GNU MathProg (AMPL)

Un lenguaje de modelado algebraico para programación lineal y entera

Pablo Yapura
ypf@agro.unlp.edu.ar

Facultad de Ciencias Agrarias y Forestales
Universidad Nacional de La Plata

Introducción a la Investigación de Operaciones
Julio del 2020

Unidad didáctica 2: Programación lineal

- Alcance: en esta unidad didáctica se introducirá el uso de modelos matemáticos como mecanismo de representación y solución para varios tipos de problemas de decisión. Por la importancia central que se le asigna en la asignatura, el problema de la programación lineal será abordado con detenimiento.
- Contenidos: (...). Formulación y resolución del problema con el lenguaje de modelado algebraico MathProg y GLPK. Problemas prototípicos de programación lineal (e.g. mezcla de productos, de la dieta, planificación de horarios, producción e inventario, cadena de abastecimiento, cartera de inversión, análisis de la envolvente de datos).

Hoja de ruta

Dos opciones muy populares

- En las planillas de cálculo el problema se formula con las herramientas estándares de la propia planilla de cálculo y para resolverlo se invoca un *solver*. Se pueden resolver problemas de tamaño moderado a grande. Las planillas son muy conocidas y proveen gran versatilidad para modelar.
- Una opción más potente consiste en formular el problema en un lenguaje que emula la notación algebraica y también encomendarle la solución a un *solver*. Se pueden resolver problemas grandes a muy grandes. Los lenguajes son relativamente fáciles de aprender y también proveen cierta versatilidad para modelar.

Solver y traducciones

- Se puede describir al *solver* como una pieza de *software* (programa o librería) que implementa algoritmos que resuelven problemas de programación (optimización) matemática.
- En ambos casos, otra pieza de *software* se encarga de las traducciones del problema formulado, ya sea en el lenguaje de las planillas de cálculo como en el de modelado algebraico, al formato que el *solver* entiende.
- Programas para resolver problemas formulados con lenguajes de modelado algebraicos hay de todo tipo, desde comerciales hasta libres y para todos los sistemas operativos. También se pueden resolver en sitios web dedicados.

La solución del *software* libre

- Una herramienta que combina una lenguaje algebraico muy potente con un *solver* capaz de resolver problemas a gran escala, que se distribuye con licencia de *software* libre, y que se ha compilado para correr tanto en Windows como en Linux es GLPK (GNU Linear Programming Kit).
- El lenguaje se llama GNU MathProg y se ha diseñado para describir modelos lineales de programación matemática. Es un subconjunto del lenguaje AMPL y su implementación en GLPK está basada principalmente en la publicación de *Fourer R, Gay DM & BW Kernighan, A Modeling Language for Mathematical Programming, Management Science 36 (1990), pp. 519-554.*

Instalación y uso de GLPK

- En el sitio oficial de GLPK está disponible el código fuente que puede ser compilado para su uso. Para ahorrarse este paso, es conveniente instalar directamente los ejecutables compilados.
- Para Windows, la recomendación es usar las compilaciones del proyecto GLPK for Windows. Para Linux, las principales distribuciones incluyen la posibilidad de instalar un paquete con el ejecutable compilado. En cualquiera de los dos sistemas operativos, el *solver* se invoca mediante la linea de comandos (para que se imprima la ayuda, en el *prompt* tipear `glpsol -h`).

La solución más sencilla

- En Windows, otra posibilidad es instalar un Entorno de Desarrollo Integrado (IDE). GUSEK (GLPK Under Scite Extended Kit) provee una interfaz gráfica para escribir los problemas en el lenguaje GNU MathProg con resaltado sintáctico y acceso simplificado a toda la funcionalidad programada en el *solver*. Naturalmente, a esa funcionalidad se accede mediante menúes.
- Junto con la instalación del entorno se instala el propio *solver* ya compilado de GLPK.
- En el sitio de GUSEK se puede acceder a la documentación y descargar el instalador.

Hoja de ruta

El problema expresado en lenguaje natural

- En una carpintería industrial, en la que se fabrican muebles de madera, se desea determinar el tiempo que se le asignará en la línea de producción a cada uno de los diferentes productos que se comercializan.
- La línea de producción está compuesta por varias máquinas, trabajando en serie y en paralelo, en las que se elaboran todas las piezas que componen los muebles a partir de tablas.
- Para simplificar, asumiremos que se consumen tablas de una única especie, escuadría y calidad; y que se producen las piezas para dos tipos de biblioteca denominadas Clásica (más cara) y Moderna (más económica).

