PROGRAMACIÓN LINEAL CON APOYO DE MATHEMATICA Y GLP

Enrique Vílchez Quesada
Escuela de Informática. Universidad Nacional de Costa Rica
evilchez@una.ac.cr
Nivel Universitario

Palabras clave: Programación. Lineal. Métodos. Símplex. *Mathematica*.

Resumen
En el contexto del curso *EIF-405 Investigación de Operaciones* impartido en la Escuela de Informática de la Universidad Nacional de Costa Rica, se han desarrollado una serie de iniciativas metodológicas para abordar el tema de programación lineal, automatizando específicamente dos métodos de resolución: el método *gráfico* y el *símplex*. Ambos se han sistematizado utilizando el lenguaje de programación del conocido software comercial denominado: *Mathematica*. En el presente trabajo se comparte el código de programación desarrollado y disponible en la dirección ULR: http://www.escinf.una.ac.cr/mathematica/index.php/archivos-de-mathematica y además, se muestran dos aplicaciones adicionales para la enseñanza y el aprendizaje de este tema, el software *GLP* que consiste en un visualizador gráfico de problemas de programación lineal en dos dimensiones y el multimedia "Programación lineal con *Mathematica*" creado por el autor de esta propuesta como un recurso complementario para el profesor o el estudiante, con la finalidad de aprender a utilizar *Mathematica* como una herramienta de apoyo de cálculo simbólico y numérico en esta área de conocimiento.

Introducción

Los problemas de programación lineal comúnmente involucran una cantidad significativa de procedimientos que en muchos casos son sumamente mecánicos y con frecuencia distraen la atención del alumno del objetivo principal que se persigue, como lo es, la toma inteligente y sistematizada de decisiones.

En este sentido, automatizar los métodos clásicos de resolución de problemas de programación lineal que involucran una cantidad relevante de variables o restricciones lineales, se torna una necesidad metodológica para cualquier docente involucrado en un curso de investigación de operaciones o álgebra lineal. Particularmente cuando estos cursos se dirigen a estudiantes de ingeniería, el enfoque pedagógico muchas veces se orienta a minimizar los errores algebraicos que podría cometer un alumno, a través del uso de software como medio de enseñanza y aprendizaje.

Con el presente trabajo se expone distintos códigos de programación desarrollados para resolver problemas de optimización lineal a través del uso del paquete comercial *Mathematica 8.0*, además de compartir otras aplicaciones de uso alternativo que podrían servir como recursos didácticos para el abordaje de este tema.

Soluciones a través del software *Mathematica*

Mathematica de la compañía *Wolfram* cuenta con tres comandos específicos para la resolución de problemas de programación lineal: *LinearProgramming*, *Maximize* y *Minimize* (*Wolfram Mathematica* 8, 2011). Cada uno ellos se explican a continuación:

- *LinearProgramming*: resuelve un problema de programación lineal donde se desea minimizar la función objetivo (si el problema es de máximos, se le invierte el signo a esta función), utilizando la siguiente sintaxis: **LinearProgramming[F, v1, v2, v3]**, siendo **F** la función objetivo, **"v1"** una matriz que contiene los coeficientes numéricos de las restricciones, **"v2"** una matriz con las constantes que acompañan a cada igualdad o desigualdad del conjunto de restricciones y **"v3"** una matriz la cual declara las condiciones de no negatividad.
- *Maximize*: calcula el valor máximo de una función y el punto de ocurrencia. La sintaxis para una función en dos variables, por ejemplo, es la siguiente: **Maximize[F[x,y],q,{x,y}]]]**, donde **q** es un conjunto de restricciones relacionadas a través de la conectiva lógica **"y"**.
- *Minimize*: calcula el valor mínimo de una función y el punto de ocurrencia. Su sintaxis para una función en dos variables, por ejemplo, es la siguiente: **Minimize[F[x,y],q,{x,y}]]]**, donde **q** es un conjunto de restricciones relacionadas a través de la conectiva lógica **"y"**.

Una limitante de estas instrucciones radica en mostrar el resultado sin los procedimientos de arrojo.

