,0000       20,0000   252419   85,29%
     96    88       53,0000   764      136,0000       20,0000   376737   85,29%
    108    95       70,9836   741      136,0000       20,0000   398699   85,29%
    124   105      103,0000   539      136,0000       20,0000   417284   85,29%
    141   131       78,0000   660      136,0000       20,0000   498016   85,29%
Elapsed time = 375,84 sec. (87194,72 ticks, tree = 2,83 MB, solutions = 6)
```

En este caso, inicialmente, llegamos a 171 tiempo de lavado. Para obtener este valor se tardó bastante más que con la solución que vimos anteriormente.

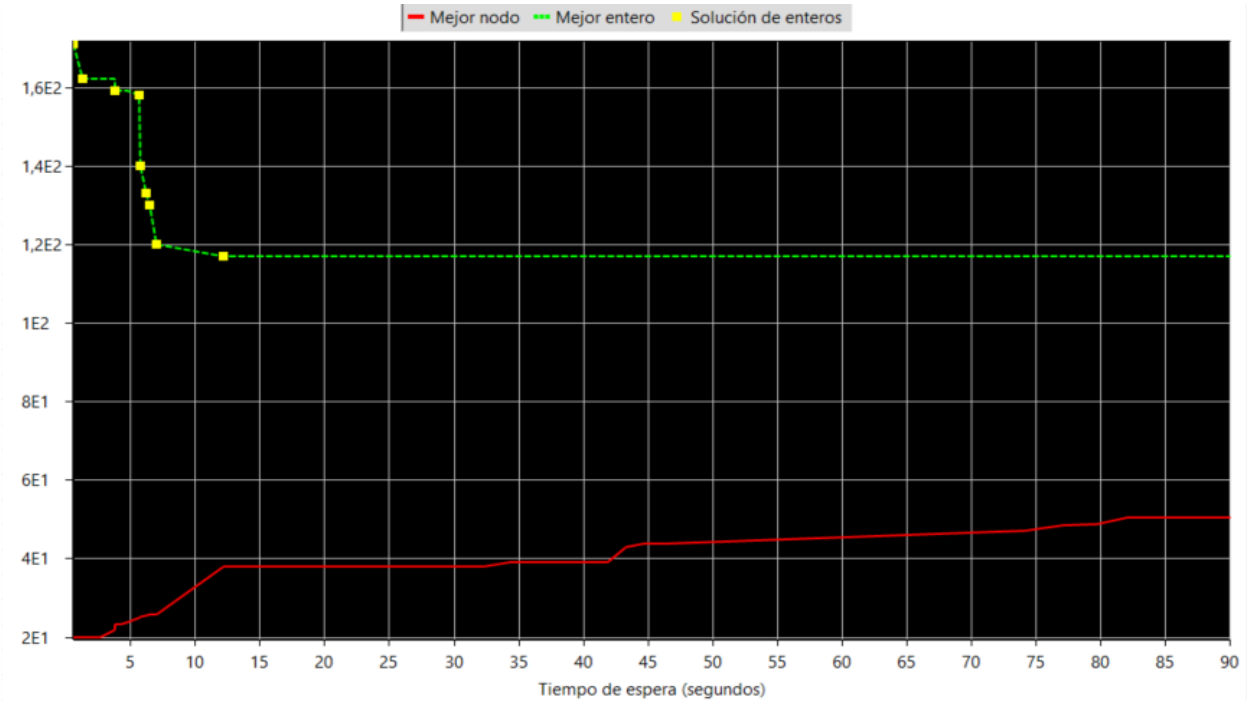
Luego esperamos 10 minutos para ver cuanto mejora, o cuanto más se puede acercar al óptimo.

