Práctica 6

Árboles de unión de menor valor

En esta práctica vamos a ver cómo podemos calcular el árbol de unión de menor valor (AUVM) utilizando MatLab. Recordemos que un árbol es un grafo no orientado conexo y acíclico. Partiremos de un grafo no orientado $G = (V, A)$ y unos pesos p de manera que cada arista e tiene asociado un peso p_e, y el objetivo será buscar un subgrafo de G que sea un árbol y que minimice el valor total de los pesos.

6.1. Resolución con MatLab

6.1.1. El comando `minspantree`

El comando con el que resolveremos el problema de encontrar un AUVM es `minspantree`. De esta manera, si hemos definido un grafo no orientado G, simplemente hemos de indicar lo siguiente:

```
>> T=minspantree(G)
```

Es conviente almacenar el resultado, en este caso le asignamos el valor `T`, puesto que la salida de esta instrucción es un nuevo grafo, de hecho un subgrafo de G, que solamente contiene los arcos que minimizan el valor total del árbol. Una vez que ya tenemos el árbol `T`, sobre él podemos obtener toda la información que deseemos, al igual que hemos hecho en la práctica 1. Por ejemplo, si queremos conocer qué aristas se han incluido en este subgrafo, utilizamos la instrucción:

```
>> T.Edges
```

Si el grafo original lo hemos definido indicando las aristas con los vectores s y t, y el vector `peso` para indicar los pesos, podemos hacer una representación gráfica del grafo y su AUVM:

```
>> G=graph(s,t,peso)
>> h=plot(G)
>> labeledge(h,1:numberedges(G),peso)
>> T=minspantree(G)
>> highlight(h,T,'EdgeColor','red')
```

Con la tercera instrucción añadimos los pesos a las aristas, mientras que con la quinta nos marcará las aristas del árbol en color rojo.

En ocasiones, podemos estar interesados en crear un AUVM partiendo de un vértice dado. En tal caso, dicho vértice se indicará como raíz. Si por ejemplo el vértice elegido es el tercero, la instrucción a utilizar será:

```
>> T=minspantree(G,'Root',3)
```

Por defecto, salvo que se indique el vértice inicial, se tomará como raíz al primer vértice.

Hemos de notar que si el grafo introducido no es conexo, el comando `minspantree` solamente calcula el AUVM asociado la componente conexa a la que pertenece su raíz (que como hemos comentado es el primer vértice, salvo que se indique otra cosa). Cuando el grafo original no es conexo, una posibilidad es calcular el bosque de menor valor. En tal caso, MatLab calculará el AUVM para cada una de las componentes conexas. Para calcular el bosque en vez del árbol, podemos utilizar la siguiente instrucción:

```
>> T=minspantree(G,'Type','forest')
```

Ejemplo 6.1. *Consideremos el grafo no orientado que aparece en la Figura 6.1.*

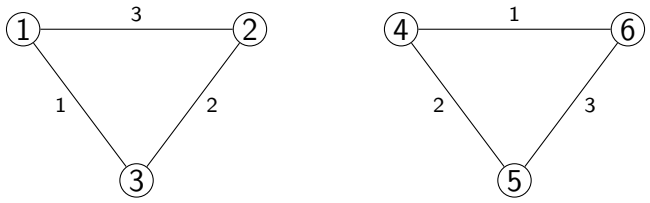

Figura 6.1: Grafo del Ejemplo 6.1.

Definamos este grafo en MatLab*:*

```
>> s=[1 1 2 4 4 5];
>> t=[2 3 3 5 6 6];
>> peso=[3 1 2 1 2 3];
>> G=graph(s,t,peso)

G =

  graph with properties:

    Edges: [6x2 table]
    Nodes: [6x0 table]
```

Obviamente, este grafo no es conexo, como podemos comprobar:

```
>> conncomp(G)

ans =

     1     1     1     2     2     2
```

Si utilizamos el comando `minspantree` *sin especificar la raíz, tomará como tal al vértice 1, obteniendo el siguiente resultado:*

```
>> T1=minspantree(G)

T1 =

   graph with properties:

     Edges: [2x2 table]
     Nodes: [6x0 table]
>> T1.Edges

ans =

     EndNodes    Weight
     --------    ------

     1    3      1
     2    3      2
```

