Universidad de Buenos Aires
Facultad De Ingeniería
Año 2022 - 2do Cuatrimestre

Modelos y Optimización (71.14)

TRABAJO PRÁCTICO Final
TEMA: Análisis de solución
FECHA:

INTEGRANTES:
Alvarez, Ernesto - #102221
<eralvarez@fi.uba.ar>

Índice

1. Introducción

En el presente trabajo se demostrará los diferentes métodos de resolución del siguiente problema:

- Una lavandería tiene que lavar prendas, algunas pueden ir juntas y otras no (destiñen).
- El tiempo de cada lavado es el tiempo que lleva lavar la prenda más sucia de ese lavado.

Es un problema del tipo **"Coloreo de grafos"** en el que el grafo se puede identificar como:

- Cada **Nodo** es una prenda.
- Cada **Arista** indica si las prendas son incompatibles.

En las siguientes secciones se verán tipos de resoluciones veloces cómo lo son los algoritmos Greedy basados en heurísticas. Y métodos eficaces y precisos como lo es la programación lineal. Puede haber puntos en que ambas características, velocidad y precisión estén unidas en un mismo método si se lo utiliza correctamente.

Se comenzó buscando resultados rápidos para ir viendo los diferentes métodos. Al principio, se utilizaron métodos algorítmicos en los que se sólo se buscaba una resolución óptima, la que sea. Luego se fue perfeccionando el algoritmo, utilizando heurísticas, mejorando los resultados, en general, que este método brindaba. Más importante, el algoritmo no dejó de ser veloz aunque ante ciertos problemas, estaba lejos de la solución óptima.

Luego, con la realización de que este tipo de algoritmos, basados en heurísticas, por más de que daban resultados buenos, no llegaban al óptimo en muchos casos. Se decidió buscar alternativas, entre las cuáles se encontró la programación lineal. Bajo este método, lo primero que se buscó fue encontrar un modelo lineal. Luego, se encontró que el modelo original no llegaba a resolverse en un tiempo relativamente corto, por lo que se emplearon técnicas para óptimizar el tiempo de ejecución.

2. Modelo

2.0.1. Constantes

T_i: Tiempo de la ropa i-esima en lavarse
M: Lavado muy grande

2.0.2. Variables

$ropa_{ij}$: Es 1 si la ropa i-esima entra en el lavado j-esimo; 0 si no.
$lavado_j$: Es 1 si el lavado j-esimo efectivamente es un lavado real.
mt_j: Es el mayor tiempo del lavado j-esimo.

2.0.3. Modelo

Lavados existentes

"La ropa ij solamente puede valer 1, si el lavado j es uno que se utilizará".

$ropa_{ij} <= lavado_j$; para toda ropa i y lavado j.

Ropa incompatible

Suponiendo que tenemos una incompatibilidad entre la ropa x y la ropa y, entonces:

$ropa_{xj} + ropa_{yj} ¡= 1$; para todo lavado j.

Tiempo maximo de lavado

$mt_j <= M.lavado_j$; para todo lavado j.

$mt_j >= T_i * ropa_{ij}$; para toda ropa i y lavado j.

Una ropa en un solo lavado

$\sum_j ropa_i j = 1$: para toda ropa i. (j son los lavados)

Funcional

$minZ = \sum mt_j$; siendo j desde 1 hasta m (siendo m la cantidad de lavados)

3. Solución con heurísticas

3.1. Características

Esta sección seguirá los lineamientos de entregas anteriores, en las que el método de resolución es algorítmico, basado en heurísticas. Esta solución es categorizada como Greedy, utilizando diferentes maneras de conseguir el óptimo local, aunque este no se consiga exactamente, la solución termina siendo un aproximado en muchos casos con poca diferencia al óptimo.

Las propiedades **positivas** de este tipo de soluciones es la siguiente:

- Algoritmo sencilla de programar.
- Algoritmo sencillo de entender.
- Algoritmo veloz.

La propiedad **negativa** principal de este tipo de soluciones es que no siempre encuentra el **óptimo**, lo que en muchos casos provoca el descarte de este tipo de soluciones.

3.2. Resultado

Adjunto al trabajo, en el repositorio, se observa la solución brindada en el archivo `respuesta`.

En la solución se encontró un tiempo de lavado total de **123 segundos** y **11 lavados**.