```
Elapsed time = 585,86 sec. (117208,89 ticks, tree = 9,48 MB, solutions = 24)
   1113   734       76,0395   722      121,0000       37,0000  1875320   69,42%
   1132   738       82,0000   584      121,0000       37,0000  1882804   69,42%
   1156   740       82,0000   483      121,0000       37,0000  1889354   69,42%
   1175   765      101,0000   547      121,0000       37,0000  1950382   69,42%
   1191   770      101,0000   440      121,0000       37,0000  1957640   69,42%
   1211   827      109,0000   192      121,0000       37,0000  2116335   69,42%
*  1242+  845                          120,0000       37,0000            69,17%
*  1248   834      integral     0      119,0000       37,0000  2119224   68,91%
   1253   824      112,0000    97      119,0000       37,0000  2119453   68,91%
```

Vemos que después de un poco más de 10 minutos llega a 119, siendo una solución mucho mejor, pero que tarda demasiado para llegar a ella.

Modelo con estricciones: Óptimo de 15 lavados

Estadística de los primero 90 segundos con la restricción de los 15 lavados o menos:



En esta estadística vemos que llega a un buen resultado ya dentro de los primeros 15 segundos.

En este caso, ese valor ya era el óptimo, 117. Aun así, se ejecutó durante 10 minutos para ver si mejoraba, pero no. Esto es porque el programa ya sabe que tiene una restricción de 15 o menos lavados, por lo tanto, es mucho más rápido el algoritmo que si no se sabe, como pasó en el caso anterior.

Aquí muestro el registro de esta solución propuesta:

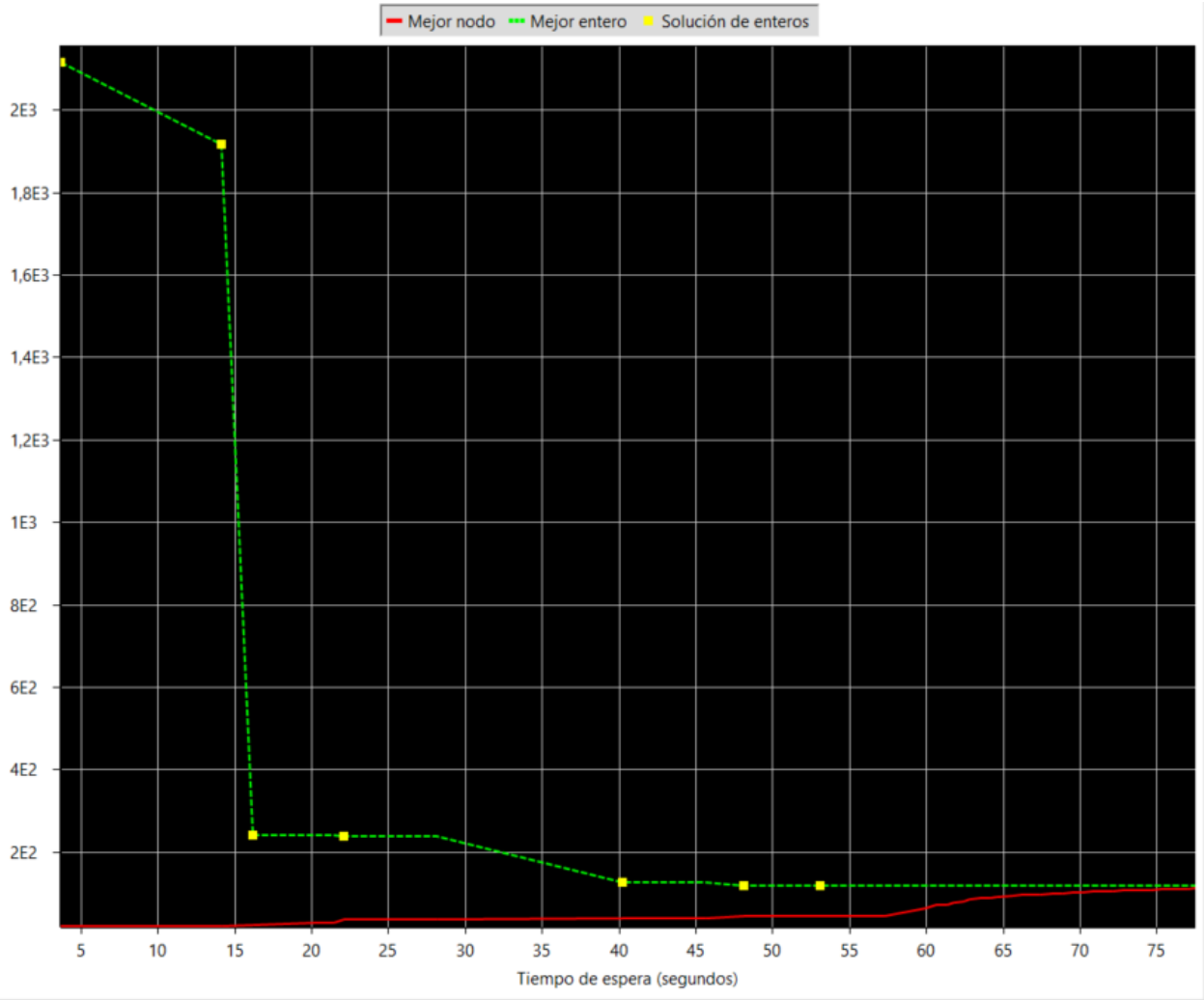
```
Reduced MIP has 3691 rows, 2025 columns, and 19179 nonzeros.
Reduced MIP has 2010 binaries, 15 generals, 0 SOSs, and 0 indicators.
Presolve time = 0,03 sec. (17,13 ticks)
Represolve time = 10,72 sec. (119,52 ticks)
  69120     0      104,0000   289      117,0000     Cuts: 148  7418189   11,11%
  69120     0      104,0000   318      117,0000     Cuts: 446  7418524   11,11%
  69120     0      104,0000   322      117,0000     Cuts: 227  7419121   11,11%
  69120     0      104,0000   275      117,0000     Cuts: 206  7419489   11,11%
  69120     0      104,0000   350      117,0000     Cuts: 411  7420044   11,11%
  69120     0      104,0000   368      117,0000     Cuts: 104  7420561   11,11%
