Modelos y Optimización I - Trabajo práctico 4

Joaquin Hojman - Padrón 102264

Enunciado

En esta cuarta entrega se pide que busquen el ÓPTIMO y lo suban a modelosuno.okapii.com, es MUY recomendable usar el código provisto CPLEX https://modelosuno.okapii.com/content/modelos_wvcp.zip

Armar un informe con cada uno de los pasos, incluir gráficos (solapas "Statistics", "Engine log", "Scripting log", etc.) y todo lo que consideren pertinente / interesante. El gráfico de "Statistics" tomarlo en los primeros 90 seg
Pasos

1) Corran su heurística sobre la instancia. Registren el resultado obtenido.
2) Prueben correr el código sin cambios, pueden detenerlo a los 10 minutos si no termina. Indicar en el informe todo lo que notan de esta corrida
3) Sabiendo que existe una solución que usa 15 lavados (se obtuvo mediante una heurística) ver cómo acelerar reduciendo el modelo (cantidad de restricciones), pueden detenerlo a los 10 minutos si no termina. Indicar en el informe todo lo que notan de esta corrida
4) Volviendo al modelo original (sin el límite de 15 lavados), descomentar la restricción "simetría". Indicar en el informe todo lo que notan de esta corrida
5) Modificar el modelo del punto anterior para que aproveche el límite de 15 lavados. Indicar en el informe todo lo que notan de esta corrida
6) Comparar el paso 3 y el 5, repetir la prueba sabiendo que existe una solución de 11 lavados
7) Comparar en el informe la heurística (paso 1) con la solución mediante programación lineal entera

Punto 1

En este punto se nos pide correr el set de datos de esta entrega con el código python realizado en el trabajo práctico 2. En mi caso el TP2 fue realizado usando un algoritmo de coloreo de grafos con una técnica greedy para su resolución.

Corriendo el algoritmo sobre la instancia dada se halló una solución que resuelve el problema utilizando 10 lavados, los cuales tardan en total 158 unidades de tiempo. Algoritmo y separación de prendas por lavado podrán hallarse como “tp1.py” y “solucion5.txt”, respectivamente, en el repositorio.

Notamos que en la solución ofrecida por la cátedra también se realizara un coloreo para resolver el problema.

Punto 2

Ahora correremos el código entregado sin realizar modificaciones.

```
Elapsed time = 450,53 sec. (152098,91 ticks, tree = 24,01 MB, solutions = 20)
   1868  1369       56,6156   764      118,0000       37,0000  3628845   68,64%
   1886  1394      103,0000   405      118,0000       37,0000  3693169   68,64%
   1898  1397      110,0000   337      118,0000       37,0000  3700637   68,64%
   1919  1413       94,0000   517      118,0000       37,0000  3756431   68,64%
   1929  1417       94,0000   582      118,0000       37,0000  3765139   68,64%
   1944  1492      103,0000   490      118,0000       37,0000  3977103   68,64%
   1962  1498      103,0000   638      118,0000       37,0000  3985540   68,64%
   1991  1477       68,0000   638      118,0000       37,0000  3950128   68,64%
   2008  1512       68,0000   955      118,0000       37,0000  4059983   68,64%
   2064  1557      113,0000   358      118,0000       37,0000  4198517   68,64%
```

Vemos que según esta tabla se obtuvo una solución de 118 unidades de tiempo para realizar todos los lavados.

En este caso tuvimos que frenar la ejecución ya que llevaba más de 10 minutos sin cambios, y el gap se mantenía fijo en 68,64%. Es probable que necesitemos cambiar el problema, agregando o quitando restricciones para llegar a una solución que pueda terminar y darnos la información de cuántos colores tendremos que usar y a que color pertenece cada prenda.

Observamos que el tiempo para llegar a esta solución fue de 450 segundos.

Observamos también el gráfico de estadísticas durante los primeros 90 segundos de la corrida.

Punto 3

Sabiendo que existe una solución que usa 15 lavados una forma de reducir el modelo es justamente poner el limite de colores en 15. Hasta ahora, el limite de colores era N, la cantidad de prendas del lavado, caso que podría darse si todas las

prendas fueran incompatibles entre sí, entonces necesariamente tendríamos un grafo completo y por ende N colores.

