Modelos y Optimización I (91.04)

Informe Final Trabajo Práctico

1er Cuatrimestre 2024

Alumno: Giancarlo Puquio
Padrón: 101946

Enunciado

- Una lavandería tiene que lavar prendas, algunas prendas 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.

En resumen tenemos prendas y se tiene que asignar cada prenda a un lavado, no pueden ir prendas que sean incompatibles a un mismo lavado.

Entrega N°1

Para esta primera entrega se me ocurrió resolver el problema usando POO en Python. Desarrolle una clase llamada DistribuidorPrendas, esta clase tiene un método asignarPrendasALavados dónde está la lógica de distribución de prendas en los lavados, de manera que no haya prendas incompatibles en un mismo lavado.

Lo primero que probé fue recorrer una lista de prendas (las prendas de esta lista están en el orden en el cual fue leída del archivo de entrada), y asignar las prendas a un lavado, siempre y cuando en el lavado a asignar no haya prendas que sean incompatibles con la prenda a añadir. Para los datos que nos dieron para esta primera entrega con 20 prendas y 210 incompatibilidades, el número de lavados fue 8.

Lo segundo que probé fue que como tenía de dato el tiempo de lavado de cada prenda, y además el enunciado decía que el tiempo de cada lavado es el tiempo que lleva lavar la prenda más sucia osea seria el tiempo más alto de todas las prendas que tiene el lavado.

Agregue la siguiente lógica de distribución de prendas: Si una prenda puede ir a más de un lavado, irá al lavado que tenga un tiempo más cercano al tiempo de lavado de la prenda a agregar.

Por Ejemplo: Si tenemos un lavado 1 de tiempo 10, y un lavado 2 de tiempo 5, si queremos agregar una prenda de tiempo de lavado 9. Esta prenda irá al lavado 1, ya que tiene el tiempo más cercano al tiempo de lavado.

Con el mismo ejemplo anterior, si queremos agregar otra prenda con un tiempo de lavado de 6, esta prenda irá al lavado 2.

Si queremos agregar una prenda con un tiempo de lavado de 20 y que tiene la opción de elegir ambos lavados 1 y 2, esta prenda irá al lavado 1, en este caso, el tiempo total de lavado del lavado 1 pasa a ser 20.

Con esto último agregado, para los datos, el número de lavados fue de 9, empeoró el resultado anterior por lo cual opte por quedarme por la primera opción.

Entrega N°2

Para esta segunda entrega pedían que mejorará el desarrollo de la primera entrega. Con los temas vistos en las clases teóricas sobre Problemas Combinatorios fui observando cada tipo de problemas y viendo cual sería el ideal para implementarlo con el trabajo práctico. Vi que me ayudaría mucho el problema de coloreo de grafos.

Antes de probar con un algoritmo de coloreo de grafos, probe si mejoraba la solución ordenando primero la lista de prendas en función de la cantidad de prendas incompatibles que tenían (En la lista irán primero las prendas que más incompatibilidades tienen). Los datos que nos dieron para esta segunda entrega fue de 385 prendas y 19095 incompatibilidades y el resultado fue 32 lavados.

Despues probe con el algoritmo de greddy de coloreo de grafos, primero pensé en cómo relacionar el problema de coloreo de grafos con el trabajo práctico, se me ocurrió usar cada prenda como vértices del grafo, y que cada vértice (prenda) esté relacionada mediante una arista con otra prenda incompatible a ella. Asi tendria un grafo, teniendo como vértices a las prendas y que cada una de las prendas conectan mediante una arista a sus prendas incompatibles, así utilice el algoritmo de coloreo de grafos greddy , que consiste en asignar colores a los vértices de manera que ningún par de vértices adyacentes compartan el mismo color, así obtuve el mínimo número de colores necesarios para colorear el grafo, que sería para nuestro problema la cantidad de lavados. El resultado me dio el mismo, se asignó 385 prendas en 32 lavados.