4. Solución con Programación Lineal

4.1. Observaciones

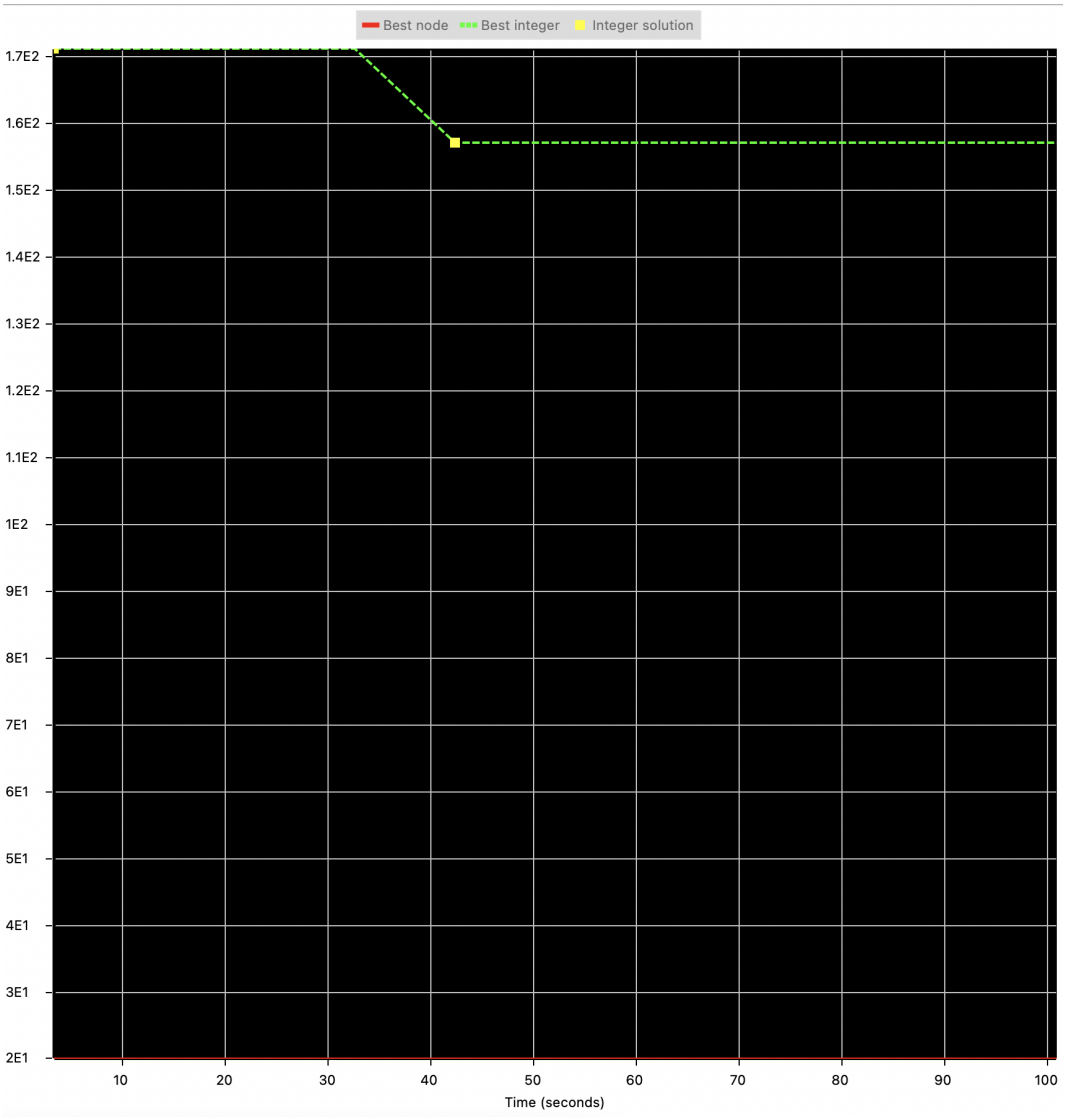

Figura 4.1: Estadísticas con OPL en CPLEX de la solución del modelo clásico

Como se puede observar en la **Figura 3.1**, a los 100 segundos de corrido el software, no se observan resultados significativos. Llegando a una función objetivo de 157 por el momento. Lejos de la solución obtenida en el punto anterior con la heurística, en la que se tardó menos de 5 segundos en correrse.

