FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico: Minimizar Tiempos de Lavado

[71.14] Modelos y Optimización I
Segundo Cuatrimestre de 2022

Alumno	Cristóbal, Sabella Rosa
Número de padrón	106440
Email	csabella@fi.uba.ar
Fecha de Entrega	21/11/2022
Práctica	Lunes

Índice

1. Introducción

La programación lineal entera es una herramienta que nos permite expresar problemas de complejidad no polinómica en forma de un modelo matemático determinista; logrando así suplir las falencias de la programación lineal continua. Aún así, este método no será mágico: la complejidad para encontrar una solución óptima seguirá siendo no polinómica, por lo que existirá entonces una gran sensibilidad temporal si no se cuidan las restricciones impuestas. Esto llevará a que existan una serie de distintas estrategias (las llamadas "heurísticas") para encontrar una solución estimada y/o recolectar evidencia para realizar conjeturas respecto a la solución óptima, para así poder restringir el problema y reducir considerablemente el tiempo de ejecución.

En este informe se detallará la experiencia al intentar resolver un mismo problema (con distinta cantidad de datos) de complejidad no polinómica, aplicando diferentes estrategias y desembocando en un modelo de programación lineal entera.

2. Enunciado

- 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.

3. El Problema

Se supondrá que lo que se quiere minimizar en este problema será el tiempo de lavado total entre todos los lavados tal que todas las prendas terminen limpias. De esta manera, puede intuirse que se trata de una variación del problema de coloración de grafos. Esto se puede visualizar si se muestra a cada prenda como un vértice del grafo (con su valor siendo su tiempo de lavado), cada desteñido como una arista, y cada lavado como un color distinto. Entonces lo que se buscaría sería encontrar una configuración de colores tal que se minimice la sumatoria del máximo valor de cada color (representando la prenda más sucia de ese lavado).

4. Planteo Formal

4.1. Objetivo

Determinar cuáles prendas y en qué número de lavado asearlas para un intervalo de tiempo p,con el objetivo de minimizar la suma total de tiempos de lavado, en base a las restricciones de desteñido.

4.2. Hipótesis

- Toda prenda deberá pertenecer a un lavado.
- Se supondrá que la suciedad de una prenda está completamente relacionada con su tiempo de lavado, por lo que "la prenda más sucia de ese lavado"(enunciado) se tomará como sinónimo de "la prenda con mayor tiempo de lavado".
- Se buscará evitar completamente el desteñimiento. Si no se logra, se considerará que no es una solución al problema.
- l tiempo de lavado de una prenda será una constante, y no se verá afectada por ningún factor (como, por ejemplo, la cantidad de prendas con las que comparte lavado).
- El tiempo de un lavado respetará a rajatabla el enunciado, y no se verá afectado por ningún factor (como, por ejemplo, el cargado de jabón según la cantidad de prendas que contenga).

- Un lavado no podrá fallar.
- Un lavado tendrá capacidad infinita: siempre y cuando se cumplan las restricciones, podrá contener cualquier cantidad de prendas.
- La cantidad total de lavados no se considerará a minimizar.
- No se considerarán gastos por electricidad, agua, etcétera.
- Una vez una prenda forma parte de un lavado, esta
  - Tendrá que lavarse durante todo su tiempo de lavado
  - No podrá ser realocada hacia otro lavado ("lavarse de a partes")
- A la hora de minimizar, no se considerará la posibilidad de realizar múltiples lavados a la vez. Es decir, se minimizará el tiempo de lavado efectivo, independientemente de si es en paralelo con múltiples lavadoras o no.

5. Primera Entrega: Heurística Trivial

5.1. Una Primera Impresión

Desde un primer momento el problema fue encarado a sabiendas de que se trataba de una variación de coloración de grafos. Es decir, se supo que este tendría una complejidad no polinómica, y que intentar encontrar una solución exacta por medio de un algoritmo eficiente sería una imposibilidad. Fue entonces que se buscó aplicar un algoritmo greedy sencillo, es decir, un algoritmo con una heurística simple; para poder así encontrar una solución aproximada, aún cuando una solución a fuerza bruta podría haber funcionado.