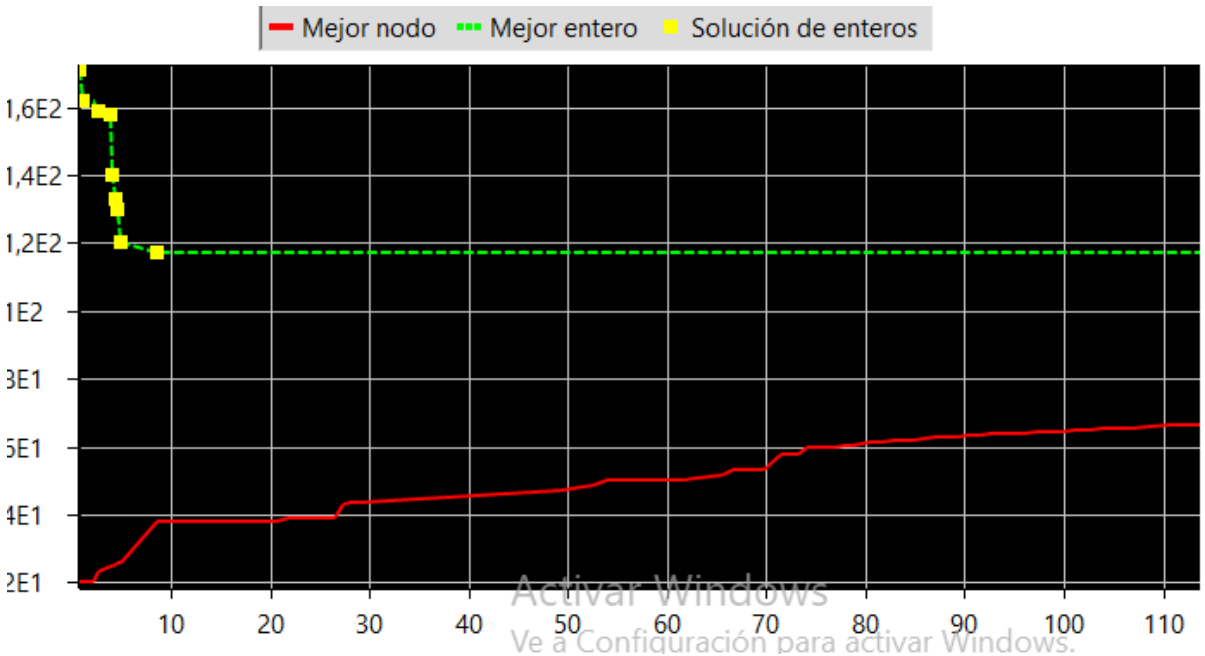
Ahora sabemos que no precisamos sino 15 colores, y por eso la variable “colores” valdrá en un rango de 1 a 15.

Limitando entonces la cantidad de colores a 15 y corriendo el modelo, luego de 10 minutos podemos observar que hay una solución que consigue lavar todas las prendas en 117 unidades de tiempo.

```
 117726 40266      116,0000   235      117,0000      106,0000 12099362    9,40%
 118688 40964      108,8000   331      117,0000      106,0000 12230411    9,40%
 119873 42452      112,0000   374      117,0000      106,0000 12461158    9,40%
 121046 43046      114,0000   311      117,0000      106,0000 12548961    9,40%
 122139 43749      110,8235   305      117,0000      106,0000 12671847    9,40%
 123399 44862      109,4880   402      117,0000      106,0000 12845324    9,40%
 124527 45983      112,0000   282      117,0000      106,0000 13011998    9,40%
 125462 46897      110,9682   401      117,0000      106,0000 13167602    9,40%
 126487 47548      113,5839   333      117,0000      106,0000 13267103    9,40%
 127297 48521      114,0000   307      117,0000      106,0000 13435641    9,40%
Elapsed time = 513,38 sec. (303437,20 ticks, tree = 1578,41 MB, solutions = 14)
```

La ejecución se detuvo luego de 10 minutos, como ya se comentó, y se logró percibir que el gap se mantenía fijo en 9,40% durante las ultimas iteraciones antes de cortar.

Comparando el valor de la solución con la del ítem anterior, sin la restricción de la cantidad de colores, vemos que la solución mejora en 1 unidad con esta restricción de tan solo 15 colores.