Probé ordenar las prendas en la lista por tiempo de lavado de menor a mayor, y aplicar al algoritmo greedy, pero esto me dio un resultado de 51 lavados.

Conclusión, obtuve un mejor resultado tanto ordenando la lista de prendas, que vayan al principio de la lista las prendas que tienen más prendas incompatibles, como aplicando el algoritmo de greedy.

Entrega N°3

Para esta tercera entrega, nos piden buscar el óptimo, para eso utilizamos el IDE CPLEX Optimization Studio.

Lo primero que hice fue leer el código provisto y relacionarlo con el trabajo práctico, y llegue a esta conclusión:

- Nodos: Representa las prendas.
- Aristas: Representan las incompatibilidades entre las prendas.
- Colores: Representan a los distintos lavados.
- Peso de los nodos: Representa el tiempo en que tarda en lavar la prenda.
- Peso de los colores: Representa el tiempo de lavado.

Además el objetivo es minimizar el tiempo total de lavado (minimizar la suma de todos los tiempos de lavados)

Los datos ahora tienen 138 prendas (cantidad de nodos) y 986 incompatibilidades (cantidad de aristas).

Paso 1: Corran su heurística sobre la instancia. Registren el resultado obtenido.

Corriendo la heurística de coloreo de grafos greedy, para los datos del problema 3, la cantidad de lavado es 11, y el tiempo total de lavado 194.

Paso 2: Prueben correr el código sin cambios.

Al correr el código sin cambios y detenerlo a los 12 minutos (También se probó con un tiempo más amplio y se pudo observar que a partir del minuto 12 hacia adelante, el mejor resultado encontrado fue el mismo), se obtuvo como resultado 118 el tiempo total de lavado. También se obtuvo como cantidad de lavados un valor de 11.

```
Elapsed time = 708,36 sec. (135404,13 ticks, tree = 10,35 MB, solutions = 15)
   1025   862       50,5064  1115      118,0000       20,0000  1726276   83,05%
   1032   879       98,0000   688      118,0000       20,0000  1776094   83,05%
```

Engine Log (Paso 2)

Podemos ver en el log que el valor de la columna Gap es de 83,05%, esto quiere decir que la solución actual está muy lejos de la óptima. También podemos ver se encontraron múltiples soluciones enteras, 15 en total, y la mejor solución entera encontrada es 118. Algo que me llama la atención es el valor del mejor límite inferior, tiene un valor de 20, es

por eso que el valor del Gap nos da alto (83,05%) ya que es la diferencia relativa entre la mejor solución entera y el mejor límite inferior.

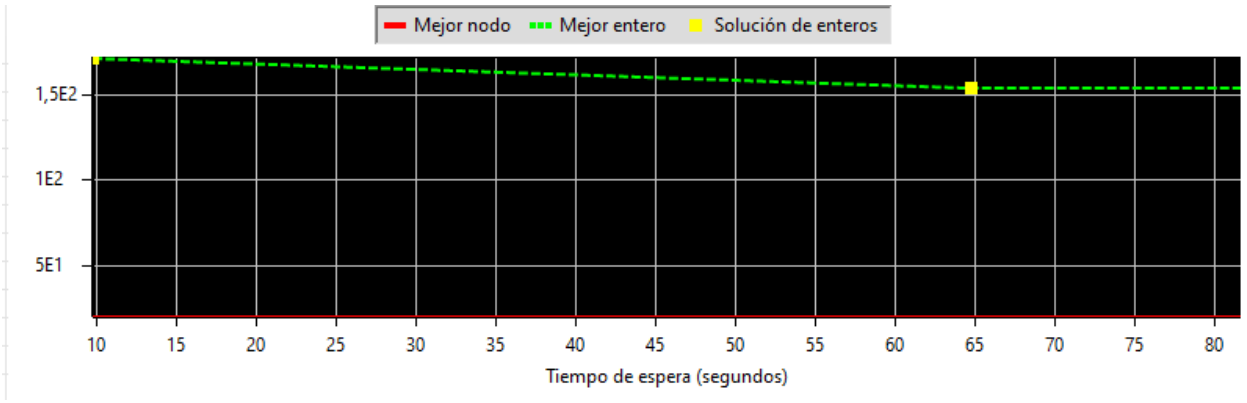

Gráfico Statistics (paso 2)