Como vemos, el árbol creado solamente hace referencia a los vértices de la primera componente conexa, formada por los vértices 1, 2 y 3, dejando a los vértices 4, 5 y 6 como vértices aislados. Por contra, si ahora buscamos el AUVM indicando como raíz al vértice 4, obtenemos lo siguiente:

```
>> T2=minspantree(G,'Root',4)

T2 =

   graph with properties:

     Edges: [2x2 table]
     Nodes: [6x0 table]
>> T2.Edges

ans =

     EndNodes    Weight
```

```
    --------    ------
    4    5      1
    4    6      2
```

En este caso, el árbol solamente hace referencia a los vértices de la segunda componente conexa (vértices 4, 5 y 6). En este segundo caso, los vértices 1, 2 y 3 han quedado aislados. Si por contra, en vez de solicitar un árbol, solicitamos el bosque, el resultado es el siguiente:

```
>> F=minspantree(G,'Type','forest')

F =

  graph with properties:

    Edges: [4x2 table]
    Nodes: [6x0 table]
>> F.Edges

ans =

    EndNodes    Weight
    --------    ------

    1     3     1
    2     3     2
    4     5     1
    4     6     2
```

En este último caso, nos ha calculado el AUVM en cada una de las componentes, y juntos forman el bosque de unión de valor mínimo. En la Figura 6.2 podemos ver la representación de los grafos `T1`, `T2` *y* `F` *obtenidos en este ejemplo.*

6.1.2. Opciones del comando `minspantree`

En el apartado anterior ya hemos visto algunas de las opciones que tiene el comando `minspantree`, como son las siguientes:

a) **Root.** Nos permite especificar la raíz del árbol.

b) **Forest**. Para grafos inconexos, calcula el AUVM en cada componente conexa y forma con ellos un bosque.

Otras opciones interesantes son las siguientes:

a) **Tipo de algoritmo.** Por defecto, el comando `minspantree` utiliza el algoritmo de Prim, que comienza en la raíz y va añadiendo aristas mientras recorre el grafo. Si queremos especificar el

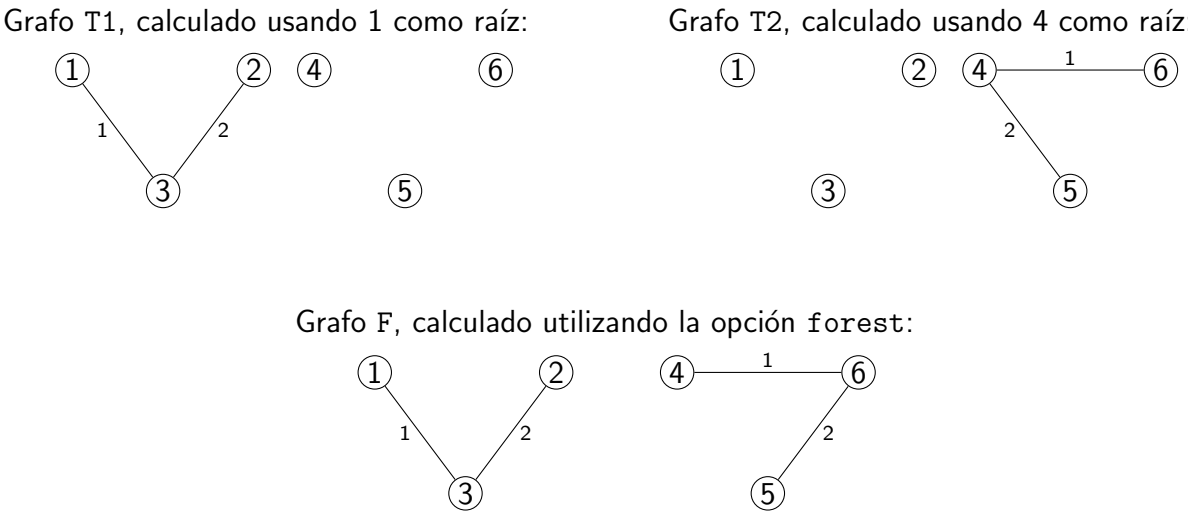

Figura 6.2: Grafos T1 and T2 obtenidos utilizando el comando minspantree con los nodos 1 y 4 como raíces, respectivamente (figura de arriba a la izquierda y derecha, respectivamente). En la figura inferior se muestra el bosque de unión de valor mínimo.

