La geometría del método Simplex y sus aplicaciones utilizando Matlab®

Recepción: Agosto de 2007 / Aceptación: Noviembre de 2007

(1) Eduardo Raffo Lecca
(2) Edgar Ruiz Lizama

RESUMEN

El artículo revisa los conceptos de la geometría computacional aplicados al método simplex utilizando MATLAB®. Dado un modelo de programación lineal o PL, una solución factible es un vector que especifica un valor para cada variable en el problema, el cual sustituyéndolo satisface todas las restricciones; incluidas las de signo. El trabajo revisa los conceptos de la geometría computacional, el método simplex y presenta la manera de aplicar la forma gráfica a un problema PL. Se incluyen archivos M-File (de MATLAB, versión 7.0). Finalmente se discuten dos casos de aplicación de un problema PL.

Palabras Clave: Programación lineal, modelos matemáticos, algoritmo simplex, cerco convexo, geometría computacional, puntos extremos, soluciones básicas factibles.

THE GEOMETRY OF THE SIMPLEX METHOD AND HIS APPLICATIONS USING MATLAB®
ABSTRACT

The article review concepts of computacional geometry apply to simplex methods using MATLAB®. Do it programming linear PL, a feasible solution is a vector who specify all restrictions, includes that sign. The work check concepts of computational geometry, the simplex method, and show the maner to apply graph form one problem PL. To include meta files (of MATLAB, version 7.0) . Finally to argue two case of application of a problem PL.

Key Words: Linear Programming, Mathematical Models, Simplex Algorithm, Euclidean space, Convex Hull, Computational Geometry, Extreme Points, Basic Feasible Solutions.

LA PROGRAMACIÓN LINEAL Y EL MÉTODO SIMPLEX

Durante e inmediatamente después de la segunda guerra mundial, trabajos en planeación de acciones militares a gran escala (*planning large-scale military*) como la ruta de barcos entre los puertos, se precedieron de modo independiente, hasta que en 1947 se unificaron con la construcción del método computacional: *el algoritmo símplex*, para formular tales problemas de modo explícito y determinar sus soluciones. Esta clase de problemas se conoce como Programación Lineal o PL[1].

La programación lineal tiene el propósito de construir modelos para una clase de problemas de decisión, que es muy solicitada en la industria, el gobierno, la economía y la ingeniería, siendo su estructura la más simple[2]. La programación lineal (Término sugerido por T.C. *Koopmans*) està relacionada con la descripción de las interrelaciones de los componentes de un sistema. Representando esta estructura en términos matemáticos, resulta un sistema de inecuaciones y ecuaciones lineales, llamado modelo de programación lineal.

La teoría de la programación lineal es concerniente con procedimientos científicos para arribar al mejor diseño desde el requerimiento específico y el estado de un objetivo.

Para George B. Dantzig, la programación lineal es un sistema con un conjunto de funciones elementales: las actividades. Una actividad es una caja negra (black box) en donde el flujo ingresa (hombres, materiales, equipos) y después sale como un producto.

Un problema de PL se representa como una formulación cuya estructura se presenta en la figura 1.

Los PL's que envuelven dos variables, pueden ser resueltos dibujando un diagrama en el plano cartesiano. Dado un PL, una solución factible es un vector que especifica un valor para cada variable en el problema, el cual sustituyéndolo satisface todas las restricciones incluidas las de signo.

Una alternativa es la representación de un PL en la forma canónica (para la maximización), y es como sigue:

$$\text{Max } z = \mathbf{cx}$$
$$\text{sujeto a}$$
$$\mathbf{Ax\ b}$$
$$\mathbf{x\ 0}$$

(1) Ingeniero Industrial. Profesor del Departamento Académico de Ingeniería de Sistemas e Informática, UNMSM.
E-mail: craffolecca@yahoo.es
(2) Magister en Informática: Ingeniero Industrial. Profesor del Departamento Académico de Ingeniería de Sistemas e Informática, UNMSM.
E-mail: cruizl@unmsm.edu.pe

Figura 1. Un problema de PL [1]

$$Max\ z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

sujeto a

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \le b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \le b_2$$

$$\ldots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \le b_m$$

$$x_j \ge 0 \qquad \forall j$$

Un sistema lineal se dice que es un sistema homogéneo, si todas las constantes del lado derecho (RHS) son ceros. Entonces un sistema como $\boldsymbol{Ax = 0}$, es sistema lineal de ecuaciones homogéneo.