Una vez llegados los 10 minutos de ejecución, se detuvo la misma. En este momento se consiguió una solución que llegaba a tener funcional de valor **119 segundos** (Visualizado en la **Figura 3.2** en la 5ta columna).

```
Elapsed time = 269.48 sec. (87299.50 ticks, tree = 11.27 MB, solutions = 21)
   1146   804      116.0000   126      121.0000       20.0000  1981099   83.47%
   1176   808       72.3015   994      121.0000       20.0000  2015821   83.47%
   1199   910       83.1527   620      121.0000       20.0000  2266724   83.47%
   1224   873       57.0000   732      121.0000       20.0000  2178851   83.47%
   1257   939       89.6830   339      121.0000       20.0000  2336292   83.47%
   1293   986       96.0000   463      121.0000       20.0000  2474222   83.47%
   1324   992      104.0000   570      121.0000       20.0000  2481610   83.47%
*  1364+ 1000                         120.0000       20.0000            83.33%
*  1365  1025      integral     0      119.0000       20.0000  2610082   83.19%
   1366  1034      107.6556   330      119.0000       20.0000  2572163   83.19%
   1404  1007      105.7613   412      119.0000       20.0000  2521201   83.19%
   1436  1017      118.0000   169      119.0000       20.0000  2529332   83.19%
```

Figura 4.2: Logs al finalizar los 10 minutos de ejecución del modelo clásico

4.2. Conclusiones

Fue una ejecución lenta, en la que no se avanzó lo suficientemente rápido como para terminar antes de los 10 minutos. Tuvo una aproximación mayor a la hecha por la solución con heurística, aún así, se espera de un modelo de programación lineal encontrar el óptimo. Por lo que en secciones siguientes se verá cómo mejorar la velocidad de este tipo de soluciones para poder llegar al óptimo en un tiempo relativamente acotado.

5. Solución sabiendo CAP de lavados

5.1. Observaciones

A partir de este momento, sabiendo que la solución "clásica" del punto anterior no llegó a finalizar en 10 minutos, intentaremos mejorar su velocidad y *performance* agregando algunas restricciones y/o meta-conocimientos que favorecerán al algoritmo.

En este caso se añadió el conocimiento, adquirido a través de una heurística, de que la cantidad de lavados, para llegar al óptimo, es de 15 lavados o menos.

Esto se traduce a reducir la cantidad de variables bivalentes que se utilizan para determinar si un lavado efectivamente se hace, de ***n*** a ***15***. Esto provoca la reducción del universo de soluciones posibles drásticamente.

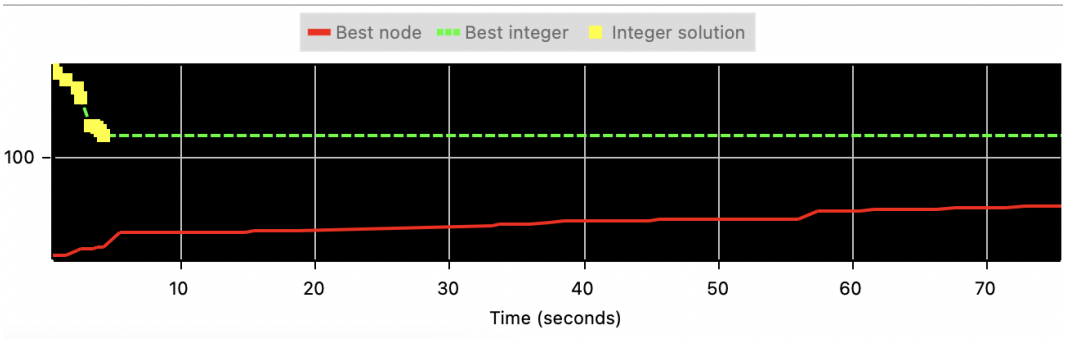

Figura 5.1: Estadísticas con OPL del modelo con 15 lavados o menos

Tanto en la **Figura 4.1** como en la **Figura 4.2** se ve que llega a la solución de un funcional con valor **117** que más adelante se verá que es la solución óptima. Sin embargo el algoritmo, en 10 minutos de ejecución, no determinó que efectivamente esta era la solución óptima. Siguió probando combinaciones.

5.2. Conclusiones

Si bien se puede ver que la velocidad mejoró sustancialmente, reduciendo el universo de posibilidades, esto no alcanzó para llegar a un tiempo total de ejecución menor a 10 minutos.