Punto 4

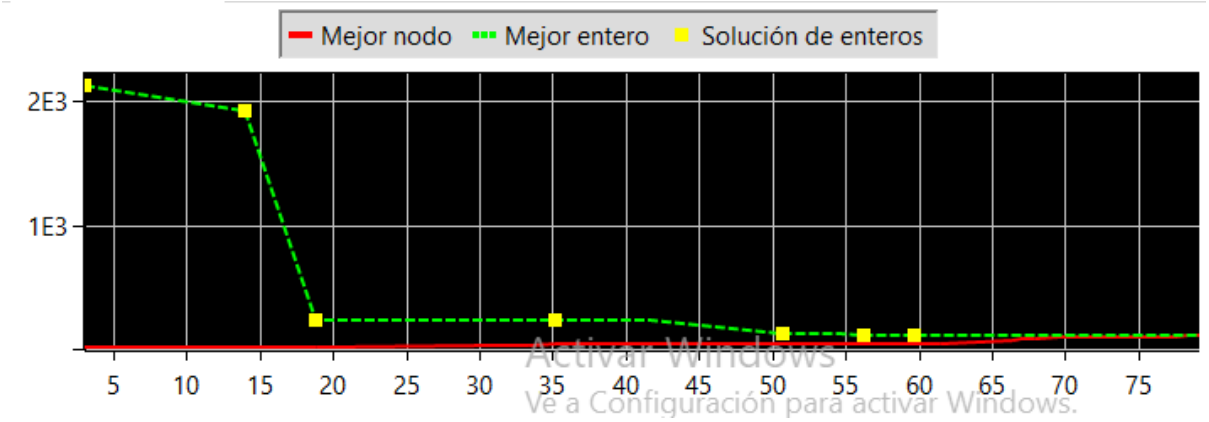
Para el modelo original volvemos a buscar soluciones. Ahora hay una restricción de simetría que dice que el peso de un color debe ser mayor al peso del siguiente y menor al peso del anterior.

```
Elapsed time = 66,31 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
```

En este caso la solución tarda 117 unidades de tiempo en lavar todas las prendas de ropa. Son necesarios 11 colores para resolver el problema (esto lo veo en la pestaña de Registro de scripts). En el software se informa también a que color o lavado pertenece cada prenda.

Vemos que la solución mejora respecto a cuando no implementaremos la restricción de simetría. El tiempo necesario para obtener los resultados en esta corrida fue de 66,31 segundos.