tipo de algoritmo, debemos de utilizar la opción dense, para el algoritmo de Prim, y sparse, para el algoritmo de Kruskal, que ordena las aristas según su peso y a continuación añade los vértices. Por ejemplo, para especificar el algoritmo de Kruskal, debemos utilizar la expresión:

```
>> T=minspantree(G,'Method','sparse')
```

b) **Lista de predecesores.** Además de obtener el AUVM como salida, también podemos solicitar un segundo vector que contiene los predecesores de cada nodo, comenzando a partir de la raíz. En tal caso, la raíz no tiene predecesor, y por tanto le asigna el valor 0.

```
>> [T,pred]=minspantree(G)
```

Ejemplo 6.2. *Continuando con el grafo del Ejemplo 6.1 representado en la Figura 6.1, podemos volver a solicitar el bosque de unión de valor mínimo añadiendo pidiendo también la lista de predecesores:*

```
>> [F,pred]=minspantree(G,'Type','forest')

F =

   graph with properties:

      Edges: [4x2 table]
      Nodes: [6x0 table]

pred =

     0     3     1     0     4     4
```

El significado del vector `pred` *es el siguiente: los vértices 1 y 4 tienen asignado el valor 0, puesto que ambos han sido tomados como raíces (cada uno de su componente conexa); el predecesor del vértice 2 es el 3, y el predecesor del vértice 3 es el 1; el vértice 4 es el predecesor de los vértices 5 y 6.*

6.2. Ejemplo ilustrativo

Consideremos un ejemplo descrito en *Introduction to Mathematical Programming, F.S.Hillier y G.J.Lieberman, McGraw-Hill Publishing Company*: la compañía Seervada Park tiene que determinar en qué carreteras ha de instalar las líneas telefónicas con el objetivo de conectar todas las estaciones con una menor longitud de línea. La información sobre las estaciones, las posibles conexiones y sus longitudes de cables se pueden ver en la Figura 6.3.

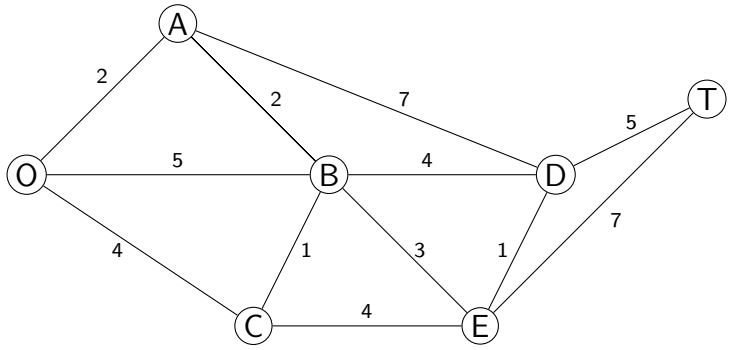

Figura 6.3: Sistema de carreteras para Seervada Park.

El objetivo es por tanto minimizar la longitud de la línea, lo que reducirá costes tanto de instalación como de mantenimiento. En nuestros términos, lo que hemos de hacer es encontrar un AUVM. Comencemos definiendo el grafo:

```
>> s=[1 1 1 2 2 3 3 4 5 5 6];
>> t=[2 3 4 3 5 4 5 6 6 7 7];
>> peso=[2 5 4 2 7 1 4 4 1 5 7];
>> G=graph(s,t,peso,{'O','A','B','C','D','E','T'})

G =

  graph with properties:

    Edges: [11x2 table]
    Nodes: [7x1 table]
>> h=plot(G)
>> labeledge(h,1:numedges(G),peso)
```

