Curso de Investigación de Operaciones

Instituto Mexicano del Petróleo

Rodrigo Alexander Castro Campos

Universidad Autónoma Metropolitana Azcapotzalco
Departamento de Sistemas

Otoño de 2017
Ciudad de México, Mexico

Tabla de contenido

Universidad Autónoma Metropolitana
Casa abierta al tiempo Azcapotzalco

Solucionadores de programación lineal y entera

GLPK es un software gratuito y de código abierto para resolver problemas de optimización lineal y entera. Es una alternativa a otros paquetes de software comerciales como CPLEX o Gurobi, los cuales son mucho más eficientes (tanto en velocidad de cálculo como en el consumo de memoria) pero sólo pueden usarse con fines no académicos tras adquirir una licencia que puede llegar a costar miles o decenas de miles de dólares.

Solucionadores de programación lineal y entera

GLPK puede llegar a resolver algunas instancias con cientos de variables y miles de restricciones en un tiempo razonable, por lo que es un adecuado primer acercamiento a este tipo de software. GLPK está disponible tanto en Linux como en Windows. La página oficial del proyecto es `https://www.gnu.org/software/glpk/` y los ejecutables para Windows están disponibles en `http://gusek.sourceforge.net/gusek.html`.

GLPK, GLPSOL y GUSEK

GLPK implementa un conjunto de rutinas para resolver problemas de optimización lineal y entera además de que provee de un pequeño intérprete (para aquéllos que sepan programar). Por otra parte, GLPSOL es un programa pensado en ejecutarse directamente desde la terminal o línea de comandos (por ejemplo, *cmd.exe* en Windows) al cual se le debe proveer de un modelo de programación lineal ya escrito en un archivo de texto. GUSEK es un editor de texto que permite invocar a GLPSOL mediante una interfaz gráfica en lugar de usar la línea de comandos.

Formatos de modelos para GLPK

Un modelo de programación lineal o entera se puede expresar de una manera sencilla siguiendo el formato LP, aunque éste presenta algunos inconvenientes menores. Si bien no es el formato preferido por GLPK (el cual es MathProg e incluye un pequeño lenguaje de programación), el formato LP es ampliamente soportado por otros paquetes de software. Modelaremos problemas de optimización siguiendo el formato LP y usando GLPK con la interfaz gráfica GUSEK para obtener sus soluciones.

Sobre el uso de modelos y programas

Disclaimer No es viable escribir a mano un modelo que contenga cientos de variables y miles de restricciones. Los modelos de optimización de tamaño industrial se generan mediante programas que consumen los datos de entrada y le entregan el modelo al software de optimización. ¡Incluso algunos modelos son tan grandes que no es viable generarlos completos!

Sobre el uso de modelos y programas

Muchos solucionadores tienen la capacidad de interactuar con otros programas *durante* la optimización para que éstos cooperen en la búsqueda de mejores soluciones o para que le entreguen al solucionador nuevas restricciones cuando éstas se vuelvan relevantes. A final de cuentas, la computadora es la evolución de la calculadora y es importante aprender a usarla: muchas tareas se vuelven viables de esta manera.

El formato LP

Un archivo en formato LP tiene extensión *.lp* y su sintaxis puede consultarse en `https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.4.0/ilog.odms.cplex.help/CPLEX/File_formats_reference/topics/LP.html`. La primera línea del archivo debe incluir la palabra **maximize** o **minimize** para indicar el sentido de la optimización, seguida de una expresión lineal que es la función objetivo. El producto de una variable por un coeficiente constante se escribe con la constante seguida del nombre de la variable. Por ejemplo:

$$\begin{aligned} &\text{maximize} \\ &3\,x + 2\,y + 4\,z \end{aligned} \tag{1}$$

El formato LP

Después de la función objetivo se indican las restricciones del problema, cuya sección se inicia con una línea que contenga las palabras **subject to** para posteriormente listar las restricciones una a una. Se admiten los operadores relacionales $=, <, <= (\leq), >, >= (\geq)$ pero una restricción un tanto molesta del formato LP es que todas las variables deben aparecer del lado izquierdo del operador relacional y todas las constantes deben aparecer del lado derecho. Por ejemplo, la restricción *3 x - 5 <= y* es inválida y debe reescribirse como *3 x - y <= 5*. Aunque GLPK permite no poner espacios entre el coeficiente y la variable, se recomienda.

$$\begin{aligned} &\text{subject to} \\ &2\,x + 4\,y + z <= 5 \end{aligned} \tag{2}$$