Programación en *Mathematica* del *método gráfico*

El *método gráfico* es una forma clásica de resolución de un problema de programación lineal en dos variables. Si bien es cierto, los comandos expuestos en la sección anterior resuelven cualquier problema de esta naturaleza, no permiten visualizar la región factible asociada, ni cada uno de los vértices vinculados. En esta dirección, el siguiente código de programación faculta al estudiante a analizar paso a paso el *método gráfico* y del mismo modo, resolver de manera automática cualquier problema en dos dimensiones con solución única o múltiples soluciones. Veamos:

```
F[x_, y_] = Input["Digite la función objetivo: "];
o = Input["Digite un 1 si es un problema de máximos o un 0 si es de mínimos"];
n = Input["Digite la cantidad de restricciones, sin considerar x>=0 y y>=0: "];
Desigualdades = {};
For[h = 1, h ≤ n,
  Desigualdades = Insert[Desigualdades, Input["Restricción, no incluya x>=0 y y>=0: "], -1];
  h++];
Desigualdades = Insert[Desigualdades, x ≥ 0, -1];
Desigualdades = Insert[Desigualdades, y ≥ 0, -1];
q = Desigualdades[[1]];
For[i = 2, i ≤ Length[Desigualdades], q = q && Desigualdades[[i]]; i++];
x2 = Input["Valor máximo de x en la región: "];
y2 = Input["Valor máximo de y en la región: "];
RegionPlot[q, {x, 0, x2}, {y, 0, y2}]
m = Binomial[n, 2];
```

