DISEÑO ÓPTIMO DE RED DE CABLEADO ESTRUCTURADO PARA LOS LABORATORIOS DE INGENIERÍA ELECTRÓNICA EN LA UNIVERSIDAD DE IBAGUÉ USANDO TRES MÉTODOS DECISIONALES Y BASADOS EN SPSS PARA LA ESCOGENCIA DEL MEJOR

Luis Leonardo Rivera, Felipe A. Arias, Diego A. Martínez, Julián F. Arena *

RESUMEN

Se describen tres algoritmos que calculan rápidamente las longitudes mínimas de cables y canaletas para el tendido de una red de cableado estructurado entre dos laboratorios del programa de ingeniería electrónica de la Universidad de Ibagué. Los algoritmos se desarrollaron utilizando tres métodos principales que son: el método de programación lineal, búsqueda exhaustiva y Dijkstra **[1]**, todos basados en el método del camino mínimo o ruta más corta. Se procedió primero a aplicarlos de forma manual siguiendo las indicaciones paso a paso y posteriormente se implementó un programa en **MATLAB [2]** para el desarrollo de cada uno de ellos. Para la escogencia del programa más eficiente se utilizó **SPSS [3]** para el análisis estadístico de los tiempos de procesamiento de cada programa en una serie de 10 iteraciones para cada uno puesto que la red era muy pequeña para que se mostraran resultados diferentes en cuanto a la eficacia o valores de Z resultantes. Los resultados muestran que para una red pequeña como sobre la cual se trabajó el mejor programa fue el basado en el algoritmo de Dijkstra. Adicionalmente, se utilizó **WinQSB** para procesar los datos mediante programación lineal y de forma tabular.

Palabras clave: *Programación lineal, búsqueda exhaustiva, Dijkstra, SPSS, MATLAB, WinQSB, PASW.*

ABSTRACT

This paper describes three algorithms that quickly calculate the minimum lengths of gutters and cables for the laying of a network wiring and cables structured between two laboratories in the program of electronic engineering at the University of Ibagué. Algorithms were developed using three major methods are: linear programming, comprehensive search and Dijkstra, all based on the shortest path or shortest route method. First proceeded to apply them manually by following the step-by-step directions and subsequently implemented a **MATLAB** program for the development of each one. The choice of the most efficient program was used for **SPSS** for the statistical analysis of each program processing times in a series of 10 iterations for each one of them because the network was very small for obtaining different results in terms of efficiency or resulting Z values. The results show that for a small network on which worked the best program was based on Dijkstra algorithm. In addition, **WinQSB** was used to process data using linear programming and tabular form.

Keywords: *Linear programming, comprehensive search, Dijkstra, SPSS, MATLAB, WinQSB, PASW.*

INTRODUCCIÓN

Uno de los problemas engorrosos y molestos al que se tiene que hacer frente en el diseño de redes de cableado estructurado es la escogencia de una ruta corta de cableado en la cual se necesite de menos cantidad de cable y también de canaletas por las cuales tender este mismo. Para atacar este problema se pueden implementar distintos algoritmos que brindan respuestas basadas en el camino mínimo o la ruta más corta. Debido a la cantidad de nodos en la red a trabajar y a las distancias entre nodos se decidió trabajar utilizando tres métodos básicos que fueron el de programación lineal (PL), la búsqueda o enumeración exhaustiva (BE) y el método de Dijkstra. Cada uno de estos métodos permite por sí mismo hallar una solución ideal en cuánto al recorrido más corto necesitándose para ello menor cantidad de material pero para poder hacer una elección subjetiva en cuanto a cuál de los tres métodos es más eficiente para la aplicación se decide tomar en cuanto la eficiencia de cada uno de ellos en términos de tiempos de computación para cada iteración. Basado en esto como primer paso se toman los tres algoritmos y se les aplica un desarrollo manual paso a paso según lo sugiere cada uno de los métodos, esto con la finalidad de conocer el fundamento exacto de funcionamiento de todos ellos. Posteriormente y conociendo su funcionamiento se creó un programa basado en **MATLAB** (matrix lab.) para cada uno basando PL en el vecino más cercano. Estos programas debían tener las siguientes características: ser amigable con el usuario, que se pudiera variar sobre él sin mayores complicaciones, que se adaptara a redes con más nodos y diferente configuración y que permitiera la visualización del tiempo de computación necesario para cada una de las corridas de los programas.

Posterior a esto se necesitaba utilizar una herramienta que nos permitiera escoger entre los tres programas y métodos cuál era el mejor para la aplicación en cuanto al uso de recursos. Para este caso optamos por utilización de menores tiempos de computación. Para este objeto se utilizó el programa Statistical Package for the Social Sciences **(SPSS**), que permite hacer un análisis estadístico de las variaciones en estos tiempos de procesamiento y escoger entre los tres la mejor opción mediante el uso de herramientas estadísticas confiables y de gran precisión. Para hacer más preciso el análisis de los datos y obtener y aprehender el conocimiento de nuevas herramientas computacionales desarrollamos los métodos de programación lineal y tabular usando **WinQSB**.

MÉTODOS

Desarrollo manual. Como se mencionó anteriormente, los métodos escogidos fueron tres: PL, BE y Dijkstra. A continuación se describen los métodos y luego el desarrollo manual mediante cada uno de ellos.

1. Programación lineal. Es un procedimiento o algoritmo matemático mediante el cual se resuelve un problema indeterminado, formulado a través de ecuaciones lineales, optimizando la función objetivo, también lineal.

Consiste en optimizar (minimizar o maximizar) una función lineal, denominada función objetivo, de tal forma que las variables de dicha función estén sujetas a una serie de restricciones que expresamos mediante un sistema de inecuaciones lineales.

Para el caso específico de este trabajo el procedimiento desarrollado para este método de programación lineal fue el siguiente:

- Análisis de la distribución sobre planos de los laboratorios de ingeniería electrónica como se ve en la **Figura 1**.

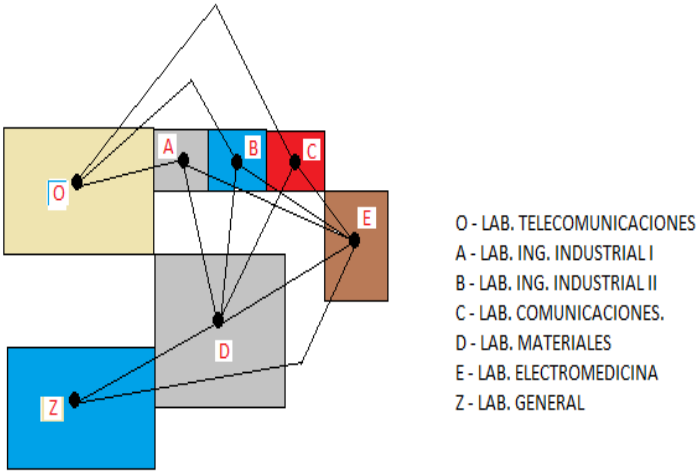


Figura 1. Distribución de los laboratorios de ingeniería.

- Se define la variable binaria de la siguiente manera:

$$X_{i,j} = \left\{ \begin{matrix} 0,\ arco\ i \to j \notin a\ la\ ruta\ más\ corta \\ 1,\ arco\ i \to j \notin a\ la\ ruta\ más\ corta \end{matrix} \right\}$$

- Trazado del grafo correspondiente al plano como se ve en la **Figura 2**.

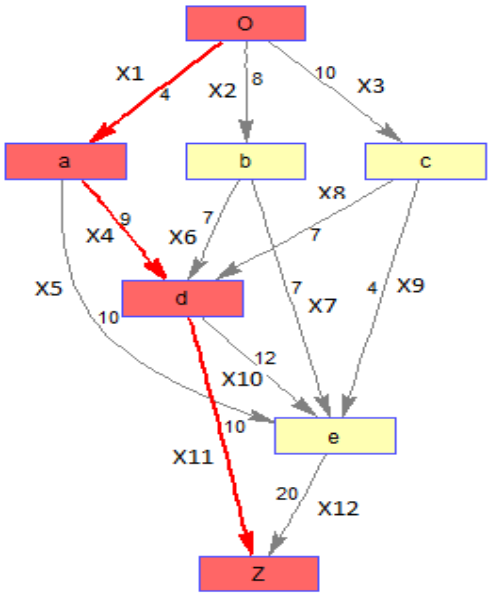


Figura 2. Grafo del trazado de los laboratorios.

Entonces la ecuación a minimizar será la siguiente:

$$\textbf{Min } Z = 4X1 + 8X2 + 10X3 + 9X4 + 10X5 + 7X6 + 7X7 + 7X8 + 4X9 + 12X10 + 10X11 + 20X12$$

- Posteriormente se asignan los posibles valores que tomará la variable al minimizar:

$$X_j = \left\{ \begin{matrix} 0,\ no\ asigna\ el\ arco\ j-ésimo\ \ la\ ruta\ más\ corta \\ 1,\ \ asigna\ el\ arco\ j-ésimo\ \ a\ la\ ruta\ más\ corta \end{matrix} \right\}$$

- Análisis de entrada y salida de rutas a cada uno de los nodos.