5.2. El Algoritmo

El algoritmo es simple:

```
Se ordenan las prendas de la que mas tarda en limpiarse a la que menos,
Se van agregando a cada uno de los lavados:
  Si puede agregarse al primer lavado,
    perfecto, se agrega;
  si no puede (debido a la limitacion del destenido),
    se prueba agregar al lavado siguiente, sucesivamente;
  si no puede agregarse a ningun lavado existente,
    se crea un nuevo lavado y se agrega.
```

De esta forma, las prendas que más tardan en lavarse tendrán menos restricciones a la hora de encontrar dónde, lo cual facilitará que en un mismo lavado hayan prendas de duraciones similares (por ejemplo, que en un lavado hayan dos prendas de 10m y una de 9m). También, siempre que se pueda meter una prenda en un lavado para el que no incrementaría el tiempo, se intentaría lograr.

5.3. Resultado y Conclusiones

Previamente a la búsqueda de la solución estimada, se probó de encontrar el óptimo por medio de fuerza bruta. Como se previó, el tiempo resultó excesivo, por lo que se cortó la ejecución y se consideró más correcto y escalable realizar la entrega utilizando el algoritmo diseñado.

El resultado final fue de 62 minutos, con 9 lavados distintos. Fue un resultado cercano al óptimo de 61 minutos (y tardando sólo milésimas en procesarse), por lo que el desempeño de la heurística elegida resultó completamente satisfactorio. Un problema sumamente complejo puede simplificarse para estimarlo, si así se necesitara.

6. Segunda Entrega: Heurística Mejorada

6.1. Una Primera Impresión

La diferencia respecto a la entrega anterior radicó en el volumen de los datos: en vez de 20 prendas por lavar, esta vez se trataban de 385. Esto imposibilitaba el uso de un algoritmo de fuerza bruta. El primer impulso fue utilizar la misma heurística que en la entrega anterior, logrando un resultado rápido, con una solución aproximada de 493 minutos. Este resultado era satisfactorio, pero finalmente, aún cuando posiblemente podría haber formado una entrega válida, se decidió fine-tunear el algoritmo para lograr un resultado más interesante.

6.2. El Algoritmo

El algoritmo era exactamente el mismo que en la entrega pasada, con un detalle de implementación cambiado: ahora el primer paso (el ordenamiento) tendría un factor de aleatoriedad. Es decir, un ordenamiento de prendas de mayor a menor tiempo de lavado podría devolver órdenes distintos a lo largo de múltiples ejecuciones, logrando así diferentes variaciones de la heurística y cubriendo una mayor cantidad de soluciones. Esto se logró creando una suerte de API por consola (véase el README en la entrega 2), para poder ejecutar varias veces el algoritmo o para replicar una corrida determinada utilizando una semilla.

6.3. Resultado y Conclusiones

El algoritmo se dejó correr durante múltiples ejecuciones (sólo por ego, un agregado de alrededor de 600000. Tardó bastante, pero posiblemente muchísimo menos que buscando por fuerza bruta), logrando mejorar el resultado considerablemente: el óptimo pasó de 493 a 439 minutos.

Esto demostró que aún una misma heurística podría tener múltiples variaciones por explorarse, con resultados variopintos y potencialmente más cercanos al óptimo.

7. Tercera Entrega: Modelo de Programación Lineal Entera

7.1. Una Primera Impresión

El problema era exactamente igual al primero, pero esta vez la solución no era importante, sino el planteo. Se propuso un modelo entonces, utilizando las técnicas de Programación Lineal Entera vistas a lo largo de la cursada.

Nota: El modelo no se detallará en este informe, para no saturarlo. Si se quiere visualizar, se puede encontrar en la carpeta Entrega 3, en el Markdown.

7.2. Conclusiones

Gracias al contenido aprendido durante la cursada, se pudo expresar el problema en forma de un modelo de Programación Lineal Entera. Esto se logró mediante la utilización de variables bivalentes para representar que una prenda iría a un lavado (similar a representar que sean de un color determinado), al agregado de restricciones pertinentes (una prenda i y una j no pueden tener ambas una bivalente k si no pueden lavarse juntas), y a la operación MAX planteada en términos de programación lineal para conseguir los tiempos de lavado.
El mayor problema con este planteo es que la resolución seguiría siendo de complejidad no polinomial (ya que métodos de resolución de Programación Lineal Entera como Planos Cortantes o Branch & Bound también lo son!). Es entonces que, cuando se quiera plasmar este modelo en un solver, habrá que ingeniárselas para encontrar alguna forma de reducir las soluciones posibles.