El formato LP

La siguiente sección es la de cotas, la cual es opcional (ya que todas las cotas se podrían incluir en la sección anterior como si fueran restricciones). Esta sección se introduce con la palabra **bounds** seguida de una lista de cotas con la variable siempre del lado izquierdo.

$$\begin{aligned} &\text{bounds} \\ &x <= 1 \\ &1 <= y <= 2 \\ &z <= 1 \end{aligned} \tag{3}$$

El formato LP

Posteriormente se pueden iniciar las secciones opcionales **integer**, **general**, **integers** o **generals** para listar los nombres de las variables que deban ser enteras. De la misma manera, se pueden iniciar las secciones **binary** o **binaries** para listar los nombres de las variables que deban ser binarias. Para que una variable sea continua, debe excluirse de estas secciones.

$$\begin{array}{c} \text{integers} \\ \text{y} \\ \text{binaries} \\ \text{x z} \end{array} \tag{4}$$

El formato LP

Para variables binarias es innecesario indicar sus cotas. El archivo puede terminar opcionalmente con la palabra **end**. Un comentario de línea inicia con la barra invertida \. Para resolver un modelo LP, basta con abrir el archivo desde GUSEK y elegir la opcion *Tools* $\rightarrow$ *Go*. Tras resolver el modelo, GLPSOL (que es invocado por GUSEK) emitirá el valor óptimo de la función objetivo pero no emitirá directamente los valores de variables que llevan a alcanzar dicho óptimo. Hay que habilitar la opción *Generate Output File on Go* para ver los valores de las variables.

El problema Bin Packing

Este problema consiste en guardar N objetos, cada uno con cierto volumen, usando la cantidad posible de mochilas de capacidad C. Es fácil ver que una instancia del problema no tiene solución si algún objeto tiene volumen $> C$ y también es fácil ver que en caso contrario, cada objeto podría ir en su propia mochila (pero por supuesto, el objetivo es usar la menor cantidad de mochilas).

El problema Bin Packing

Nosotros modelaremos el problema de la siguiente manera. La familia de variables binarias $x_{i,j}$ indicarán si el objeto i se guardará en la mochila j. La variable binaria m_j indicará si la mochila j fue usada por algún objeto (asumiremos que tenemos el mismo número de mochilas que de objetos por el argumento presentado anteriormente). Dadas las restricciones de capacidad de las mochilas y la restricción de que cada objeto debe ir en alguna mochila, minimizaremos la suma $m_1 + m_2 + \cdots + m_N$.

El problema Bin Packing

Las restricciones de capacidad de las mochilas son fáciles de modelar: para una mochila j dada, la suma de los volúmenes de los objetos i por la variable $x_{i,j}$ respectiva debe ser $\leq C$. La restricción de que cada objeto debe ir en alguna mochila también es fácil de modelar: para un objeto i dado, la suma de las variables $x_{i,j}$ debe ser 1. Las variables m_j sólo son un poco más difíciles de modelar: m_j debe ser 1 si cualquier variable $x_{i,j}$ es 1.

El problema Bin Packing

Ejercicio: para el problema Bin Packing, resuelva la instancia en la que los volúmenes de los seis objetos a guardar son $(11, 10, 8, 6, 3, 2)$ y las mochilas tienen capacidad 20.

El problema de acoplamiento bipartito de costo mínimo

Este problema consiste en construir la mayor cantidad de parejas sin elementos en común, donde el primer elemento de cada pareja debe pertenecer al conjunto C_1 y el segundo elemento debe pertenecer al conjunto C_2, con C_1 y C_2 disjuntos. Al mismo tiempo, se busca minimizar la suma de los costos de las parejas elegidas. El costo de cada posible pareja puede ser elegido de manera arbitraria.

El problema de acoplamiento bipartito de costo mínimo

Este problema puede visualizarse como la asignación de tareas a trabajadores, donde cada tarea sólo puede estar a cargo de un trabajador y cada trabajador sólo puede atender una tarea. Se busca completar la mayor cantidad de tareas a un costo total mínimo. El costo de asignar una tarea a un trabajador u otro puede no ser el mismo y algunos trabajadores no pueden realizar ciertas tareas.