O) X1+X2+X3=1

a) X1=X4+X5

b) X2=X6+X7

c) X3=X8+X9

d) X11=X4+X6+X8-X10

e) X12=X5+X7+X9+X10

Z) X11+X12=1

Quedando entonces:

X4+X5-X1=0

X6+X7-X2=0

X8+X9-X3=0

X4+X6+X8+X10-X11=0

X5+X7+X9+X10-X12=0

Del análisis de las ecuaciones anteriores entonces nos queda que:

X1, X4, X11=1 y **X2, X3, X4, X6, X7, X8, X9, X10, X12**=0

De donde

Min Z=4(1)+9(1)+10(1)

➔**Z=** 23

Luego el recorrido mínimo es: **O – a – d – Z**

2. Búsqueda exhaustiva. Consiste en la enumeración y trazado de todas las posibles soluciones de la red mediante el seguimiento desde el primer nodo y realizando la suma de las distancias entre cada uno de ellos conectado con el siguiente. Este algoritmo requiere de gran cantidad de memoria y puede ser poco útil para redes de gran tamaño.

Para la solución de este algoritmo tenemos en cuenta tanto el plano de la **Figura 1** como el grafo de este de la **Figura 2.**

Luego enumeramos cada uno de los posibles caminos de solución de nuestro sistema y sumamos las distancias entre los nodos:

O – a – e – d - Z = 36

O – a – d – e – Z =35

O – a – d – Z = 23 <=

O – a – e – Z = 34

O – b – d – e – Z = 47

O – b - e – d – Z = 37

O – b – e – Z = 45

O – b – d – Z = 25

O – c – e – d – Z = 36

O – c – d – e – Z = 49

O – c – d – Z = 27

O – c – e – Z = 34

Luego de trazar todos los caminos posibles y realizar las sumas por simple inspección nos damos cuenta que el camino más corto y la ruta a seguir serán:

O – a – d – Z = 23 <=

3. Algoritmo de Dijkstra. El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo dirigido y con pesos en cada arista. La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene. El algoritmo es una especialización de la búsqueda de costo uniforme, y como tal, no funciona en grafos con aristas de costo negativo (al elegir siempre el nodo con distancia menor, pueden quedar excluidos de

la búsqueda nodos que en próximas iteraciones bajarían el costo general del camino al pasar por una arista con costo negativo).

Para llevar a cabo este algoritmo seguimos los siguientes pasos.

- Se retoma la **Figura 2** para tener en cuenta las distancias entre nodos.
- Se realizan las diferentes iteraciones utilizando la fórmula:

E(j)={(anterior etiqueta temporal), (última etiqueta temporal + distancia que pasó de etiqueta permanente al **j**)}

- De cada iteración se escoge el menor valor.
- Este menor valor escogido de cada iteración se va ubicando como se ve en la **Tabla 1** a continuación.

TABLA 1. Valores de las iteraciones y respuesta al camino mínimo por Dijkstra.

I	**e**	**a**	**b**	**c**	**d**	**e**	**Z**	**jmin**
0	**0**	**4**	8	10	∞	∞	∞	a
1	**4**	-	**8**	10	13	14	∞	b
2	**8**	-	-	**10**	13	14	∞	c
3	**10**	-	-	-	**13**	14	∞	d
4	**13**	-	-	-	-	**14**	23	e
5	**14**	-	-	-	-	-	**23**	Z
Ej		4	8	10	13	14	**23**	

A continuación mostramos las iteraciones realizadas para este problema.

I1= e(b)=8, 4+∞ = 8

e(c)=10, 4+∞ = 10

e(d)= ∞, 4+9 = 13

e(e)= ∞, 4+10 = 14

e(Z)= ∞, 4+∞= ∞

I2= e(c)=10, 8+V= 10

e(d)=13, 8+7= 13

e(e)=14, 8+7= 14

e(Z)= ∞, 8+∞= ∞

I3= e(d)=13, 10+7= 13

e(e)=14, 10+4= 14

e(Z)= ∞, 10+∞= ∞

I4= e(e)=14, 13+12= 14

e(Z)= ∞, 13+10=23

I5= e(Z)=23, 14+20= 23

- Luego de realizar las iteraciones y llenar la tabla buscamos la forma de comprobar el camino más corto devolviéndonos desde Z hasta el nodo de origen de la siguiente manera:

E(Z) – E(d)= 23-13= 10

E(Z) – E(e)= 23-14= 9

- Vemos cuál de los dos resultados coincide con el valor de la distancia entre esos nodos, vemos que éste es entre "Z y d" y seguimos desde el nodo d:

E(d) – E(a)= 13-4= 9

E(d) – E(b)= 13-8= 5

E(d) – E(c)= 13-10= 3

- Hacemos lo mismo y vemos que la distancia que coincide es entre "d y a".
- Luego de "a hasta O" sólo hay un camino por lo que podemos decir que la solución es:

Z 10 d 9 a 4 O=>23, ó

O – a – d – Z, con Z= 23

Se puede ver que con todos los métodos el resultado de Z es el mismo y como es una red sencilla siempre que se apliquen los mismos algoritmos se podrá observar el mismo resultado, es por esto que la variable de decisión elegida para determinar cuál de todos estos métodos es el que más conviene no fue el valor de Z sino el tiempo de computación o procesamiento de cada uno de estos algoritmos en **MATLAB**.

Desarrollo en MATLAB. Mediante el uso de **MATLAB** se crearon programas específicos para tres algoritmos: vecino más cercano, búsqueda exhaustiva y Dijkstra. A continuación se presentan los códigos y se da una pequeña explicación de cada uno. Es de anotar, además, que para la creación de estos códigos se utilizaron librerías de MATLAB y se depuró el código para adaptarlo a las necesidades específicas del problema. Esto se hizo así para el algoritmo de Dijkstra y para el de búsqueda exhaustiva, el del vecino más cercano fue creación absoluta de los integrantes del grupo de trabajo.

En el documento anexo se incluyen todos los programas completos en **MATLAB** y además todas las explicaciones paso a paso de cómo fueron creados estos.

1. Vecino más cercano (para programación lineal)

```
% ECUACIÓN PROPIA


MINZ=4X1+8X2+10X3+9X4+10X5+7X6+7X7+7X8+4X9+12X10+10X11+12X12
```

Para hacerlo más amigable al usuario se pueden ingresar los valores mediante teclado para además ver variaciones en los resultados con el cambio de los valores de las ramas.

Instrucciones de limpiado de pantalla. Se utilizan en los tres programas.

```
clc;
clear;

% ENTRADA DE LAS DISTANCIAS ENTRE NODOS

X1 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 1 - 2 :   ');
X2 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 1 - 3 :   ');
X3 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 1 - 4 :   ');
X4 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 2 - 5 :   ');
X5 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 2 - 6 :   ');
X6 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 3 - 5 :   ');
X7 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 3 - 6 :   ');
X8 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 4 - 5 :   ');
X9 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 4 - 6 :   ');
X10 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 5 - 6 :   ');
X11 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 5 - 7 :   ');
X12 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 6 - 7 :   ');

% X1+X2+X3=1
 Se utiliza el tic-toc para  medir los tiempos de procesamiento del
programa durante cada corrida. Se ubica el tic al inicio de donde se
quiere comenzar a sensar el programa y el toc al final del mismo.

tic

if (X1<X2)&&(X1<X3)
    msgbox('EL CAMINO MÍNIMO ES X1')
    A=X1
elseif (X2<X1)&&(X2<X3)
    msgbox('EL CAMINO MÍNIMO ES X2')
    A=X2
elseif (X3<X2)&&(X3<X1)
    msgbox('EL CAMINO MÍNIMO ES X3')
    A=X3
end

% X4+X5-X1=0

if (X4<X5)
    msgbox('EL CAMINO MÍNIMO ES X4')
    B=X4
elseif (X5<X4)
    msgbox('EL CAMINO MÍNIMO ES X5')
    B=X5
```

```
end

% X6+X7-X2=0

if (X6<=X7)
    msgbox('EL CAMINO MÍNIMO ES X6')
    C=X6
elseif (X7<X6)
    msgbox('EL CAMINO MÍNIMO ES X7')
    C=X7
end

% X8+X9-X3=0

if (X8<X9)
    msgbox('EL CAMINO MÍNIMO ES X8')
    D=X8
elseif (X9<X8)
    msgbox('EL CAMINO MÍNIMO ES X9')
    D=X9
end

% X4+X6+X8-X10-X11=0

F=X12

if (X10+X12)<X11
    E=X10+X12
elseif(X10+X12)>X11
    E=X11
end


if (A==X1)&&(B==X4)&&((X10+X12<X11))
    msgbox('EL  CAMINO MÍNIMO ES 0 - A - D - E - Z')
    G=A+B;
    H=G+E;
elseif (A==X1)&&(B==X4)&&((X10+X12>X11))
    msgbox('EL  CAMINO MÍNIMO ES 0 - A - D - Z')
    G=A+B;
    H=G+E;
elseif (A==X1)&&(B==X5)
    msgbox('EL  CAMINO MÍNIMO ES 0 - A - E - Z')
    G=A+B;
    H=G+F;
elseif (A==X2)&&(C==X6)&&((X10+X12<X11))
    msgbox('EL  CAMINO MÍNIMO ES 0 - B - D - E - Z')
    G=A+C;
    H=G+E;
elseif (A==X2)&&(C==X6)&&((X10+X12>X11))
    msgbox('EL  CAMINO MÍNIMO ES 0 - B - D - Z')
    G=A+C;
    H=G+E;
elseif (A==X2)&&(C==X7)
    msgbox('EL  CAMINO MÍNIMO ES 0 - B - E - Z')
    G=A+C;
    H=G+F;
elseif (A==X3)&&(D==X8)&&((X10+X12<X11))
```

```
    msgbox('EL  CAMINO MÍNIMO ES 0 - C - D - E - Z')
    G=A+D;
    H=G+E;
elseif (A==X3)&&(D==X8)&&((X10+X12>X11))
    msgbox('EL  CAMINO MÍNIMO ES 0 - C - D - Z')
    G=A+D;
    H=G+E;
elseif (A==X3)&&(D==X9)
    msgbox('EL  CAMINO MÍNIMO ES 0 - C - E - Z')
    G=A+D;
    H=G+F;
end

MINIMOZ=sparse(H)

toc
```

