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

Laboratorio 4: Generación de Columnas.
Flujo Multiservicio

1. Flujo multiservicio fraccional (multicommodity flows).

La entrada del *problema de flujo multiservicio* consiste en un digrafo $G = (V, A)$, una función de capacidad $u\colon A \to \mathbb{Z}_+$, una función de costo por unidad $c\colon A \to \mathbb{R}_+$ y un conjunto finito K de *servicios*, donde cada $k \in K$ está definido por un nodo origen $s_k \in V$, un nodo destino $t_k \in V$ y una demanda $d_k \in \mathbb{Z}_+$.

En este problema se desea mandar para cada servicio k, d_k unidades de *servicio* de tipo k desde el origen s_k hasta el destino t_k a través del digrafo G, respetando que la cantidad total de servicios (es decir la suma sobre todos los $k \in K$) que pasan por cada arco está limitada por su capacidad. Deseamos hacer esto al menor costo posible. Este problema se modela como un programa lineal (MCF: *multicommodity flow problem*).

$$\text{(MCF)} \quad \min \sum_{e\in A} c_e \sum_{k\in K} x_e^k$$

$$\text{s.a.} \quad \sum_{k\in K} x_e^k \le u_e. \qquad \forall e \in A \tag{1}$$

$$x^k(\delta^+(v)) - x^k(\delta^-(v)) = \begin{cases} d_k & \text{si } v = s_k \\ -d_k & \text{si } v = t_k \\ 0 & \text{si } v \in V \setminus \{s_k, t_k\} \end{cases} \qquad \forall k \in K \tag{2}$$

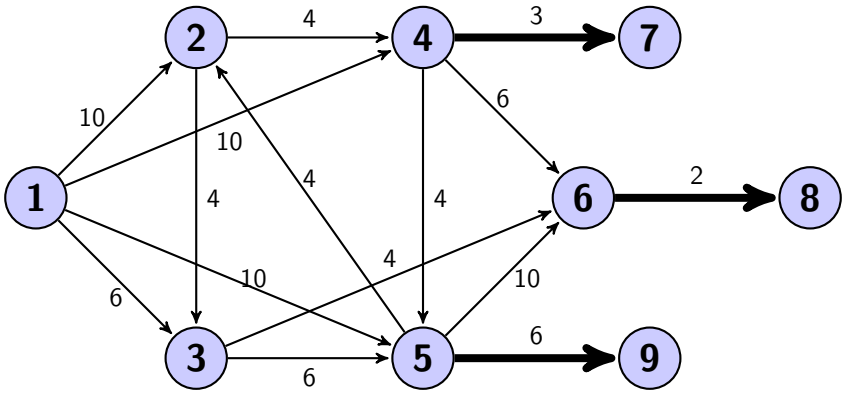
$$x^k \in \mathbb{R}_+^A \qquad \forall k \in K.$$

Acá, la variable x_e^k representa la cantidad de flujo del servicio k que pasa por el arco $e \in A$.

Si además pedimos que cada flujo por separado sea integral, es decir que $x^k \in \mathbb{Z}_+$, entonces el problema se conoce como (MCIF: *multicommodity integer flow problem*).

1.1. Modelo directo y *warmstart* solutions

Para evitar problemas de factibilidad, supondremos también que para cada par origen-destino s_k, t_k existe un arco artificial (s_k, t_k) con capacidad igual a la suma de las demandas y con costo muy alto (para efectos del problema a estudiar, estos arcos adicionales ya estarán incluidos en los archivos de entrada). En la siguiente instancia de ejemplo, los valores en los arcos corresponden a sus costos, los arcos con linea delgada tienen una capacidad de 4, los arcos con linea gruesa tienen capacidad de 10.



Hay 4 servicios descritos en la tabla a continuación **como fue descrito antes, estos introducen arcos artificiales con capacidad igual a la suma de las demandas 17 y costo alto igual a 100**.

origen	destino	demanda
1	8	5
1	7	4
5	8	3
1	9	5

Warmstart. Como sabemos que existe una solución factible a priori, resulta útil pasársela a julia/gurobi como incumbente antes de comenzar a optimizar. Hay varias formas de hacerlo, una relativamente simple consiste en hacerlo al momento de la creación del modelo. Por ejemplo, si tenemos tres variables y_1, y_2, y_3 enteras no negativas en un modelo `mimodelo` y deseamos que tome una solución inicial $(a_1, a_2, a_3) = (6, 0, 1)$, podemos escribir

```
a=[6 0 1]
@variable(mimodelo, y[i=1:3]>=0, Int, start = a[i])
```

A este proceso se le conoce como *warmstarting*.

1.2.

Para instancias pequeñas, tanto MCF como MCIF problema se puede resolver directamente con bastante rapidez. Sin embargo a medida que las instancias crecen se hace conveniente usar métodos de decomposición para resolverlas.