El problema de acoplamiento bipartito de costo mínimo

Nosotros modelaremos el problema de la siguiente manera. La familia de variables binarias $x_{i,j}$ indicarán si la tarea i se asignará al trabajador j. Si hay menos trabajadores que tareas, pueden crearse trabajadores ficticios que puedan atender cualquier tarea pero a un costo muy alto. De manera similar, si el trabajador j no puede atender la tarea i entonces se puede asignar un costo muy alto a la pareja (i, j). Para cada tarea i, la suma de las variables $x_{i,j}$ debe ser 1 mientras que para cada trabajador j, la suma de las variables $x_{i,j}$ debe ser ≤ 1. Se busca minimizar el producto de los costos de cada pareja por la variable $x_{i,j}$ correspondiente.

El problema de acoplamiento bipartito de costo mínimo

Ejercicio: para el problema de acoplamiento bipartito de costo mínimo, resuelva la siguiente instancia en la que hay cuatro tareas, cuatro trabajadores y los costos $C(i,j)$ de las parejas son:

- $C(1,1) = 2$
- $C(2,1) = 5$
- $C(2,2) = 4$
- $C(2,3) = 1$
- $C(3,2) = 7$
- $C(3,4) = 8$
- $C(4,1) = 5$.

Universidad Autónoma Metropolitana
Casa abierta al tiempo Azcapotzalco

El problema del agente viajero

Este problema consiste en encontrar una ruta que visite los N vértices de una gráfica sin pasar por el mismo vértice más de una vez, regresando al vértice de origen y minimizando el costo de la ruta usada. El costo de la ruta es la suma de los costos de las aristas usados durante el recorrido. Nosotros modelaremos el problema de la siguiente manera. La familia de variables binarias $x_{i,j}$ indicarán si el vértice j se visita inmediatamente después del vértice i. Para cada vértice i, la suma de las variables $x_{i,j}$ debe ser 1 mientras que lo mismo debe ocurrir para cada vértice j.

El problema del agente viajero

Para evitar que existan ciclos independientes, incluiremos una restricción de potencial: elegimos un vértice k cualquiera y al resto de los vértices les debemos asignar un potencial u_i entre 1 y $n-1$. La siguiente fórmula determina cómo se asignan los potenciales.

$$u_i - u_j + n x_{i,j} \leq n - 1 \qquad \forall i, j \geq 1, i, j \neq k \tag{5}$$

Cuando $x_{i,j} = 0$, la ecuación se cumple trivialmente. Cuando $x_{i,j} = 1$ entonces $u_i < u_j$ y todos los vértices consecutivos de un ciclo deberán cumplir las restricciones de potencial de forma creciente (por ejemplo, potenciales 1, 2, 3, ...) hasta llegar al vértice k, para el cual la restricción de potencial ya no aplica. Si cualquier ciclo no pasa por el vértice k, entonces la cadena creciente estricta llega a una contradicción. Todo ciclo entonces debe pasar por el vértice k, por lo que sólo existe un ciclo.

El problema del agente viajero

Ejercicio: para el problema del agente viajero, resuelva la siguiente instancia en la que hay cuatro ciudades con coordenadas en el plano.

- $P_1 = (1, 2)$
- $P_2 = (2, 2)$
- $P_3 = (3, 3)$
- $P_4 = (4, 3)$

El problema del agente viajero con cuello de botella

Este problema consiste es una variante del anterior y consiste en minimizar el costo de la arista más costosa del ciclo, en lugar de minimizar la suma de de los costos de las aristas del ciclo. Para modelar este problema hay que usar un truco, ya que no podemos pedir explícitamente lo que deseamos. Introduciremos una variable adicional z y minimizaremos dicha variable.

El problema del agente viajero con cuello de botella

La familia de restricciones que agregaremos son de la forma $z \geq C_{i,j}x_{i,j}$ donde $C_{i,j}$ es el costo de la arista entre los puntos P_i y P_j. Un valor suficientemente grande de z volverá factible el sistema, pero el optimizador intentará atrapar a z en un valor muy bajo.

Ejercicio: para el problema del agente viajero con cuello de botella, resuelva la instancia anterior.

Expresiones lógicas binarias

Cuando se está trabajando con variables binarias, es común querer expresar el *AND*, el *OR* y el *NOT*. Estas son algunas maneras de reescribir estos operadores en programación lineal.

- $y = x_1 \wedge x_2 \wedge \cdots \wedge x_n$: $y \leq x_1, y \leq x_2, \ldots, y \leq x_n$, $y \geq x_1 + x_2 + \cdots + x_n - (n-1)$.
- $y = x_1 \vee x_2 \vee \cdots \vee x_n$: $y \geq x_1, y \geq x_2, \ldots, y \geq x_n$, $y \leq x_1 + x_2 + \cdots + x_n$.
- $y = \neg x_1$: $y = 1 - x_1$.