Utilizar las soluciones con métodos que utilizan heurísticas puede ser de utilidad. De manera indirecta, la solución que provee puede ser utilizada para acotar el modelo, y el universo de combinaciones de este, para lograr mayor *performance* en su ejecución.

Por lo que la combinación de ambas técnicas es de gran ayuda en la resolución de problemas.

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Bound	ItCnt	Gap
*	0+	0			300.0000	0.0000		100.00%
*	0+	0			171.0000	0.0000		100.00%
	0	0	20.0000	1078	171.0000	20.0000	1897	88.30%
*	0+	0			167.0000	20.0000		88.02%
	0	0	20.0000	1176	167.0000	Cuts: 386	2954	88.02%
*	0+	0			162.0000	20.0000		87.65%
	0	0	21.8697	956	162.0000	Cuts: 525	6779	86.50%
	0	0	23.2893	875	162.0000	Cuts: 830	9711	85.62%
	0	0	24.0642	906	162.0000	Cuts: 739	12290	85.15%
*	0+	0			155.0000	24.0642		84.47%
	0	0	24.2936	976	155.0000	Cuts: 634	13750	84.33%
	0	0	24.7470	963	155.0000	Cuts: 655	15151	84.03%
*	0+	0			153.0000	24.7470		83.83%
*	0+	0			151.0000	24.7470		83.61%
*	0+	0			147.0000	24.7470		83.17%
	0	0	-1.00000e+75	0	147.0000	24.7470	15151	83.17%
	0	0	25.0667	969	147.0000	Cuts: 602	17185	82.95%
	0	0	25.3217	947	147.0000	Cuts: 570	18142	82.77%
	0	0	25.5120	1016	147.0000	Cuts: 592	18830	82.64%
	0	0	25.7516	927	147.0000	Cuts: 495	19487	82.48%
*	0+	0			125.0000	25.7516		79.40%
	0	0	25.8562	936	125.0000	Cuts: 426	20379	79.32%
	0	0	25.9439	925	125.0000	Cuts: 540	21134	79.24%
*	0+	0			124.0000	25.9439		79.08%
	0	0	25.9944	876	124.0000	Cuts: 490	21814	79.04%
	0	0	26.0302	964	124.0000	Cuts: 474	22343	79.01%
*	0+	0			123.0000	26.0302		78.84%
	0	0	26.0709	930	123.0000	Cuts: 426	23266	78.80%
	0	0	26.0841	952	123.0000	Cuts: 470	23808	78.79%
*	0+	0			121.0000	26.0841		78.44%
	0	0	26.0940	968	121.0000	Cuts: 464	24336	78.43%
*	0+	0			120.0000	26.0951		78.25%
*	0+	0			117.0000	26.0951		77.70%
	0	0	26.1121	1015	117.0000	Cuts: 448	24926	77.66%
	0	0	26.1361	1002	117.0000	Cuts: 369	25352	77.66%
	0	0	26.2033	949	117.0000	Cuts: 421	25953	77.54%
	0	0	26.2244	959	117.0000	Cuts: 354	26275	77.54%
	0	0	26.2376	938	117.0000	Cuts: 320	26608	77.45%
	0	0	26.2517	951	117.0000	Cuts: 307	26949	77.45%
	0	0	26.2740	1059	117.0000	Cuts: 310	27322	77.40%
	0	0	26.2994	997	117.0000	Cuts: 262	27599	77.40%
	0	0	26.3323	931	117.0000	Cuts: 271	27989	67.52%
	0	0	26.3575	955	117.0000	Cuts: 457	28487	67.52%
	0	2	26.3575	931	117.0000	38.0000	28487	67.52%

Figura 5.2: Logs al finalizar los 10 minutos de ejecución del modelo con 15 lavados o menos

6. Solución detectando simetría

6.1. Observaciones

Otra manera de reducir el universo de combinaciones, es restringiendo al modelo aún más.

En este caso, reduciremos el modelo sabiendo que es indistinto si se termina utilizando el lavado "número 1" que el lavado "número 2", si al final de la ejecución, en distintas soluciones, contienen las mismas prendas.