En el gráfico Statistics, podemos ver la evolución de las soluciones encontradas, Mejor nodo (línea roja) es la mejor solución continua, Mejor enteró (línea verde) indica la evolución de la mejor solución entera encontrada.

En el gráfico Statistics tomado en los primeros 90 segundos, la mejor solución encontrada tiene un valor de 153.

Paso 3: Sabiendo que existe una solución que usa 15 lavados (se obtuvo mediante una heurística) ver cómo acelerar reduciendo el modelo (cantidad de restricciones).

Al reducir el límite de colores a 15 el modelo tiene un espacio de búsqueda más reducido y optimizado, llevando a cabo una mejor asignación de colores y pesos dentro de las restricciones impuestas. La mejor solución encontrada es de 117 y se utilizaron 11 colores. Se detuvo la ejecución a los 12 minutos.

```
Elapsed time = 684,11 sec. (244198,94 ticks, tree = 1,21 MB, solutions = 17)
  49702   169      104,0000   441      117,0000      104,0000  5603118   11,11%
  50143   312      108,9596   381      117,0000      104,0000  5621104   11,11%
  51275  1238      111,0000   320      117,0000      104,0000  5692802   11,11%
```

Engine Log (Paso 3)

El Gap bajó considerablemente a diferencia del paso anterior, ahora tiene un valor de 11,11% ya que ahora el mejor límite inferior es 104, esto indica que la solución encontrada está cerca de la óptima.

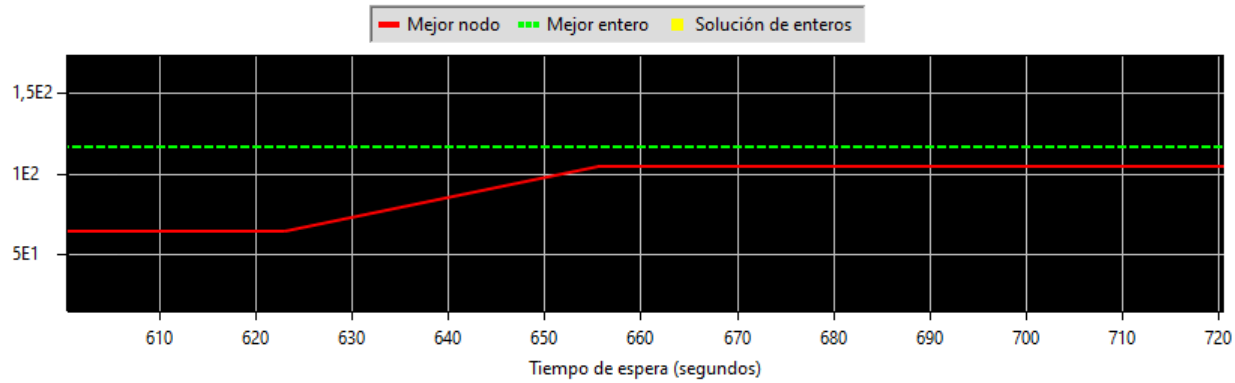

Gráfico Statistics (paso 3)

La corrida en este paso lo detuve en 12 minutos, en el gráfico estadístico, pude observar que a partir del segundo 620, el mejor nodo (línea roja) aumenta y se acerca al mejor entero (linea verde), en este intervalo el Gap paso de 45.30% a 11,11%, al igual que el Best Bound que pasó de 64 a 104.