El objetivo de este laboratorio es implementar la descomposición Dantzig Wolfe de (MCF) es decir, de la versión fraccional.

Para esto observamos que las restricciones (2) están naturalmente descritas por bloques (las restricciones asociadas a cada servicio k involucran variables distintas), mientras que las restricciones (1) son restricciones de enlace entre los distintos bloques.

Para cada $k \in K$, llamemos X_k al poliedro.

$$X_k = \{\, x^k \in \mathbb{R}_+^A : \quad x^k(\delta^+(v)) - x^k(\delta^-(v)) = d_k \cdot ([\![v = s_k]\!] - [\![v = t_k]\!]) \,\},$$

donde $[\![\cdot]\!]$ se evalúa a 1 si la expresión que está dentro del corchete es verdadero. Notamos que X_k es el conjunto de todos los s_k - t_k flujos de valor d_k.

Sea $\mathcal{P}_k$ el conjunto de todos los s_k - t_k caminos en G. Sea además $\mathcal{C}$ el conjunto de todos los ciclos (dirigidos) en G. Usando el teorema de descomposición de flujos (del curso anterior) sabemos que para todo $x^k \in X_k$, existen coeficientes $y_{k,p} \colon p \in \mathcal{P}_k$ y $z_{k,c} \colon c \in \mathcal{C}$ tal que

$$\begin{aligned} x^k &= \sum_{p \in \mathcal{P}_k} y_{k,p} \chi^p + \sum_{c \in \mathcal{C}} z_{k,c} \chi^c \\ \sum_{p \in \mathcal{P}_k} y_{k,p} &= d_k \\ y_{k,p}, z_{k,c} &\geq 0 \; \forall p \in \mathcal{P}_k, \forall c \in \mathcal{C}_k \end{aligned}$$

La idea de la descomposición de Dantzig-Wolfe (DW) en este caso consiste en reescribir el sistema

$$\begin{aligned} \text{(MCF)} \quad \text{mín} \sum_{e \in A} c_e \sum_{k \in K} x_e^k & \\ \text{s.a.} \quad \sum_{k \in K} x_e^k \leq u_e. \quad & \forall e \in A \\ x^k \in X_k \quad & \forall k \in K. \end{aligned} \tag{1}$$

reemplazando las variables x^k por las variables $y_{k,p}$ y $z_{k,c}$ descritas antes; y luego usar métodos de generación de columna

Para simplificar notación, llamemos además $\mathcal{P} := \bigcup_{k \in K} \mathcal{P}_k$ (entendido como unión disjunta) y para cada $p \in \mathcal{P}$, definamos el costo del camino p como $c_p = \sum_{e \in p} c_e$.

2. Problemas

Su grupo debe entregar un archivo jupyter con todas las respuestas a este laboratorio.

Compromiso ético Las respuestas o el trabajo parcial entregado solo puede ser trabajado al interior de su grupo. No debe compartir resultados parciales o código, con integrantes de otro grupo. Al entregar sus respuestas se está comprometiendo a cumplir los requisitos de este párrafo, en caso que el equipo docente encuentre demasiadas similitudes entre las entregas de dos grupos, podrá solicitar a cualquiera de los integrantes que explique el código entregado, pudiendo la respuesta cambiar la nota de la entrega para el grupo completo.

Los datos de este ejercicio vienen en un formato texto plano.

Ejercicio 1: Escriba una función `red(nombrearchivo)` que sea capaces de leer entradas para el problema de este laboratorio y guardarla en un formato útil. Estudie el archivo **mcftest.txt** que viene adjunto para comprender el formato de las filas y columnas. La información viene con el siguiente formato:

- La primera linea contiene 2 enteros, $|V|$ k;
- Las siguientes k lineas son de la forma s_i t_i d_i, donde i varía de 1 a k.
- Luego viene la descripción de los arcos del grafo, estos vienen en una lista de 0 o más líneas, cada línea representa un arco. El arco (i, j) de costo c_{ij} y capacidad u_{ij} viene descrito con 4 enteros: i j c_{ij} u_{ij}.

Ejercicio 2:

(a) Escriba una función `modelo_MCF(..., warm, entero)` que recibe los datos una instancia (en el formato que usted haya usado de acuerdo al ejercicio anterior) un valor booleano `warm`, y un valor booleano `entero`. Esta función debe devolver el modelo (MCF/MCIF) en variables $(x^k)_{k\in[K]}$ (se recomienda internamente en el modelo usar el formato $x[i,j,k]$ donde $(i,j)\in A; k\in K$).