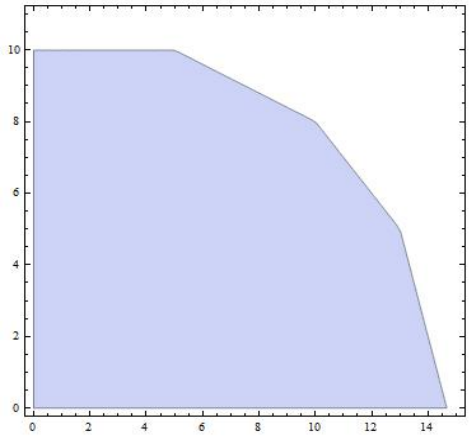
```
(* Se crean las ecuaciones *)
Ecuaciones = {};
For[i = 1, i ≤ Length[Desigualdades] - 2, vs = Characters[ToString[Desigualdades[[i]]]];
  For[j = 1, j ≤ Length[vs], If[vs[[j]] == ">", vs[[j]] = "="]; If[vs[[j]] == "<", vs[[j]] = "="];
   j++]; Ecuaciones = Insert[Ecuaciones, ToExpression[StringJoin[vs]], -1]; i++];
vs = {};
P = {};
l = 0;
For[i = 1, i ≤ m, For[j = i + 1, j ≤ Length[Ecuaciones],
   vs = Solve[{Ecuaciones[[i]], Ecuaciones[[j]]}, {x, y}];
   (* Se prueba si las intersecciones están en la región factible *)
   For[h = 1, h ≤ Length[Desigualdades],
    If[(Desigualdades[[h]] /. {vs[[1, 1]], vs[[1, 2]]}) == False, l = 1]; h++];
   If[l ≠ 1, P = P⋃{{x /. vs[[1, 1]], y /. vs[[1, 2]]}}]; l = 0; j++]; i++];
(* Intersecciones con el eje x *)
InterX = {};
For[i = 1, i ≤ Length[Ecuaciones], vs = {}; vss = {};
  vs = Solve[Ecuaciones[[i]], y]; If[vs ≠ {}, vss = Solve[(y /. vs[[1]]) == 0, x];
   If[vss ≠ {}, InterX = Insert[InterX, vss[[1]], -1]], vs = Solve[Ecuaciones[[i]], x];
   If[vs ≠ {}, InterX = Insert[InterX, vs[[1]], -1]]]; i++];
   (* Se prueba si las intersecciones están en la región factible *)
   Inter = {};
   l = 0;
   For[i = 1, i ≤ Length[InterX], Inter = Insert[Inter, x /. InterX[[i, 1]], -1]; i++];
   For[i = 1, i ≤ Length[Inter], For[h = 1, h ≤ Length[Desigualdades],
      If[(Desigualdades[[h]] /. {x → Inter[[i]], y → 0}) == False, l = 1]; h++];
     If[l ≠ 1, P = P⋃{{x /. x → Inter[[i]], y /. y → 0}}]; l = 0; i++];
   (* Intersecciones con el eje y *)
   InterY = {};
   For[i = 1, i ≤ Length[Ecuaciones], vs = {}; vss = {};
     vs = Solve[Ecuaciones[[i]], x]; If[vs ≠ {}, vss = Solve[(x /. vs[[1]]) == 0, y];
      If[vss ≠ {}, InterY = Insert[InterY, vss[[1]], -1]], vs = Solve[Ecuaciones[[i]], y];
      If[vs ≠ {}, InterY = Insert[InterY, vs[[1]], -1]]]; i++];
   (* Se prueba si las intersecciones están en la región factible *)
   Inter = {};
   l = 0;
   For[i = 1, i ≤ Length[InterY], Inter = Insert[Inter, y /. InterY[[i, 1]], -1]; i++];
   For[i = 1, i ≤ Length[Inter], For[h = 1, h ≤ Length[Desigualdades],
      If[(Desigualdades[[h]] /. {x → 0, y → Inter[[i]]}) == False, l = 1]; h++];
     If[l ≠ 1, P = P⋃{{x /. x → 0, y /. y → Inter[[i]]}}]; l = 0; i++];
   (* Se prueba si el origen está en la región factible *)
   l = 0;
   For[h = 1, h ≤ Length[Desigualdades],
    If[(Desigualdades[[h]] /. {x → 0, y → 0}) == False, l = 1]; h++]; If[l ≠ 1, P = P⋃{{0, 0}}];
   w = {};
   j = 1;
   For[i = 1, i ≤ Length[P], w = Insert[w, F[P[[i, j]], P[[i, j + 1]]], -1]; i++]
   Print["Los vértices son: ", P];
   Print["Sus imágenes son respectivamente: ", w];
```

```
cont = 1;
For[i = 1, i ≤ Length[w], For[j = i + 1, j ≤ Length[w], If[w[[i]] == w[[j]], cont = cont + 1] ; j++];
  i++];
If[o == 1,
  Pos = {};
  For[i = 1, i ≤ Length[w], If[Max[w] == w[[i]], Pos = Insert[Pos, i, -1]]; i++];
  If[cont == 1, Print["Hay una única solución que ocurre en: ", P[[Pos[[1]]]],
    " y corresponde a: ", w[[Pos[[1]]]]],
   Print[ "Hay múltiples soluciones dadas por la ecuación: ",
    Simplify[P[[Pos[[1]]]] + t (P[[Pos[[2]]]] - P[[Pos[[1]]]])], " 0<=t<=1"]]];
If[o == 0,
Pos = {};
For[i = 1, i ≤ Length[w], If[Min[w] == w[[i]], Pos = Insert[Pos, i, -1]]; i++];
If[cont == 1, Print["Hay una única solución que ocurre en: ", P[[Pos[[1]]]],
  " y corresponde a: ", w[[Pos[[1]]]]],
 Print[ "Hay múltiples soluciones dadas por la ecuación: ",
  Simplify[P[[Pos[[1]]]] + t (P[[Pos[[2]]]] - P[[Pos[[1]]]])], " 0<=t<=1"]];
```


Estas líneas de código generan paso a paso la metodología que se usa en el *método gráfico*, solicitando al usuario las condiciones del problema de programación lineal como expresiones algebraicas (la función objetivo y las restricciones). En Álgebra lineal apoyada con Mathematica (Vílchez, 2012), se comparte un programa similar donde se solicita el ingreso de la función objetivo y las restricciones lineales como coeficientes numéricos de matrices y no como expresiones algebraicas. Por ejemplo, al correr en cualquiera de los dos programas el problema:

$$\begin{aligned}&\text{Maximizar } Z = 2x_1 + x_2 \text{ sujeta a:}\\ &x_2 \leq 10\\ &2x_1 + 5x_2 \leq 60\\ &x_1 + x_2 \leq 18\\ &3x_1 + x_2 \leq 44\\ &x_1 \geq 0, x_2 \geq 0\end{aligned}$$