8. Última Entrega: Resolución con Programación Lineal Entera

8.1. Una Primera Impresión

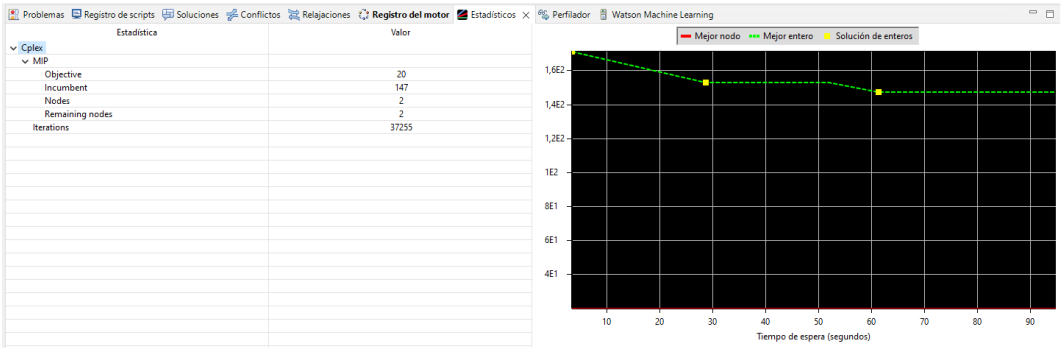
Esta vez el problema es de 138 prendas, por lo que una ejecución descuidada del modelo seguramente resultaría en tiempos de ejecución imposibles. Definitivamente tendrá que utilizarse alguna suerte de heurística para llegar a alguna clase de resultado satisfactorio.

8.2. Paso 1: Heurística propia

Se buscó una solución utilizando la heurística de la Entrega 2. El resultado, a lo largo de varias ejecuciones, resultó de 123 minutos de lavado total (otra vez, con un tiempo de ejecución en el orden de las milésimas). Nótese que el resultado final tiene 11 lavados totales. Esto podría ser utilizado como heurística si se quisieran minimizar tiempos de ejecución, como un límite posible de lavados (aunque cabe destacar que se trata de una reducción que podría eliminar una posible solución óptima).

8.3. Paso 2: Intento de Resolución "sin Heurísticas"

Al empezar la ejecución, puede visualizarse que empieza desde la solución trivial de asignar cada lavado a una única prenda, iterando lentamente entre solución y solución hasta encontrar la óptima. Esto lo hace con una técnica llamada Branch & Bound (más puntualmente, Branch & Cut, que relaja con Planos Cortantes en algún paso de la ejecución), que se encarga de ir encontrando soluciones posibles en un árbol de ellas (formadas agregando restricciones que "enterizan" a las variables y descartando ramas que no tengan soluciones mejores a alguna ya encontrada).



A los 90 segundos, va acercándose lentamente a una solución preliminar de 147 minutos. Como puede apreciarse más detallado en el engine log:

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Bound	ItCnt	Gap
*	0+	0			2760,0000	0,0000		100,00 %
*	0+	0			1467,0000	0,0000		100,00 %
*	0+	0			1048,0000	0,0000		100,00 %
*	0+	0			171,0000	0,0000		100,00 %
	0	0	20,0000	7779	171,0000	20,0000	16	88,30 %
*	0+	0			153,0000	20,0000		86,93 %
	0	0	20,0000	1608	153,0000	Impl Bds: 103	6422	86,93 %
	0	0	20,0000	2266	153,0000	Cuts: 1425	16339	86,93 %
	0	0	20,0000	1552	153,0000	Cuts: 166	23266	86,93 %
*	0+	0			152,0000	20,0000		86,84 %
*	0+	0			147,0000	20,0000		86,39 %
	0	0	−1,00000e+75	0	147,0000	20,0000	23266	86,39 %
	0	0	20,0000	1894	147,0000	Cuts: 1495	30919	86,39 %
	0	2	20,0000	1894	147,0000	20,0000	30919	86,39 %

Nota: El log contiene información excesiva para este análisis. Por ejemplo, sobre cada vez que encuentra nodos con posibles soluciones, objetivos que si no se cumplen el nodo es descartado, información sobre los bounds y los planos de corte y un porcentaje Gap, que con un cálculo detallado

en la documentación de CPLEX, da una idea del porcentaje de iteración.