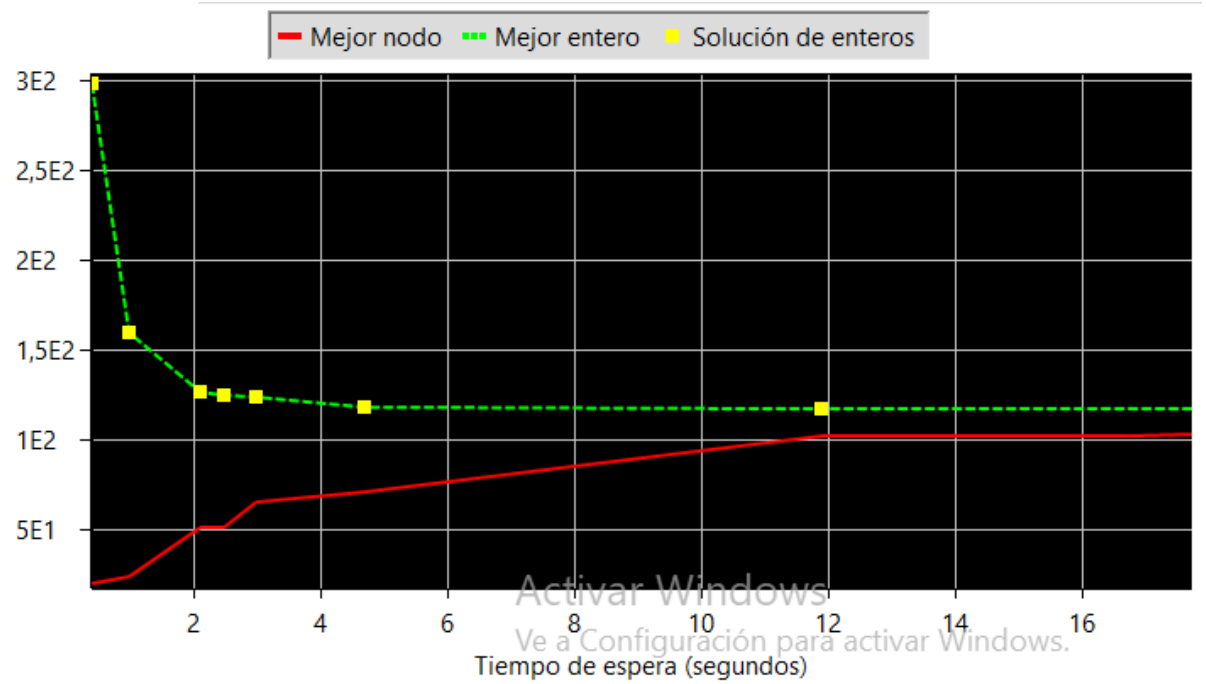


Punto 5

Se nos solicita ahora una nueva corrida del modelo, manteniendo la restricción de simetría y agregando la restricción de como máximo 15 lavados que habíamos utilizado en el ítem 3.

```
Elapsed time = 12,95 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%
```

Al correr el modelo volvemos a obtener una solución de 117 unidades de tiempo, y que utiliza 11 colores diferentes para poder realizar los lavados (es decir, 11 lavados bastaran para lavar todas las prendas). El tiempo necesario para obtener la solución fue de 12,95 segundos.



Punto 6

El paso 2 nos había dado una solución de 118 unidades de tiempo para lavar todas las prendas. Luego conseguimos mejorar esta solución a 117 unidades de tiempo tanto en el ítem 3 como en los ítems 4 y 5. La radical diferencia entre las ultimas 3 corridas mencionadas es el tiempo tardado en lograr esta solución.

En el ítem 3 requerimos mas de 10 minutos y ni siquiera logro terminar la ejecución, debimos cortarla manualmente, y eso que tenia la limitación de 15 colores como máximo. Luego para los ítems siguientes agregamos la restricción de simetría, que acelero impresionantemente los tiempos y logro que la ejecución efectivamente terminara, dándonos además la distribución de las prendas en cada lavado.

No solo la restricción de la simetría provoco mejoras en la ejecución: la restricción de 15 colores como máximo también jugo su papel, y eso podemos verlo comparando los tiempos de ejecución entre los ítems 4 y 5. En el ítem 4

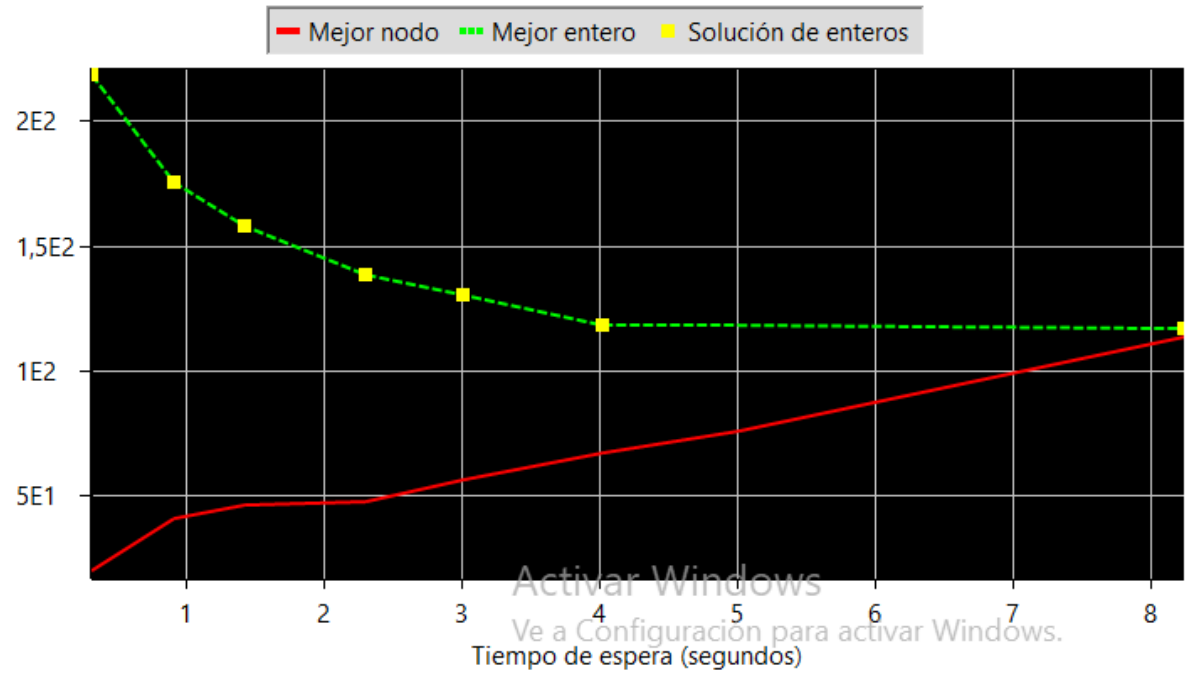
necesitamos algo mas de un minuto para obtener los resultados buscados, pero en el ítem 5 tan solo hicieron falta 12 segundos para obtener la misma solución.