  69120     0      104,0000   304      117,0000     Cuts: 197  7420989   11,11%
  69120     0      104,0000   350      117,0000     Cuts: 376  7421319   11,11%
  69120     0      104,0000   244      117,0000      Cuts: 68  7421576   11,11%
  69120     0      104,0000   309      117,0000     Cuts: 482  7422056   11,11%
  69120     2      104,0000   185      117,0000      104,0000  7422056   11,11%
Elapsed time = 453,80 sec. (150626,33 ticks, tree = 0,02 MB, solutions = 14)
  69136    12      106,0000   376      117,0000      104,0000  7427525   11,11%
  69231    75      106,0000   359      117,0000      104,0000  7437018   11,11%
  69374   193      112,0000   158      117,0000      104,0000  7463015   11,11%
  70699   711      114,0000   143      117,0000      105,0000  7522245   10,26%
  73790  3852      113,6667   331      117,0000      105,0000  7751827   10,26%
  76412  5988      109,0000   359      117,0000      105,0000  7929199   10,26%
  78221  7476      106,1140   417      117,0000      105,0000  8096240   10,26%
  80069  9232      111,0000   387      117,0000      105,0000  8289648   10,26%
  81783 10520      114,5455   297      117,0000      105,0957  8456645   10,17%
  83743 12335      111,5839   324      117,0000      105,1623  8660634   10,12%
Elapsed time = 537,56 sec. (188870,21 ticks, tree = 316,44 MB, solutions = 14)
  85794 14111      116,0000   370      117,0000      105,1623  8844635   10,12%
  87754 15326      116,0000   248      117,0000      105,2117  8993968   10,08%
  89690 17167      109,8700   362      117,0000      105,3740  9194196    9,94%
  91670 19075        cutoff             117,0000      105,5410  9397957    9,79%
  93378 20242      111,3066   352      117,0000      105,6510  9555903    9,70%
  95506 22168      112,0000   314      117,0000      105,7681  9770771    9,60%

Clique cuts applied:  25
Implied bound cuts applied:  90
Flow cuts applied:  196
Mixed integer rounding cuts applied:  672
Zero-half cuts applied:  42
Lift and project cuts applied:  3
Gomory fractional cuts applied:  48

Root node processing (before b&c):
  Real time             =   12,02 sec. (6296,80 ticks)
Parallel b&c, 8 threads:
  Real time             =  588,84 sec. (207792,24 ticks)
  Sync time (average)   =   56,92 sec.