En un principio de la ejecución, además, nos muestra la siguiente línea:

```
MIP Presolve eliminated 120051 rows and 0 columns.
MIP Presolve modified 12429 coefficients.
Reduced MIP has 35199 rows, 19182 columns, and 124639 nonzeros.
Reduced MIP has 19044 binaries, 138 generals, 0 SOSs, and 0 indicators.
```

Es decir, el MIP Solver de CPLEX resuelve utilizando una heurística, reduciendo las soluciones posibles, y logrando así tiempos de ejecución más potables. Posteriormente, realiza un "Performing Restart", que vuelve a realizar un recortado de soluciones posibles desde 0 (pero con la memoria de recordar los nodos previamente recorridos). Nótese que en este paso esto último nunca llegó a ocurrir, pero en los posteriores sí.

La ejecución fue cortada a los 10m.

```
Elapsed time = 485,52 sec. (157098,00 ticks, tree = 9,89 MB, solutions = 20)
   1071    721        39,0000  1121       119,0000        20,0000  1658772   83,19%
   1076    730        39,0000  1223       119,0000        20,0000  1708634   83,19%
   1079    721        37,0000  1255       119,0000        20,0000  1686851   83,19%
   1083    734        39,0000  1105       119,0000        20,0000  1720860   83,19%
   1089    736        39,0000  1274       119,0000        20,0000  1726428   83,19%
   1091    741        37,0000  1371       119,0000        20,0000  1773415   83,19%
   1097    752        39,0000   960       119,0000        20,0000  1798574   83,19%
   1101    755        37,0000  1397       119,0000        20,0000  1814960   83,19%
   1105    756        37,0000  1339       119,0000        20,0000  1818550   83,19%
   1120    770        39,0000   977       119,0000        20,0000  1876837   83,19%
Elapsed time = 557,02 sec. (170535,56 ticks, tree = 10,93 MB, solutions = 20)
   1133    790        39,0000   854       119,0000        20,0000  1965012   83,19%
   1150    801        20,0000  1401       119,0000        20,0000  2017222   83,19%

Implied bound cuts applied:  3460
Zero-half cuts applied:  229
Gomory fractional cuts applied:  92

Root node processing (before b&c):
  Real time             =   92,25 sec. (64204,80 ticks)
Parallel b&c, 4 threads:
  Real time             =  508,06 sec. (115795,06 ticks)
  Sync time (average)   =   38,04 sec.
  Wait time (average)   =    0,00 sec.
                          ----------------
Total (root+branch&cut) =  600,31 sec. (179999,86 ticks)
```

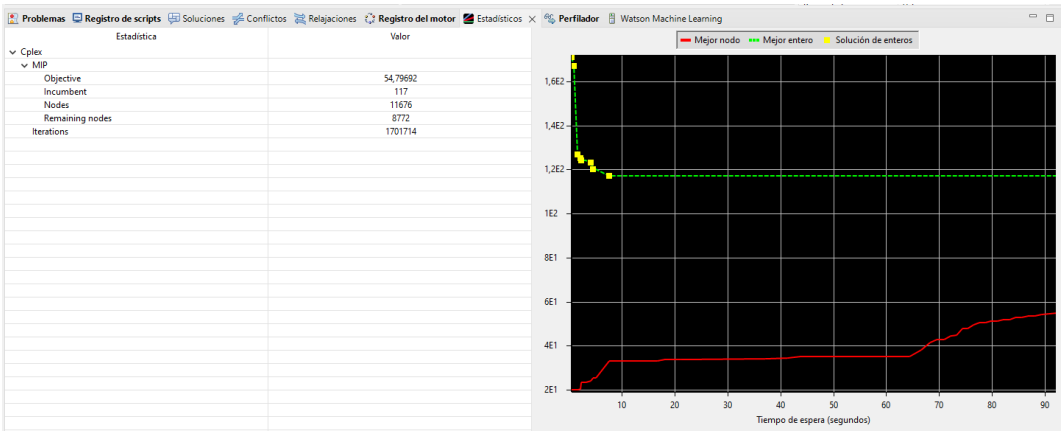
Nótese que, al terminar, brinda información sobre los cortes realizados y benchmarks de ejecución del CPU. También, la solución encontrada resultó de 119 minutos. Aún así, al haberse cortado la ejecución manualmente, esta última solución no pudo ser obtenida.

