MA4702. Programación Lineal Mixta : Teoría y Laboratorio. 2024.
Profesor: José Soto.
Profesor Auxiliar: Álvaro Márquez
Profesor Ayudante: Paolo Martiniello

Laboratorio 3: Instrucciones

- **Reglas:** Se recomienda avanzar en todos los ejercicios de este laboratorio durante los bloques del miércoles. Sin embargo, para efecto de revisión, este laboratorio está dividido en dos partes.
- Debe entregar un .ipynb con los ejercicios 1, 2, 3 y 4 resueltos antes de las 18:00 del día de hoy. Puede dejar la salida ordenada de la parte (c) de cada ejercicio para la entrega final.
- Luego, debe entregar un archivo .ipynb con el laboratorio completo (todos los ejercicios resueltos) antes de las 23:59 del día lunes 20, de mayo. En esta entrega puede (si lo desea) mejorar la entrega de los cuatro ejercicios ya entregados, pero de hacerlo necesita además crear una celda donde explique los cambios realizados.
- **Entregables:**
  - En ambas ocasiones debe entregar un archivo .ipynb. Antes de entregarlo, límpielo y ejecute TODAS sus celdas (RESTART KERNEL AND RUN ALL CELLS). Esto puede tomar tiempo.

1. Datos grupos.

Complete los datos de esta celda con todos los integrantes de su grupo.

2. Preparacion.

Esta sección contiene instrucciones. Ejecute todas las celdas y comprenda sus funciones.

3. Ejercicios

Ejercicio 1 Escriba las funciones

`leeragendamientogeneral(nombrearchivo)`

`leergrafo(nombrearchivo)`

`leerintervalos(nombrearchivo)`

descritas en el marco teórico. Puede suponer que todos los datos involucrados son números enteros.

(Nota: No usaremos `leeragendamientosimple(nombrearchivo)` por lo que no necesita entregarlo. Las funciones anteriores serán usadas para leer los archivos de entrada provistos.)

Los ejercicios (2, 3, 4, 6, 7, 8) tienen el siguiente esquema de 3 partes cada uno

ESQUEMA

(a) Modelar el problema solicitado como un Programa Lineal Mixto y escribir en markdown/latex en la celda correspondiente del archivo jupyter el modelo

(b) Escribir una función `modelo_N(...)` donde `N` es el número de ejercicio, y ... es una secuencia de arreglos que contiene los parámetros (datos) del modelo. Esta función debe devolver el modelo de la parte (a) implementado en JuMP/Gurobi.

(c) Usando la función creada y las funciones del ejercicio 1, optimice la(s) instancia(s) indicadas, usando como tiempo límite de optimización 1 minuto. Muestre en pantalla **la solución obtenida** (independiente si es óptima) *de la manera más legible posible*. Por ejemplo, si la solución es un coloreo c, muestre una lista que tenga para cada $i \in V$ un par $(i, c(i))$, o si la solución es un agendamiento, muestre para cada $j \in J$ el trío (j, i, S_j). **Adicionalmente** muestre una tabla/dataframe que tenga el **valor objetivo** obtenido, su **gap**, el **tiempo de creación del modelo**, **tiempo de optimización**, **número de variables** y el **número de restricciones**)

Ejercicio 2 Aplique el esquema al problema (**color**) de coloreo de un grafo $G = (V, E)$ con la menor cantidad de colores. Aquí los datos son (n, m, E) con $n = |V|, m = |E|$ y E el conjunto de aristas del grafo. Use como instancias de prueba `color1.txt` y `color2.txt`

Ejercicio 3 Aplique el esquema al problema (**tardanza**) que consiste en minimizar la suma de las tardanzas de un problema de agendamiento con datos $(n, m, E, (p_{ij})_{ij}, (r_j)_j, (D_j)_j, \preceq)$. En esta ocasión, suponga que todos los datos son números enteros pequeños y llame $Q = \max_j r_j + \max_i \sum_{ij} p_{ij}$. Es claro que existe un agendamiento óptimo para (**tardanza**) cuyo makespan es a lo más Q. Como todos los datos son enteros, usted puede discretizar el tiempo de cada máquina en *celdas* unitarias (1,2,..., $|Q|$) indexadas por su tiempo final. Use (entre otras) variables x_{ijt} binarias que indiquen si el trabajo j es asignado a la máquina i, con tiempo de inicio $S_i = t$.