Este programa se desarrolla mediante comparaciones sucesivas entre los valores entre nodos y eliminando las variables nulas y al final sumando las restantes.

El resultado para este problema con el camino mínimo, el valor de Z y el tiempo de procesamiento se ve en la **Figura 3** como aparece en **MATLAB**.

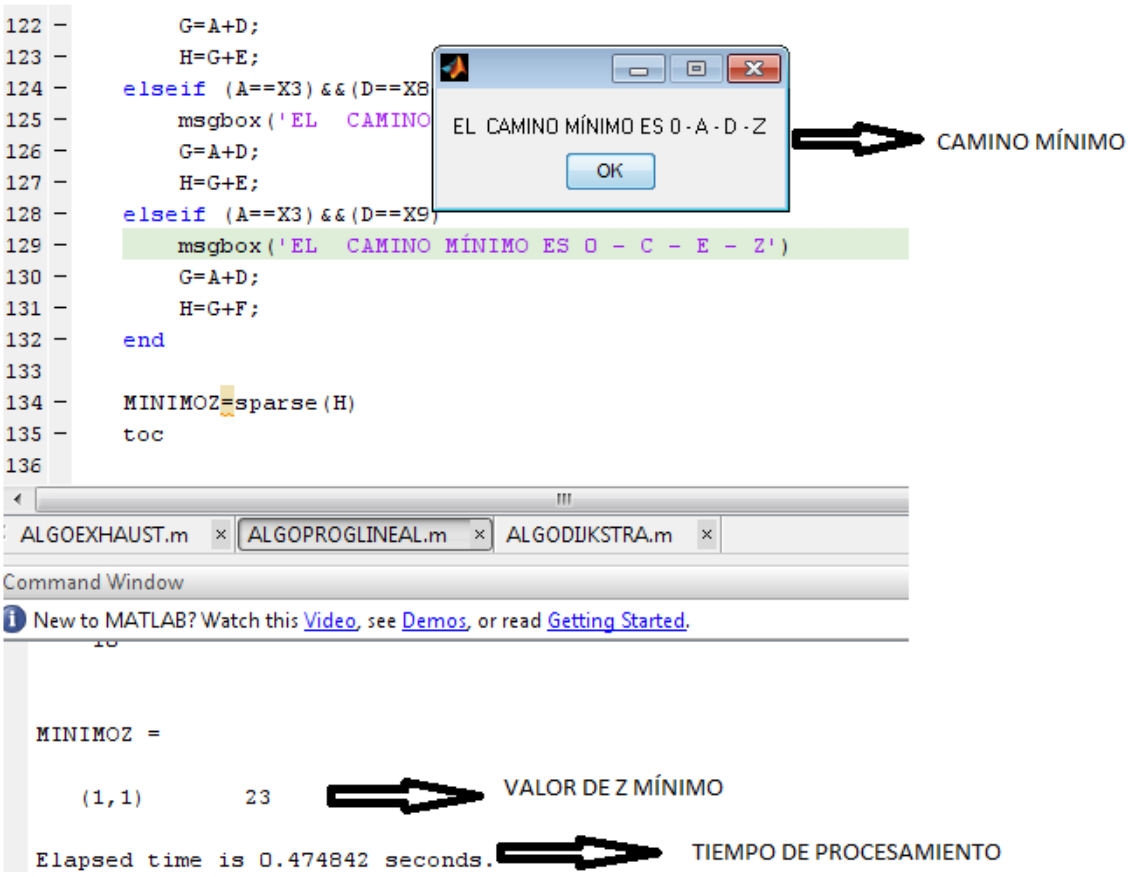


FIGURA 3. Pantalla de **MATLAB** que muestra el resultado para PL.

2. Búsqueda exhaustiva.

```
clc;
clear;

% INTRODUCCIÓN Y EQUIVALENCIA DE NODOS

O=1;
A=2;
B=3;
C=4;
D=5;
E=6;
Z=7;

% INGRESO DEL NÚMERO DE CONEXIONES QUE TIENE CADA NODO

C1 =input('INGRESE EL PRIMER NODO CONECTADO AL NODO 1:       ');
C2 =input('INGRESE EL SEGUNDO NODO CONECTADO AL NODO 1:      ');
C3 =input('INGRESE EL TERCER NODO CONECTADO AL NODO 1:       ');
C4 =input('INGRESE EL PRIMER NODO CONECTADO AL NODO 2:       ');
C5 =input('INGRESE EL SEGUNDO NODO CONECTADO AL NODO 2:      ');
C6 =input('INGRESE EL PRIMER NODO CONECTADO AL NODO 3:       ');
C7 =input('INGRESE EL SEGUNDO NODO CONECTADO AL NODO 3:      ');
C8 =input('INGRESE EL PRIMER NODO CONECTADO AL NODO 4:       ');
C9 =input('INGRESE EL SEGUNDO NODO CONECTADO AL NODO 4:      ');
C10=input('INGRESE EL PRIMER NODO CONECTADO AL NODO 5:       ');
C11=input('INGRESE EL SEGUNDO NODO CONECTADO AL NODO 5:      ');
C12=input('INGRESE EL PRIMER NODO CONECTADO AL NODO 6:       ');

% ENTRADA DE LAS DISTANCIAS ENTRE NODOS

X1 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 1 - 2 :    ');
X2 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 1 - 3 :    ');
X3 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 1 - 4 :    ');
X4 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 2 - 5 :    ');
X5 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 2 - 6 :    ');
X6 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 3 - 5 :    ');
X7 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 3 - 6 :    ');
X8 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 4 - 5 :    ');
X9 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 4 - 6 :    ');
X10 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 5 - 6 :    ');
X11 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 5 - 7 :    ');
X12 =input('INGRESE LA DISTANCIA ENTRE LOS NODOS 6 - 7 :    ');

% SE UTILIZA EL NOMBRE "cm" PARA CREAR LA MATRIZ CON LA CUAL VAMOS A
TRABAJAR
% COMO PRIMER VECTOR SE INTRODUCEN EL NÚMERO DE SALIDAS QUE TIENE CADA
NODO
% MEDIANTE LA REITERACIÓN DEL NÚMERO DEL NODO DEL QUE PARTIMOS Y ASÍ
UNO A
% CONTINUACIÓN DEL OTRO.
% LUEGO COMO OTRO VECTOR SE UBICAN LOS NODOS A LOS QUE SE DIRIGEN CADA
UNA
% DE LAS SALIDAS DEL ANTERIOR.
% Y POSTERIORMENTE, SE AGREGA OTRO VECTOR CON EL PESO (DISTANCIA DEL
CABLE
% EN NUESTRO CASO) O LONGITUD DE LA CONEXIÓN ENTRE LOS NODOS.
```

```
% FINALMENTE SE DECLARA HASTA QUÉ NODO SE DESEA QUE SE REALICE EL
PROCESO.

% SE UTILIZA tic - toc PARA MARCAR EL PRINCIPIO Y EL FINAL DEL PROCESO
AL
% QUE SE LE QUIERE MEDIR EL TIEMPO DE PROCESAMIENTO, tic AL INICIAR Y
toc
% AL FINALIZAR.

tic


% SE UTILIZA LA INSTRUCCIÓN "sparse" PARA EXTRAER LOS ELEMENTOS NULOS
DE LA
% MATRIZ Y HACERLA MÁS COMPACTA PARA HALLAR LAS DISTANCIAS ENTRE
NODOS.

cm = sparse([O O O A A B B C C D D E],[C1 C2 C3 C4 C5 C6 C7 C8 C9 C10
C11 C12],[X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12],7,7)

% LUEGO DE CREAR LA MATRIZ SE PROCEDE A LA UBICACIÓN DE LAS DISTANCIAS
% ENTRE LOS NODOS CONECTADOS ENTRE SÍ Y HACIENDO TODAS LAS SUMAS QUE
SON
% PERMITIDAS. LO ANTERIOR SE HACE MEDIANTE LA INSTRUCCIÓN "dist" QUE
ES LA
% QUE HALLA LAS DISTANCIAS EUCLIDIANAS EN LA MATRÍZ.
% LUEGO LA MISMA INSTRUCCIÓN UBICA EL VALOR MÍNIMO RECORRIDO Y LO
% DESPLIEGA EN PANTALLA COMO UNA LISTA DE LOS NODOS INTERCONECTADOS Y
SUS
% VALORES ENTRE SÍ.

% POSTERIORMENTE LA INSTRUCCIÓN "path" RECORRE LOS NODOS POR LOS
CUALES SE
% HIZO ESTA SUMA Y OBTIENE EL RESULTADO DEL CAMINO SEGUIDO.

% FINALMENTE MEDIANTE "graphshortestpath" AGRUPANDO "dist" Y "path"
COMO
% UNA MATRÍZ ([dist,path]) SE VISUALIZA EL VALOR DE LA DISTANCIA
MÍNIMA
% RECORRIDA Y LOS NODOS QUE SE SIGUIERON PARA CONSEGUIR ÉSTA DESDE EL
% ORÍGEN HASTA EL PUNTO QUE SE QUIERA SENSAR. ESTO SE HACE UBICANDO
LUEGO
% DE graphshortestpath(nombre de la matriz, nodo de origen, nodo de
destino)

[dist,path] = graphshortestpath(cm,O,7)

% LA INSTRUCCIÓN SIGUIENTE PERMITE VISUALIZAR EL GRAFO CON LOS NODOS Y
% PESOS DE CADA RAMA.

h = view(biograph(cm,[],'ShowWeights','on'))

% TRAZA LA RUTA ESCOGIDA EN EL GRAFO Y MARCA CON COLOR LOS NODOS Y LOS
% CAMINOS ELEGIDOS.

    set(h.Nodes(path),'Color',[1 0.4 0.4])
    edges = getedgesbynodeid(h,get(h.Nodes(path),'ID'));
```

```
    set(edges,'LineColor',[1 0 0])
    set(edges,'LineWidth',1.5)

% SE ANOTA QUE ESTE PROGRAMA SE USA PARA GRAFOS DIRIGIDOS PUESTO QUE
PARA
% NO DIRIGIDOS SE UTILIZA OTRO CÓDIGO.
% SE ESCOGIÓ DIRIGIDO PUESTO QUE LA RUTA DE COMUNICACIÓN ES
% UNIDIRECCIONAL ENTRE EL LAB. DE TELECOMUNICACIONES Y EL GENERAL.
toc
```

