Introducción a GLPK en *Optimización de Problemas de Producción*

Docentes:
Fernando Islas – Joaquín Correa – Carlos Testuri

Departamento de Investigación Operativa. Instituto de Computación.
Facultad de Ingeniería. Universidad de la República

2024

Sistema informático para la resolución de problemas de programación lineal y programación lineal entera mixta, integrado por

- Lenguaje de modelado algebraico (*MathProg*)
- Solver (*glpsol*) utiliza diferentes algoritmos: métodos simplex primal y dual, punto interior primal-dual y branch and cut.

- Los problemas se escriben en un archivo de texto plano utilizando los elementos del lenguaje
- **Descripción del modelo**: Formado por una secuencia de **sentencias**
- **Datos del modelo**: Formado por una secuencia de **bloques de datos**

Descripción del modelo: Sentencias

- Las sentencias son la la unidad básica del lenguaje
- Toda sentencia finaliza con un punto y coma (**;**)
- **Sentencias Declarativas**: Sirven para definir **objetos del modelo**
- **Sentencias Funcionales**: Para realizar acciones específicas

Descripción del modelo: Objetos

- Es definido mediante un nombre simbólico que lo identifica
- Los objetos pueden ser: escalares o n-dimensionados ($n \geq 1$)
- **Objetos Escalares**: Los referenciamos por medio del nombre; por ejemplo t
- **Objetos Vectoriales**(1-dimensión): Los referenciamos por medio del nombre y un subíndice entre corchetes; por ejemplo: *nutriente[i]*
- **Objetos *n*-Dimensionados**: Los referenciamos por medio del nombre y n subíndices entre corchetes; por ejemplo para n=2 (matriz), *contenido[i,j]*

- Los **conjuntos** definen el dominio de los índices utilizados en el modelo
- Los **parámetros** representan datos de la realidad
- Las **variables** representan magnitudes que están bajo el control de quién toma las decisiones

Definición y Sintáxis	Ejemplo
Conjuntos	
set *nombre*;	**set** *A*;
Parámetros	
param *nombre dominio*;	**param** *precio{i in A}*;
Variables de Decisión	
var *nombre dominio expresión*;	**var** *x{i in A} >= 0*;

- Las **restricciones** representan exigencias que debe cumplir la solución que buscamos

Definición y Sintáxis
Restricciones
s.t. *nombre dominio* : *expresión* $=$ *expresión*;
s.t. *nombre dominio* : *expresión* $\leq$ *expresión*;
s.t. *nombre dominio* : *expresión* $\geq$ *expresión*;

Ejemplo
s.t. *demanda{j in N}* :
*sum{i in A} x[i] * contenido[i,j] >= requisito[j]*;

- La **función objetivo** representa la expresión que queremos optimizar

Definición y Sintáxis	Ejemplo
Función Objetivo	
maximize *nombre* :	
expresión;	
minimize *nombre* :	**minimize** *costo* :
expresión;	*sum{i in A} precio[i] * x[i]*;

Descripción del modelo: Comentarios

- Es una construcción muy útil al momento de documentar nuestros modelos
- Sirven para hacer anotaciones legibles
- Estas anotaciones son ignoradas por el intérprete que procesa el modelo
- **Comentarios de línea**: El carácter numeral (#) comenta todo lo que se encuentre a su derecha hasta el final de la línea
- **Comentarios de bloque**: La secuencia barra asterisco (/*) comenta todos los caracteres que se encuentren hasta la secuencia asterisco barra (*/)

Datos del modelo

- La primera forma de proporcionarle datos al modelo es por medio del **operador de asignación :=** (dos puntos igual) en la declaración (definición) del objeto. Por ejemplo:

  set A **:=** Leche Carne Arroz Papa Tomate;

- La segunda es por medio de bloques de datos en la **sección de datos del modelo**

Datos del modelo: Sección de datos

- La sección de datos del modelo puede estar definida en el mismo archivo que la descripción del modelo, luego de la sentencia **data**;
- En esta sección las expresiones no están permitidas
- Por medio de **bloques de datos** se asignan valores a los *Conjuntos* y a los *Parámetros*

```
sentencia;
      . . .
sentencia;
data;
bloque de datos;
      . . .
bloque de datos;
end;
```