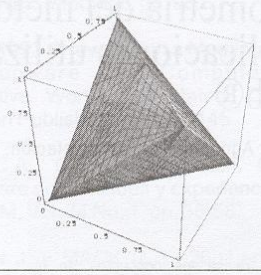
Si, x^1 y x^2 son soluciones factibles de este sistema $Ax^1 = 0$, $Ax^2 = 0$ se tiene que $A(\alpha_1x^1 + \alpha_2x^2) = 0$ para todo α_1, α_2. Entonces cua lquier combinación de soluciones de un sistema lineal de ecuaciones homogéneas es también una solución de este sistema.

Considere un sistema lineal de ecuaciones no homogéneo, con variables no negativas $Ax = b, x \ge 0$. Si x^1, x^2 son soluciones factibles de este sistema, se tiene $Ax^1 = Ax^2 = b, x^1, x^2 \ge 0$, entonces $A(\alpha_1x^1 + \alpha_2x^2) = b, \alpha_1x^1 + \alpha_2x^2 \ge 0$, y provee que $\alpha_1 + \alpha_2 = 1, \alpha_1, \alpha_2 \ge 0$. Entonces una combinación convexa de soluciones factibles de este sistema es también factible.

A *simplex* o *n*- símplex es el *convex hull* de un conjunto de (*n* +1) puntos. Un símplex es un *n*-dimensional análogo a un triángulo.

En la teoría de optimización el algoritmo símplex, creado por el matemático norteamericano George B. Dantzig 1947, es una técnica popular para dar solución a un problema de PL

Figura 2. Un 3-Simplex



El concepto de un símplex, es un politopo de $n+1$ vértices en E^n (ver figura 2); así por ejemplo: un tetraedro en un espacio de tres dimensiones; también son:

- El 1-simplex es un segmento de línea.
- El 2-simplex es un triángulo.
- El 3-simplex es un tetraedro.

En geometría un politopo significa, la generalización a cualquier dimensión de un polígono bidimensional, y un poliedro tridimensional. El término fue creado por *Alicia Boole Stott*, hija del lógico matemático G. Boole[3].

CONVEX HULL

En matemáticas, el ***convex hull*** o cerco envolvente para un conjunto de puntos *X* es un espacio vectorial real *V*, con el mínimo conjunto convexo que contiene a *X*.

El convex hull de *X* puede ser descrito como el conjunto de combinaciones convexas de puntos desde *X*. El cerco convexo de *X* es:

$$H_{convex}(X) = \left\{ \sum_{i=1}^{k} \alpha_i x_i \;\middle|\; \alpha_i \in \Re, x_i \in X, \alpha_i \ge 0, \sum_{i=1}^{k} \alpha_i = 1, k = 1,2,\ldots \right\}$$

Figura 3. Ejemplos de *convex hull*

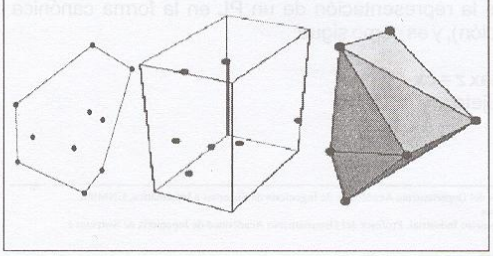


Figura 4. El *convex hull* para un conjunto de puntos *X*

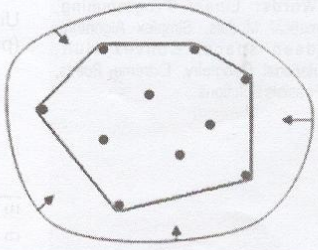
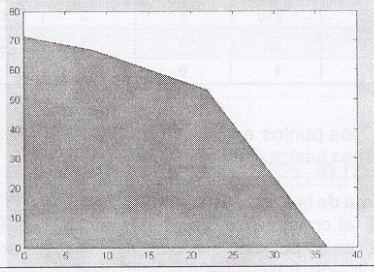


Figura 5. El convex hull obtenido con Matlab.

```
>> x=[21.8750  8.3333  36.3636      0      0];
>> y=[ 53.1250  66.6667       0  71.4286      0];
>> k=convhull(x,y);
>> fill(x(k),y(k),'r')
```

Figura 6. El convex hull obtenido con Matlab.



con X un subconjunto del espacio vectorial *n*-dimensional (Ver figura 3.)

El convex hull de un conjunto de puntos, es la intersección de todos los conjuntos convexos. En particular, un conjunto es convexo, si todas las combinaciones convexas de pares de puntos .