Al final se pueden observar también el tiempo, la distancia y el sendero escogidos en la **Figura 4** y en la **Figura 5** el grafo y el camino mínimo coloreado.

```
dist =

    23


path =

     1     2     5     7

Biograph object with 7 nodes and 12 edges.
Elapsed time is 0.564697 seconds.
>>
```

FIGURA 4. Nodos escogidos para el camino y tiempo de computación o procesamiento del programa.

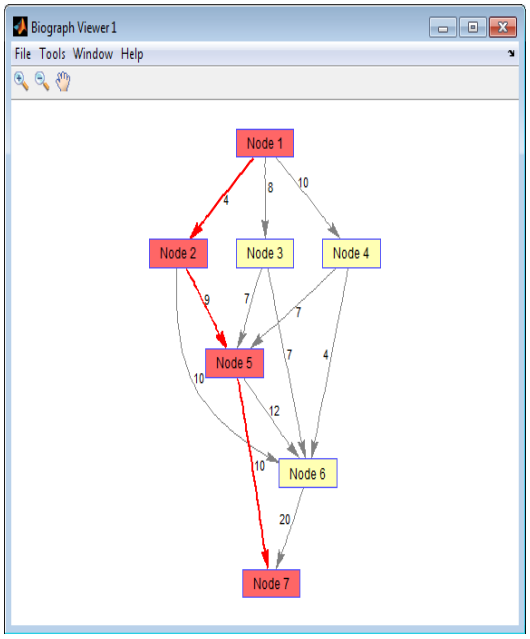


FIGURA 5. Pantalla de **MATLAB** en el resultado del algoritmo de búsqueda exhaustiva.

3. Algoritmo de Dijkstra.

```
% INGRESO DE VALORES DE LA MATRÍZ.
% SE INGRESAN LAS DISTANCIAS ENTRE C/U DE LOS NODOS. DEBERÁ HACERSE
% EN FORMA DE VECTORES SEPARADOS POR PUNTO Y  COMA DE LA SIGUIENTE
% MANERA PARA UNA MATRÍZ DE 3x3, A= [1 2 3; 2 4 5; 2 4 5]
% SI EL NODO ESTÁ CONECTADO CONSIGO MISMO O NO TIENE CONEXIÓN CON EL
% SIGUIENTE NODO SE DEBERÁ PONER CERO (0) EN ESA POSICIÓN

A=input('INGRESE LOS VALORES DE LA MATRÍZ:  ');

% SE UTILIZA tic - toc PARA MARCAR EL PRINCIPIO Y EL FINAL DEL PROCESO
AL
% QUE SE LE QUIERE MEDIR EL TIEMPO DE PROCESAMIENTO, tic AL INICIAR Y
toc
% AL FINALIZAR.

tic

% SE UTILIZA LA INSTRUCCIÓN "sparse" PARA EXTRAER LOS ELEMENTOS NULOS
DE LA
% MATRIZ Y HACERLA MÁS COMPACTA PARA HALLAR LAS DISTANCIAS ENTRE
NODOS.

G=sparse(A)

% SE INICIALIZAN LAS MATRICES PARA NODOS VISITADOS, CAMINO MÍNIMO
(path) Y
% DISTANCIA EUCLIDIANA (dist) EN CEROS TODOS LOS ELEMENTOS.

visitados=[0 0 0 0 0 0 0];
path=[0 0 0 0 0 0 0];
dist=[0 0 0 0 0 0 0];

% LA MATRÍZ VISTA ES COMO SE MUESTRA A CONTINUACIÓN:

%   1   2   3   4   5   6   7
% 1 0   4   8   10  0   0   0
% 2 0   0   0   0   9   10  0
% 3 0   0   0   0   7   7   0
% 4 0   0   0   0   7   4   0
% 5 0   0   0   0   0   12  10
% 6 0   0   0   0   0   0   20
% 7 0   0   0   0   0   0   0

% MEDIANTE sparse SE ELIMINAN LOS ELEMENTOS NULOS DE LA MATRÍZ Y SE VE
ASÍ:

%   1   2   3   4   5   6   7
% 1     4   8   10
% 2                 9   10
% 3                 7   7
% 4                 7   4
% 5                     12  10
% 6                         20
% 7
```

```
% SE COMIENZA LUEGO A ANALIZAR DESDE LA POSICIÓN 1,1 PARA LAS
DISTANCIAS
% ENTRE NODOS ESCOGIENDO EL MÍNIMO DE LA MANERA EN LA QUE SE VE EN LA
% SECUENCIA;

% - SE EMPIEZA DESDE LA FILA 1 Y SE BUSCA EL ELEMENTO DE MENOR VALOR.
% - LUEGO SE UBICA EL VALOR DE LA COLUMNA EN QUE ESTABA.
% - ESTE VALOR SE COGE Y SE UBICA DESDE LA FILA DEL MISMO VALOR Y SE
ESCOGE
%   DE NUEVO EL MÍNIMO ENTRE LOS VALORES ALLÍ UBICADOS.
% - SE VUELVE A UBICAR LA COLUMNA DE ESTE MÍNIMO VALOR Y SE VUELVE A
TOMAR
%   EL MISMO DE LAS COLUMNAS.
% - SE ESCOGE EL MENOR Y SE SIGUE CON EL MISMO PROCEDIMIENTO HASTA
LLEGAR
%   AL PUNTO ELEGIDO PARA FINALIZAR.
% - ESTOS VALORES ELEGIDOS COMO MÍNIMOS SE SUMAN Y ESTE SERÁ EL VALOR
DE
%   "Z"
% - LOS NODOS VISITADOS SE OBTIENEN DE LOS VALORES DE LAS COLUMNAS EN
LAS
%   CUALES ESTABAN ESTOS VALORES MÍNIMOS.

%   1   2   3   4   5   6   7
% 1     4   8   10
% 2                 9   10
% 3                 7   7
% 4                 7   4
% 5                     12  10
% 6                         20
% 7                         0


% ESTE PROCESO ES EL QUE SE DESARROLLA POR MEDIO DE LOS CICLOS for E
if QUE
% SIGUEN A CONTINUACIÓN.
inicio=1;
visitados(inicio)=1;
dest=7;

for i=1:length(G)
    d(i)=G(inicio,i);
    path(i)=inicio;
end

for i=1:length(G)
   min = inf;
   u=1;
   for j=1:length(G)
       if visitados(j)==0
           if d(j)<min
                min = d(j);
                u = j;
           end
       end
   end


visitados(u)=1;
```

```
      for v=1:length(G)
         if visitados(v)==0
             if ((d(u)+G(u,v))<d(v))
                 d(v)=d(u)+G(u,v);
                 path(v)=u;
             end
         end
      end
end

% MEDIANTE "graphshortestpath" AGRUPANDO "dist" Y "path" COMO
% UNA MATRÍZ ([dist,path]) SE VISUALIZA EL VALOR DE LA DISTANCIA
MÍNIMA
% RECORRIDA Y LOS NODOS QUE SE SIGUIERON PARA CONSEGUIR ÉSTA DESDE EL
% ORÍGEN HASTA EL PUNTO QUE SE QUIERA SENSAR. ESTO SE HACE UBICANDO
LUEGO
% DE graphshortestpath(nombre de la matriz, nodo de origen, nodo de
% destino)

[dist,path] = graphshortestpath(G,1,7)

% SE UTILIZA ESTA INSTRUCCIÓN PARA OBSERVAR EL GRAFO

h = view(biograph(G,[],'ShowWeights','on'))

% ESTAS INSTRUCCIONES PERMITEN VISUALIZAR SOBRE EL GRAFO EL CAMINO
ESCOGIDO,
% LOS NODOS,  COLOREARLOS Y DARLES EL ANCHO DE LA LÍNEA DEL TRAZO.

  set(h.Nodes(path),'Color',[1 0.4 0.4])
    edges = getedgesbynodeid(h,get(h.Nodes(path),'ID'));
    set(edges,'LineColor',[1 0 0])
    set(edges,'LineWidth',1.5)

toc
```

