



UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

2DO CUATRIMESTRE DE 2022

[71.14] MODELOS Y OPTIMIZACIÓN

Cuarta Entrega

Integrantes:

Lucía Pardo <lpardo@fi.uba.ar>

Padrón:

99999

Observaciones:

20 de noviembre de 2022

Índice

1. Heurística entrega anterior	2
1.1. Análisis del problema:	2
1.2. Objetivo:	2
1.3. Modelo:	2
1.4. Hipótesis:	2
1.5. Variables:	2
1.6. Resolución	2
2. Soluciones con distintas heurísticas algorítmicamente	3
2.1. Cplex modelo inicial	3
2.2. Mejora performance	4
2.3. simetría	5
2.4. Simetría con capacidad de lavados	9
2.5. Conclusiones	9

1. Heurística entrega anterior

1.1. Análisis del problema:

Se trata de un problema de Armado en donde una lavandería tiene prendas que destiñen y son incompatibles con otras, teniendo un tiempo de lavado de determinado cada una.

1.2. Objetivo:

Se quiere minimizar el tiempo de lavado de todas las prendas respetando las incompatibilidades entre prendas.

1.3. Modelo:

1.4. Hipótesis:

Los tiempos de cada prenda son constantes

1.5. Variables:

Y_{il} : Prenda i usada en el lavado l (Bivalente, 1 para usada, 0 para no usada en el lavado l)

T_i : Tiempo de la prenda i (constante)

TL_{il} : Tiempo de lavado de la prenda i en el lavado l [entera]

$Tmax_l$: Tiempo máximo del lavado l [entera]

$Ymax_{il}$: Si la prenda i en el lavado l es el tiempo máximo [bivalente]

L_i : Lavado i [entera]

1.6. Resolución

Si por ejemplo tomamos archivo vemos que la prenda 1 no es compatible con la prenda 10. Basados en eso sabemos que no queremos que se encuentren en el mismo lavado por lo que:

$$\sum Y_{1l} + Y_{10l} \leq 1$$

$$\sum Y_{2l} + Y_{19l} \leq 1$$

Hacemos lo mismo y para todas las prendas y sus otras prendas incompatibles
Además me aseguro que una prenda esté en un solo lavado:

$$\sum Y_{il} = 1$$

Idem para cada prenda

Defino el Lavado i con la cantidad de prendas que contiene:

$$L1 = \sum_{1, \text{cant} \text{ prendas}} Y_{i1}$$

Quiero que si la prenda i está en el lavado l tome valor, sino sea 0:

$$TL_{il} = T_i Y_{il}$$

Ahora es necesario encontrar el tiempo máximo de ese lavado: Para el lavado l :

$$TL_{il} \leq T_{max_l} \leq TL_{il} + M(1 - Y_{max_{il}})$$

$$\sum Y_{max_{il}} = 1$$

$$MIN(FUNCZ) = \sum T_{max_l}$$

$$l = 1..cant_{lavados}$$

2. Soluciones con distintas heurísticas algorítmicamente

Primero setteamos todo el entorno para poder trabajar con CPLEX y el proyecto brindado por la cátedra. Se utiliza la heurística de las anteriores entregas dando un resultado de 13 lavados con un tiempo de 171

```
lulix@lulix: /media/lulix/OS/Lu/FIUBA/modelos/Modelos [main] python3 lavanderia.py
tiempo de lavados es 171 con 13 lavados
```

Figura 2.1 – corrida del primer algoritmo implementado

2.1. Cplex modelo inicial

Ahora sobre el programa se corre la solución propuesta, se saca una captura para observar las soluciones encontradas durante los primeros 90 segundos.