El convex hull para un conjunto de puntos X, es el mínimo conjunto convexo, que contiene los puntos de X (ver figura 4).

Debido a sus múltiples aplicaciones, la computación del convex hull apoyada por la geometría computacional, ha sido estudiado intensivamente; y los campos como el reconocimiento de patrones, cutting stock y allocation, son sólo algunas de sus aplicaciones [4].

Los algoritmos de Graham y Jarvis, son dos algoritmos muy utilizados para determinar el *convex* hull[5].

MATLAB®, incluye en su biblioteca de funciones, la función **convhull**, esto es "The Quickhull Algorithm for Convex Hull", que combina el Quickhull Algorithm y el Beneath-Beyond Algorithm; muy similar al algoritmo aleatorio y de incrementos para la triangulación de Delaunay[5]. Una búsqueda aleatoria por árbol, en contraste con las búsquedas de los árboles balanceados, es que la conducta depende de los valores producidos por generadores de números aleatorios, antes que por el orden de entrada[6].

La función **convhull**, posee las siguientes sintaxis:

```
k = convhull(x,y)
k = convhull(x,y,options)
[k,a] = convhull(...)
```

donde k = **convhull**(x,y) retorna los índices de los vectores x y y de los puntos en el convex hull. Ejecutando los comandos de MATLAB, que se presentan en la figura 5, se consigue invocar y ejecutar la función **convhull**. Esta función necesita de los puntos *x* e *y*, entregando el cerco convexo como resultado.

SOLUCIONES BÁSICAS FACTIBLES

De la forma canónica de la PL, se transforma a la forma estándar, que incluye restricciones de igualdad y variables no negativas:

Max $z = \mathbf{cx}$
sujeto a:
$\mathbf{Ax = b}$
$\mathbf{x}\ 0$

Esto se consigue modificando las restricciones:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq b_i$$

e introduciendo una nueva variable(de holgura o *slack*)

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n + s_i = b_i$$

Para el caso de desigualdades de mayor o igual que, se escribe como:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n - s_i = b_i$$

Dado el siguiente PL:

$$\max z = 143x_1 + 60x_2$$
$$sa:$$
$$120x_1 + 210x_2 \leq 15000$$
$$110x_1 + 30x_2 \leq 4000$$
$$x_1 + x_2 \leq 75$$
$$x_1, x_2 \geq 0$$

La conversión a la forma estándar para las restricciones de menor que o igual, se consigue adicionado variables de holgura (Ver la figura 7).

Figura 7. La matriz en la forma estándar

120	210	1	0	0
110	30	0	1	0
1	1	0	0	1

En PL, los puntos extremos, son también llamados soluciones básicas factibles.

Teorema de la solución básica factible
Sea K el conjunto de soluciones factibles al PL, si $x \in K$ es un punto extremo de K, entonces este es una solución básica factible o BFS(Basic Feasible Solution).

El sistema de ecuaciones del PL, consta de *m* igualdades y *n* restricciones. Se asume que el *rank* de ***A*** es *m*; cada base consiste de *m* vectores columnas de ***A***. El número total de distintas bases para el PL en la forma estándar es menor que o igual la combinación binomial de *n* en *m*:

$$\binom{n}{m} = \frac{n!}{(n-m)!m!}$$

Cada BFS esta asociada con una o más bases para el PL. El número total de distintas BFS no puede exceder de esta combinación, la cual es finita. Ver las figuras 8 y 9 donde se implementa la función en MATLAB **corner.m** y su asociado **deleteCol.m**.

El conjunto de puntos extremos o BFS se muestran en la figura 10.

Figura 8. Archivo corner.m

```
function puntos=corner(A,b)
% puntos extremos para un PL
% E. Raffo Lecca
%
    [m,n]=size(A);
    puntos=[ ];
    if  n>=m
       combin = nchoosek(n,m) ;% coeficientes binomiales
       v=nchoosek(1:n,m);
       %Matrix v, contains todas las     combinaciones
       % n rows and  m columns
       for k=1:combin
          y=zeros(n,1);
          x=inv(A(:,v(k,:)))*b;
          if all( x >= 0 & (x ~= inf & x ~= -inf ) )
                y(v(k,:)) = x;
                puntos = [puntos y];
          end
       end
    else
        error('Numero de ecuaciones es > numero de variables ');
    end
    puntos
    % delete puntos repetidos
    puntos=deleteCol(puntos);
```

Teorema de la Resolución
Cada solución factible del PL, puede ser expresada como la suma de:

(1) Una combinación convexa de los BFS del PL y
(2) Una solución homogénea correspondiente al PL.