Datos del modelo: Sección de datos (2)

- Aunque por lo general vamos a querer tener los datos del modelo en un archivo diferente, de modo de poder instanciar un mismo modelo con distintos juegos de datos

Archivo del modelo	Archivo de datos

sentencia`;`
sentencia`;`
`. . .`
sentencia`;`
`end;`
	`data;`
	bloque de datos`;`
	bloque de datos`;`
	`. . .`
	bloque de datos`;`
	`end;`

Datos del modelo: Bloques de datos

Bloque de datos	Ejemplo
Conjuntos	
set *nombre* := $t_1, \ldots, t_n$;	**set** *A* := *Leche Carne Arroz Papa Tomate*;
Parámetros (Escalar)	
param *nombre* := v;	**param** *T* := *4*;
Parámetros (Vector)	
param *nombre* := t_1, v_1,	**param** *precio* := Leche 24
t_2, v_2,	Carne 190
$\ldots$	Arroz 32
t_{n-1}, v_{n-1},	Papa 25
t_n, v_n;	Tomate 50;

Datos del modelo: Bloques de datos (2))

Bloque de datos

Parámetros (Matriz)

$$
\begin{array}{llccccl}
\textbf{param}\ \textit{nombre} & : & c_1 & c_2 & \dots & c_n & := \\
 & r_1 & a_{11} & a_{12} & \dots & a_{1n} & \\
 & r_2 & a_{21} & a_{22} & \dots & a_{2n} & \\
 & & . & . & . & . & \\
 & r_m & a_{m1} & a_{m2} & \dots & a_{mn}; &
\end{array}
$$

Ejemplo

Parámetros (Matriz)

param *contenido* :

:	Gr	Pr	Ca	:=
Leche	20	34	49	
Carne	250	180	0	
Arroz	2,5	67	780	
Papa	1	21	170	
Tomate	2	11	47;	

Formulación básica (MathProg)

$$
\begin{array}{ll}
\max & \sum_{j \in J} c_j x_j \\
\text{s.a} & \sum_{j \in J} A_{ij} x_j \leq b_i, \forall i \in I \\
 & x_j \geq 0, \text{entero}, \forall j \in J.
\end{array}
$$

Formulación básica (MathProg)

$$
\begin{array}{ll}
\max & \sum_{j \in J} c_j x_j \\
\text{s.a} & \sum_{j \in J} A_{ij} x_j \leq b_i, \forall i \in I \\
 & x_j \geq 0, \text{entero}, \forall j \in J.
\end{array}
$$

```
maximize obj:
  sum{j in J} c[j] * x[j];

s.t. rest{i in I}:
  sum{j in J} A[i,j] * x[j] <= b[i];
```

Formulación básica (MathProg modelo completo)

Conjuntos

I

J

Parámetros

c_j

b_i

A_{ij}

Variable de decision

x_j

$$
\begin{array}{ll}
\max & \sum_{j \in J} c_j x_j \\
\text{s.a} & \sum_{j \in J} A_{ij} x_j \leq b_i, \forall i \in I \\
 & x_j \geq 0, \text{entero}, \forall j \in J.
\end{array}
$$

Formulación básica (MathProg modelo completo)

Conjuntos

I

J

Parámetros

c_j

b_i

A_{ij}

Variable de decision

x_j

$$
\begin{array}{ll}
\max & \sum_{j\in J} c_j x_j \\
\text{s.a} & \sum_{j\in J} A_{ij}x_j \leq b_i, \forall i \in I \\
 & x_j \geq 0, \text{entero}, \forall j \in J.
\end{array}
$$

```
set I;
set J;

param c{J};
param b{I};
param A{I,J};

var x{J} >= 0, integer;

maximize obj:
  sum{j in J} c[j] * x[j];

s.t. rest{i in I}:
  sum{j in J} A[i,j] * x[j]
    <= b[i];

end;
```