Los datos del problema

- La eficiencia de la línea de producción es diferente para cada producto (bibliotecas por hora):
  - Clásica 16
  - Moderna 25
- Las bibliotecas retornan diferentes contribuciones al beneficio económico (pesos por biblioteca):
  - Clásica 310
  - Moderna 230
- Y en base a los pedidos, las expectativas de venta son (producción máxima de bibliotecas):
  - Clásica 375
  - Moderna 500

El razonamiento para la formulación

- Si la línea de procesamiento funciona durante 40 horas semanales (e.g. 8 horas diarias en un solo turno), la cuestión a resolver es ¿cuántas bibliotecas clásicas y cuántas bibliotecas modernas se deben producir para obtener el máximo beneficio económico?
- Si se definen X_C y X_M como las cantidades de bibliotecas de la serie clásica y moderna, respectivamente, a producir en una semana, entonces se puede formular una ecuación para la contribución al beneficio económico que se desea maximizar: `(beneficio por biblioteca Clásica)*`X_C `+ (beneficio por biblioteca Moderna)*`X_M, o sea:

$$310 * X_C + 230 * X_M$$

El razonamiento para la formulación (continuación)

- Con idéntico razonamiento, el tiempo semanal disponible en la línea de procesamiento se puede escribir como: `(tiempo de producción de una biblioteca Clásica)*`X_C `+ (tiempo de producción de una biblioteca Moderna)*`X_M, es decir:

$$(1/16) * X_C + (1/25) * X_M \leqslant 40$$

- Finalmente, los límites de la producción se pueden escribir como:

$$0 \leqslant X_C \leqslant 375$$
$$0 \leqslant X_M \leqslant 500$$

Hoja de ruta

El primer modelo matemático

$$
\begin{array}{ll}
\text{Maximizar} & z = 310\,X_C + 230\,X_M \\
\text{Sujeto a} & (1/16)\,X_C + (1/25)\,X_M \leqslant 40 \\
& 0 \leqslant X_C \leqslant 375 \\
& 0 \leqslant X_M \leqslant 500
\end{array}
$$

- En el que z representa la contribución al beneficio económico y todas las demás variables y constantes ya fueron definidas.
- Coloquialmente, el problema puede leerse como:

  Encontrar los valores de X_C y X_M que hagan máximo el valor de z y que, simultáneamente, cumplan todas las restricciones (inecuaciones).

Un modelo algebraico generalizado

Dados P: un conjunto de productos (bibliotecas)
a_j: cantidad del producto j procesado por hora, $\forall j \in P$
b: tiempo (horas) disponible en la línea de procesamiento
c_j: contribución al beneficio (pesos) por unidad del producto j, para cada $j \in P$
u_j: producción máxima del producto j, para cada $j \in P$

Se define X_j: cantidad del producto j a producir, para cada $j \in P$

Maximizar $$z = \sum_{j \in P} c_j X_j$$

Sujeto a $$\sum_{j \in P} (1/a_j) X_j \leqslant b$$

$$0 \leqslant X_j \leqslant u_j \qquad (\forall j \in P)$$

Hoja de ruta

La primera codificación para GNU MathProg

El primer modelo matemático, transcripto verborrágicamente a GNU MathProg, se codifica en un archivo de texto que se puede resolver invocando al solver glpsol:

```
### 1_produccion.mod ###
var X_C;
var X_M;
maximize Beneficio: 310 * X_C + 230 * X_M;
subject to Tiempo: (1/16) * X_C + (1/25) * X_M <= 40;
subject to Clasica_lim: 0 <= X_C <= 375;
subject to Moderna_lim: 0 <= X_M <= 500;
end;
```

Componentes fundamentales del modelo algebraico

- Objetos elementales de la codificación del lenguaje algebraico GNU MathProg (AMPL)
  - Conjuntos, como el de los productos
  - Parámetros, como la eficiencia de la línea de producción y las contribuciones al beneficio económico
  - Variables, que constituyen las incógnitas a determinar
  - un Objetivo, que se debe maximizar o minimizar
  - Restricciones, que la solución debe verificar
- Objetos auxiliares de la codificación
  - Índices, para recorrer los miembros de los objetos elementales