Podemos observar el tiempo, la distancia mínima o valor de Z y el camino desde el origen al final en la **Figura 6**.

```
dist =

    23


path =

     1     2     5     7

Biograph object with 7 nodes and 12 edges.
Elapsed time is 0.486583 seconds.
fx >>
```

FIGURA 6. Tiempo y nodos escogidos en Dijkstra.

La **Figura 7** muestra el grafo final con el camino mínimo marcado.

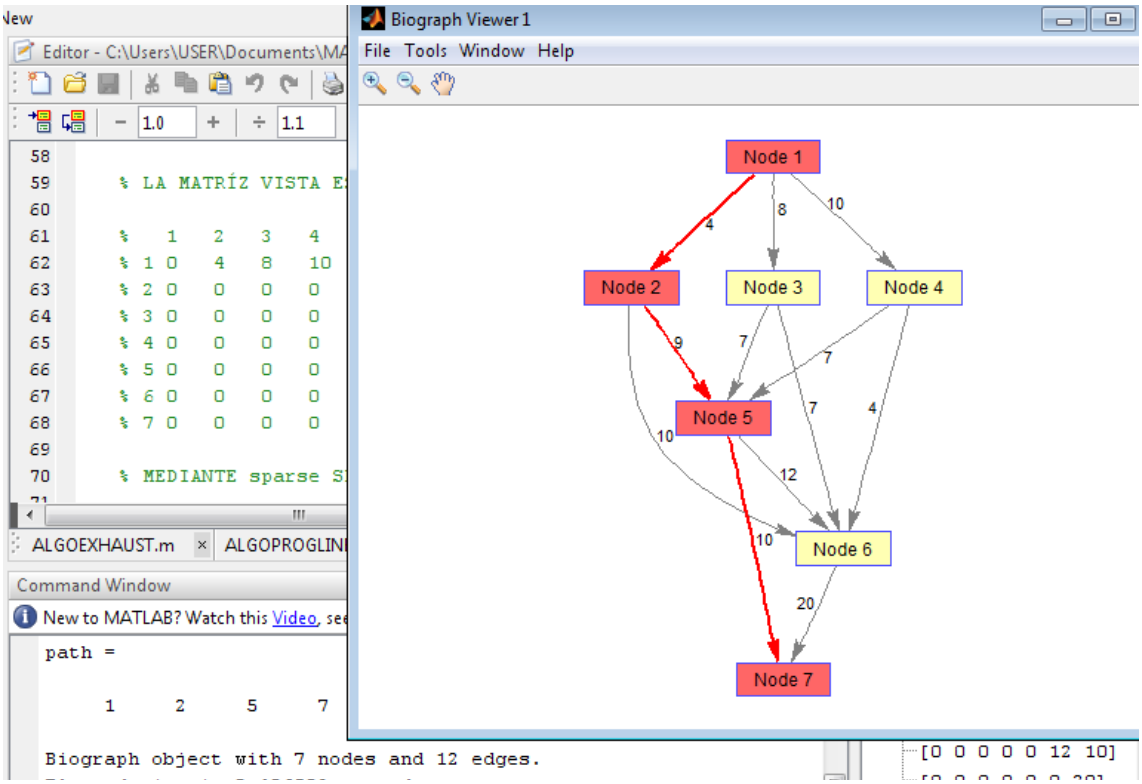
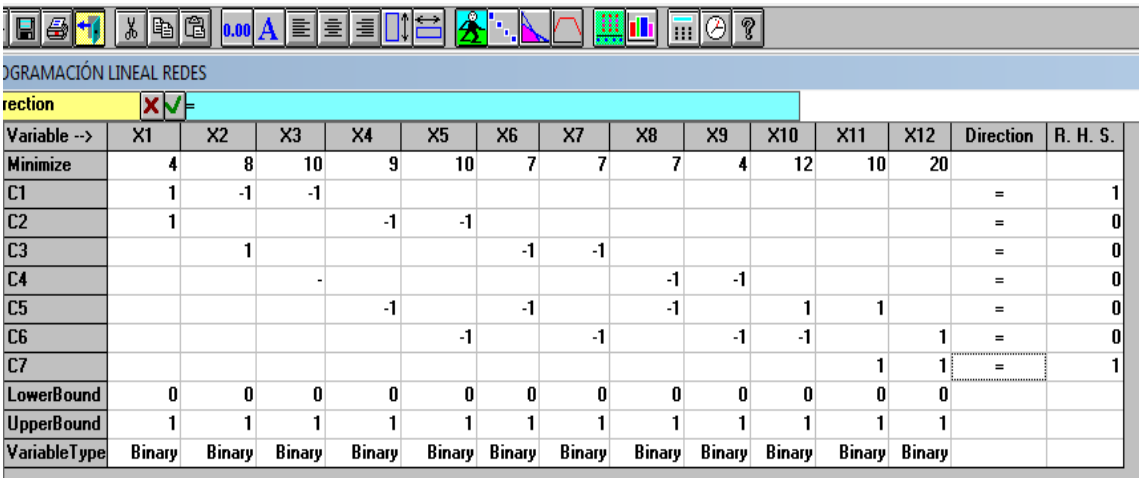


FIGURA 7. Pantalla de **MATLAB** en el resultado del algoritmo de Dijkstra que muestra el grafo y el camino escogido resaltado en color.

SIMULACIÓN EN WinQSB

WinQSB es un sistema interactivo de ayuda a la toma objetiva de decisiones que contiene herramientas muy útiles para resolver distintos tipos de problemas en el campo de la investigación de operaciones pero también se puede extrapolar para cualquier otro campo. El sistema está formado por distintos módulos, uno para cada tipo de modelo o problema. Entre ellos se eligió para simular aquí mediante programación lineal y mediante el método tabular. A continuación se mostrarán los distintos pasos para cada una de las dos.

En la **Figura 8** se ve la pantalla principal del menú para programación lineal. En este menú se ingresan las variables, las restricciones, los coeficientes de las variables y los valores binarios de las variables para cada restricción. En la **Figura 9** se puede ver el resultado arrojado por **WinQSB** en el que se pueden apreciar el valor mínimo de Z, las variables básicas, los límites superior e inferior y las variables que toman valor en esta ecuación.