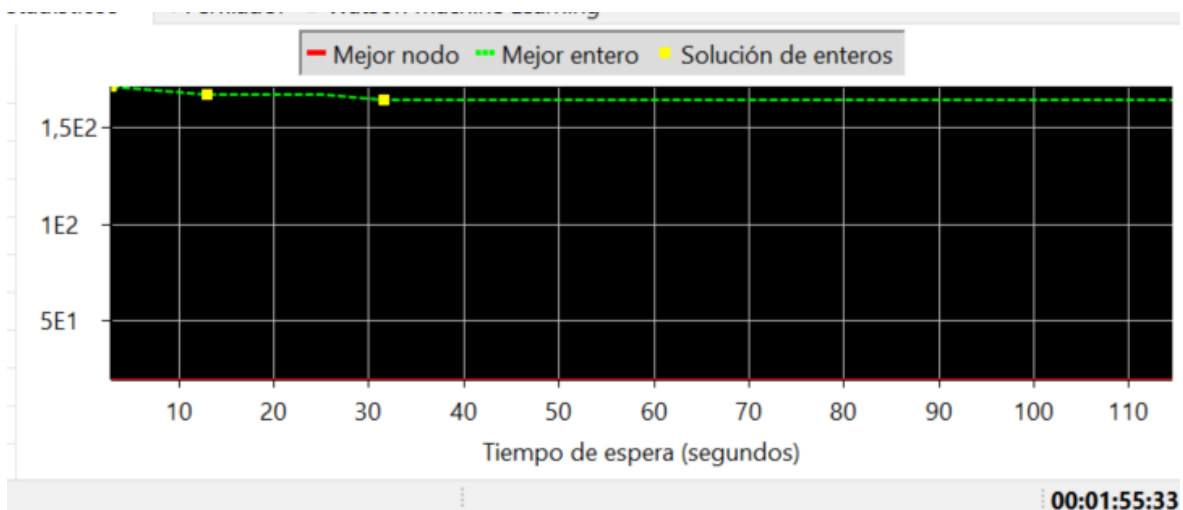


Figura 2.2 – corrida del primer algoritmo en cplex primeros 90 segundos

Se puede apreciar que durante los primeros segundos encuentra 3 soluciones y a partir de los 30 ya tarda mucho más en encontrar nuevas soluciones.