Formulación básica (modelo y datos)

basico.mod

```
set I;
set J;

param c{J};
param b{I};
param A{I,J};

var x{J} >= 0, integer;

maximize obj:
  sum{j in J} c[j] * x[j];

s.t. rest{i in I}:
  sum{j in J} A[i,j] * x[j] <= b[i];

end;
```

basico.dat

```
set I := 1, 2, 3;
set J := 1, 2, 3, 4;

param c := 1 11,
           2 22,
           3 33,
           4 44;

param b :=
  1 100, 2 200, 3 300;

param A :
     1   2   3   4 :=
 1   5   6   7   8
 2   9  10  11  12
 3  13  14  15  15 ;

end;
```

Ejemplo: Conjuntos indizados y parámetros de más de dos dimensiones

prod.mod

```
set P;          # productos
set A{P};       # areas de
    mercado
param h;        # n mero
    semanas
set T := 1..h; # semanas

# l mite de producto por
    semana
param market{p in P, A[p], T}
  >= 0;

# producto vendido
var Sell{
  p in P, a in A[p], t in T
} >= 0, <= market[p,a,t];

...
```

prod.dat

```
param h := 3;
set P := b c;
set A[b] := east north;
set A[c] := east west export;

param market :=
[b,*,*]:     1     2     3 :=
east      2000  2000  1500
north     4000  4000  2500

[c,*,*]:     1     2     3 :=
east      1000   800  1000
west      2000  1200  2000
export    1000   500   500  ;

...

end;
```

Problema de la dieta

El problema trata de la selección de alimentos que satisfacen requisitos nutricionales a costo mínimo. Dado un conjunto de alimentos, con su información nutricional y precio, el objetivo es seleccionar las cantidades de alimentos a adquirir de forma de minimizar el costo de la dieta, mientras se cumple con requisitos nutricionales. Estos requisitos se establecen como cotas mínimas de algunos componentes nutricionales.

Cuadro: Contenido de nutrientes y precios por alimento

Alimento	Grasas(g)	Proteínas(g)	Carbohidratos(g)	Precio($)
Leche (L)	20	34	49	24
Carne (Kg)	250	180	0	190
Arroz (Kg)	2.5	67	780	32
Papa (Kg)	1	21	170	25
Tomate (Kg)	2	11	47	50
Requisitos (/día)	60	120	280	

Problema de la dieta (Formulación)

El objetivo es determinar la cantidad diaria de cada alimento a adquirir mediante las variables x_L, x_C, x_A, x_P, x_T.
Para cada nutriente existe una restricción de cantidad mínima.

$$
\begin{cases}
min & 24x_L + 190x_C + 32x_A + 25x_P + 50x_T & \\
s.a. & 20x_L + 250x_C + 2{,}5x_A + x_P + 2x_T \geq 60, & \text{(Grasas)} \\
 & 34x_L + 180x_C + 67x_A + 21x_P + 11x_T \geq 120, & \text{(Proteinas)} \\
 & 49x_L + 780x_A + 170x_P + 47x_T \geq 280, & \text{(Carbohidratos)} \\
 & x_L, x_C, x_A, x_P, x_T \geq 0. &
\end{cases}
$$

Implementación del Problema de la dieta (Conjuntos y Parámetros)

```
# PROBLEMA DE LA DIETA
#

set A;
/* Alimentos */

set N;
/* Nutrientes */
```

```
param precio{i in A};
/* Precio por alimento */

param requisito{j in N};
/* Requisitos nutricionales diarios */

param contenido {i in A, j in N};
/* Contenido nutricional j por alimento i */
```

Implementación del Problema de la dieta (Variables, restricciones y objetivo)

```
var x{i in A} >= 0;
/* Cantidades a ser adquiridas */
```

```
minimize cost: sum{i in A} precio[i] * x[i];
/* Costo minimo */
```

```
s.t. demanda{j in N}: sum{i in A} x[i] * contenido[i,j] >=
    requisito[j];
/* Satisfaga los requisitos diarios para el nutriente i */
```