  Wait time (average)   =    0,08 sec.
                          ------------
Total (root+branch&cut) =  600,86 sec. (214089,04 ticks)
```

Modelo con restricciones: Simetría

La estadística para los primeros 90 segundos con la restricción de simetría:



Observamos que justo antes de los 90 segundos encuentra bastante soluciones cercanas al óptimo, llegando a este muy rápidamente.

Llegamos al primer caso hasta el momento en el que la ejecución termina sola, al minuto y 20 segundos. Mucho antes de lo que veníamos cortándola manualmente.

```
      0     0       44,0000   361      118,0000      Cuts: 86    38011   62,71%
      0     0       44,0000   379      118,0000     Cuts: 289    38955   62,71%
      0     0       44,0000   399      118,0000      Cuts: 98    40075   62,71%
*     0+    0                          117,0000       44,0000            62,39%
      0     0  -1,00000e+75     0      117,0000       44,0000    40075   62,39%
Detecting symmetries...
      0     2       44,0000   338      117,0000       44,0000    40370   62,39%
Elapsed time = 55,14 sec. (41602,42 ticks, tree = 0,02 MB, solutions = 24)
      2     4       52,2204   342      117,0000       44,0000    41392   62,39%
      8     7      110,2632   124      117,0000       44,0000    44439   62,39%
     29    13       65,5631   331      117,0000       45,0000    49012   61,54%
    101    43      115,9980   135      117,0000       48,7500    59604   58,33%
    236   103      101,0000   205      117,0000       48,7500    68507   58,33%
    386   162      114,0000   137      117,0000       49,5000    72915   57,69%
    534   252      116,0000   149      117,0000       54,4500    78509   53,46%
    612   351        cutoff             117,0000       62,0500    87735   46,97%
    681   374      114,0739   161      117,0000       71,8016    92849   38,63%
   1445   787      113,2467   226      117,0000       83,8759   114786   28,31%
Elapsed time = 62,67 sec. (44809,58 ticks, tree = 76,65 MB, solutions = 24)
   1981  1026      109,2778   150      117,0000       91,7011   138584   21,62%
   2609  1259      112,0921   231      117,0000       97,4807   164931   16,68%
   3258  1486      111,0000   136      117,0000      101,3444   192079   13,38%
   4065  1697        cutoff             117,0000      105,7273   217807    9,63%
   4819  1551      113,0000   154      117,0000      108,1250   240382    7,59%
   5707  1385    infeasible             117,0000      110,5000   267637    5,56%
   6989   376        cutoff             117,0000      114,7222   292041    1,95%

Clique cuts applied:  5
Implied bound cuts applied:  1571
Mixed integer rounding cuts applied:  3
Zero-half cuts applied:  30
Lift and project cuts applied:  1
Gomory fractional cuts applied:  5

Root node processing (before b&c):
  Real time             =   54,89 sec. (41316,55 ticks)
Parallel b&c, 8 threads:
  Real time             =   25,00 sec. (10559,30 ticks)
  Sync time (average)   =    4,28 sec.
  Wait time (average)   =    0,00 sec.