Paso 4: Volviendo al modelo original (sin el límite de 15 lavados), descomentar la restricción "simetría".

Al ejecutar el modelo agregando la restricción de simetría, el modelo terminó de correr en 246,16 sec. Y se obtuvo un resultado de 117 y la cantidad de lavados es de 11, el valor del Gap es 6,72%

```
Elapsed time = 231,64 sec. (76713,46 ticks, tree = 4,61 MB, solutions = 26)
   5453  1381      107,1786   152      117,0000       94,8637   446625   18,92%
   5961  1347      112,1204   192      117,0000       99,1375   479890   15,27%
   6412  1150    infeasible            117,0000      102,4162   508227   12,46%
   7149   692      111,9864   208      117,0000      109,1408   539524    6,72%
```

Engine Log (Paso 4)

```
« Registro de scripts (suelte el código de script aquí para ejecutarlo)
solution: 117 /size: 138 /time: 2246349.031
Cantidad de colores usados: 11
Nodo 1: 1
Nodo 2: 1
Nodo 3: 1
Nodo 4: 1
Nodo 5: 1
Nodo 6: 1
Nodo 7: 4
Nodo 8: 7
Nodo 9: 1
Nodo 10: 2
```

Registro de scripts (Paso 4)

Al ver el Registro de scripts note que se usaron 11 colores para colorear los nodos del grafo, pero a diferencia de las otras corridas, estos colores tienen ahora un valor entre 1 y 11, ya que antes también se utilizaban 11 colores, pero los colores no tomaban valores consecutivos.

Gráfico Statistics (paso 4)

En el gráfico estadístico tomado en los primero 90 sec de ejecución, vemos que en determinado momento el valor del mejor entero se acerca al valor del mejor nodo.

Paso 5: Modificar el modelo del punto anterior para que aproveche el límite de 15 lavados.

Al limitar el número de colores a 15 y agregar la restricción de simetría, el tiempo de ejecución del modelo fue de 43,50 segundos. La solución óptima aún sigue siendo de 117.

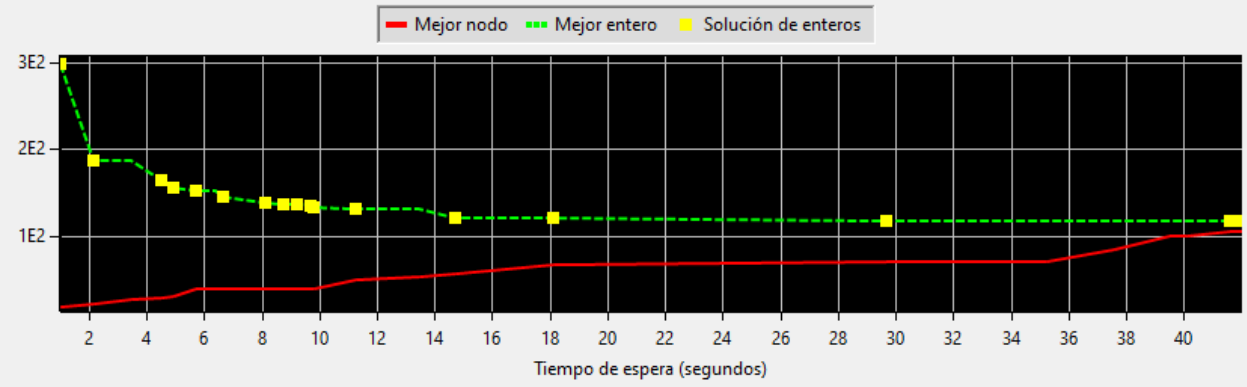

Gráfico Statistics (paso 5)

Estadística	Valor
Cplex	
MIP	
Objective	105,264321
Incumbent	117
Nodes	4780
Remaining nodes	618
Iterations	281507

Esta vez en el gráfico Estadístico vemos que ahora el valor del mejor nodo se acercó al valor del mejor entero. El valor del objetivo tiene un valor aproximado de 105,26, y la mejor solución factible encontrada es 117.