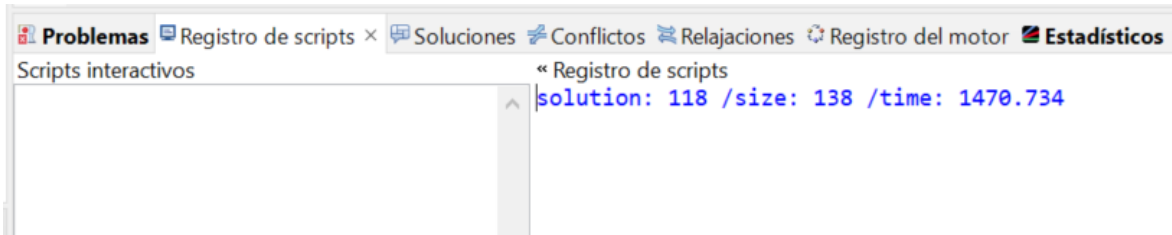


Figura 2.3 – registro scripts en cplex 10 minutos

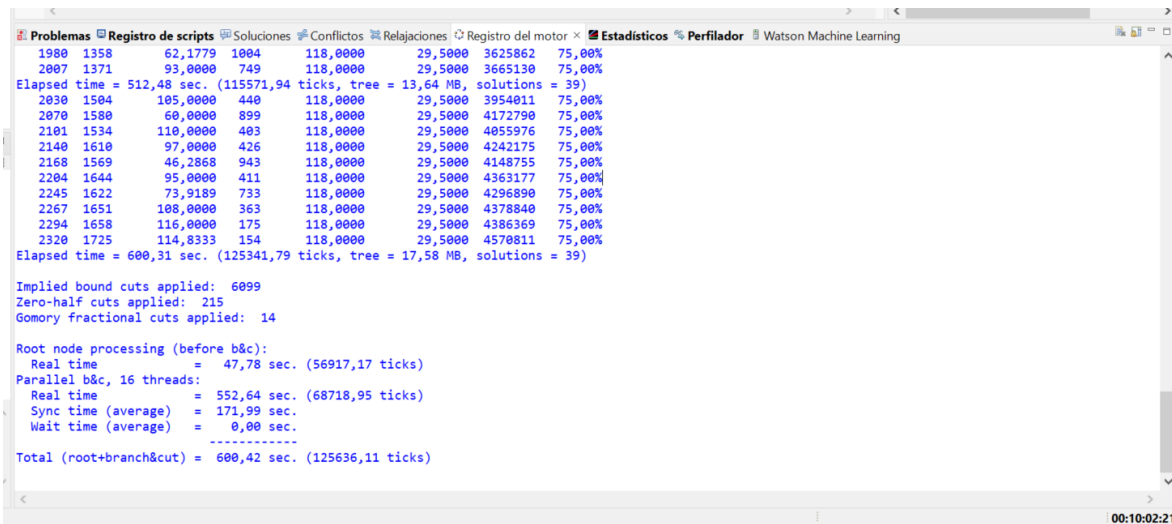


Figura 2.4 – corrida del primer algoritmo en cplex 10 minutos

Luego de 10 minutos se para el algoritmo y se observan los resultados obtenidos hasta el momento, siendo 118 , mejor que el resultado obtenido con nuestra heurística, pero con un tiempo mucho mayor para obtener el resultado.

El algoritmo a pesar de haber obtenido una mejor solución que con la heurística propia, luego de 10 minutos no pudo lograr terminar.

2.2. Mejora performance

El paso siguiente para intentar mejorar el algoritmo se intenta agregar restricciones, para esto se sabe mediante una heurística que hay un límite de 15 lavados modificando el n que se encuentra como constante global.



Figura 2.5 – corrida con 15 lavados primeros 90 segundos

A comparación con la anterior corrida sin la restricción, se observa que se encuentran las primeras soluciones mucho más rápidamente para luego quedar corriendo sin terminar o encontrar nuevas soluciones por un tiempo.

```

Elapsed time = 561,66 sec. (234052,69 ticks, tree = 1938,82 MB, solutions = 10)
154270 51157      107,0000    373      117,0000    106,0000  18447749    9,40%
155607 51828      108,0000    425      117,0000    106,0000  18619411    9,40%
157286 52454      113,0000    326      117,0000    106,0000  18786224    9,40%
158851 53892      116,0000    319      117,0000    106,0000  19079598    9,40%
160315 54690      109,3846    368      117,0000    106,0000  19253960    9,40%
161708 56312      114,4000    441      117,0000    106,0000  19632018    9,40%
162886 56826      116,0000    317      117,0000    106,0000  19742552    9,40%
164076 57989      cutoff      117,0000    106,0000  20100792    9,40%
165140 59033      108,0000    359      117,0000    106,0000  20362398    9,40%
166024 59790      109,4615    436      117,0000    106,0000  20573793    9,40%
Elapsed time = 735,67 sec. (272216,15 ticks, tree = 2437,91 MB, solutions = 10)
Nodefile size = 385,60 MB (352,81 MB after compression)
167090 60123      116,0000    265      117,0000    106,0000  20641712    9,40%

```

Figura 2.6 – corrida con 15 lavados 10 minutos

luego de 10 minutos, se llegó al resultado de 117 segundos, un poco mejor que el anterior, pero el algoritmo sigue sin terminar, y se detiene. Se intentará seguir buscando mejoras

2.3. simetría

Probando ahora, se observa nuevamente que en muy poco tiempo se empiezan a encontrar soluciones, y llegando esta vez muy rápidamente al óptimo y finalizando el algoritmo. Esto se debe a que con este paso agregamos una restricción que elimina del modelo posibles soluciones alternativas que son simétricas y por lo tanto con iguales, recortando mucho el problema y ayudando al algoritmo.