Es decir un sistema de ecuaciones, con variables no negativas del PL, y una solución homogénea correspondiente al PL, corresponde a un vector ***y*** que satisface:

$$Ay = 0$$
$$y \geq 0$$

Un sistema homogéneo correspondiente al PL, es llamado una solución homogénea extrema, si este es un BFS de:

$$Ay = 0$$
$$\sum_{j=1}^{n} y_j = 1$$
$$y_j \geq 0, \forall j$$

Lema
Cualquier 0 es la solución homogénea única, o cada solución homogénea correspondiente al PL, puede ser expresado como una combinación lineal de soluciones homogéneas extremas.

El script **graphPL.m** y la ejecución del método gráfico para un PL ejemplo se muestran en las figuras 11 y 12. Más detalles sobre *script*, ver [9].

Figura 9. Archivo deleteCol.m

```
function puntos=deleteCol(puntos)
% delete columnas repetidas
% E. Raffo Lecca
%
    [nn,L] = size(puntos);
     v = [ ];
     for i=1:L-1
          for j=i+1:L
               x= puntos(:,i) ;
               y= puntos(:,j );
               if all( x == y )
                    v= [v, j ];
               end
          end
     end
     puntos(:,v) = [ ];
```

Figura 10. Puntos extremos o BFS

21.8750	8.3333	36.3636	0	0
53.1250	66.6667	0	714286	0

Figura 11. Script para graphPL.m

```
function graphPL(A,b,c,direccion)
 % solucion grafica para un PL
% E. Raffo Lecca
%
% Linear Programming, two variables
% Datos
% A  =matriz tecnologica
% b  =vector de la mano derecha
% c  =vector de costos
% direccion =cadena de la direccion de las variables
% Resultados
%
% max z = 143x1 +   60x2
%             120x1 + 210x2  <=  15000
%             110x1 +   30x2  <=  4000
%                 x1 +      x2   <= 75
%
[m n]=size(A);
if n ~= 2
   string='El problema es solo para dos variables';
   msgbox(string,'Error Window graphPL','help');
   return
end
% crear la submatriz de holguras e invocar a corner
A = [A, eye(m)];
for k=1:m
   if  direccion(k) == '>'
       A(k,n+k)= -1;
   end
   if b(k) < 0
       A(k,:)= -A(k,:);
       b(k)  = -b(k);
   end
end
n=n+m;
vertices=corner(A,b);
if length(vertices) == 0
   error('No existe region factible');
end
vertices=vertices(1:2,:);
vertices=deleteCol(vertices);
d=corner([A;ones(1,n)],[zeros(m,1);1]);
if length(d) > 0
    error('No existe region factible acotada');
end
x1=vertices(1,:);
x2=vertices(2,:);
k=convhull(x1,x2);
hold on;
fill(x1(k),x2(k),'r');
xlabel('x_1','FontSize',15);
ylabel('x_2','FontSize',15);
z=c*vertices;
z0=max(z);
%xmax=max(vertices(1,:)); xmin=min(vertices(1,:));
j=find(z == z0);
x0=vertices(1,j); y0=vertices(2,j);
if  c(1)*c(2) ~= 0
    x=[0.9*x0,1.1*x0]; y= y0 +(-c(1)/c(2))*(x-x0);
    plot(x,y,'b--','LineWidth',2);
end
title(['Region de solucion factible, con z = ',num2str(z0)],'FontSize',15)
text(x0,y0,['   ( ' , num2str(x0), ' , ',num2str(y0),' )'],'FontSize',15)
plot(x0,y0,'o',...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','g',...
            'MarkerSize',10)
grid on
```