8.4. Paso 3: Intento de Resolución con Lavados Reducidos

Se toma la heurística de que no habrán más de 15 lavados en el óptimo. Con esta conjetura, ya se nota la reducción de las soluciones por explorar:

```
MIP Presolve eliminated 13053 rows and 0 columns.
MIP Presolve modified 1347 coefficients.
Reduced MIP has 3945 rows, 2085 columns, and 13532 nonzeros.
Reduced MIP has 2070 binaries, 15 generals, 0 SOSs, and 0 indicators.
```

Puede apreciarse que los números son súmamente menores que en el caso anterior. Esto debido a que reducir la cantidad de lavados posibles a modo de parámetro permite la evaluación de muchas menos variables bivalentes (que recordemos que, escalan en complejidad cuadráticamente). También, al graficarse los primeros 90 segundos, se puede apreciar el "boost" inicial que tiene:



Terminando con 90 segundos en una solución aún mejor a la que se llegó en el paso pasado con 10 minutos. Aún así, esta restricción no alcanzó para que la ejecución sea menor a 10 minutos, quedando entonces 117 como nueva cota para una solución posible.

```
Elapsed time = 565,06 sec. (583825,42 ticks, tree = 1945,99 MB, solutions = 11)
 115199  55067      114,0000      269      117,0000      105,4400  12649710     9,88 %
 115657  55410      112,0000      358      117,0000      105,4429  12709686     9,88 %
 116133  55937      113,0000      253      117,0000      105,4510  12795946     9,87 %
 116647  56289      112,0000      230      117,0000      105,4741  12855135     9,85 %
 117089  56712      113,0000      278      117,0000      105,4969  12923935     9,83 %
 117498  57002      113,0744      424      117,0000      105,5002  12982595     9,83 %
 117854  57175      114,6020      292      117,0000      105,5151  13018324     9,82 %
```

Igualmente puede apreciarse que el porcentaje Gap es infinitamente menor en este final de ejecución que en el anterior, por lo que se puede afirmar que efectivamente esta heurística logró el tiempo de ejecución.

8.5. Paso 4: Resolución Usando Restricción de Simetría

Para este paso, se aplica una restricción con la forma:

$$pesoColor_{i-1} >= pesoColor_i;\ \forall i\ tal\ que\ 2 <= i <= limiteColores$$

Siendo limiteColores la ya vista heurística de limitar la cantidad de lavados. Es decir, se agrega una restricción que dará una suerte de orden a los lavados, haciendo que los de índice i siempre tarden menos o igual que el lavado i - 1. Teniendo en cuenta que el orden de los lavados realmente no importa, esta restricción seguramente mejore en sobremanera el tiempo de ejecución (achicando considerablemente la cantidad de ramas de soluciones posibles, pudiendo ser detectadas y podadas rápidamente); sin comprometer la solución final (ya que un mínimo, por ejemplo y a 3 lavados [1, 2, 3], tardaría exactamente lo mismo que otro [1, 3, 2]; por lo que al menos una solución óptima siempre habría).