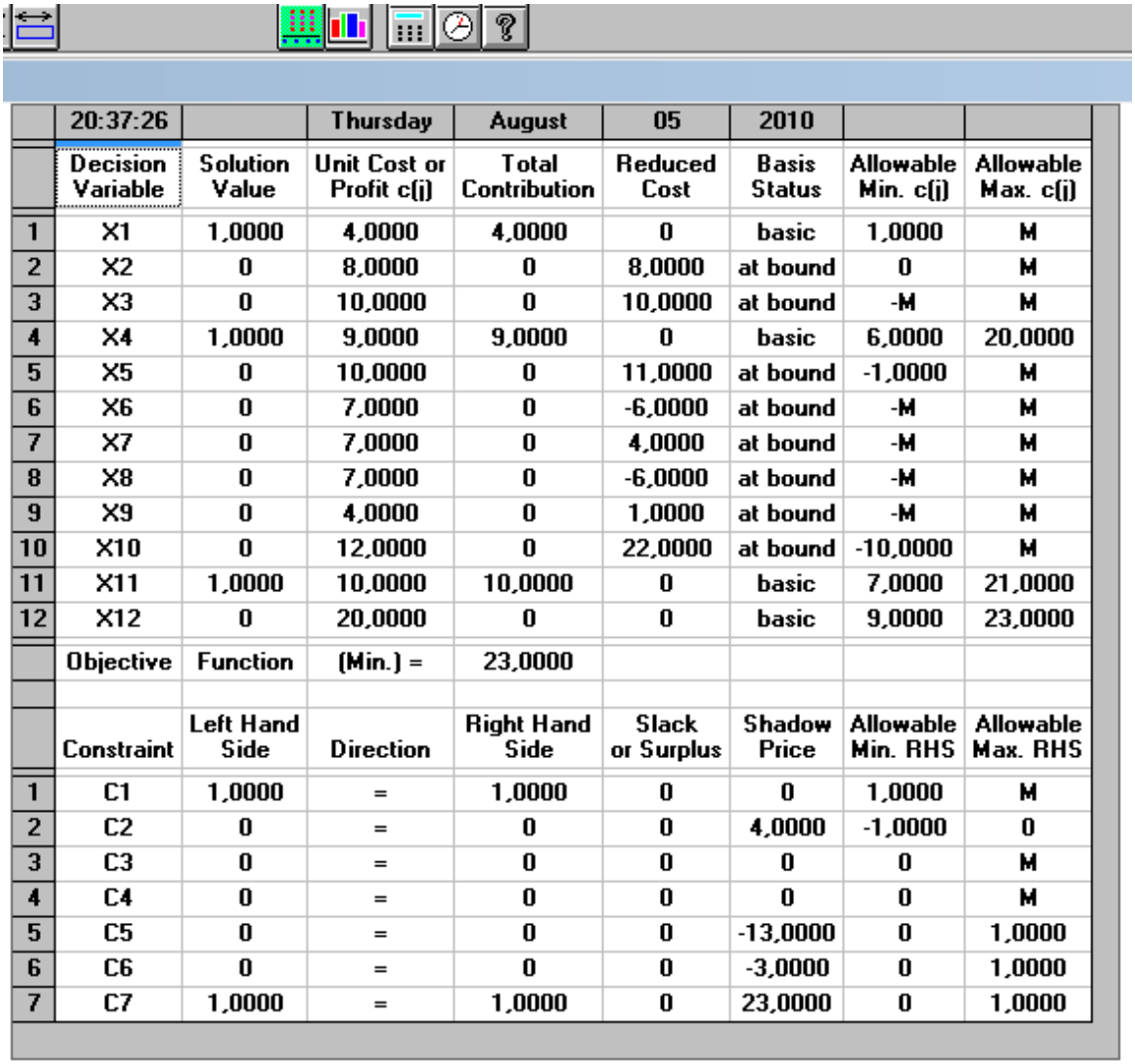


OGRAMACIÓN LINEAL REDES

rection =

Variable -->	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	Direction	R. H. S.
Minimize	4	8	10	9	10	7	7	7	4	12	10	20		
C1	1	-1	-1										=	1
C2	1			-1	-1								=	0
C3		1				-1	-1						=	0
C4			-					-1	-1				=	0
C5				-1		-1		-1		1	1		=	0
C6					-1		-1		-1	-1		1	=	0
C7											1	1	=	1
LowerBound	0	0	0	0	0	0	0	0	0	0	0	0		
UpperBound	1	1	1	1	1	1	1	1	1	1	1	1		
VariableType	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary		

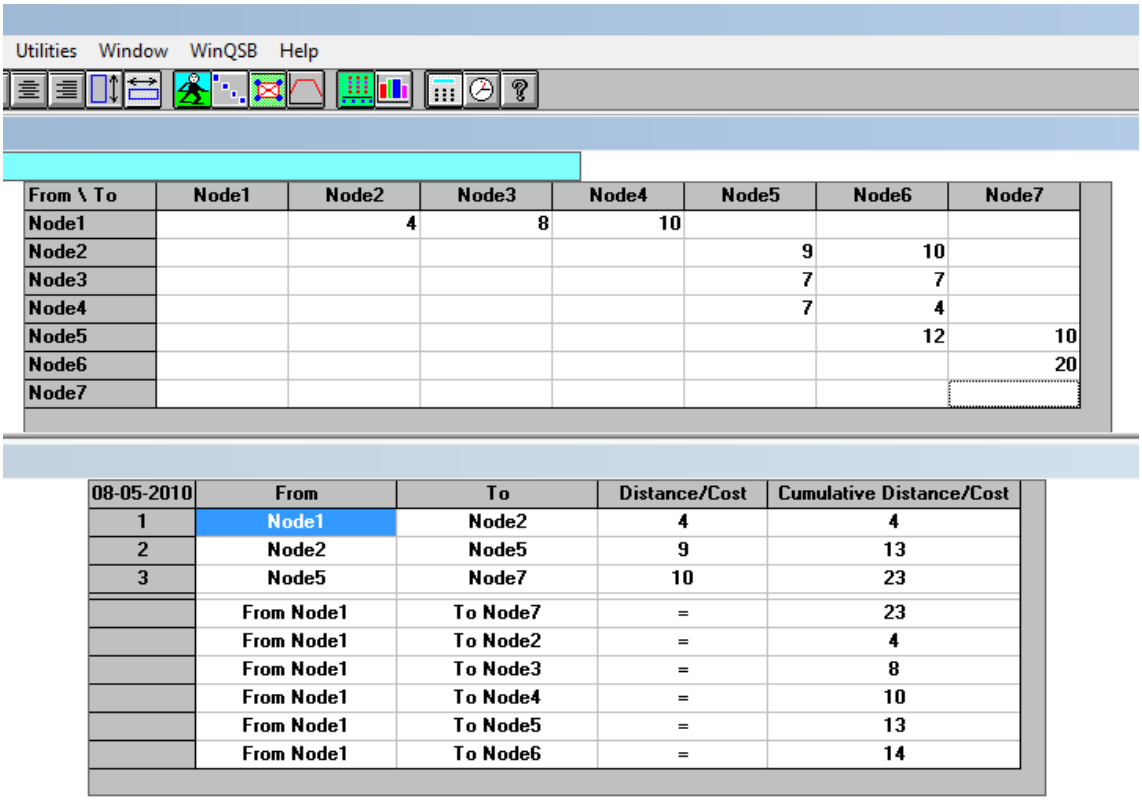
FIGURA 8. Pantalla del menú de **WinQSB** donde se deben ingresar los distintos valores de variables y restricciones.



	20:37:26		Thursday	August	05	2010		
	Decision Variable	Solution Value	Unit Cost or Profit c(j)	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c(j)	Allowable Max. c(j)
1	X1	1,0000	4,0000	4,0000	0	basic	1,0000	M
2	X2	0	8,0000	0	8,0000	at bound	0	M
3	X3	0	10,0000	0	10,0000	at bound	-M	M
4	X4	1,0000	9,0000	9,0000	0	basic	6,0000	20,0000
5	X5	0	10,0000	0	11,0000	at bound	-1,0000	M
6	X6	0	7,0000	0	-6,0000	at bound	-M	M
7	X7	0	7,0000	0	4,0000	at bound	-M	M
8	X8	0	7,0000	0	-6,0000	at bound	-M	M
9	X9	0	4,0000	0	1,0000	at bound	-M	M
10	X10	0	12,0000	0	22,0000	at bound	-10,0000	M
11	X11	1,0000	10,0000	10,0000	0	basic	7,0000	21,0000
12	X12	0	20,0000	0	0	basic	9,0000	23,0000
	Objective	Function	(Min.) =	23,0000				
	Constraint	Left Hand Side	Direction	Right Hand Side	Slack or Surplus	Shadow Price	Allowable Min. RHS	Allowable Max. RHS
1	C1	1,0000	=	1,0000	0	0	1,0000	M
2	C2	0	=	0	0	4,0000	-1,0000	0
3	C3	0	=	0	0	0	0	M
4	C4	0	=	0	0	0	0	M
5	C5	0	=	0	0	-13,0000	0	1,0000
6	C6	0	=	0	0	-3,0000	0	1,0000
7	C7	1,0000	=	1,0000	0	23,0000	0	1,0000

FIGURA 9. Tabla de resultados de la programación lineal en **WinQSB**.

Ahora se verá el **WinQSB** para el método tabular en la **Figura 10**. En esta pantallas se presenta el ingreso de datos de las distancias de los nodos válidos (no se ingresan las que valen cero o infinito) y luego la tabla de resultados donde se observa el valor mínimo de Z y el camino mínimo recorrido del origen al final.