El funcional en la solución óptima es de **117**, y llegó a 11 lavados. Se muestra la solución de la Figura 5.2 a 5.4

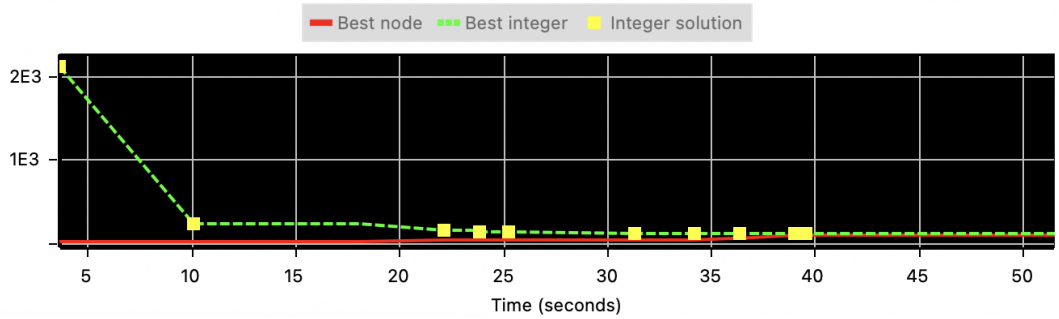

Figura 6.1: Estadísticas con OPL del modelo con la eliminación de las combinaciones simétricas

6.2. Conclusiones

Esta es la primera de las soluciones en la que se llega a la solución óptima antes de los 10 minutos. Se resuelve a los pocos segundos, en un tiempo menor al minuto.

Esto muestra la importancia de recortar las ramas del árbol de posibilidades. Esta restricción de simetrías redujo el tiempo de ejecución de algo de más de diez minutos a menos de uno.

La restricción transformó la búsqueda de la solución en iteraciones en las que se va sumando de a un lavado. Primero buscará con un solo lavado, no encuentra solución óptima, por lo que prosigue a buscar con dos lavados, que tampoco es factible. Y así hasta llegar al onceavo, en el que encuentra una solución óptima, y empieza a buscar en esa cantidad el óptimo, pero ahora en distribución de prendas.

```
solution: 117 /size: 138 /time: 1667864707.294728041
Nodo 1: 2
Nodo 2: 1
Nodo 3: 1
Nodo 4: 1
Nodo 5: 2
Nodo 6: 1
Nodo 7: 3
Nodo 8: 2
Nodo 9: 1
Nodo 10: 2
Nodo 11: 1
Nodo 12: 2
Nodo 13: 3
Nodo 14: 1
Nodo 15: 1
Nodo 16: 2
Nodo 17: 1
Nodo 18: 4
Nodo 19: 1
Nodo 20: 6
Nodo 21: 9
Nodo 22: 1
Nodo 23: 1
Nodo 24: 1
Nodo 25: 1
Nodo 26: 5
Nodo 27: 2
Nodo 28: 2
Nodo 29: 8
Nodo 30: 1
Nodo 31: 1
Nodo 32: 4
Nodo 33: 9
Nodo 34: 2
Nodo 35: 3
Nodo 36: 8
Nodo 37: 6
Nodo 38: 1
Nodo 39: 1
Nodo 40: 2
Nodo 41: 2
Nodo 42: 2
Nodo 43: 3
Nodo 44: 1
Nodo 45: 6
Nodo 46: 3
Nodo 47: 1
Nodo 48: 1
Nodo 49: 2
Nodo 50: 1
Nodo 51: 1
Nodo 52: 3
Nodo 53: 6
Nodo 54: 8
Nodo 55: 1
Nodo 56: 2
Nodo 57: 10
Nodo 58: 7
Nodo 59: 1
Nodo 60: 1
Nodo 61: 8
Nodo 62: 3
```

Figura 6.2: Solución detectando simetría - pt 1

```
Nodo 63: 1
Nodo 64: 1
Nodo 65: 1
Nodo 66: 1
Nodo 67: 1
Nodo 68: 2
Nodo 69: 2
Nodo 70: 5
Nodo 71: 2
Nodo 72: 7
Nodo 73: 1
Nodo 74: 11
Nodo 75: 1
Nodo 76: 2
Nodo 77: 2
Nodo 78: 8
Nodo 79: 1
Nodo 80: 2
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: 2
Nodo 93: 1
Nodo 94: 1
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: 2
Nodo 107: 1
Nodo 108: 6
Nodo 109: 2
Nodo 110: 1
Nodo 111: 1
Nodo 112: 1
Nodo 113: 1
Nodo 114: 1
Nodo 115: 4
Nodo 116: 9
Nodo 117: 2
Nodo 118: 2
Nodo 119: 1
Nodo 120: 2
Nodo 121: 1
Nodo 122: 2
Nodo 123: 6
Nodo 124: 2
```