Figura 12. Ejecución de graphPL.m

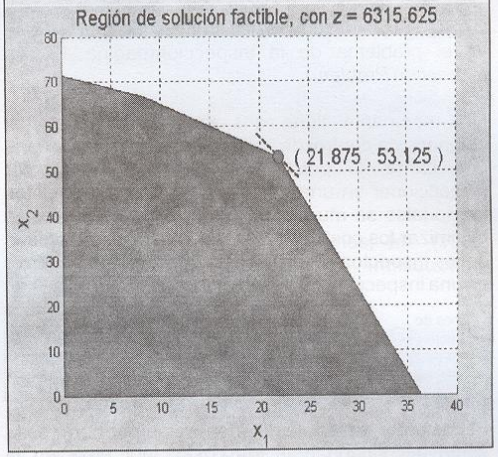


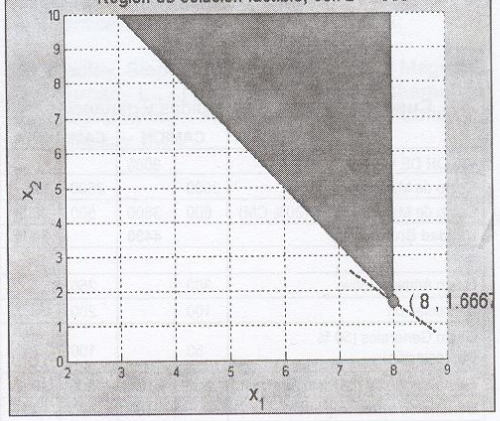
Figura 13. Invocación al programa graphPL.m

```
A = [1 0;0 1;5 3];
B = [8;10;45];
graphPL(A,b,[ -40 -36],'<<>')
```

Figura 14. Ejecución de graphPL.m

```
vertices =
    3.0000    8.0000    8.0000
   10.0000    1.6667   10.0000
```

Figura 15. Método gráfico para el caso 1



APLICACIONES

CASO 1: Tomado del clásico libro *Ravindran*[11], éste es el problema de la inspección(página 18, *An Inspection Problem*).

Una compañía tiene dos tipos de inspectores, asignados a control de calidad. Actualmente se tiene una carga de por lo menos 1800 piezas por inspeccionar en una jornada de 8 horas. Los datos adicionales se muestran en la cuadro 1. Se desea minimizar los costos totales, en la asignación óptima de los inspectores. Considerar que el costo por error en una inspección es de $2.0.

Tipos de inspectores	Tasa (Unidades/hr.)	Precisión (%)	Salario ($/hr.)	disponibilidad
1	25	98	4	8
2	15	95	3	10

El costo por desarrollar la inspección por hora, es la suma del salario más los costos por error en la inspección; esto es:

$$\text{Tipo 1: } \$4 + 2(25)(0.02) = \$5/hr.$$

$$\text{Tipo 2: } \$3 + 2(15)(0.05) = \$4{,}5/hr.$$

La función objetivo, corresponde al total de una jornada de 8 horas:

$$Z = 8(5x_1 + 4.5x_2) = \$5/hr.$$

La condición de unidades a inspeccionar en una jornada de 8 horas es:

$$8(25)x_1 + 8(15)x_2 \geq 1800 \text{ o } 5x_1 + 3x_2 \geq 45$$

El PL, para el problema es como sigue:

$$\min Z = 40x_1 + 36x_2$$

El programa presenta el *convex hull*, compuesto de

$$x_1 \leq 8$$
$$x_2 \leq 10$$
$$5x_1 + 3x_2 \geq 45$$
$$x_1, x_2 \geq 0$$

Cuadro 1. Estado de pérdidas y ganancias

	CAMION		CAMIONETA	
VALOR DE VENTA		8000		6000
Costo de Material	3000		2500	
Costo de Mano de Obra (20% CM)	600	3600	500	3000
Utilidad Bruta		**4400**		**3000**
Gasto Administrativo	300		250	
Depreciación	100		200	
Gasto Generales (50 % Depreciación)	50		100	
Gasto Variables (2 Gasto Administ.)	600	1050	500	1050
Beneficios antes Impuestos		**3350**		**1950**

tres puntos, tal como aparece en la Figura 13.

La solución queda presentada en la figura 14; donde el óptimo es:

$$x_1 = 8, x_2 = 5/3, Z = 380$$

CASO 2: El presente caso: Ensambladora de vehículos, es una adaptación del libro de Bradley[12].

La empresa ensambladora de vehículos Morillas, se encuentra ubicada en la ciudad de Trujillo. Actualmente, se encuentra planeando su producción para los siguientes meses. Sus productos son: Un camión y una camioneta.

El proceso de producción, consiste en el ensamble del tren de aterrizaje con la carrocería; y el acabado final.
Mensualmente se pueden ensamblar 3000 unidades del tren de aterrizaje entre camiones y camionetas. La capacidad de ensamblaje de carrocerías es de 2000 camiones o 4000 camionetas, la carrocería de un camión se ensambla en el doble tiempo que una camioneta. El acabado final tiene capacidad para 2000 camiones y 2500 camionetas.