From \ To	Node1	Node2	Node3	Node4	Node5	Node6	Node7
Node1		4	8	10			
Node2					9	10	
Node3					7	7	
Node4					7	4	
Node5						12	10
Node6							20
Node7							

08-05-2010	From	To	Distance/Cost	Cumulative Distance/Cost
1	Node1	Node2	4	4
2	Node2	Node5	9	13
3	Node5	Node7	10	23
	From Node1	To Node7	=	23
	From Node1	To Node2	=	4
	From Node1	To Node3	=	8
	From Node1	To Node4	=	10
	From Node1	To Node5	=	13
	From Node1	To Node6	=	14

FIGURA 10. Método tabular mediante **WinQSB**. Se muestran la pantalla inicial y la de solución del problema.

RESULTADOS

Para comenzar a aplicar el programa de elección del mejor método se siguen los siguientes pasos:

- Se usa una muestra piloto de m=9.
- Se calcula $n = \frac{t^2_{\alpha/2,m-1} * S^2_m}{d^2}$

$$y = f(f_1, f_2)$$

- Con $\alpha = 0.05$ y
- $\alpha/2 = 0.025$
- Esto significa que se trabaja con un error del 5% o un grado de factibilidad del 95%.

- De las tablas de t-student (**Tabla 2**) se obtienen los valores de para cada uno de los límites de los intervalos.

TABLA 2. Tabla de la distribución de *t*-student.

Tabla t-Student

Grados de libertad	0.25	0.1	0.05	0.025	0.01	0.005
1	1.0000	3.0777	6.3137	12.7062	31.8210	63.6559
2	0.8165	1.8856	2.9200	4.3027	6.9645	9.9250
3	0.7649	1.6377	2.3534	3.1824	4.5407	5.8408
4	0.7407	1.5332	2.1318	2.7765	3.7469	4.6041
5	0.7267	1.4759	2.0150	2.5706	3.3649	4.0321
6	0.7176	1.4398	1.9432	2.4469	3.1427	3.7074
7	0.7111	1.4149	1.8946	2.3646	2.9979	3.4995
8	0.7064	1.3968	1.8595	2.3060	2.8965	3.3554
9	0.7027	1.3830	1.8331	2.2622	2.8214	3.2498
10	0.6998	1.3722	1.8125	2.2281	2.7638	3.1693
11	0.6974	1.3634	1.7959	2.2010	2.7181	3.1058
12	0.6955	1.3562	1.7823	2.1788	2.6810	3.0545
13	0.6938	1.3502	1.7709	2.1604	2.6503	3.0123
14	0.6924	1.3450	1.7613	2.1448	2.6245	2.9768
15	0.6912	1.3406	1.7531	2.1315	2.6025	2.9467
16	0.6901	1.3368	1.7459	2.1199	2.5835	2.9208
17	0.6892	1.3334	1.7396	2.1098	2.5669	2.8982
18	0.6884	1.3304	1.7341	2.1009	2.5524	2.8784
19	0.6876	1.3277	1.7291	2.0930	2.5395	2.8609
20	0.6870	1.3253	1.7247	2.0860	2.5280	2.8453
21	0.6864	1.3232	1.7207	2.0796	2.5176	2.8314
22	0.6858	1.3212	1.7171	2.0739	2.5083	2.8188
23	0.6853	1.3195	1.7139	2.0687	2.4999	2.8073
24	0.6848	1.3178	1.7109	2.0639	2.4922	2.7970
25	0.6844	1.3163	1.7081	2.0595	2.4851	2.7874
26	0.6840	1.3150	1.7056	2.0555	2.4786	2.7787
27	0.6837	1.3137	1.7033	2.0518	2.4727	2.7707
28	0.6834	1.3125	1.7011	2.0484	2.4671	2.7633
29	0.6830	1.3114	1.6991	2.0452	2.4620	2.7564
30	0.6828	1.3104	1.6973	2.0423	2.4573	2.7500
31	0.6825	1.3095	1.6955	2.0395	2.4528	2.7440
32	0.6822	1.3086	1.6939	2.0369	2.4487	2.7385
33	0.6820	1.3077	1.6924	2.0345	2.4448	2.7333
34	0.6818	1.3070	1.6909	2.0322	2.4411	2.7284

- Para este caso se trabaja además con un estimado *d* (en tiempo) de 100ms.

- Con estas ecuaciones y con la **Tabla 3** se puede determinar que el número de iteraciones es posible de realizar y que el estudio se puede seguir llevando adelante a analizar definitivamente en **SPSS**.

TABLA 3. Valores de error para cada uno de los algoritmos con los d calculados mediante límites superior e inferior y el d fijado por los diseñadores.

	PL	**BE**	**DIJKSTRA**
d cal.	65.45ms	64.45ms	61.37ms
d. fija	100ms	100ms	100ms

- Teniendo en cuenta que si *d* fijada < *d* calculada esto implica que se pueden usar 10 corridas o iteraciones, se sigue adelante con la aplicación.

- Usando el **SPSS** de IBM llamado **PASW** ingresamos los valores del tiempo de procesamiento para cada iteración y cada uno de los programas y se pueden ver los siguientes pantallazos en la siguiente página.

- Se puede observar en la **Tabla 4** que se ingresan las tres variables a trabajar con seis decimales, en modo escala, de tipo numérico, de carácter de entradas y con una etiqueta que describe el origen de la variable.

- Luego en la **Tabla 5** se pueden ver todos los valores de cada una de las diez iteraciones para los tres programas en **MATLAB**.

TABLA 4 . Pantalla del **PASW** para ingreso de variables, nombres, etiquetas, entrada o salida y demás datos propios de cada una de las variables.

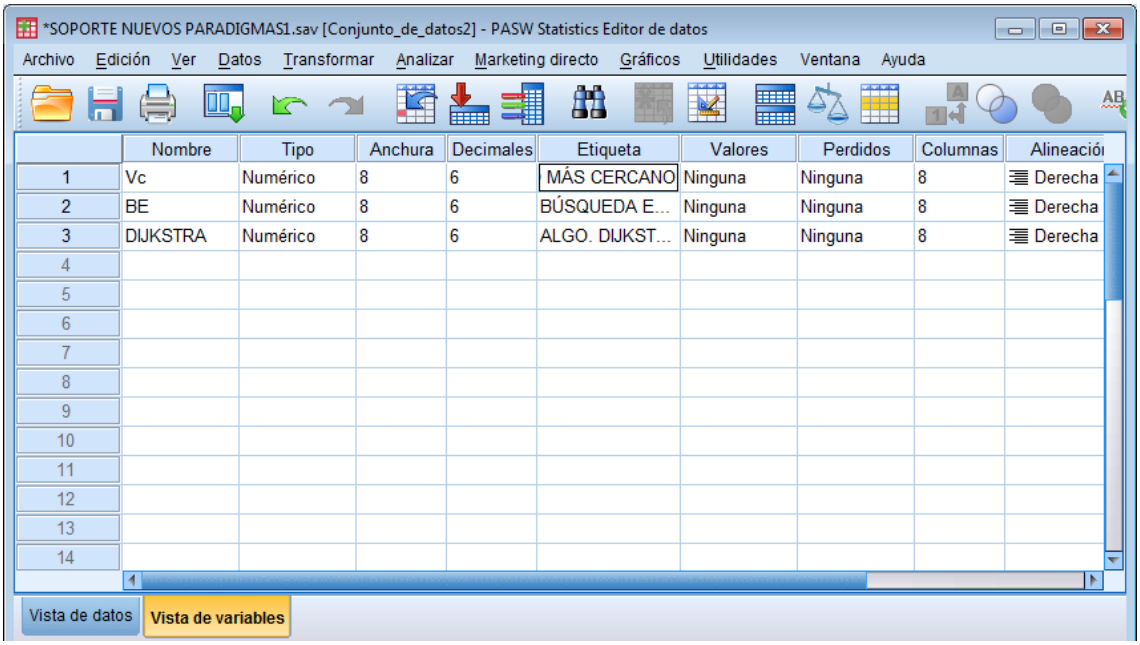
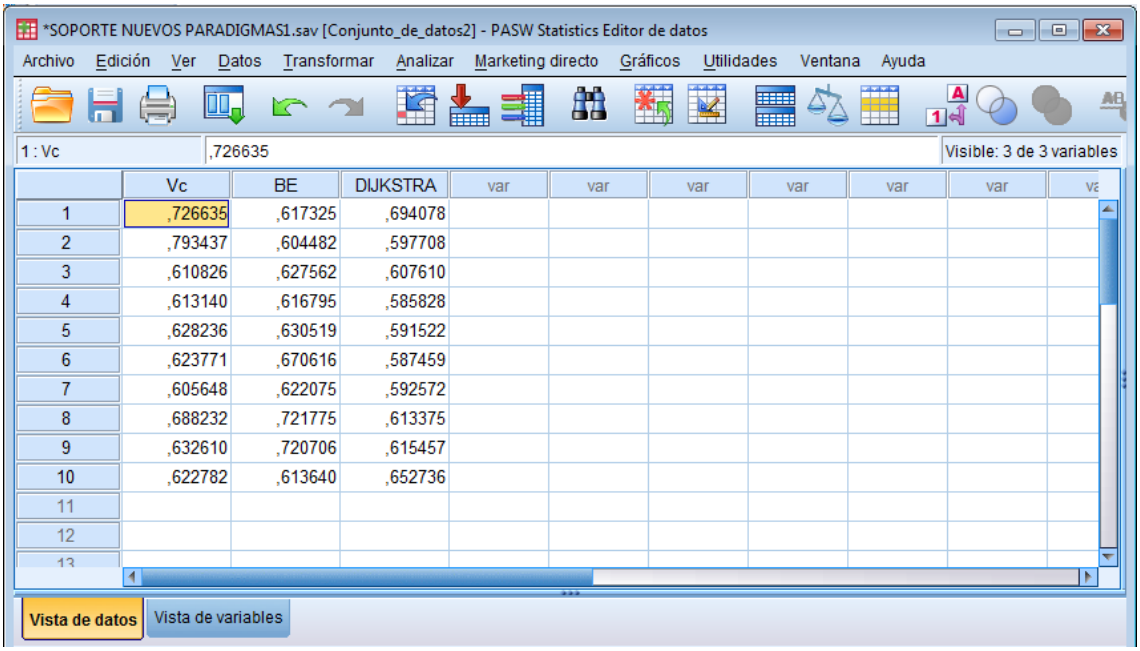