De esta manera hemos definido el grafo y hemos obtenido la representación gráfica de la Figura 6.4. Busquemos a continuación el AUVM:

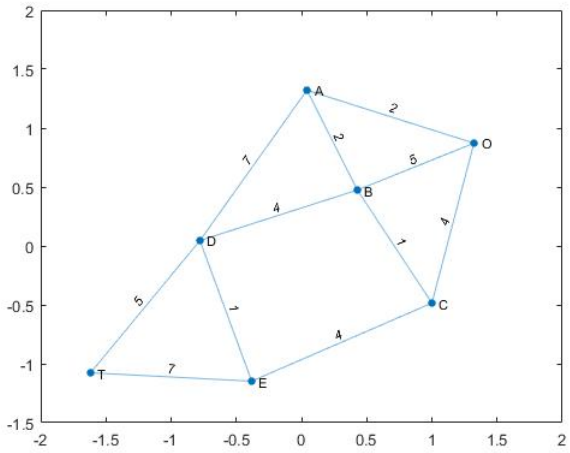

Figura 6.4: Representación gráfica de MatLab sistema de carreteras de Seervada Park.

```
>> T=minspantree(G)

T =

  graph with properties:

    Edges: [6x2 table]
    Nodes: [7x0 table]
>> T.Edges

ans =

     EndNodes       Weights

    ----------     ---------

    'O'    'A'      2
    'A'    'B'      2
    'B'    'C'      1
    'B'    'D'      4
    'D'    'E'      1
    'D'    'T'      5
```

Con el último de los comandos, `T.Edges`, solicitamos la información referente a las aristas que ha seleccionado para formar parte del árbol. También podríamos solicitar la matriz de adyacencia mediante el commando `adjacency(T)` o, si queremos la matriz completa, mediante `full(adjacency(T))`. Si queremos representar gráficamente el AUVM, podemos utilizar las siguientes instrucciones:

```
>> highlight(h,T,'EdgeColor','red')
```

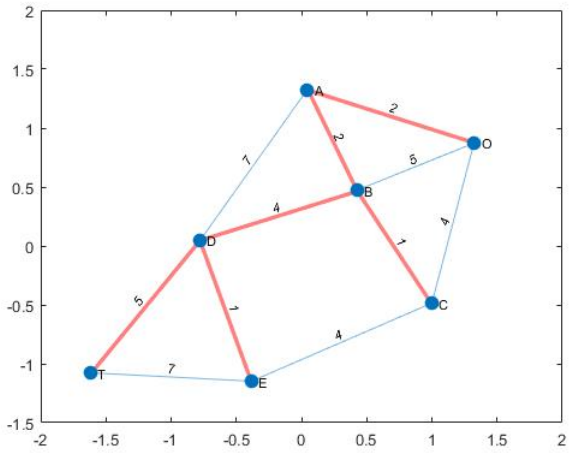

Figura 6.5: Árbol de unión de menor valor para el sistema de carreteras de Seervada Park.

El resultado lo podemos ver en la Figura 6.5, con la salida de MatLab, o en la Figura 6.6, utilizando la representación original del grafo.

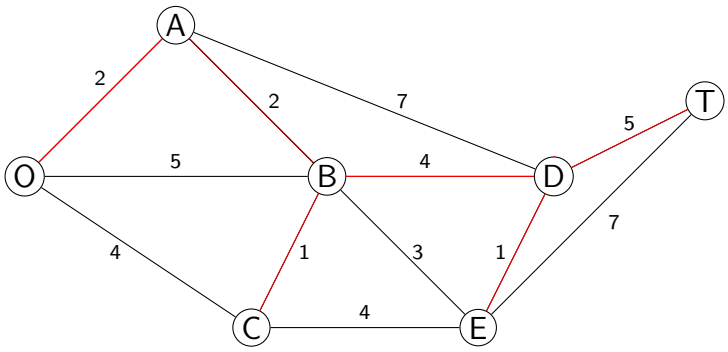

Figura 6.6: Árbol de unión del sistema de carreteras para Seervada Park.