Figura 6.3: Solución detectando simetría - pt 2

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

Figura 6.4: Solución detectando simetría - pt 3

7. Solución detectando simetría y sabiendo CAP de lavados

7.1. Observaciones

Esta es la solución más rápida hasta ahora. Ya que combina restricciones para acelerar la *performance* y el conocimiento que brindan los algoritmos basados en heurísticas, como se vieron en las **secciones 4 y 5**. Se utiliza como máxima cantidad de lavados, al igual que en la **sección 4**, 15 lavados.

En este caso se llega también al óptimo, **117 segundos** es el tiempo total entre todos los **11 lavados**. Este resultado se consiguió en 9 segundos.

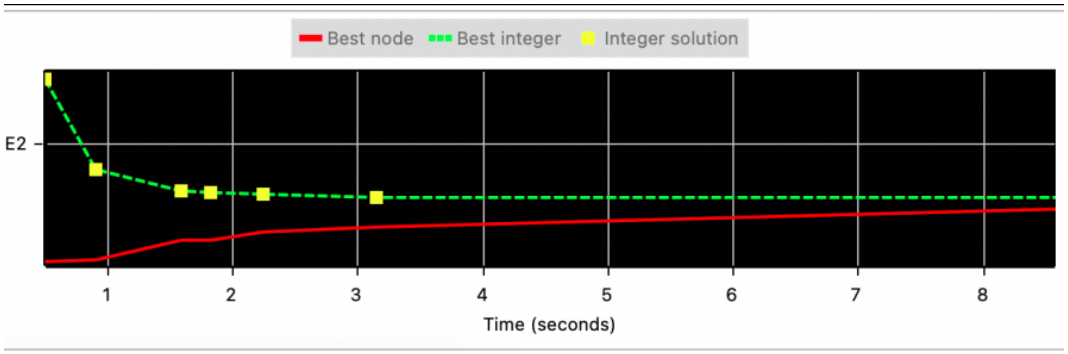

Figura 7.1: Estadísticas de la solución detectando simetría y sabiendo el CAP de lavados

7.2. Conclusiones

Cuanto más restricciones se le impongan al modelo, y sobretodo a las variables bivalentes, más veloz será el software al momento de resolverlo.

```
solution: 117 /size: 138 /time: 1667865123.003772974
Nodo 1: 1
Nodo 2: 1
Nodo 3: 1
Nodo 4: 1
Nodo 5: 1
Nodo 6: 5
Nodo 7: 3
Nodo 8: 1
Nodo 9: 1
Nodo 10: 2
Nodo 11: 1
Nodo 12: 2
Nodo 13: 3
Nodo 14: 1
Nodo 15: 2
Nodo 16: 1
Nodo 17: 1
Nodo 18: 4
Nodo 19: 1
Nodo 20: 6
Nodo 21: 1
Nodo 22: 1
Nodo 23: 6
Nodo 24: 1
Nodo 25: 1
Nodo 26: 5
Nodo 27: 2
Nodo 28: 2
Nodo 29: 2
Nodo 30: 1
Nodo 31: 1
Nodo 32: 4
Nodo 33: 2
Nodo 34: 2
Nodo 35: 5
Nodo 36: 8
Nodo 37: 2
Nodo 38: 1
Nodo 39: 1
Nodo 40: 2
Nodo 41: 2
Nodo 42: 2
Nodo 43: 4
Nodo 44: 2
Nodo 45: 6
Nodo 46: 1
Nodo 47: 1
Nodo 48: 1
Nodo 49: 3
Nodo 50: 2
Nodo 51: 1
Nodo 52: 4
Nodo 53: 3
Nodo 54: 6
Nodo 55: 2
Nodo 56: 1
Nodo 57: 9
Nodo 58: 8
Nodo 59: 1
Nodo 60: 3
Nodo 61: 4
Nodo 62: 3
```