Implementación del Problema de la dieta (Bloques de datos)

```
data;

set A := Leche Carne Arroz Papa Tomate;

set N := Grasas Proteinas Carbohidratos;

param precio := Leche     24 /*($ por litro)*/
                Carne     190  /*($ por Kg)*/
                Arroz     32 /*($ por Kg)*/
                Papa      25 /*($ por Kg)*/
                Tomate    50;  /*($ por Kg)*/

param requisito := Grasas         60   /*(g diarios)*/
                   Proteinas       120  /*(g diarios)*/
                   Carbohidratos 280; /*(g diarios)*/
```

Implementación del Problema de la dieta (Bloques de datos) (2)

```
param contenido :   Grasas  Proteinas  Carbohidratos :=
#                   (g)       (g)          (g)
            Leche    20        34           49
            Carne   250       180            0
            Arroz   2.5        67          780
            Papa      1        21          170
            Tomate    2        11           47;
end;
```

Resolución

El programa `glpsol` es ejecutado desde la línea de comandos. Para obtener la consola de línea de comandos en MS-Windows hay que ejecutar el programa `cmd.exe`.
En `glpsol`, para indicarle al intérprete que el archivo de entrada corresponde a una descripción de un modelo, se utiliza la opción `model`. Por medio de la opción `output` se le indica el archivo en donde se almacenará la salida con los resultados. Las opciones son antecedidas por dos guiones o signo de menos.

```
C:\> glpsol --model dieta.mod --output dieta.sol
```

Interpretación de los resultados (Información sobre el problema)

```
Problem:    dieta
Rows:       4
Columns:    5
Non-zeros:  19
Status:     OPTIMAL
Objective:  cost = 80.3187251 (MINimum)
```

El solver ha encontrado un valor óptimo (mínimo en este caso) al problema *dieta* y vale \$ 80.3187251.

Interpretación de los resultados (Información sobre el estado de la función objetivo y las restricciones)

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 cost         B        80.3187
     2 demanda[Grasas]
                    NL            60            60                    0.414343
     3 demanda[Proteinas]
                    NL           120           120                    0.462151
     4 demanda[Carbohidratos]
                    B        368.988           280
```

Analizando la tabla vemos que el estado (columna `St`) de los requisitos nutricionales en Grasas esta acotado inferiormente (`NL`). Su valor marginal (valor dual o precio sombra) vale \$ 0.4143 por lo que si la restricción fuera relajada, la función objetivo mejoraría ese valor. Con los requisitos de Proteínas sucede algo similar. Sin embargo los requisitos nutricionales en Carbohidratos no están acotados (`St` es `B`) por lo que la restricción esta inactiva. Relajarla no mejorará el valor objetivo; su valor dual es cero.

Interpretación de los resultados (Información de la solución)