Formule un programa lineal para la producción y resuelva, contando con los datos del cuadro 1.

Sean:
X_1= Producción de camiones
X_2= Producción de camionetas

$$\text{Max. } Z = 3350\,X_1 + 1950\,X_2$$

sujeto a Restricción de ensamble del tren

$$X_1 + X_2 \leq 3000$$

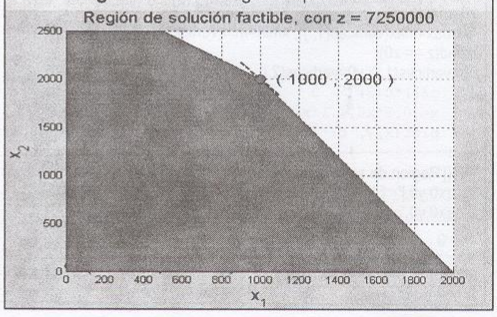
Restricción de ensamblaje de la carrocería

$$\frac{X_1}{2000} + \frac{X_2}{4000} \leq 1$$

Equivalente de: % prod. Camión + % prod. Camioneta $\leq$ 100%

Es decir:

Figura 16. Método gráfico para el caso 2



2t unidades prod. Camión + t unidad prod. Camioneta

$$2t\,X_1 + t\,X_2 \leq 4000 \quad o$$
$$2\,X_1 + X_2 \leq 4000$$

Restricción acabado final

$$X_1 \leq 2000$$
$$X_2 \leq 2500$$

En la figura 16, se presenta la solución usando la forma gráfica, haciendo uso del programa escrito en MATLAB: **graphPL.m**.

De la salida se concluye, que la ganancia máxima es de $7250.000, con la siguiente asignación:

Tipo	Producción
Camiones	1000
Camionetas	2000

CONCLUSIONES

El conjunto finito de candidatos a soluciones en un PL, es el conjunto de vértices de un politopo, definido por las restricciones lineales. La forma de solución gráfica en un PL, permite en otras cosas visualizar el conjunto convexo y el punto extremo con la función objetivo más alta. Para la situación específica de dos variables de decisión, esta forma es la más solicitada.

Existen eficientes programas de computadoras cuando el número de variables es de más de dos variables de decisión.

La Programación Lineal, es una aplicación de los conjuntos convexos. El método gráfico en PL, es fácil de implementar haciendo uso del convex hull.

El algoritmo de *simplex*, está basado en la idea del mejoramiento de los costos, por el movimiento de un vértice a otro en el politopo. El software Matlab por poseer funciones incorporadas y un ambiente gráfico permite la aplicación de la geometría del método simplex para solucionar problemas de PL tal como se ha visto en los dos casos aquí presentados.

REFERENCIAS BIBLIOGRAFICAS

1. Dantzig, George B., (1963). Linear Programming and Extensions. Princeton University Press, New Jersey. U.S.A.
2. Raffo Lecca, Eduardo, (1999). Investigación de Operaciones. Raffo Lecca editores. Lima, Perú.
3. Wikipedia. (2007). La enciclopedia libre, http://es.wikipedia.org/.
4. Preparata, Franco P. y Shamos, M., (1985) Computational Geometry An Introduction. Springer-Verlag.
5. Raffo Lecca, Eduardo, (2005). Geometría Computacional: El problema del cerco convexo. Industrial Data, Vol. (8), 2:pp. 69-76(2005), ISSN 1560-9146.
6. Barber, C. B., D.P. Dobkin, and H.T. Huhdanpaa. The Quickhull Algorithm for Convex Hulls. ACM Transactions on Mathematical Software, Vol. 22, No. 4, Dec. 1996, p. 469-483.
7. Laszlo, Michael J., (1996). Computational Geometry and Computer Graphics in C++. Prentice-Hall. U.S.A.
8. Murty, Katta G., (1983). Linear Programming. John Wiley & Sons. U.S.A.
9. Raffo Lecca, Eduardo, (2005). Métodos Numéricos Para Ciencia e Ingeniería con MATLAB. Raffo Lecca editores. Lima, Perú.
10. Papadimitriou, Christos H., (1982). Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Inc. U.S.A.
11. Phillips, Don T., Ravindran, A., Solberg, James., (1976). Operations Research: Principles an Practice. John Wiley & Sons, Inc. U.S.A.
12. Bradley, Stephen P., Hax, Arnoldo C., Magnanti, Thomas L., (1977). Applied Mathematical Programming. Addison-Wesley Publishing Company. U.S.A.