                          ------------
Total (root+branch&cut) =   79,89 sec. (51875,85 ticks)
```

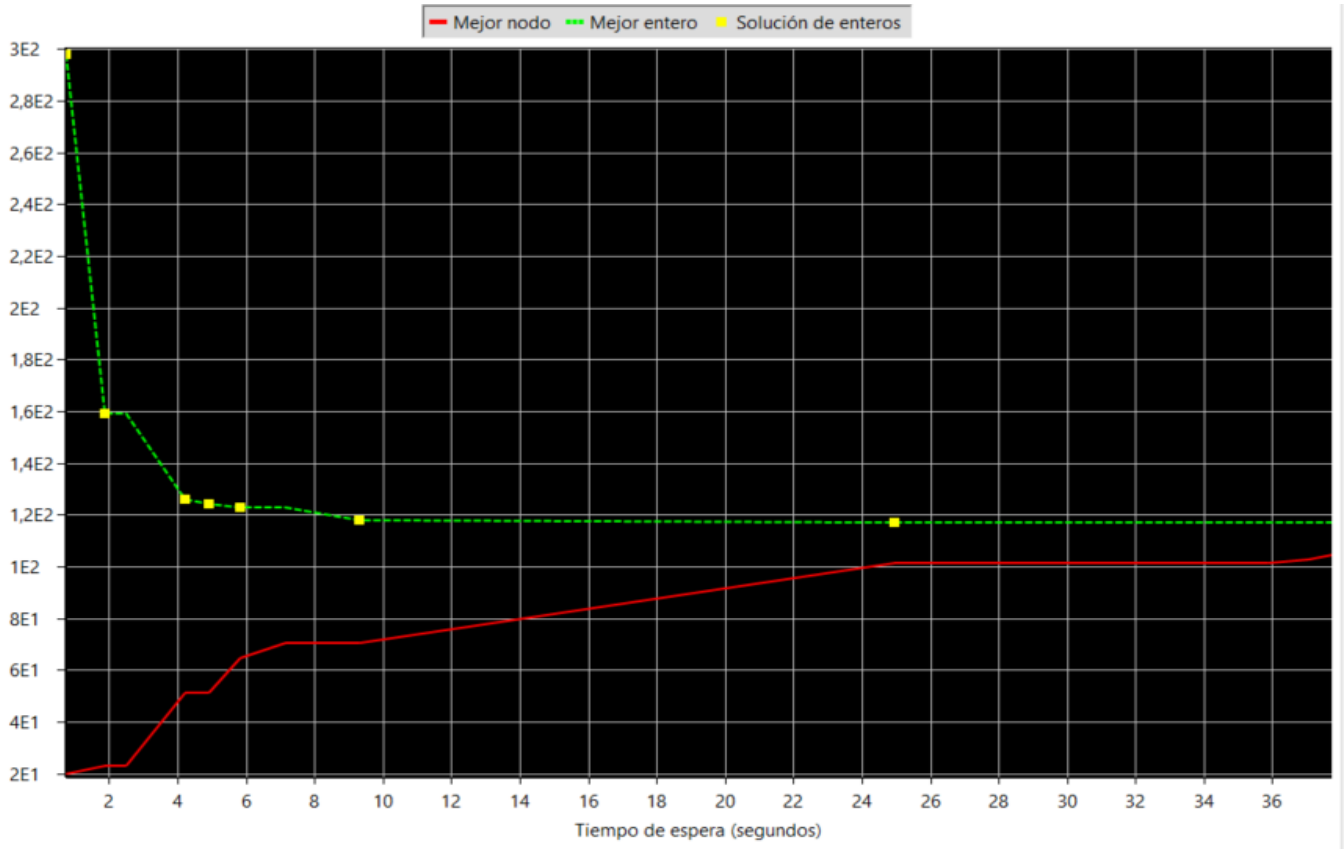
Vemos que, con simetría, o sea, con mas restricciones, la velocidad del algoritmo incrementa notablemente, llegando al óptimo en cuestión de minutos.

Mostramos las prendas y sus respectivos lavados, siendo el nodo el número de la prenda y luego el número de lavado al que esta va.