```
   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x[Leche]     B        2.96414             0
     2 x[Carne]     NL             0             0                     3.22709
     3 x[Arroz]     B       0.286853             0
     4 x[Papa]      NL             0             0                     14.8805
     5 x[Tomate]    NL             0             0                     44.0876
```

Vemos en la columna `Activity` la solución óptima del problema. Las que presentan estado `B` son las variables básicas. La solución es entonces Leche = 2.96414 L y Arroz 0.286853 kg. Deberíamos adquirir esas cantidades diariamente para minimizar los costos y cumplir con los requisitos nutricionales. La columna `Marginal` indica los costos reducidos de las variables no básicas, ante lo que se puede confirmar que se ha alcanzado el mínimo.

Se busca establecer un plan de distribución diario que minimice los costos en el traslado de un producto desde plantas hasta almacenes.

Se cuenta con dos plantas de productos en Seattle y San Diego, que producen 500 y 750 unidades por día, respectivamente. La distribución desde las plantas atiende tres almacenes ubicados en New York, Chicago y Topeka, que demandan cada uno 300 unidades diarias.

Se desea determinar el número de unidades diarias que se distribuirán desde las dos plantas a los tres almacenes, con el objetivo de minimizar el costo total de transporte.

Problema de distribución (2)

En la siguiente tabla se muestran los costos de transporte por unidad desde cada planta a cada almacén.

Plantas	Almacenes		
	New York	Chicago	Topeka
Seattle	2.5	1.7	1.8
San Diego	2.5	1.8	1.4

Problema de distribución: Conjuntos, decisiones y objetivo

Conjuntos:

P : Plantas

A : Almacenes

Decisiones:

$x_{i,j}$ Cantidad de unidades a transportar de la planta $i \in P$ al almacén $j \in A$

Objetivo:

$$\min \sum_{i \in P} \sum_{j \in A} transporte_{i,j} x_{i,j}$$

Problema de distribución: Restricciones

- Para cada planta, la cantidad de producto distribuida debe ser menor o igual a la cantidad producida.
$$\begin{aligned} x_{Seattle,NewYork} + x_{Seattle,Chicago} + x_{Seattle,Topeka} &\leq 500 \\ x_{SanDiego,NewYork} + x_{SanDiego,Chicago} + x_{SanDiego,Topeka} &\leq 750 \end{aligned}$$
- Se debe satisfacer la demanda de los almacenes.
$$\begin{aligned} x_{Seattle,NewYork} + x_{SanDiego,NewYork} &\geq 300 \\ x_{Seattle,Chicago} + x_{SanDiego,Chicago} &\geq 300 \\ x_{Seattle,Topeka} + x_{SanDiego,Topeka} &\geq 300 \end{aligned}$$
- No negatividad de las variables de decisión:
$x_{Seattle,NewYork} \geq 0$, $x_{Seattle,Chicago} \geq 0$, $x_{Seattle,Topeka} \geq 0$,
$x_{SanDiego,NewYork} \geq 0$, $x_{SanDiego,Chicago} \geq 0$, $x_{SanDiego,Topeka} \geq 0$

Problema de distribución: Formulación

$$\begin{cases} \min & \sum_{i\in\{Seattle,SanDiego\}} \sum_{j\in\{NewYork,Chicago,Topeka\}} transporte[i,j] \times x[i,j] \\ s.a. & \sum_{j\in\{New\ York,Chicago,Topeka\}} x_{Seattle,j} \leq 500 \\ & \sum_{j\in\{NewYork,Chicago,Topeka\}} x_{SanDiego,j} \leq 750 \\ & \sum_{i\in\{Seattle,SanDiego\}} x_{i,NewYork} \geq 300 \\ & \sum_{i\in\{Seattle,SanDiego\}} x_{i,Chicago} \geq 300 \\ & \sum_{i\in\{Seattle,SanDiego\}} x_{i,Topeka} \geq 300 \\ & x_{i,j} \geq 0\ con\ i \in \{Seattle, SanDiego\}\ y\ j \in \{NewYork, Chicago, Topeka\} \end{cases}$$

Implementación del Problema de distribución (Conjuntos y Parámetros)

```
# PROBLEMA DE TRANSPORTE
#

set P;
/* Plantas */

set A;
/* Almacenes */
```

```
param capacidad{i in P};
/* Capacidad de la planta i */

param demand{j in A};
/* Demanda del almacen j */

param transporte{i in P, j in A};
/* Costo de transporte unitario */
```

Implementación del Problema de distribución (Variables, restricciones y objetivo)

```
var x{i in P, j in A} >= 0;
/* Cantidades a ser distribuidas */
```

```
minimize cost: sum{i in P, j in A} transporte[i,j] * x[i,j];
/* Costos totales de transporte */
```

```
s.t. supply{i in P}: sum{j in A} x[i,j] <= capacidad[i];
/* Sujeto a la capacidad de la planta i */

s.t. demand{j in A}: sum{i in P} x[i,j] >= demanda[j];
/* Satisfaga la demanda del almacen j */
```

Implementación del Problema de distribución (Bloques de datos)

```
data;

set P := Seattle San-Diego;

set A := New-York Chicago Topeka;

param capacidad := Seattle      500
                   San-Diego    750;

param demand := New-York     300
                Chicago      300
                Topeka       300;

param transporte :      New-York    Chicago    Topeka :=
             Seattle      2.5          1.7        1.8
             San-Diego    2.5          1.8        1.4    ;

end;
```

Interpretación de los resultados
(Información sobre el estado de la función objetivo y las restricciones)