Nos piden ahora que reduzcamos la cantidad de colores máximos de 15 a 11. Esto tiene sentido, recordemos que en los ítems 4 y 5 la solución ofrecida usaba 11 lavados y no 15, por lo que podemos reducir este valor tranquilamente. Para la prueba mantendremos la restricción de simetría.

```
Elapsed time = 5,91 sec. (3794,95 ticks, tree = 0,02 MB, solutions = 7)
    105    28      110,4781   179      118,0000       80,8627    34088   31,47%
    728   326      115,5385    75      118,0000       95,7751    54124   18,83%
   1333   629        cutoff            118,0000      102,5994    67457   13,05%
   1910   713      110,6800    99      118,0000      108,1333    83465    8,36%
   2511   743        cutoff            118,0000      110,4000   106722    6,44%
*  2846+  539                          117,0000      113,0000               3,42%
*  2889   500      integral     0      117,0000      113,0000   120674    3,42%
   3224    47        cutoff            117,0000      114,3053   127394    2,30%
```

Volvimos a obtener una solución de 117 unidades de tiempo, pero tardamos la mitad del tiempo en obtenerla, tan solo 5,91 segundos.

Se usaron, lógicamente, 11 colores distintos o lavados para resolver el problema, y el software nos ofrece la distribución de las prendas en esos 11 lavados.



Si tratásemos de reducir aun mas la cantidad de colores, por ejemplo, a 10, no podríamos obtener una solución valida. Parece que llegamos al optimo con una solución de 11 colores diferentes.