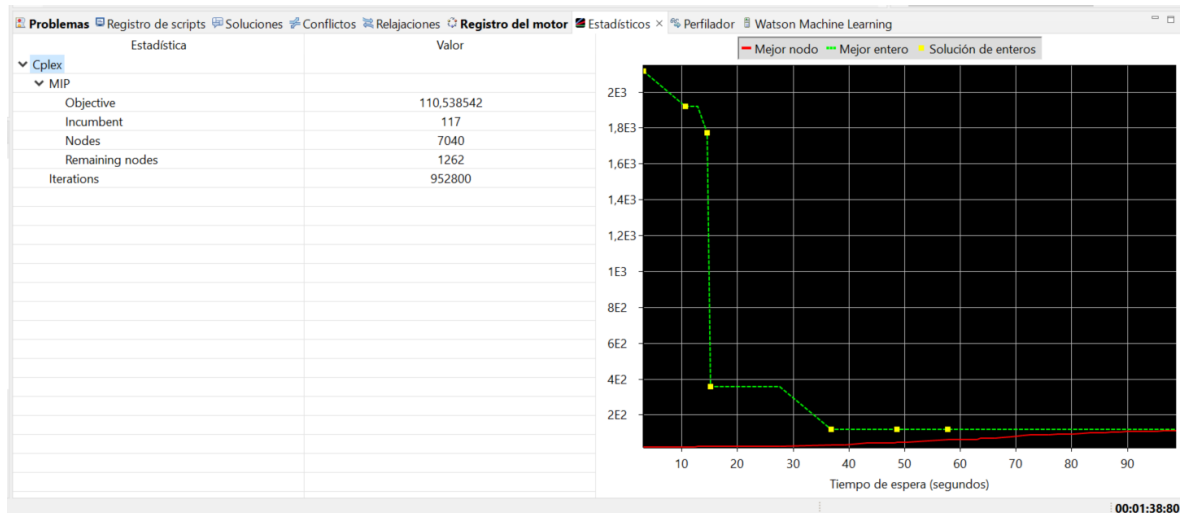


Figura 2.7 – corrida con simetría

Se presenta los logs de la solución obtenida

solution: 117 /size: 138 /time: 3142.875

Nodo 1: 1
 Nodo 2: 1
 Nodo 3: 2
 Nodo 4: 1
 Nodo 5: 1
 Nodo 6: 6
 Nodo 7: 3
 Nodo 8: 5
 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: 4
 Nodo 19: 1
 Nodo 20: 10
 Nodo 21: 9
 Nodo 22: 1
 Nodo 23: 6
 Nodo 24: 1
 Nodo 25: 1
 Nodo 26: 7
 Nodo 27: 2
 Nodo 28: 1

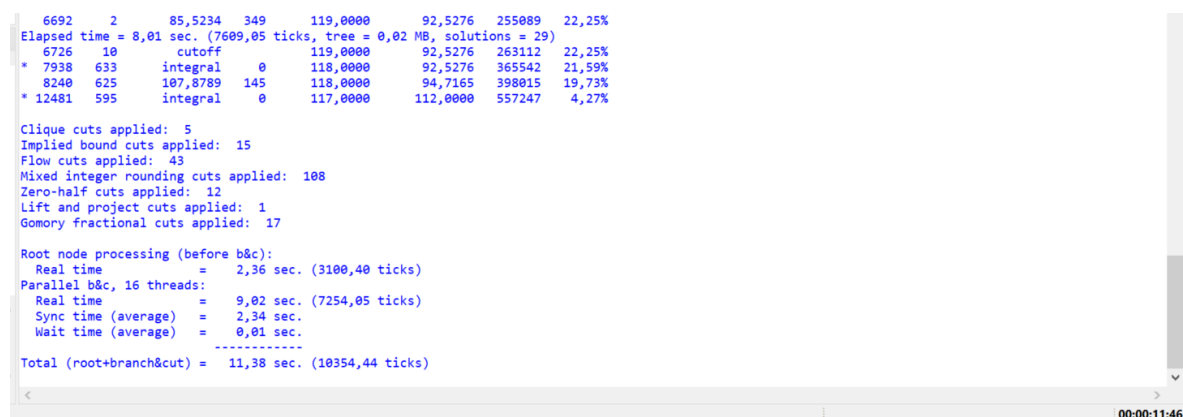
Nodo 29: 8
Nodo 30: 1
Nodo 31: 7
Nodo 32: 4
Nodo 33: 9
Nodo 34: 1
Nodo 35: 3
Nodo 36: 8
Nodo 37: 5
Nodo 38: 1
Nodo 39: 1
Nodo 40: 8
Nodo 41: 1
Nodo 42: 3
Nodo 43: 3
Nodo 44: 5
Nodo 45: 10
Nodo 46: 2
Nodo 47: 1
Nodo 48: 1
Nodo 49: 2
Nodo 50: 1
Nodo 51: 1
Nodo 52: 2
Nodo 53: 6
Nodo 54: 8
Nodo 55: 1
Nodo 56: 1
Nodo 57: 9
Nodo 58: 9
Nodo 59: 1
Nodo 60: 1
Nodo 61: 4
Nodo 62: 3
Nodo 63: 1
Nodo 64: 1
Nodo 65: 1
Nodo 66: 1
Nodo 67: 2
Nodo 68: 1
Nodo 69: 5
Nodo 70: 5
Nodo 71: 1
Nodo 72: 7
Nodo 73: 2
Nodo 74: 11
Nodo 75: 8
Nodo 76: 2