```
   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 cost         B           1680
     2 supply[Seattle]
                    NU           500                         500         < eps
     3 supply[San-Diego]
                    B            400                         750
     4 demand[New-York]
                    NL           300           300                         2.5
     5 demand[Chicago]
                    NL           300           300                         1.7
     6 demand[Topeka]
                    NL           300           300                         1.4
```

Si bien la restricción *supply[Seattle]* acota la función objetivo superiormente, el valor marginal esta muy próximo a 0 (ϵ, con $\epsilon \to 0$).
Por otro lado, las restricciones de almacén indican que si se pudiera elegir un almacén para reducir su demanda (relajar la restricción), nos convendría elegir el de New York, dado que cada unidad decrementaría los costos 2.5 puntos en el valor óptimo.

Interpretación de los resultados (Información de la solución)

```
   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x[Seattle,New-York]
                    B            200             0
     2 x[Seattle,Chicago]
                    B            300             0
     3 x[Seattle,Topeka]
                    NL             0             0                         0.4
     4 x[San-Diego,New-York]
                    B            100             0
     5 x[San-Diego,Chicago]
                    NL             0             0                         0.1
     6 x[San-Diego,Topeka]
                    B            300             0
```

Analizando la segunda tabla vemos que el programa de envío de menor costo es aquel que envía desde Seattle 200 unidades a New York, y 300 a Chicago; y desde San Diego 100 a New York y 300 a Topeka cumpliendo los límites de las plantas y las demandas de los almacenes. A su vez indica que enviar unidades desde Seattle a Topeka o de San Diego a Chicago aumenta el costo unitario en 0.4 y 0.1 unidades, respectivamente.

Problema de determinación de lotes no capacitado

El problema de determinación de lotes no capacitado se describe para un horizonte de n períodos. Para cada período, $t \in \{1, \ldots, n\}$, se debe determinar el nivel de producción de forma de satisfacer cierta demanda, $d_t \geq 0$, mientras se minimizan costos fijos de producción, f_t, costos unitarios de producción, p_t, y costos unitarios de depósito, h_t, para almacenar la producción remanente luego de atender la demanda en cada uno de los períodos.

Variables de Decisión

- x_t es la cantidad producida en el período t,
- s_t es el inventario al final del período t,
- $y_t = 1$ si se produce en el período t, $y_t = 0$ en otro caso.

Restricciones

- equilibrio de inventario según períodos:
$$\begin{aligned} s_{t-1} + x_t &= d_t + s_t, \quad t = 1, \ldots, n, \\ s_0 &= 0 \qquad\qquad\qquad\qquad , \end{aligned}$$
- activación de producción
Dado que no hay una cota de producción se asume un valor grande M para la activación de los costos fijos
$$x_t \leq My_t, \quad t = 1, \ldots, n,$$

Función Objetivo

- **minimizar** $\sum_{t=1}^{n} (p_t x_t + h_t s_t + f_t y_t)$

Problema de determinación de lotes no capacitada (Formulación)

$$\begin{cases} \min & \sum_{t=1}^{n}(p_t x_t + h_t s_t + f_t y_t) \\ \text{s.a.} & s_{t-1} + x_t = d_t + s_t,\ t = 1, \ldots, n \\ & x_t \leq M y_t,\ t = 1, \ldots, n \\ & s_0 = 0, \\ & x_t \geq 0,\ t = 1, \ldots, n \\ & s_t \geq 0,\ t = 1, \ldots, n \\ & y_t \in \{0, 1\},\ t = 1, \ldots, n \end{cases}$$

Implementación del Problema de determinación de lotes no capacitado: Conjuntos y Parámetros

```
# PROBLEMA DE DETERMINACION DE LOTES NO CAPACITADO
#

set T;
/* Periodos */

param M;

param f{t in T};
/* Costo fijo de producir en el periodo t */

param p{t in T};
/* Costo unitario de produccion en el periodo t, */

param h{t in T};
/* costo unitario de almacenamiento en el periodo t */

param d{t in T};
/* demanda en el periodo t */
```

Implementación del Problema de determinación de lotes no capacitado: Variables y Objetivo