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

   0     0    infeasible                                      1124
```

Punto 7

Compararemos ahora la solución que venimos trabajando con la solución que obtuvimos en el punto 1. Recordemos que aquella solución fue obtenida con un algoritmo greedy de coloreo de grafos.

En esta solución en cuestión se obtuvo una solución de tan solo 10 lavados, pero que lograba lavar todas las prendas en 158 unidades de tiempo. Vemos que esta solución es usa menos colores o lavados que la solución usando un modelo de programación lineal entera, que precisaba 11 lavados, pero a su vez necesita más unidades de tiempo para llevar a cabo los 10 lavados. Es decir, la solución con PLE tarda menos tiempo con mas lavados que la heurística con un coloreo greedy.

¿Por qué sucede esto? La explicación es sencilla: el algoritmo greedy tiene como función y objetivo reducir al mínimo la cantidad de colores usados, no la suma del mayor valor de cada uno de sus lavados. Y por supuesto, minimizar la cantidad de colores de ninguna manera nos garantiza que se minimizara la suma de máximos valores de cada color.

Si se quisiera que el algoritmo greedy resolviera el problema minimizando esos valores máximos probablemente habría que hacerle importantes modificaciones o directamente no usar un coloreo de grafos.

Comparación de las diversas entregas

Se realizaron 4 entregas diferentes del trabajo practico a lo largo de la materia. Inicialmente, la primera idea que surgió para tratar de resolver el problema fue una especie de fuerza bruta. Rápidamente claudique con esta idea ya que me pareció que sería más difícil resolverlo por fuerza bruta que pensar una idea inteligente para hacerlo, además de menos interesante y enriquecedor.

Pensé entonces en una solución greedy para la instancia del problema entregada por la catedra. Si bien era greedy también seguía siendo bastante artesanal: consistía en ordenar las prendas de forma descendente poniendo primera a la que mas tiempo tardaba su lavado. De esta forma se recorría el arreglo de prendas intentando poner las más grandes juntas. La solución obtenida fue aceptable, pero a la hora de subir la solución y mirar el ranking note que compañeros de curso habían logrado soluciones mejores, es decir que el algoritmo que había pensado aun podía intentar mejorarse.

Pensando que seguramente el problema de mi primer algoritmo estaba en que utilizaba muchos lavados diferentes, para la segunda solución también ordene las

prendas, pero ahora de forma descendente por aquellas que tenían mas incompatibilidades con otras prendas. Luego en orden de mayor a menor el algoritmo unía la prenda con mas incompatibilidades con aquellas que no tuvieran incompatibilidades con ella, o con el resto de las prendas ya agregadas al lavado, de mayor a menor (según el orden definido). Luego de subir esta solución obtuve un mejor tiempo de lavado para todas las prendas, valor que compartía con varios compañeros en el ranking del curso, pero no superado por nadie, con esto quedé ya satisfecho y di por concluida la resolución de este primer trabajo practico.

Luego fue momento de encarar el segundo trabajo practico, donde entregada una instancia con muchos mas valores que la anterior, se nos pedía modificar el algoritmo intentando mejorarlo. Fue de esta manera que se me ocurrió que podía usar un algoritmo de coloreo de grafos (aprendí bastante sobre este tipo de algoritmos al cursar la materia Teoría de Algoritmos I). El problema de esto es que un algoritmo de coloreo de grafos es NP-Completo, por lo cual temía que encontrar la solución usando este tipo de algoritmo insumiera una gran cantidad de tiempo. Es por eso que utilice una solución greedy a este tipo de algoritmos, que el inconveniente que tiene es que utiliza algunos cálculos random, por lo cual diversas corridas pueden ofrecer distintos resultados.

Lo que tuve que hacer fue realizar varias corridas del algoritmo a fin de quedarme con la mínima que encontrar y fuera un valor que considerara aceptable. Cuando compare la mejor solución del algoritmo greedy de coloreo con la solución que me daba el algoritmo hecho para el tp1 para esta instancia, la solución del coloreo era muchísimo mejor, por lo cual subí esa solución al ranking del curso donde se mantuvo durante todo el periodo de entrega como la mejor solución. A pesar de esto ya tenia mis dudas sobre que fuera la “mejor” solución posible, pues sabia que reducir la cantidad de colores no es reducir el tiempo máximo de cada color sumados y por ende podía haber soluciones mejores con más lavados.

Para la tercera entrega en principio no hacia falta programar, sino pasar el modelo realizado al papel como un modelo de programación lineal. En el mismo, intente adaptar mi solución de coloreo de grafos de mínima cantidad de colores a mínima suma entre los valores máximos de cada color. Luego tomé la instancia de ese tp y corrí sobre el mi algoritmo greedy de coloreo, notando previamente que era la misma instancia del tp1. Asumí que iría a mejorar el tiempo de lavado que había obtenido con mi primer algoritmo, pero contrario a esto, la solución del algoritmo greedy de coloreo era peor que la solución ordenando por incompatibilidades. Esto resulto algo descorazonador, si bien para problemas grandes aparentemente el algoritmo greedy de coloreo era mejor, para problemas pequeños parecía convenir usar el ordenamiento por incompatibilidades.

Durante esta ultima entrega, quedo patente que el algoritmo de coloreo de grafos mediante greedy podía dar una solución aceptable, con la que podríamos manejarnos y estar satisfechos, pero lejos de ser la mejor. El asunto de la reducción de colores y no del tiempo de lavado máximo de cada color genera una insalvable brecha entre la solución provista y la óptima, lograda con PLE. ¿Tiene alguna ventaja? Probablemente para muchas personas sea mas fácil codear un algoritmo greedy en python (o buscarlo en alguna biblioteca de internet) antes que aprender a utilizar un software menos utilizado para el común como CPLEX. Si obtener el

optimo fuera VITAL para algún propósito, convendría fuertemente utilizar si es posible una solución por PLE que asegura el óptimo, a menos que se desee dedicar largas y tortuosas horas para modificar el algoritmo de coloreo de grafos para que busque otra cosa (minimizar la suma de tiempos máximos de cada color).

Justamente esto ultimo es lo que hace el algoritmo de PLE provisto por la catedra, usando un modelo de optimización utiliza varios componentes de un algoritmo de coloreo de grafos (colores, aristas, nodos) a lo que precisamos para resolver el problema planteado.

Concluimos de esta forma con los trabajos prácticos de esta materia. Considero muy útil el conocimiento aprendido ya que ofrece una nueva forma de no solo pensar, sino de resolver problemas complejos que quizás utilizando heurísticas diseñadas en algún lenguaje de programación pueden no conseguir el resultado optimo fácilmente.