La transcripción del modelo algebraico

Con los objetos del lenguaje y separando el modelo de los datos:

```
### 2_produccion.mod ###
set P;
param a {j in P};
param b;
param c {j in P};
param u {j in P};
var X {j in P};
maximize Beneficio: sum {j in P} c[j] * X[j];
subject to Tiempo: sum {j in P} (1/a[j]) * X[j] <= b;
subject to Limite {j in P}: 0 <= X[j] <= u[j];
end;
```

La codificación de los datos

Los datos se codifican con los mismos objetos usados en la descripción del modelo y juntos constituyen una instancia particular del problema. Se pueden codificar en el mismo archivo del modelo o por separado:

```
### 2_produccion.dat ###
data;
set P := clasica moderna;
param:              a        c        u        :=
        clasica  16       310      375
        moderna  25       230      500      ;
param b := 40;
end;
```

Un modelo mejorado, i.e. bien documentado

```
### Extracto de 3_produccion.mod ###
   set productos;
# diferentes modelos de biblioteca
   param efi_linea {productos} > 0;
# eficiencia de la linea de procesamiento
# [bibliotecas/hora]
   var prd_biblioteca {p in productos}
  >= 0, <= prd_maxima[p];
# produccion de cada modelo de bibliotecas [bibliotecas]
   maximize Beneficio: sum {p in productos}
   con_beneficio[p] * prd_biblioteca[p];
# Objetivo: maximizar la contribucion al beneficio y los
```

Los datos del modelo bien documentado

```
### 3_produccion.dat ###
data;
set productos := clasica moderna;
param:     efi_linea  con_beneficio  prd_maxima  :=
  clasica      16          310           375
  moderna      25          230           500      ;
param tmp_disponible := 40;
end;
```

¿Nuevos productos?

Dado que el modelo es general, esta situación es apenas una instancia diferente y sólo se necesita codificar un nuevo archivo de datos con el producto agregado:

```
### 3_1_produccion.dat ###
data;
set productos := clasica moderna rustica;
param:     efi_linea  con_beneficio  prd_maxima  :=
   clasica      16          310          375
   moderna      25          230          500
   rustica      21          285          475      ;
param tmp_disponible := 40;
end;
```

¿Variables de decisión doblemente acotadas?

```
### Extracto de 4_produccion.mod ###
   set productos;
# diferentes modelos de biblioteca
   param prd_minima {productos} >= 0;
# produccion minima de cada modelo [bibliotecas]
   param prd_maxima {productos} >= 0;
# produccion maxima de cada modelo [bibliotecas]
   var prd_biblioteca {p in productos}
  >= prd_minima[p], <= prd_maxima[p];
# produccion de cada modelo de bibliotecas [bibliotecas]
   maximize Beneficio: sum {p in productos}
   con_beneficio[p] * prd_biblioteca[p];
```

¿Procesos en la línea de producción?

Para representar nuevas restricciones a los recursos, es necesario revisar el modelo:

```
### Extracto de 5_produccion.mod ###
set procesos;
  # diferentes procesos de produccion
param efi_linea {productos,procesos} > 0;
  # eficiencia de la linea de procesamiento
  # [bibliotecas/hora]
param tmp_disponible {procesos} >= 0;
subject to Tiempo {q in procesos}: sum {p in productos}
  (1/efi_linea[p,q]) * prd_biblioteca[p]
  <= tmp_disponible[q];
```

Procesos en la línea de producción

Y también los datos:

```
### Extracto de 5_produccion.dat ###
data;
set productos :=  clasica moderna rustica;
set procesos := corte ensamble;
param efi_linea:  corte ensamble  :=
        clasica     16       25
param:     con_beneficio prd_minima  prd_maxima  :=
  rustica      285            250          475      ;
param tmp_disponible := corte 35  ensamble  40;
end;
```

Un modelo algebraico más generalizado

Dados P, Q: un conjunto de productos y otro de procesos

a_{jq}: cantidad del producto j que se elabora en el proceso q por hora, $\forall j \in P$ y $\forall q \in Q$

b_q: tiempo del proceso q disponible en la línea, $\forall q \in Q$

c_j: contribución al beneficio por unidad del producto j, $\forall j \in P$

l_j, u_j: producción mínima y máxima del producto j, $\forall j \in P$

Se define X_j: cantidad del producto j a producir, $\forall j \in P$

Maximizar $$z = \sum_{j \in P} c_j X_j$$

Sujeto a $$\sum_{j \in P} (1/a_{jq}) X_j \leqslant b_q \qquad (\forall q \in Q)$$

$$l_j \leqslant X_j \leqslant u_j \qquad (\forall j \in P)$$

Hoja de ruta

Especificación del problema

- En la formulación de alimentos balanceados para aves se usan estándares específicos que establecen diferentes requerimientos nutricionales.
- Para su producción se emplean varias materias primas, incluyendo granos, harinas, sales y un concentrado comercial, entre otros.
- Es posible presentar los requerimientos junto con los contenidos nutricionales de las diferentes materias primas y su valor comercial en forma de tablas.
- ¿Cómo deben mezclarse las materias primas para minimizar el costo de producir 1 kg de balanceado y cumplir los requerimientos estandarizados?