Si se recibe un valor `warm=true` entonces se usa la solución inicial indicada en el texto como incumbente, si no, se debe ejecutar sin solución inicial. Si se recibe un valor `entero=true` entonces se debe exigir que las variables x_e^k sean enteras, si no se dejan simplemente como reales no negativas.

(b) Escriba una función que, dada una solución $x=(x^k)_{k\in[K]}$ (obtenida del modelo) y entregue un dataframe similar al que sigue.

Row	cola	cabeza	d1	d2	d3	d4	d5	d6
	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64
1	1	2	1	2	3	4	5	6
2	2	1	3	4	1	2	3	4
...								

Es decir, cada fila es un arco del grafo, las primera y segunda columna son el inicio y fin del arco, y las siguientes corresponden a la cantidad de flujo de cada tipo de servicio que pasa por ese arco.

(c) Escriba una función `resuelve1(nombrearchivo, warm, entero)` que use las partes anteriores para resolver una instancia `nombrearchivo` (use un límite de tiempo de 5 minutos), y que reporte en pantalla (idealmente en un dataframe), **adicionalmente** el **valor objetivo** obtenido, su **gap**, el **número de iteraciones simplex usadas**, el **tiempo de creación del modelo**, **tiempo de optimización**, **número de variables** y el **número de restricciones**). Además usando la parte anterior debe reportar la mejor solución encontrada.

Pruebe su función en la instancia `mcftest.txt`, debería obtener una solución de valor 246. Hágalo tanto para warm=true/false y para entero=true/false.

Ejercicio 3:

(a) Escriba una demostración (directamente en el archivo jupyter) de que al aplicar el método de DW descrito en este documento a (MCF), se obtiene el siguiente sistema:

$$\begin{aligned}
\text{(DW)}\quad \text{mín} \sum_{k\in K}\sum_{p\in\mathcal{P}_k} c_p y_p & \\
\text{s.a.} \sum_{k\in K}\sum_{p\in\mathcal{P}_k:\, e\in p} y_p \le u_e. \quad \forall e\in A & \qquad (3)\\
\sum_{p\in\mathcal{P}_k} y_p = d_k. \quad \forall k\in K & \qquad (4)\\
y\in\mathbb{R}_+^{\mathcal{P}} & \qquad (5)
\end{aligned}$$

Justifique porqué no es necesario usar las variables z_c en esta formulación.

Para aplicar generación de columnas, requerimos entender el master problem asociado a la relajación lineal restringida a subconjuntos $\overline{\mathcal{P}}_k \subseteq \mathcal{P}_k$. Más precisamente llamemos

$$(\mathrm{MP}(\overline{\mathcal{P}})) \quad \min \sum_{k\in K} \sum_{p\in\overline{\mathcal{P}}_k} c_p y_p$$

$$\text{s.a.} \sum_{k\in K} \sum_{p\in\overline{\mathcal{P}}_k :\, e\in p} y_p \le u_e. \quad \forall e \in A \tag{6}$$

$$\sum_{p\in\overline{\mathcal{P}}_k} y_p = d_k. \quad \forall k \in K \tag{7}$$

$$y \in \mathbb{R}_+^{\overline{\mathcal{P}}} \tag{8}$$

(b) Calcule y escriba el dual del PL anterior, $(\mathrm{DP}(\overline{\mathcal{P}}))$ con variables $\{\alpha_e\}_{e\in A}$ y $\{\beta_k\}_{k\in K}$, donde los α_e son no-positivos y los β_k son irrestrictos.

(c) Llame $(\bar{y})$ y $(\bar{\alpha}, \bar{\beta})$ a un par de soluciones óptimas para los problemas MP y DP asociados a un conjunto fijo $\overline{\mathcal{P}} = \bigcup_{i=1}^{k} \overline{\mathcal{P}}_i$ de columnas. Demuestre que el **pricing problem** (el problema de determinar que columna(s) agregar) se puede ver por bloques, y que para cada bloque $k \in K$ el problema equivale a encontrar un camino $p^* \in \overline{\mathcal{P}}_k$ de largo mínimo con respecto a la función de largo $g\colon A \to \mathbb{R}_+$, dada por

$$g_e := c_e - \bar{\alpha}_e \ge 0$$

y luego revisar si $g(p^*) = \sum_{e\in P^*} g_e$ es menor que $\bar{\beta}_k$. ¿En qué caso debe agregar p^* como columna?

Ejercicio 4 Si bien el problema de camino de largo mínimo en un grafo con largo no negativos se puede resolver con algoritmos combinatoriales muy eficientes (como Dijkstra), en este ejercicio usted tendrá que implementarlo mediante el uso de un PL (que de hecho será integral).