Se obtiene:



Los vértices son: $\left\{\{0, 0\}, \{0, 10\}, \{5, 10\}, \{10, 8\}, \{13, 5\}, \left\{\frac{44}{3}, 0\right\}\right\}$

Sus imágenes son respectivamente: $\left\{0, 10, 20, 28, 31, \frac{88}{3}\right\}$

Hay una única solución que ocurre en: {13, 5} y corresponde a: 31

Programación del *método símplex* a través de *Mathematica*

En Bronson (1996), Hillier y Liberman (1998) y Taha (1997) el lector encontrará en detalle el algoritmo de resolución de problemas de programación lineal a través del *método símplex*. En esta sección se comparte una implementación de *símplex* utilizando el software *Mathematica*. Ésta considera únicamente restricciones de la forma ≤ en las cuales no existen soluciones degeneradas. El programa no resuelve problemas con soluciones múltiples; además, si existe un cociente mínimo mayor a 1000, en el código se debe modificar esta constante por un número mayor al resultado del cociente. Por medio del código compartido, se insta al lector a crear la implementación del *método símplex* con variables artificiales.


```
m = Input["Digite la cantidad de incógnitas del problema de programación lineal: "];
MFO = {};
For[j = 1, j ≤ m,
  MFO =
   Insert[MFO, Input["Digite el coeficiente correspondiente de la función objetivo: "],
    j]; j++];
n = Input["Digite la cantidad de restricciones, sin considerar x>=0 y y>=0: "];
o = Input["Digite un 1 si es un problema de máximos o un 0 si es de mínimos: "];
If[o == 0,  For[i = 1, i ≤ Length[MFO], MFO[[i]] = -1 * MFO[[i]]; i++]];
MCN = ConstantArray[0, {n, m}];
For[i = 1, i ≤ n,
  For[j = 1, j ≤ m,
   MCN = ReplacePart[MCN, Input["Digite el coeficiente de la restricción respectiva: "],
     {i, j}]; j++]; i++];
MTC = ConstantArray[0, {n, 1}];
For[i = 1, i ≤ n,
  MTC = ReplacePart[MTC, Input["Digite el término constante correspondiente: "], {i, 1}];
  i++];
TablaSimplex = ConstantArray[0, {n + 1, m + n + 2}];
(* Me construye la tabla símplex inicial *)
For[i = 1, i ≤ n + 1,
  For[j = 1, j ≤ m + n + 2,
   If[i ≤ n && j ≤ m, TablaSimplex = ReplacePart[TablaSimplex, MCN[[i, j]], {i, j}]];
   If[i ≤ n && j > m && j ≤ n + m,
    If[i == j - m, TablaSimplex = ReplacePart[TablaSimplex, 1, {i, j}]]];
   If[i ≤ n && j > m + n + 1, TablaSimplex = ReplacePart[TablaSimplex, MTC[[i, 1]], {i, j}] ];
   If[i == n + 1 && j ≤ m, TablaSimplex = ReplacePart[TablaSimplex, -MFO[[j]], {i, j}]];
   If[i == n + 1 && j == m + n + 1, TablaSimplex = ReplacePart[TablaSimplex, 1, {i, j}]]; j++];
  i++];

(* Método símplex *)
vc = {}; (* Vector de cocientes *)
vop = {};
pos = 1;
cont = 0;
k = 0;
w = 0;
```

```
(* j representa la columna en la que nos fijamos,
i representa la iteración i del método símplex  y t la variación de las filas *)
For[j = 1, j ≤ m + n,
  Print["Tabla símplex ", j]; Print[MatrixForm[TablaSimplex]];
  For[i = 1, i ≤ n,
   If[TablaSimplex[[i, j]] != 0 && (TablaSimplex[[i, m + n + 2]] / TablaSimplex[[i, j]]) > 0,
    vc = Insert[vc, TablaSimplex[[i, m + n + 2]] / TablaSimplex[[i, j]], {pos}],
    vc = Insert[vc, 1000, {pos}]]; pos++; i++]; Print["El vector de cocientes es: ", vc];
  (* Encuentra la ubicación del elemento pivote *) minpos = Position[vc, Min[vc]];
  Print["Sale la variable: ", minpos[[1, 1]]];
  vop = Insert[vop, minpos[[1, 1]], -1];
  If[Length[minpos] == 1, TablaSimplex[[minpos[[1, 1]]]] =
    1 / TablaSimplex[[minpos[[1, 1]], j]] * TablaSimplex[[minpos[[1, 1]]]];
   For[i = 1, i ≤ n + 1, If[i ≠ minpos[[1, 1]],
     TablaSimplex[[i]] = -TablaSimplex[[i, j]] * TablaSimplex[[minpos[[1, 1]]]] +
       TablaSimplex[[i]]]; Print[MatrixForm[TablaSimplex]]; i++], cont = 1;
   Print["Solución degenerada o no hay solución óptima"]]; If[cont == 1, Break[]];
  For[h = 1, h ≤ m + n, If[TablaSimplex[[n + 1, h]] ≥ 0, k++]; h++];
  If[k == n + m, Print["Tabla símplex ", j + 1]; Print[MatrixForm[TablaSimplex]];
   Break[], k = 0]; vc = {}; pos = 1; j++];
If[o == 1 && cont ≠ 1, Print["El valor máximo es: ", TablaSimplex[[n + 1, m + n + 2]]],
  If[count ≠ 1, Print["El valor mínimo es: ", -1 * TablaSimplex[[n + 1, m + n + 2]]]]];
If[cont ≠ 1, If[Length[vop] == 1, Print[x1, "→", TablaSimplex[[vop[[1]], m + n + 2]]],
  For[i = 1, i < Length[vop], For[j = i + 1, j ≤ Length[vop],
    If[vop[[i]] == vop[[j]], Print[xi, "→", 0]; w = 1]; j++];
   If[w == 0, Print[xi, "→", TablaSimplex[[vop[[i]], m + n + 2]]]]; w = 0; i++]]];
If[m == Length[vop], Print[xi, "→", TablaSimplex[[vop[[i]], m + n + 2]]]];
```