Usando GLPSOL

La interfaz gráfica GUSEK es fácil de usar, pero no permite indicar fácilmente todos los parámetros configurables de GLPSOL. Estos parámetros pueden ser bastante útiles. Por ejemplo, es posible indicar qué algoritmo queremos usar durante la resolución de los programas lineales (continuos) así como qué cortes o heurísticas usar durante la resolución de un programa entero. También podemos especificar el nombre del archivo de salida que contendrá el resultado de la optimización, entre otros.

Usando GLPSOL

La manera más sencilla de entrar a la línea de comandos en Windows (aunque no la más intuitiva) es con la combinación de tecla "Windows" + *R*. En caso de que el sistema operativo pregunte qué aplicación deseamos ejecutar, debemos teclear *cmd.exe*. Sin realizar más configuraciones, debemos posicionarnos en la carpeta donde está *glpsol.exe*, por lo que los siguientes comandos pueden resultar útiles:

- *dir*: Muestra el contenido del directorio actual.
- *cd nombre*: Cambia al directorio hijo *nombre*.
- *cd ..*: Cambia al directorio padre.

Parámetros de GLPSOL

La lista de parámetros de GLPSOL pueden consultarse con *glpsol.exe --help* y en `https://en.wikibooks.org/wiki/GLPK/Using_GLPSOL`. Ejemplos de algunos parámetros generales son:

- *--lp nombre*: Resuelve el modelo LP del archivo *nombre*.
- *--log nombre*: Escribe la bitácora de ejecución en el archivo *nombre*.
- *--output nombre*: Escribe el resultado en el archivo *nombre*.
- *--tmlim n*: Limita el tiempo de optimización a *n* segundos.

Parámetros de GLPSOL

Algunos parámetros que sirven para controlar aspectos de la resolución de programas lineales (continuos) son:

- --*primal*: Indica que el cálculo debe realizarse con simplex primal.
- --*dual*: Indica que el cálculo debe realizarse con simplex dual.
- --*nopresol*: Deshabilita el presolucionador continuo.
- --*exact*: Indica que el cálculo debe realizarse con aritmética exacta.
- --*interior*: Indica que el cálculo debe realizarse con métodos de punto interior.

Parámetros de GLPSOL

Algunos parámetros que sirven para controlar aspectos de la resolución de programas lineales enteros son:

- --*nomip*: Indica que se debe ignorar la integralidad de las variables.
- --*nointopt*: Deshabilita el presolucionador entero.
- --*gomory*: Habilita los cortes de Gomory.
- --*mir*: Habilita los cortes MIR.
- --*cover*: Habilita los cortes de cubrimiento.
- --*clique*: Habilita los cortes de clan (de gráfica).
- --*cuts*: Habilita todos los cortes.
- --*mipgap n*: Indica que la brecha entre la mejor cota y la mejor solución debe ser al menos n para considerar la solución como óptima.

Programación cuadrática

Varios solucionadores como CPLEX y Gurobi (aunque no GLPK) tienen soporte para resolver modelos con funciones objetivo y restricciones cuadráticas siempre y cuando el espacio de búsqueda sea convexo. Hay que notar que cuando la programación cuadrática involucra productos de variables binarias, entonces el programa puede reescribirse como un programa lineal entero, ya que el producto de variables binarias corresponde con su *AND* (si alguna de las dos es cero, entonces el *AND* es cero tal como en la multiplicación).

Programación cuadrática

El formato LP ha sido extendido en los solucionadores anteriores para poder describir expresiones cuadráticas. Las expresiones cuadráticas deben aparecer después de los términos lineales y deben escribirse dentro de corchetes [. . .]. Además de eso:

- El producto de dos variables debe denotarse con el operador binario *.
- El producto de una variable por sí misma puede escribirse con ^2.
- Cuando la función objetivo tiene una subexpresión cuadrática, la misma debe terminar con un factor /2 después de los corchetes.

Programación cuadrática

Un ejemplo de programa cuadrático es

$$\begin{gathered} \text{maximize} \\ x + [\ 4\ y\ \hat{}\ 2\ ]\ /\ 2 \\ \text{subject to} \\ [\ 5\ x\ *\ y\ ] \leq 1 \end{gathered} \tag{6}$$

La referencia del formato LP extendido se encuentra en `http://www.gurobi.com/documentation/7.5/refman/lp_format.html`.

Enlaces de interés

- Open Research Challenge 2015 (Discrete Optimization): `https://openresearchchallenge.org/discreteOptimization/ChairofEconomics/The+Challenge`.

Gracias por su atención