La tabla de requerimientos y contenidos

Materia prima	Precio $.kg^{-1}$	Proteína cruda	Fibra bruta	Lisina	Calcio	Fósforo	Sal
		.....	.....	..... $g.kg^{-1}$	.....	.....	.....
Granos de maíz	6,8	78,0	16,0	2,3	0,7	0,30	
Granos de trigo	7,2	114,0	22,0	3,4	0,6	0,34	
Salvado de trigo	2,3	142,0	95,0	6,0	0,3	10,0	
Afrechillo de arroz	2,2	117,0	72,0	6,5	1,0	13,0	
Concentrado comercial	230,0	800,0		546,0		1,10	
Harina de huesos	5,6				300,0	140,0	
Carbonato de calcio	11,2				400,0		
Sal	4,2						1000,0
Requerimiento		135-145	30-50	$\geqslant$ 5,6	23-40	4,6-6,5	3,7–6,0

La versión verborrágica

```
### Extracto de balanceado_aves_costo_minimo.mod ###
var x_gma >= 0; var x_gtr >= 0; var x_svd >= 0;
var x_afr >= 0; var x_ccm >= 0; var x_hhu >= 0;
minimize  Costo: 6.8 * x_gma + 7.2 * x_gtr +
 2.3 * x_svd + 2.2  * x_afr + 230.0  * x_ccm +
 5.6 * x_hhu + 11.2 * x_cca + 4.2 * x_sal;
subject to Fosforo: 0.3 * x_gma + 0.4 * x_gtr +
 10.0 * x_svd + 13.0 * x_afr + 1.1 * x_ccm +
 140 * x_hhu >= 4.6;
end;
```

El modelo algebraico

```
### Extracto de dieta_costo_minimo.mod ###
set mat_primas;
set req_nutritivos;
param mezcla_base >= 0;
param contenido{mat_primas, req_nutritivos} >= 0;
minimize Costo_total:
  sum {i in mat_primas} costo[i] * peso[i];
data;
set mat_primas := grn_maiz, grn_trigo, svd_trigo,
afr_arroz, con_comercial, har_huesos, car_calcio, sal;
end;
```

Hoja de ruta

Mezclas, turnos, equilibrio económico

Con el mismo modelo simbólico y con otra semántica, se pueden analizar otros problemas:

- Supongamos que a las materias primas las llamamos insumos y a los requerimientos los denominamos productos. Entonces, de cada insumo `i` se debe decidir la cantidad `X[i]`, entre `min_insumo[i]` y `max_insumo[i]` que se empleará.
- Como resultado, se incurrirá en un costo igual a `costo[i]*X[i]` y se producirán `ins_prd[i,p]*X[i]` unidades de cada producto `p`.
- El objetivo es encontrar la combinación de insumos de costo mínimo que permita producir una cantidad, para cada producto `p`, entre `min_prd[p]` y `max_prd[p]`.

Otras aplicaciones: mezclas

- En un tipo común de aplicaciones de este modelo, los insumos son las materias primas que se mezclan y los productos son las cualidades de la mezcla resultante.
- Las materias primas pueden ser derivados del petróleo crudo mezclado para producir naftas, carbones vegetales y minerales para alimentar un alto horno o fibras cortas y largas de diferentes especies para producir papel.
- Las cualidades pueden ser contenidos o concentraciones, o incluso propiedades complejas como el octanaje, dureza y la resistencia al rasgado (advertencia: ¿proporcionalidad?).

Otras aplicaciones: equilibrio económico

- En otra aplicación muy conocida los insumos son las actividades de producción de cierto sector de la economía y los productos son sus respectivos productos.
- Las parámetros `min_ins` y `max_ins` son límites para las actividades mientras que los `min_prd` y `max_prd` son regulados por la demanda.
- Entonces, el objetivo es determinar los niveles de las actividades que satisfacen la demanda al mínimo costo.