Use como instancias de prueba `agenda1.txt` y `agenda2.txt`

Ejercicio 4 Vuelva a aplicar el esquema para el mismo problema (**tardanza**), pero ahora escriba un modelo que no discretice el tiempo. En particular, su modelo debería considerar S_j (o C_j o ambas) como una variable continua no negativa.

Use como instancias de prueba `agenda1.txt` y `agenda2.txt`

Ejercicio 5 El problema **(intervalos)** de coloreo de intervalos se puede modelar como coloreo de grafos. Escriba una función `crearGrafo(intervalos)` que reciba los intervalos entregados por la salida de su función `leerintervalos(nombrearchivo)` y devuelva un grafo G (similar a la salida de su función `leergrafos(nombrearchivo)`) tal que, cualquier k-coloreo de G corresponda a una asignación de intervalos disjuntos a k máquinas.

Use ahora las funciones creadas en el ejercicio 2 para encontrar un coloreo mínimo para los intervalos de las instancias `intervalo1.txt`, `intervalo2.txt` siguiendo la parte (c) del esquema.

Ejercicio 6 Agregando restricciones de manera adecuada, aplique una técnica simple para quebrar simetría en el problema de coloreo **(color)** obteniendo un nuevo modelo **(color-simetria)**. Repita el ejercicio 2 para este nuevo modelo, y optimice las instancias `color1.txt`,`color2.txt`,`intervalo1.txt` e `intervalo2.txt`.

Ejercicio 7 Elija uno de los modelos de tardanza (ejercicio 3 o 4) y modifíquelo para que permita resolver el problema **(intervalos)** de coloreo de intervalos, para esto reescriba el modelo siguiendo las siguientes indicaciones:

1. Considere $|V|$ máquinas idénticas
2. Considere los intervalos (s_i, t_i) como trabajos con tiempo de liberación $r_{ij} = s_i$ para todo s_i y deadline $t_i = D_i$,
3. Imponga que la tardanza sea 0.
4. Agregue alguna variable que represente si cada máquina es o no es usada y use esto para escribir el objetivo.
5. Cuando termine, **simplifique el modelo lo más que pueda**, y aplique un quiebre de simetría entre las máquinas idénticas.

Aplique el esquema al modelo resultante y pruébelo en los archivos `intervalo1.txt` e `intervalo2.txt`.

Ejercicio 8 Finalmente, vuelva a modelar el problema **(intervalos)** de coloreo de intervalos desde cero, tomando prestado ideas de coloreo y de agendamiento. Hágalo de la manera más eficiente y simple que pueda. Aplique el esquema al modelo resultante probándolo en los archivos `intervalo1.txt` e `intervalo2.txt` .

Las siguientes ideas les pueden permitir eliminar por completo la simetría:

1. Si se considera $\mathbb{R}$ como una linea de tiempo, algunos intervalos preceden por completo a otros en cualquier agendamiento. Además, si a y b son intervalos tales que ninguno precede a otro (i.e. son incomparables) entonces no pueden agendarse en la misma máquina (pues se intersectan).
2. En vez de recordar *en qué máquina / de qué color* está cada intervalo a. Piense en una estrategia que use para cada par de intervalos a y b donde a precede a b, una variable $x_{a,b}$ indicatriz del evento b *es el siguiente intervalo agendado en la máquina donde está agendado* a.
3. Interprete las variables x como caminos vértice-disjuntos que están asignados a máquinas distintas. Agregando una fuente y un sumidero artificial e ideas de flujo en redes, idee un programa lineal entero que minimice el número de caminos.