(a) Dado un digrafo $G(V, A)$, un origen $s \in V$, un destino $t \in V$, y una función de largo $g\colon A \to \mathbb{R}_+$ en sus arcos. Escriba una función `resuelveflujominimo(...)` que encuentre la manera (un flujo) de mandar una unidad de flujo de s a t en el grafo G, a costo mínimo (puede poner capacidades idénticamente igual a 1 en los arcos), resolviendo un PL puro.

(b) Demuestre (argumente rápidamente) que cualquier solución extrema x^* del PL anterior debe ser entero.

(c) Llame $S(x^*)$ al conjunto de arcos e tal que $x^*_e = 1$. Del teorema de descomposición de flujos sabemos que x^* es la suma de una indicatriz de un solo camino P junto con 0 o más indicatrices de ciclos de costo 0. El camino P será el camino de costo mínimo de s a t. Escriba un método `encuentracamino(...)` que dado un vector x^* de este estilo, encuentre dicho s-t camino P. Para esto tiene dos opciones (usted puede elegir cual implementar)

a) Escriba un método que haga BFS sobre el conjunto $S(x^*)$ y devuelva la lista ordenada de vértices del camino encontrado.

b) O bien, escriba un PL que calcule nuevamente un s-t flujo de costo mínimo en el grafo $(V, S(x^*))$ donde los costos son ahora idénticamente iguales a 1 sobre todo $S(x^*)$. La solución de este PL serán un vector que vale 1 sobre los arcos del camino. Devuelva la lista ordenada de los vértices del camino encontrado.

Ejercicio 5

(a) Complete en el archivo .ipynb la función `gencol-fraccional(nombrearchivo)` de generación de columnas que le permitirá resolver el master problem $(\mathrm{MP}(\mathcal{P}))$. La función debe usar como columnas iniciales los

caminos asociados a los arcos directos entre las fuentes y destinos. En cada iteración, debe agregar columnas en todos los bloques $k \in K$ donde el pricing problem lo detecte. Recuerde que los problemas internos son todos PL puros (no hay restricciones de integralidad). Este método debe entregar como salida cuatro objetos: el número de iteraciones usadas, la colección `columnas` completa de columnas encontradas (la lista de caminos), el tiempo total utilizado, y una solución asociada. Si lo desea puede limitar el número de iteraciones en el código a un valor alto (por ejemplo 500).

(b) Complete en el archivo .ipynb la función `iteracion-final(columnas)` que reciba las columnas generadas por el proceso anterior y resuelva la versión entera del master problem con dichas columnas. Esta es la solución final que entrega el método.

(c) Escriba una función `gencol-entero(nombrearchivo)` que use las partes anteriores para encontrar una solución entera en la instancia `nombrearchivo` (use un límite de tiempo de 5 minutos), y que reporte en pantalla el **valor objetivo** obtenido encontrado, su **gap**, el **tiempo total del proceso** y la mejor solución encontrada (tanto en variables y_p, describiendo sus caminos, como en variables x_e^k).

Pruebe su función en la instancia `mcftest.txt`, debería obtener una solución de valor 246. Por ejemplo, su solución podría provenir de:

k	origen	destino	caminos	costo total
1	1	8	$4 \cdot (1,3,6,8) + 1 \cdot (1,2,4,6,8)$	$4 \cdot 12 + 1 \cdot 22$
2	1	7	$4 \cdot (1,4,7)$	$4 \cdot 13$
3	5	8	$3 \cdot (5,6,8)$	$3 \cdot 12$
4	1	9	$4 \cdot (1,5,9) + 1 \cdot (1,2,4,5,9)$	$4 \cdot 16 + 1 \cdot 24$

Ejercicio 6 Realice un análisis comparativo simple del **tiempo de ejecución** y el **número total de iteraciones simplex realizadas** entre cada uno de los 6 algoritmos para MCF/MCIF creados en este laboratorio, corriendo las instancias **`mcfinstXX.txt`** donde **`XX`** va de 1 a 20. Le recomendamos crear funciones adicionales que automaticen el proceso (entregue esas funciones también).

Su análisis debe resolver todas las instancias los siguientes algoritmos:

- Usar el modelo `modelo-MCF` para las 4 posibilidades de sus parámetros booleanos.
- Generación de columnas fraccional `gencol-fraccional`.
- Generación de columnas entero `gencol-entero`.

Incluya estadísticas apropiadas (promedio, por ejemplo) y **una tabla con los valores objetivos de cada instancia encontrados** (solo el valor).

Nota: Como mínimo usted debe correr 18 de las 20 instancias con los 3 algoritmos. Si no logra correrlas todas con alguno de los algoritmos debe explicar porqué.

Nota: Los valores objetivos pueden dar diferentes entre si, la versión entera y fraccional pueden tener valores objetivos distintos. Más aún, la versión de generación de columnas entero es una restricción del problema MCIF (solo retorna una solución factible basada en las columnas encontradas, no necesariamente la solución entera óptima de MCIF)