Por ejemplo, al resolver:

$$\begin{aligned}&\text{Maximizar } Z = 3x_1 + 4x_2 + 5x_3 \text{ sujeta a:}\\ &3x_1 + x_2 + 5x_3 \leq 150\\ &x_1 + 4x_2 + x_3 \leq 120\\ &2x_1 + 2x_3 \leq 105\\ &x_1 \geq 0, x_2 \geq 0, x_3 \geq 0\end{aligned}$$

Se obtiene:

Tabla símplex 4

$$\begin{pmatrix} \frac{11}{19} & 0 & 1 & \frac{4}{19} & -\frac{1}{19} & 0 & 0 & \frac{480}{19} \\ \frac{2}{19} & 1 & 0 & -\frac{1}{19} & \frac{5}{19} & 0 & 0 & \frac{450}{19} \\ \frac{16}{19} & 0 & 0 & -\frac{8}{19} & \frac{2}{19} & 1 & 0 & \frac{1035}{19} \\ \frac{6}{19} & 0 & 0 & \frac{16}{19} & \frac{15}{19} & 0 & 1 & \frac{4200}{19} \end{pmatrix}$$

El valor máximo es: $\frac{4200}{19}$

$x_1 \to 0$

$x_2 \to \frac{450}{19}$

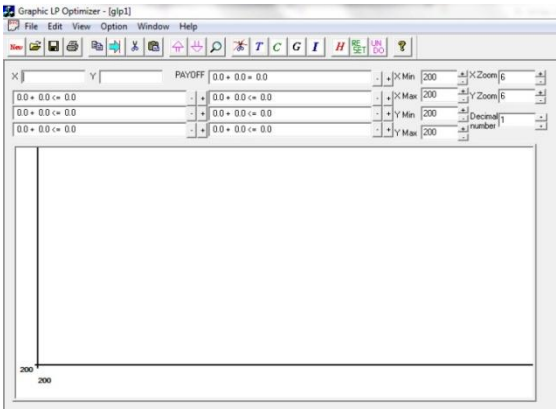
$x_3 \to \frac{480}{19}$

El programa arroja todas las *tablas símplex* requeridas en el procedimiento, aunque la salida anterior solo muestra la última.