```
Nodo 1: 1
Nodo 2: 1
Nodo 3: 1
Nodo 4: 1
Nodo 5: 1
Nodo 6: 4
Nodo 7: 4
Nodo 8: 2
Nodo 9: 1
Nodo 10: 2
Nodo 11: 1
Nodo 12: 2
Nodo 13: 2
Nodo 14: 1
Nodo 15: 1
Nodo 16: 1
Nodo 17: 1
Nodo 18: 3
Nodo 19: 1
Nodo 20: 7
Nodo 21: 1
Nodo 22: 1
Nodo 23: 1
Nodo 24: 1
Nodo 25: 1
Nodo 26: 1
Nodo 27: 2
Nodo 28: 6
Nodo 29: 8
Nodo 30: 1
Nodo 31: 1
Nodo 32: 8
Nodo 33: 2
Nodo 34: 3
Nodo 35: 2
Nodo 36: 8
Nodo 37: 2
Nodo 38: 1
Nodo 39: 1
Nodo 40: 1
Nodo 41: 1
Nodo 42: 4
Nodo 43: 2
Nodo 44: 2
Nodo 45: 11
Nodo 46: 1
Nodo 47: 1
Nodo 48: 1
Nodo 49: 3
Nodo 50: 1
Nodo 51: 2
Nodo 52: 3
Nodo 53: 4
Nodo 54: 8
Nodo 55: 1
Nodo 56: 1
Nodo 57: 11
Nodo 58: 7
Nodo 59: 4
Nodo 60: 2
Nodo 61: 6
Nodo 62: 2
Nodo 63: 1
Nodo 64: 6
Nodo 65: 1
Nodo 66: 1
Nodo 67: 2
Nodo 68: 2
Nodo 69: 3
Nodo 70: 5
Nodo 71: 1
Nodo 72: 7
Nodo 73: 2
Nodo 74: 10
Nodo 75: 1
Nodo 76: 1
Nodo 77: 1
Nodo 78: 8
Nodo 79: 1
Nodo 80: 1
Nodo 81: 11
Nodo 82: 1
Nodo 83: 6
Nodo 84: 2
Nodo 85: 1
Nodo 86: 1
Nodo 87: 1
Nodo 88: 1
Nodo 89: 4
Nodo 90: 1
Nodo 91: 1
Nodo 92: 4
Nodo 93: 1
Nodo 94: 8
Nodo 95: 4
Nodo 96: 2
Nodo 97: 2
Nodo 98: 1
Nodo 99: 7
Nodo 100: 5
Nodo 101: 3
Nodo 102: 1
Nodo 103: 1
Nodo 104: 1
Nodo 105: 1
Nodo 106: 2
Nodo 107: 1
Nodo 108: 6
Nodo 109: 2
Nodo 110: 1
Nodo 111: 8
Nodo 112: 2
Nodo 113: 1
Nodo 114: 1
Nodo 115: 3
Nodo 116: 6
Nodo 117: 1
Nodo 118: 6
Nodo 119: 1
Nodo 120: 3
Nodo 121: 1
Nodo 122: 8
Nodo 123: 7
Nodo 124: 2
Nodo 125: 1
Nodo 126: 2
Nodo 127: 2
Nodo 128: 2
Nodo 129: 1
Nodo 130: 6
Nodo 131: 4
Nodo 132: 1
Nodo 133: 1
Nodo 134: 2
Nodo 135: 9
Nodo 136: 2
Nodo 137: 1
Nodo 138: 5
```

Modelo con restricciones: Óptimo de 15 lavados y simetría

La estadística en este caso, con ambas restricciones, para los primeros 90 segundos es la siguiente:



Observamos que la ejecución por poco pasa los 40 segundos, llegando al óptimo poco después.

Por lo que podemos llegar a la conclusión de que, con mayor cantidad de restricciones, el algoritmo aumenta muchísimo su velocidad, llegando al óptimo en cuestión de segundos.

Vemos el screen del loggeo del mismo, que llega a 117 en 40,16 segundos.