Otras aplicaciones: planificación de horarios

- En una aplicación más abstracta, los insumos son turnos y los productos son las horas trabajadas en ciertos días del mes.
- Para un horario particular `i`, `ins_prd[i,p]` será el número de horas que el trabajador que cumple el turno `i` trabajará el día `p`, el `costo[i]` será el salario mensual del trabajador que cumple el turno `i` y `X[i]` será el número de trabajadores asignados al turno `i`.
- El objetivo es minimizar el costo total de pagar los salarios mensuales mientras que las restricciones establecerán que para cada día `p`, el número total de trabajadores asignados debe estar entre `min_prd[p]` y `max_prd[p]`. Naturalmente las unidades de tiempo pueden ser otras (advertencia: ¿continuidad?).

Otras aplicaciones: corte del stock

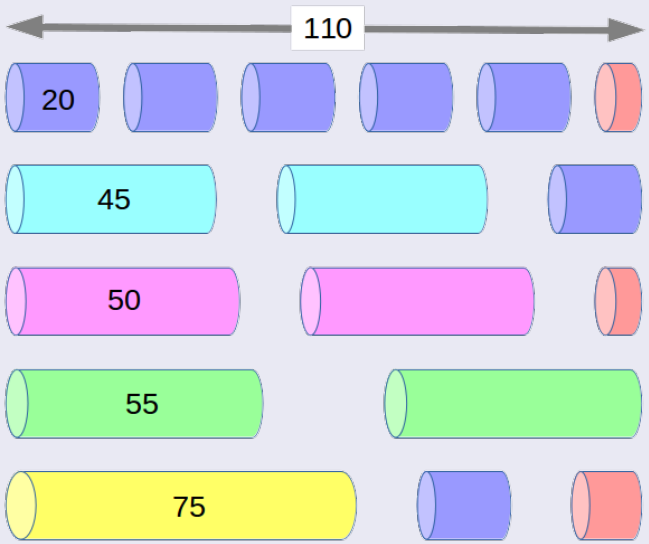
- En otra aplicación igualmente abstracta, el problema más general es conocido como corte del stock (existencias) y como tal se presenta en muchas industrias y en la logística. En la industria del papel es conocido como el problema del (re)corte de rollos y busca minimizar los desperdicios en el proceso de producir las cantidades requeridas de rollos de longitud estandarizada a partir de la subdivisión de rollos más grandes.

Otras aplicaciones: corte del stock

- Por ejemplo: en una fábrica de papel el producto es un rollo estándar de 110 pulgadas de longitud, del cual se deben cortar rollos de menor longitud para su comercialización en base a pedidos específicos, e.g. rollos de 20, 45, 50, 55 y 75 pulgadas de longitud.
- Entonces, dado un conjunto de pedidos concretos y consolidados, es decir los números de rollos que se requieren detallados por longitudes, se necesita determinar el número mínimo de rollos de 110 pulgadas que se deben fraccionar y los correspondientes patrones de corte a emplear para cumplir los pedidos.

Otras aplicaciones: corte del stock

- Un patrón de corte consiste en extraer un cierto número de rollos más pequeños sin exceder la longitud del rollo grande. Entonces 2 rollos de 50 pulgadas constituyen un patrón aceptable (con un desperdicio de 10 pulgadas), al igual que dos rollos de 55 pulgadas (sin desperdicios), entre otros.



Otras aplicaciones: corte del stock

- En este problema, los patrones de corte desempeñan el rol de las materias primas o insumos y las restricciones establecen los límites inferiores en los pedidos por ancho de corte, desempeñando el rol de los productos (como los requerimientos nutricionales).

```
### Extracto de corte_rollo_papel.mod ###
set longitudes;
param pedidos{longitudes} >= 0;
param num_patrones integer >= 0;
set patrones := 1..num_patrones;
param num_rollos{patrones,longitudes} integer >= 0;
var cortar{patrones} integer >= 0;
```

Hoja de ruta

Make Titles Informative.

Make Titles Informative.

Make Titles Informative.

Hoja de ruta

Make Titles Informative.

Make Titles Informative.

Make Titles Informative.

Resumen

- The first main message of your talk in one or two lines.
- The second main message of your talk in one or two lines.
- Perhaps a third message, but not more than that.

- Outlook
  - Something you haven't solved.
  - Something else you haven't solved.

For Further Reading I


A. Author.
Handbook of Everything.
Some Press, 1990.


S. Someone.
On this and that.
Journal of This and That, 2(1):50–100, 2000.