Paso 6: Comparar el paso 3 y el 5, repetir la prueba sabiendo que existe una solución de 11 lavados.

El paso 3 se ejecutó sin la restricción de simetría y con el límite de colores de 15, la ejecución se tuvo que detener a los 12 minutos porque el modelo seguía corriendo, en cambio en el paso 5, con la restricción de simetría y el límite de colores de 15, la ejecución duró 43.50 segundos. Esto sucede ya que la restricción de simetría reduce el espacio de búsqueda, evita que se exploren soluciones que son equivalentes bajo permutaciones de colores.

Poniendo ahora un límite de colores de 11 (poner este límite tiene sentido, ya que en las corridas anteriores pude observar que se usaban exactamente 11 colores para colorear el grafo) y con la restricción de simetría, la ejecución terminó en 30 segundos.

```
Elapsed time = 26,14 sec. (9594,70 ticks, tree = 0,76 MB, solutions = 10)
   3300   276      114,3333    74      117,0000      110,9023   176522    5,21%
```

Engine Log (Paso 6)

El valor del Gap es de 5,21%, esto quiere decir que el mejor valor encontrado hasta el momento está aproximadamente un 5,21% por encima del valor objetivo. La mejor solución encontrada sigue teniendo un valor de 117.

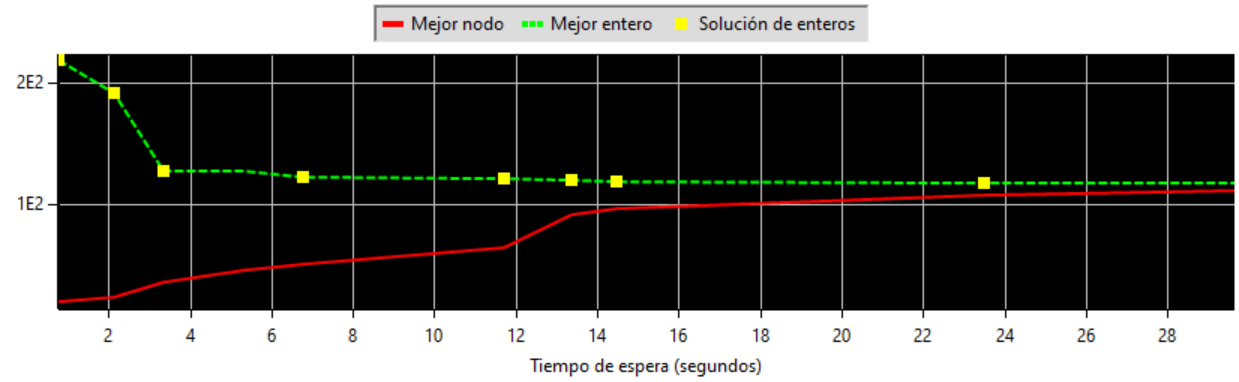

Gráfico Statistics (paso 6)

El gráfico estadístico nos dice que al final de la ejecución, el mejor nodo (línea roja) y el mejor enteró (línea verde) convergen, lo que nos indica que la solución encontrada es una solución cercana a la óptima.

Paso 7: Comparar en el informe la heurística (paso 1) con la solución mediante programación lineal entera.

En el paso 1 con la heurística greedy nos dio como resultado 194 el tiempo de lavado, y con la solución mediante la programación lineal entera se obtuvo un resultado de 117, una gran diferencia. Cabe aclarar que cuando se utiliza la heurística de coloreo de grafo greedy, estábamos minimizando la cantidad de colores a usar y no el tiempo total de lavado.

Fue de mucha ayuda agregar la restricción de simetría, y también limitar la cantidad de colores a usar a 11, ya que reducían considerablemente el espacio de búsqueda del modelo.