```
var x{t in T} >= 0;
/* Cantidades a producirse en el periodo t */

var s{i in (T union {0})} >= 0;
/* inventario al final del periodo t */

var y{t in T}, binary;
/* 1 si se produce en el periodo t, 0 en otro caso */
```

```
minimize costo: sum{t in T} p[t] * x[t] + h[t] * s[t] + f[t] *
    y[t];
/* Costo minimo */
```

Implementación del Problema de determinación de lotes no capacitado: Restricciones

```
s.t. equilibrio{t in T}: s[i-1] + x[t] = d[t] + s[t];
/* Equilibrio de inventario */

s.t. activ{t in T}: x[t] <= M * y[t];
/* Activacion de la produccion */

s.t. inicial: s[0] = 0;
/* Inventario inicial */
```

Modelo

```
param n;
set I := 1 .. n; # objetos

param c{I}; # valor
param a{I}; # tama o
param b;    # capacidad

var x{I} binary; # llevar si/no

... modelo mochilero ...

solve;

display{i in I}: i, x[i];
display:
  sum{i in I} a[i] * x[i], b;

end;
```

Salida

```
Display statement at line 18
i = 1
x[1].val = 1
i = 2
x[2].val = 0
i = 3
x[3].val = 0
i = 4
x[4].val = 0
i = 5
x[5].val = 1
i = 6
x[6].val = 1
i = 7
x[7].val = 1
i = 8
x[8].val = 1
Display statement at line 19
82
b = 101
```

printf

Modelo

```
param n;
set I := 1 .. n; # objetos

param c{I}; # valor
param a{I}; # tama o
param b;    # capacidad

... modelo mochilero ...

solve;

printf{i in I}:
  "objeto %s, decisi n %d\n", i, x[i
    ];
printf:
  "volumen %.2f, \ncapacidad %d\n",
  sum{i in I} a[i] * x[i], b;

end;
```

Salida

```
objeto 1, decisión 1
objeto 2, decisión 0
objeto 3, decisión 0
objeto 4, decisión 0
objeto 5, decisión 1
objeto 6, decisión 1
objeto 7, decisión 1
objeto 8, decisión 1
volumen 82.00,
capacidad 101
```

Instalación

GLPK for Windows:
`http://winglpk.sourceforge.net/`

GLPK for Windows:
`http://winglpk.sourceforge.net/`

El proyecto *GLPK for Windows* provee ejecutables pre-compilados para Windows.

GLPK for Windows:
`http://winglpk.sourceforge.net/`

El proyecto *GLPK for Windows* provee ejecutables pre-compilados para Windows.

El zip de la distribución contiene los ejecutables para 32 y 64 bits en las carpetas *w32* y *w64* respectivamente. Se recomienda agregar la carpeta correspondiente al *PATH* del sistema para poder invocar al *solver* desde cualquier carpeta.

Verificación

Para verificar instalación, ejecutar desde línea de comandos:

```
glpsol --version
```

Para verificar instalación, ejecutar desde línea de comandos:

```
glpsol --version
```

Salida esperada:

```
GLPSOL: GLPK LP/MIP Solver, v4.55

Copyright (C) 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008,
2009, 2010, 2011, 2013, 2014 Andrew Makhorin, Department for Applied
Informatics, Moscow Aviation Institute, Moscow, Russia. All rights
reserved. E-mail: <mao@gnu.org>.

This program has ABSOLUTELY NO WARRANTY.

This program is free software; you may re-distribute it under the terms
of the GNU General Public License version 3 or later.
```

Documentación GNU MathProg

Las distintas distribuciones tienen incluida la documentación de GLPK.

En particular, el manual de referencia de *MathProg* es el archivo `gmpl.pdf`.

También se puede encontrar *online*, por ejemplo en:
`http://gusek.sourceforge.net/gmpl.pdf`