```
   3484     0       94,2258   358      117,0000     Cuts: 201   246679   19,47%
   3484     0       94,3245   369      117,0000     Cuts: 234   247650   19,38%
   3484     0       94,3602   350      117,0000     Cuts: 259   248066   19,35%
   3484     0       94,4397   341      117,0000     Cuts: 196   248558   18,88%
   3484     0       94,5067   363      117,0000     Cuts: 169   249022   18,88%
   3484     0       94,5605   333      117,0000     Cuts: 179   249518   18,88%
   3484     0       94,6441   317      117,0000     Cuts: 144   249911   18,88%
   3484     0       94,6470   338      117,0000     Cuts: 200   250081   15,73%
   3484     0       94,7191   320      117,0000      Cuts: 46   250691   15,73%
   3484     0       94,7248   300      117,0000     Cuts: 121   250971   15,73%
   3484     0       94,7248   325      117,0000      Cuts: 88   251287   13,29%
   3484     2       94,7248   296      117,0000      101,4461   251288   13,29%
   3498     5      104,3611   261      117,0000      101,4461   255808   13,29%
Elapsed time = 26,55 sec. (10141,01 ticks, tree = 0,02 MB, solutions = 8)
   3536    15      101,5000   279      117,0000      101,4461   265760   13,29%
   3714    83      105,4286   274      117,0000      101,4461   281747   13,29%
   4341   351      106,5106   256      117,0000      101,4461   333315   13,29%
   5699   659        cutoff             117,0000      106,0240   427378    9,38%

Clique cuts applied:  5
Implied bound cuts applied:  8
Flow cuts applied:  53
Mixed integer rounding cuts applied:  162
Zero-half cuts applied:  13
Gomory fractional cuts applied:  6

Root node processing (before b&c):
  Real time             =    6,95 sec. (2816,53 ticks)
Parallel b&c, 8 threads:
  Real time             =   33,20 sec. (11630,53 ticks)
  Sync time (average)   =    5,77 sec.
  Wait time (average)   =    0,01 sec.
                          ------------
Total (root+branch&cut) =   40,16 sec. (14447,06 ticks)
```

A continuación, la composición de los lavados:

```
solution: 117
Nodo 1: 1
Nodo 2: 1
Nodo 3: 1
Nodo 4: 1
Nodo 5: 1
Nodo 6: 1
Nodo 7: 4
Nodo 8: 1
Nodo 9: 1
Nodo 10: 2
Nodo 11: 1
Nodo 12: 2
Nodo 13: 1
Nodo 14: 1
Nodo 15: 1
Nodo 16: 1
Nodo 17: 1
Nodo 18: 3
Nodo 19: 1
Nodo 20: 9
Nodo 21: 6
Nodo 22: 10
Nodo 23: 1
Nodo 24: 1
Nodo 25: 1
Nodo 26: 5
Nodo 27: 2
Nodo 28: 1
Nodo 29: 2
Nodo 30: 1
Nodo 31: 1
Nodo 32: 8
Nodo 33: 2
Nodo 34: 1
Nodo 35: 2
Nodo 36: 8
Nodo 37: 5
Nodo 38: 1
Nodo 39: 1
Nodo 40: 1
Nodo 41: 2
Nodo 42: 2
Nodo 43: 1
Nodo 44: 1
Nodo 45: 6
Nodo 46: 2
Nodo 47: 1
Nodo 48: 1
Nodo 49: 2
Nodo 50: 2
Nodo 51: 2
Nodo 52: 1
Nodo 53: 4
Nodo 54: 6
Nodo 55: 1
Nodo 56: 1
Nodo 57: 10
Nodo 58: 11
Nodo 59: 1
Nodo 60: 1
Nodo 61: 3
Nodo 62: 2
Nodo 63: 1
Nodo 64: 9
Nodo 65: 5
Nodo 66: 1
Nodo 67: 2
Nodo 68: 1
Nodo 69: 7
Nodo 70: 5
Nodo 71: 1
Nodo 72: 7
Nodo 73: 2
Nodo 74: 10
Nodo 75: 1
Nodo 76: 1
Nodo 77: 1
Nodo 78: 8
Nodo 79: 2
Nodo 80: 1
Nodo 81: 11
Nodo 82: 1
Nodo 83: 6
Nodo 84: 2
Nodo 85: 1
Nodo 86: 1
Nodo 87: 1
Nodo 88: 1
Nodo 89: 4
Nodo 90: 1
Nodo 91: 1
Nodo 92: 11
Nodo 93: 1
Nodo 94: 2
Nodo 95: 4
Nodo 96: 1
Nodo 97: 2
Nodo 98: 1
Nodo 99: 6
Nodo 100: 5
Nodo 101: 3
Nodo 102: 1
Nodo 103: 2
Nodo 104: 1
Nodo 105: 1
Nodo 106: 2