Método gráfico* mediante el uso del software *GLP

El software *GLP* es un visualizador gráfico de problemas de programación lineal en dos dimensiones, se distribuye gratuitamente con el texto Investigación de operaciones en la ciencia administrativa (Eppen, G. y otros, 2000).

La pantalla principal del *GLP* es la siguiente:

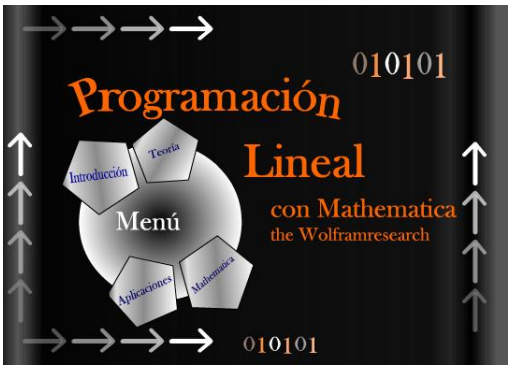


Cada uno de los campos de texto que se observan en la parte superior permite ingresar la función objetivo y las restricciones correspondientes, con la limitante de seis restricciones a lo sumo. Las flechas de color rosado hacia arriba y hacia abajo, maximizan o minimizan la función lineal, mostrándose simultáneamente la región factible en el área de graficación de la pantalla.

Multimedia "Programación lineal con *Mathematica*"

"Programación lineal con *Mathematica*" fue elaborado por el autor de esta propuesta y el profesor Eric Padilla Mora colega de la Universidad Estatal a Distancia en Costa Rica, utilizando *Adobe Flash*. El diseño educativo se fundamentó en crear una herramienta con la intención de servir de apoyo al profesor o al alumno para aprender principios básicos de programación lineal y uso de software en la resolución de problemas dentro de este campo

de conocimiento. Es de uso gratuito con fines pedagógicos. El multimedia posee la siguiente interfaz:



El botón "Introducción" recarga el *splash* de inicio de la aplicación, "Teoría" brinda una explicación sobre los principios fundamentales de un problema de programación lineal, además se proporciona al usuario un recorrido histórico y algunas actividades interactivas, tales como: juegos y quices. "Mathematica" resume los comandos más importantes con los que cuenta el software *Mathematica* (que ya fueron explicados en la sección 2) para tratar el tema de programación lineal y brinda una serie de videos demostrativos al alumno.

Conclusiones

La resolución de problemas de programación lineal demanda la necesidad de programar los métodos clásicos de esta teoría, con la finalidad de centrar los procesos de enseñanza y aprendizaje en el análisis, la interpretación y la toma de decisiones. Lo anterior permite optimizar el tiempo de clase y el esfuerzo intelectual del alumnado, en los aspectos esenciales de esta área de la investigación de operaciones, sin caer en el desgaste de aplicar rutinas mecánicas desarrolladas en papel y lápiz.

El software *Mathematica 8.0* ofrece muchas facilidades de programación y el código compartido mediante el presente trabajo, ofrece una clara oportunidad de implementación asistida por computadora, para aquellos docentes interesados en desarrollar este tipo de temáticas complementando el enfoque tradicional con el uso de la tecnología educativa.

Referencias Bibliográficas

Bronson, R. (1996) *Investigación de operaciones*. México: Serie Schaum, Mc Graw Hill.

Eppen, G. y otros. (2000) *Investigación de operaciones en la ciencia administrativa*. México: Prentice Hall.

Hillier, F. y Liberman, G. (1998) *Introducción a la investigación de operaciones*. México: Mc Graw Hill.

Taha, H. (1997) *Investigación de operaciones*. México: Prentice Hall.

Vílchez, E. (2012) *Álgebra lineal apoyada con Mathematica*. Costa Rica: Editorial Tecnológica.

Wolfram Mathematica 8: Documentation Center (2011) *Mathematica functions and tutorials*. Obtenido el 1 de enero del 2011: http://reference.wolfram.com/mathematica/guide/Mathematica.html.