Figura 7.2: Solución detectando simetría y sabiendo máximo de lavados - pt 1

```
Nodo 63: 1
Nodo 64: 1
Nodo 65: 1
Nodo 66: 1
Nodo 67: 2
Nodo 68: 2
Nodo 69: 2
Nodo 70: 5
Nodo 71: 1
Nodo 72: 7
Nodo 73: 2
Nodo 74: 10
Nodo 75: 1
Nodo 76: 2
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: 3
Nodo 87: 1
Nodo 88: 1
Nodo 89: 3
Nodo 90: 2
Nodo 91: 1
Nodo 92: 7
Nodo 93: 1
Nodo 94: 1
Nodo 95: 3
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: 2
Nodo 111: 3
Nodo 112: 1
Nodo 113: 1
Nodo 114: 3
Nodo 115: 5
Nodo 116: 9
Nodo 117: 1
Nodo 118: 7
Nodo 119: 1
Nodo 120: 4
Nodo 121: 1
Nodo 122: 2
Nodo 123: 5
Nodo 124: 2
```

Figura 7.3: Solución detectando simetría y sabiendo máximo de lavados - pt 2

```
Nodo 125: 1
Nodo 126: 4
Nodo 127: 1
Nodo 128: 1
Nodo 129: 1
Nodo 130: 3
Nodo 131: 1
Nodo 132: 9
Nodo 133: 4
Nodo 134: 2
Nodo 135: 7
Nodo 136: 2
Nodo 137: 1
Nodo 138: 5
```