Nodo 107: 1
Nodo 108: 6
Nodo 109: 2
Nodo 110: 3
Nodo 111: 2
Nodo 112: 1
Nodo 113: 1
Nodo 114: 1
Nodo 115: 5
Nodo 116: 9
Nodo 117: 3
Nodo 118: 2
Nodo 119: 1
Nodo 120: 3
Nodo 121: 1
Nodo 122: 3
Nodo 123: 6
Nodo 124: 2
Nodo 125: 1
Nodo 126: 6
Nodo 127: 1
Nodo 128: 2
Nodo 129: 1
Nodo 130: 4
Nodo 131: 1
Nodo 132: 6
Nodo 133: 10
Nodo 134: 2
Nodo 135: 7
Nodo 136: 2
Nodo 137: 1
Nodo 138: 5
```

Comparación de modelos

Comparación I:

- Restricción de 15 lavados como máximo vs la misma restricción sumada a la simetría.

En la primera de ellas, se observa que se llega al óptimo en cuestión de segundos, pero sigue probando todas las combinaciones posibles. Esto hace que el programa, llegado a los 10 minutos de ejecución, siga corriendo, ya que sigue comparando y buscando mejores soluciones. La razón de esto es porque busca con 2^{15}combinaciones, siendo la mayoría repetidas. Esto, en el segundo modelo no sucede, ya que, al sumarle otra restricción, el problema se acota de forma que se llega a una solución de una forma más rápida, y sin probar 2^{15}combinaciones.

Comparación II:

- Heurística propuesta (la primera, realizada en el lenguaje de programación Python) vs la programación lineal entera.

  Se puede apreciar que, como método de resolución, la heurística propuesta no está lejos de la solución buscada, y es rápida. Una vez que ya esta programada, intentar resolver un problema se vuelve sencillo. Sólo debe ingresar el archivo con el mismo, y esperar el resultado en segundos. Ahora, si este script no está resuelto, resolver el problema lleva un poco mas de tiempo, pero no mucho, ya que el programador lo resuelve de una forma que es fácil y sencilla de entender, y capaz de reutilizar código para llegar a una solución decente.

  La programación lineal nos mostro que, agregando restricciones a nuestros modelos, las soluciones llegan de forma mucho más rápida que si no lo hacemos, y en estos casos la velocidad puede asemejarse con la de la heurística.

  En los casos de no tener las restricciones que nos proporcionen esta característica, la solución puede llegar a tardar demasiado, y no es algo que nos llame para utilizar este tipo de herramientas. Lo que sí podemos sumar la idea de que no necesitamos cambiar todo un código entero para llegar a una solución, sino que cambiamos las variables y sus restricciones, y de esta forma el programa busca el óptimo con tan solo con esta información.

Entre ambas opciones, capaz la más 'segura' es la programación lineal, ya que con ella sabemos que podemos llegar a la solución óptima. En cambio, con la heurística vimos que aunque intentemos, capaz hay cosas que el programador no llega a pensar o ver en su cabeza como para tratar ciertas restricciones o variables y poder así llegar al óptimo, sino que llega a una solución que puede acercarse lo suficiente (según sea el caso), pero no es la mejor. O puede darse que sí llega a la mejor solución para un caso exacto, pero si se agregan o sacan diferentes factores externos, como aumentar la cantidad de nodos notablemente, o las incompatibilidades entre ellos, su solución puede no ser más la mejor con su misma heurística y deba cambiar su código para cada caso particular.