Puede apreciarse que tuvo un comienzo más tosco que los pasos anteriores, pero cerca de los 45 segundos logra un salto a una solución más aceptable. Nótese que la cantidad de soluciones a evaluar no fue reducida respecto al segundo paso:

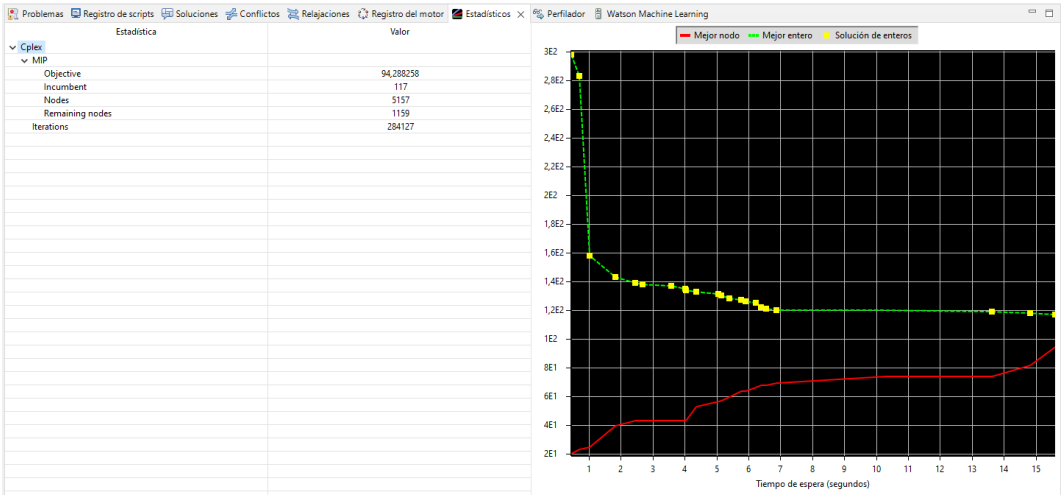
```
MIP Presolve eliminated 120138 rows and 0 columns.
MIP Presolve modified 12342 coefficients.
Reduced MIP has 35249 rows, 19182 columns, and 124427 nonzeros.
Reduced MIP has 19044 binaries, 138 generals, 0 SOSs, and 0 indicators.
```

Pero igualmente, debido al poder descartar nodos con mayor facilidad por la restricción de simetría, finalmente la ejecución logró terminar sin un corte externo, ¡a los 250 segundos! (poco más de 4 minutos). El resultado mínimo óptimo es de 117 minutos, con no más de 11 lavados. Ahora que finalmente se llegó a una solución, en el scripting log queda detallada, con el formato n° prenda ->n° lavado:

```
solution: 117 /size: 138 /time: 5852.109
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: 2
Nodo 12: 2
Nodo 13: 2
Nodo 14: 1
Nodo 15: 2
Nodo 16: 2
Nodo 17: 1
Nodo 18: 3
...
```

8.6. Paso 5: Resolución Usando Restricción de Simetría y Lavados Reducidos

Se procede entonces a aplicar ambas mejoras a la vez: se eliminan simetrías de soluciones (con óptimos repetidos) y se fuerza a que hayan menos de 15 lavados. La ejecución habla por sí sola:

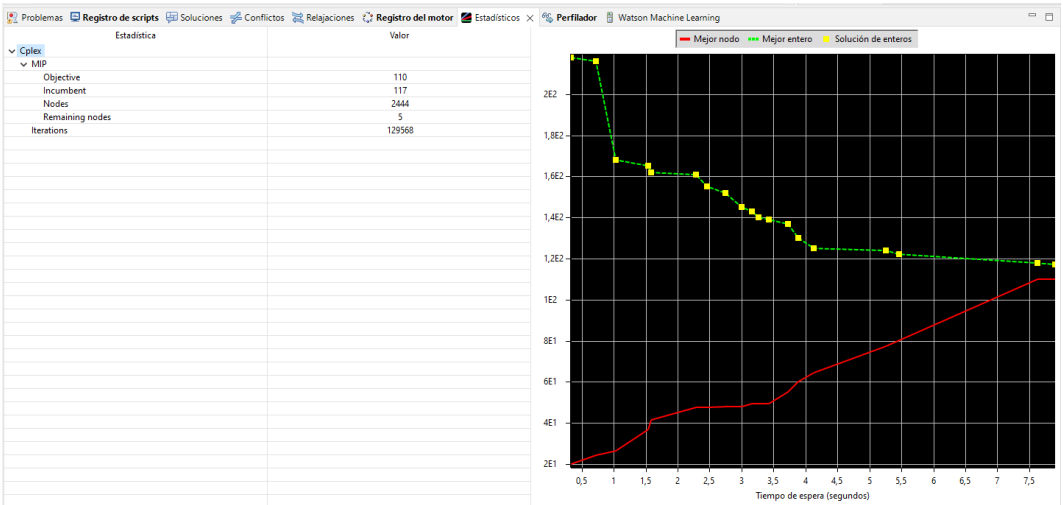


A poco más de 15 segundos, se llegó al óptimo de 117. ¿Por qué ocurrió esto? Como ya se sabía que el óptimo tiene 11 lavados, cualquier reducción de que podamos hacer será inconsecuente respecto a eliminar resultados buscados; pero el hecho de recorrer menos soluciones posibles, permite que la resolución sea infinitamente más rápida.

8.7. Paso 6: Comparación entre Pasos 3 y 5. Evaluación a no más de 11 lavados

Ambos pasos (el 3 y el 5) evalúan la misma cantidad de nodos (soluciones), ya que parten de la misma restricción de no más de 15 lavados. La diferencia radica en, como se mencionó previamente, la restricción de eliminar simetrías, que evita que nodos sean innecesariamente evaluados. Se puede observar que para un óptimo de N lavados existen otras N! soluciones iguales simplemente desordenando, y recortar algo que escala factorialmente es mucho más potente que algo que escala exponencialmente (y es por esto que la solución del paso 4 fue considerablemente mejor que la del 3, aún sin límite de lavados).

Volviendo a refinar el modelo, se propone realizar la ejecución con la restricción de simetría y con, esta vez, no más de 11 lavados. Se sabe que esto desembocará en el óptimo, ya que, como se ha mencionado varias veces, es esta misma cantidad la mínima necesaria. La ejecución completamente eficiente puede visualizarse:



La ejecución finalizó en el óptimo a menos de 8 segundos.

8.8. Paso 7: Conclusión Final. Comparación entre Heurística y Programación Lineal Entera

Ahora bien, si puede reducirse la búsqueda de un óptimo a una cantidad de tiempo completamente razonable, ¿para qué son necesarias las heurísticas? La respuesta puede intuirse según el desarrollo de este informe. La programación lineal entera no es mágica, sino que está fuertemente condicionada (y con mucha sensibilidad) respecto a las restricciones utilizadas. Falta de ellas desembocan en una solución no acotada o tiempos de ejecución extremadamente altos, mientras que un exceso de ellas podría resultar en una solución incompatible o en la eliminación accidental de la solución óptima. Es por ello que se necesita realizar una fuerte exploración del espacio del problema para determinar qué conjeturas resultarían menos perjudiciales. Sería gracias a una exploración con sentido común que la restricción de eliminar simetrías surge, y permite evitar cálculos fuertemente innecesarios; mientras que es una exploración con heurísticas la que podría determinar una cota superior para la cantidad de lavados. Es en este último punto donde se puede apreciar el verdadero poder de las heurísticas. Como ejemplo, se puede ver en el resultado del paso 1: no sólo dio una solución aproximada (no tan errada, 6 minutos de diferencia), sino que además brindó una cantidad aproximada de lavados; es decir, más importante que una "solución", brinda información que podría servir para poder acercarse al óptimo con el menor riesgo posible (obviamente nunca convendría ir directo a limitar los 11 lavados, ya que esto podría terminar eliminando el óptimo; sino tomárselo con conservaduría y tomar una cota relativamente cercana, por ejemplo, 15).

En conclusión, las heurísticas son excelentes herramientas para comprender un problema de complejidad no polinómica desde la completa ignorancia, con el potencial de brindar importante información para lograr acercarse al valor óptimo buscado. La programación lineal entera, por su parte, es extremadamente poderosa con la información y en las manos adecuadas, logrando modelar comportamientos y problemas complejos con una fuerte simplicidad y comprensión.

Nota: Los logs completos y fotos de estadísticas se pueden encontrar en las carpetas correspondientes de cada paso.