Figura 7.4: Solución detectando simetría y sabiendo máximo de lavados - pt 3

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

*     0+    0                          300.0000        0.0000           100.00%
*     0+    0                          298.0000        0.0000           100.00%
      0     0       20.0000  1177      298.0000       20.0000     1749   93.29%
*     0+    0                          159.0000       20.0000           87.42%
      0     0       21.9681  1159      159.0000     Cuts: 669     4343   85.49%
*     0+    0                          126.0000       23.0769           81.68%
      0     0       25.8036   818      126.0000     Cuts: 873     9752   59.52%
*     0+    0                          124.0000       51.0018           58.87%
      0     0       26.8886   814      124.0000     Cuts: 610    11766   58.87%
*     0+    0                          123.0000       51.0018           58.54%
      0     0       28.8371   759      123.0000     Cuts: 552    14530   47.49%
      0     0       28.8927   798      123.0000     Cuts: 519    15393   47.49%
      0     0       29.0357   833      123.0000     Cuts: 249    16347   42.93%
*     0+    0                          121.0000       70.2013           41.98%
*     0+    0                          120.0000       71.7275           40.23%
*     0+    0                          117.0000       71.7275           38.69%
      0     0  -1.00000e+75     0      117.0000       71.7275    16404   38.69%
      0     2       30.9533   606      117.0000       71.7275    17236   38.69%
Elapsed time = 2.91 sec. (3383.23 ticks, tree = 0.02 MB, solutions = 9)
     25    18        cutoff            117.0000       71.7275    22703   38.69%
    553   247      101.3071   230      117.0000       71.7275    56373   38.69%
   1482   649       71.8917   404      117.0000       71.7275    97792   38.69%
   1803   994      101.5157   252      117.0000       71.7275   143390   38.69%
   2891  1346      103.8291   243      117.0000       71.7275   174410   38.69%
   3621  1947      113.0000   150      117.0000       71.7275   221737   38.69%
```

Figura 7.5: Logs solución detectando simetría y sabiendo máximo de lavados

8. Análisis de soluciones

8.1. Conclusiones

Analizaremos la diferencia entre las dos soluciones presentadas anteriormente; por un lado la solución que sólo pone un máximo de 15 lavados y por el otro la que utiliza este conocimiento y, además, utiliza la restricción de la simetría.

La primera de ellas muestra que llega al óptimo en los primeros 5 segundos, pero se queda, hasta pasados los 10 segundos, buscando mejores soluciones. Esto sucede porque prueba 2^{15} combinaciones de lavados posibles, cuando en la realidad muchas de esas combinaciones se repiten. No importa el numero de lavado que se utilice, solo importa la cantidad y las prendas que van en cada uno. Y de esto se aprovecha la restricción "simetría", haciendo que importe la cantidad de lavados y no cuál de los potenciales lavados se utiliza.

8.2. Sabiendo la cantidad de lavados

Utilizando ahora el conocimiento que tenemos de resultados anteriores, se intentará ver qué resultado brinda el software dándole exactamente la cantidad de lavados que se necesitan.

Lo que se logra es una solución con una tiempo de ejecución reducido. En lo que llega al óptimo en menos de 5 segundos Esto ocurre porque lo único que tiene que probar es todas las combinaciones posibles de ropas que entran en los lavados, y al haber tantas restricciones para la ropa, la velocidad del software es significativa.

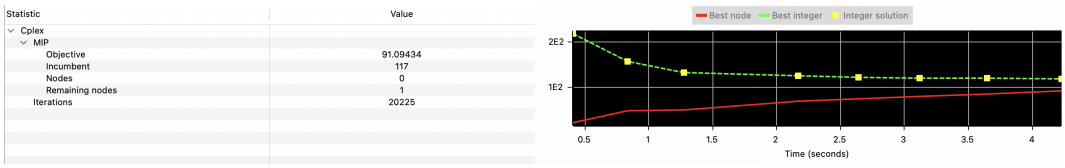

Figura 8.1: Estadísticas con OPL de la solución sabiendo exactamente la cantidad de lavados

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

*     0+    0                          220.0000        0.0000           100.00%
*     0+    0                          217.0000        0.0000           100.00%
      0     0       20.0000   915      217.0000       20.0000     1435   90.78%
*     0+    0                          155.0000       20.0000           87.10%
      0     0       26.3014   794      155.0000     Cuts: 605     3511   70.04%
*     0+    0                          132.0000       46.4384           64.82%
      0     0       37.0000   686      132.0000     Cuts: 687     5518   62.95%
*     0+    0                          124.0000       48.9027           60.56%
      0     0       41.8134   652      124.0000     Cuts: 833    10062   45.97%
*     0+    0                          120.0000       67.0000           44.17%
      0     0       44.2343   637      120.0000     Cuts: 804    12501   38.60%
*     0+    0                          119.0000       73.6803           38.08%
      0     0       49.2038   606      119.0000     Cuts: 833    14961   34.45%
*     0+    0                          118.0000       78.0000           33.90%
      0     0       62.0000   524      118.0000     Cuts: 833    17737   28.81%
      0     0       62.9387   499      118.0000     Cuts: 565    18978   28.81%
      0     0       62.9431   529      118.0000     Cuts: 648    19527   28.81%
*     0+    0                          117.0000       84.0000           28.21%
      0     0       62.9807   434      117.0000     Cuts: 303    20358   22.14%
      0     0       62.9807   509      117.0000     Cuts: 522    21124   19.66%
      0     2       63.1082   311      117.0000       97.0000    21698   17.09%
Elapsed time = 4.40 sec. (3514.57 ticks, tree = 0.02 MB, solutions = 9)
    332     9      114.0000   170      117.0000       97.0000    47717   17.09%
    992    47    infeasible             117.0000       97.0000    88385   17.09%

Clique cuts applied:  2
Implied bound cuts applied:  689
Mixed integer rounding cuts applied:  139
Zero-half cuts applied:  47
Multi commodity flow cuts applied:  9

Root node processing (before b&c):
  Real time             =    4.36 sec. (3478.60 ticks)
Parallel b&c, 12 threads:
  Real time             =    0.72 sec. (595.49 ticks)
  Sync time (average)   =    0.26 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    5.08 sec. (4074.09 ticks)
```

Figura 8.2: Logs solución sabiendo exactamente la cantidad de lavados

9. Conclusión sobre metodologias

Como se fue hablando a lo largo del Trabajo, las metodologías tienen valores positivos y valores negativos.

Para programadores, puede tentar utilizar algoritmos Greedy como método de resolución de este tipo de problemas, sobretodo si el problema varia dinámicamente a lo largo de su vida, como puede ser un típico caso en el software industrial.

Sin embargo, la programación lineal es una herramienta poderosa y flexible que permite resolver los más diversos tipos de problemas. El problema de esta es que, en muchos casos, tarda demasiado tiempo en terminar de ejecutarse, y, sin las técnicas adecuadas, puede volverse imposible esperar a su finalización.

Uno de los aspectos importantes de los algoritmos Greedy es su simpleza, tanto para programarlo como para entenderlo. Esto genera un atractivo grande. Pero su mayor defecto, en algunos problemas, no en todos, es que puede dar una solución aproximada y no exacta. Esto quiere decir que hay tipos de problemas que no encontraremos la solución óptima mediante este tipo de algoritmos. En este momento es cuando entra la programación lineal, que su mayor fuerte es la precisión con la que brinda una solución.