TABLA 5. Pantalla del **PASW** en donde se muestra el ingreso de los valores numéricos de las variables para comenzar a procesar los resultados.



- Luego de ingresar los datos al programa estadístico se seleccionan los ítems que se desean estudiar para la escogencia del mejor de los tres para trabajar en el diseño de la red. Para este caso se escogieron todos los de tendencia central y de dispersión que se ven en la **Tabla 6.**

TABLA 6. Estadísticos de la muestra de trabajo con diez iteraciones para cada programa.

Estadísticos

		VECINO MÁS CERCANO	BÚSQUEDA EXHAUSTIVA	ALGO. DIJKSTRA
N	Válidos	10	10	10
	Perdidos	0	0	0
Media		,65453170	,64454950	,61383450
Mediana		,62600350	,62481850	,60265900
Moda		,605648[a]	,604482[a]	,585828[a]
Desv. típ.		,062154507	,044103254	,034454431
Varianza		,004	,002	,001
Rango		,187789	,117293	,108250
Mínimo		,605648	,604482	,585828
Máximo		,793437	,721775	,694078
Suma		6,545317	6,445495	6,138345

a. Existen varias modas. Se mostrará el menor de los valores.

- En las siguientes **Tablas 7, 8 y 9** se pueden observar las distribuciones de frecuencia de los tiempos para cada una de las diez iteraciones y de los tres programas

TABLA 7. Distribuciones de frecuencia para **Vecino más Cercano** en **MATLAB**.

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	,605648	1	10,0	10,0	10,0
	,610826	1	10,0	10,0	20,0
	,613140	1	10,0	10,0	30,0
	,622782	1	10,0	10,0	40,0
	,623771	1	10,0	10,0	50,0
	,628236	1	10,0	10,0	60,0
	,632610	1	10,0	10,0	70,0
	,688232	1	10,0	10,0	80,0
	,726635	1	10,0	10,0	90,0
	,793437	1	10,0	10,0	100,0
	Total	10	100,0	100,0	

TABLA 8. Distribuciones de frecuencia para **Búsqueda Exhaustiva** en **MATLAB**.

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	,604482	1	10,0	10,0	10,0
	,613640	1	10,0	10,0	20,0
	,616795	1	10,0	10,0	30,0
	,617325	1	10,0	10,0	40,0
	,622075	1	10,0	10,0	50,0
	,627562	1	10,0	10,0	60,0
	,630519	1	10,0	10,0	70,0
	,670616	1	10,0	10,0	80,0
	,720706	1	10,0	10,0	90,0
	,721775	1	10,0	10,0	100,0
	Total	10	100,0	100,0	

TABLA 9. Distribuciones de frecuencia para **DIJKSTRA** en **MATLAB**.

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	,585828	1	10,0	10,0	10,0
	,587459	1	10,0	10,0	20,0
	,591522	1	10,0	10,0	30,0
	,592572	1	10,0	10,0	40,0
	,597708	1	10,0	10,0	50,0
	,607610	1	10,0	10,0	60,0
	,613375	1	10,0	10,0	70,0
	,615457	1	10,0	10,0	80,0
	,652736	1	10,0	10,0	90,0
	,694078	1	10,0	10,0	100,0
	Total	10	100,0	100,0	

- Del análisis de las medidas de tendencia central de cada uno de los tres programas y de las medidas de dispersión podemos afirmar que el de **DIJKSTRA** es el que mejor desempeño presenta frente a los demás con respecto al tiempo de procesamiento de los datos y obtención de la solución. Demostró al menos la mitad de la variabilidad y dispersión que el segundo algoritmo y un cuarto del tercero.
- Como se dijo al principio en este trabajo se trataba de escoger el mejor en cuestión de tiempo y no de exactitud en cuanto a la **Z** pues en este aspecto todos se desempeñaron de igual manera.

CONCLUSIONES

SPSS es una herramienta importante para realizar análisis entre diferentes variables y poder mediante ella escoger la más favorable para la actividad que se va a emprender. Además el uso de varios algoritmos decisionales provenientes de la investigación de operaciones brinda una gran ventaja al momento de escoger objetivamente la opción más adecuada en cualquier tipo de toma de decisión en la ingeniería.

En la actualidad se pueden encontrar múltiples herramientas de software para la aplicación estadística del tipo **SPSS** como **PASW** que se utilizó para este trabajo, todos ellos con el mismo entorno de opciones y muy pocas variantes gráficas lo que hace que la utilización de uno u otro sea de menor interés para el usuario siendo sí de gran importancia como ya se anotó anteriormente.

La programación en **MATLAB** permite llevar a cabo un sinnúmero de pruebas y llevar a la práctica algoritmos que en otro entorno sería engorrosos y de difícil ejecución. La correcta utilización de las librerías y las funciones permitió la elaboración fácil de los programas que se anexan a continuación de este documento y sobre los que se trabajó estadísticamente.

Otra cosa a resaltar es que una de las grandes ventajas en el campo de la ingeniería es adaptar lo que se halla en el medio para conseguir los resultados que se buscan y no perder tiempo en crearlos de cero, así sobre programas existentes es posible realizar nuevas consultas y adaptaciones que facilitan la consecución del resultado final de manera eficaz y eficiente.

Aunque el objeto de este trabajo no fue utilizar software ya diseñado para simular estos métodos, se pudo observar que el **WinQSB** es una herramienta sumamente importante y de fácil aplicación que brinda resultados exactos y de sencilla lectura. Además, posee variados métodos de aplicación desde la programación lineal hasta Dijkstra y Floyd, pasando por programación dinámica y métodos tabulares lo que lo convierte en una de las herramientas más versátiles en este campo.

Por último, aunque el **SPSS** y los algoritmos brindan variadas opciones y muy objetivas, el poder de decisión final está en el ingeniero o la persona encargada de tomar sobre sí la responsabilidad de escoger entre todas las opciones la mejor respecto a la situación planteada y si esta persona no cuenta con la preparación necesaria o no sabe analizar las herramientas que se le brindan, se habrá perdido una gran oportunidad y desechado una ventaja competitiva importante.

RECONOCIMIENTOS

El planteamiento del problema y la creación de los diferentes programas para los algoritmos y toda la dinámica usada en este trabajo fue propuesto como evaluación para el curso **Nuevos Paradigmas Decisionales en el campo de la Ingeniería** por el Dr. **Javier Asencio García**, profesor titular y doctor en Ciencias Técnicas de la Universidad Otto Von Guericke, Magdeburg, Alemania y miembro de la Comisión Nacional de la Carrera de Ing. Industrial de la República de Cuba. Miembro del tribunal nacional de doctorados de Ing. Industrial de la República de Cuba.

La evaluación y orientación final de este trabajo estuvo dirigida por el Ing. Francisco Alfonso Chediak, profesor titular de la Universidad de Ibagué, magister en

Investigación Operativa y Estadística de la Universidad Tecnológica de Pereira y especialista en Matemática Aplicada con Énfasis en Investigación.

BIBLIOGRAFÍA

[1]DERFLER, Frank. Redes LAN & WAN. LAN, WAN e intranets. Prentice Hall, Colombia, 1999.

[2]www.mathworks.com

[3]www.spss.com

www.networkandcommunications.com

www.proing.com.co

www.unizar.es

www.IBM.com

*** Estudiantes de Ingeniería Electrónica de la Universidad de Ibagué.**