Nodo 77: 1
Nodo 78: 8
Nodo 79: 5
Nodo 80: 1
Nodo 81: 10
Nodo 82: 1
Nodo 83: 6
Nodo 84: 2
Nodo 85: 1
Nodo 86: 1
Nodo 87: 1
Nodo 88: 1
Nodo 89: 3
Nodo 90: 2
Nodo 91: 1
Nodo 92: 7
Nodo 93: 1
Nodo 94: 2
Nodo 95: 3
Nodo 96: 2
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: 1
Nodo 107: 3
Nodo 108: 6
Nodo 109: 2
Nodo 110: 1
Nodo 111: 1
Nodo 112: 3
Nodo 113: 1
Nodo 114: 1
Nodo 115: 11
Nodo 116: 9
Nodo 117: 3
Nodo 118: 2
Nodo 119: 1
Nodo 120: 4
Nodo 121: 1
Nodo 122: 2
Nodo 123: 2
Nodo 124: 2

Nodo 125: 1
 Nodo 126: 6
 Nodo 127: 2
 Nodo 128: 5
 Nodo 129: 1
 Nodo 130: 9
 Nodo 131: 4
 Nodo 132: 1
 Nodo 133: 10
 Nodo 134: 2
 Nodo 135: 7
 Nodo 136: 2
 Nodo 137: 1
 Nodo 138: 5

2.4. Simetría con capacidad de lavados

Ahora se intenta combinar ambas soluciones, tanto con simetría como con una capacidad de lavados.



```

6692      2      85,5234      349      119,0000      92,5276      255089      22,25%
Elapsed time = 8,01 sec. (7609,05 ticks, tree = 0,02 MB, solutions = 29)
6726      10      cutoff      0      119,0000      92,5276      263112      22,25%
* 7938      633      integral      0      118,0000      92,5276      365542      21,59%
8240      625      107,8789      145      118,0000      94,7165      398015      19,73%
* 12481      595      integral      0      117,0000      112,0000      557247      4,27%

Clique cuts applied: 5
Implied bound cuts applied: 15
Flow cuts applied: 43
Mixed integer rounding cuts applied: 108
Zero-half cuts applied: 12
Lift and project cuts applied: 1
Gomory fractional cuts applied: 17

Root node processing (before b&c):
  Real time       = 2,36 sec. (3100,40 ticks)
Parallel b&c, 16 threads:
  Real time       = 9,02 sec. (7254,05 ticks)
  Sync time (average) = 2,34 sec.
  Wait time (average) = 0,01 sec.
-----
Total (root+branch&cut) = 11,38 sec. (10354,44 ticks)
  
```

Figura 2.8 – resultados con simetría

Como se esperaba esta fue la solución más rápida con apenas 11 segundos y obteniendo el optimo de 117 segundos y 11 lavados.

A continuación se ve que el log con ambas combinaciones de restricciones es el mismo que para el caso anterior.

2.5. Conclusiones

Al principio la heurística presentada en python de forma Greedy permitía que el algoritmo termine pero logrando una solución muy poco óptima y en caso de querer una mejor estimación tener que realizar una refactorización.

Analizando los métodos con programación lineal e intentando plantear restricciones se puede llegar a una solución a pesar de tener un problema complejo.

Agregando restricciones la extensión de posibilidades a calcular por el algoritmo se acota mucho, permitiendo una rápida y óptima solución.

Sin embargo, en caso de no tener dichas restricciones o no poder llegar a calcularlas, deja el problema sin que el algoritmo pueda terminar en un tiempo considerable, siendo algo no deseado.

Es importante analizar el tipo de problema al que se está enfrentando, decidiendo así la mejor herramienta a utilizar, teniendo en cuenta si es necesario una solución exacta, el tiempo en que se necesita y si es posible modelarlo a través de la programación lineal u otros métodos para poder conseguir lo que mejor se ajusta a dicho problema.