Networks and Complexity

Solution 1-9

This is an example solution from the forthcoming book Networks and Complexity. Find more exercises at https://github.com/NC-Book/NCB

Ex 1.9: A tale of two cities [3]

Consider the Moravian power grid from the previous exercise for a final time. Now we start again with an empty graph, but there are two power plants, which are located in B and Z. Each of the plants has the capacity to supply the whole region. Find the lines that need to be built such that each city can get power from one of the power plants and the length of lines built is minimal. (Bonus: Can you also solve this exercise without drawing the network, you might need to invent some other form of bookkeeping.)

Solution

Before we start, here is the distance matrix again

$$\mathbf{D} = \begin{pmatrix} 0 & 137 & 63 & 74 & 77 \\ 137 & 0 & 75 & 76 & 198 \\ 63 & 75 & 0 & 51 & 121 \\ 74 & 76 & 51 & 0 & 151 \\ 77 & 198 & 121 & 151 & 0 \end{pmatrix} \tag{1}$$

and the ordering of nodes

$$1 : \mathrm{B},\ 2 : \mathrm{O},\ 3 : \mathrm{L},\ 4 : \mathrm{Z},\ 5 : \mathrm{J}. \tag{2}$$

We apply a variant of Kruskal's algorithm: In addition to rejecting any link that does not reduce the number of components, we also reject any link that would connect the two components in which B and Z are located.

1. Try (L,Z) [51km] – accept
2. Try (L,B) [63km] – reject (puts B and Z in same component)
3. Try (B,Z) [74km] – reject (puts B and Z in same component)
4. Try (L,O) [75km] – accept
5. Try (O,Z) [76km] – reject (does not reduce number of components)
6. Try (J,B) [77km] – accept

Now everybody is connected to one of the power plants. The final edge set is

$$E = \{(\mathrm{L},\mathrm{Z}), (\mathrm{L},\mathrm{O}), (\mathrm{J},\mathrm{B})\}. \tag{3}$$

But how could we do these steps without drawing the network? Note that we only need to remember which components the nodes are in. Instead of drawing the network we can also remember this in a table. In our initial network where no links have been placed each node is in its own component, so the able could look like this

B	O	L	Z	J
1	2	3	4	5

So Brno is in component 1, Ostrava is in component 2 and so on. Now we try to place the link (L,Z). This connects two different components (3,4) and does not connect Connect the components of Brno and Zlin. Hence, we can place (L,Z).

Placing the link puts all nodes from component 3 and 4 into the same component. From now on this joint component is denoted by 3. We record the updated component assignments in a new row in our table, which now looks as follows:

B	O	L	Z	J
1	2	3	4	5
1	2	3	3	5

Next we try to place the link (L,B), but this would connect the component of Brno (1) to the component os Zlin (3), so we reject the link and the table stays unchanged.

The next shortest link is (B,Z) but this would even connect Brno and Zlin directly. So that's another rejection.

Next we try (L,O). This connects two different components (2,3) and does not involve connecting the component of Brno to the component of Zlin. Placing the link merges components 2 and 3. We now refer to the merged component as component 2, in the next update all 3s in the table become 2s:

B	O	L	Z	J
1	2	3	4	5
1	2	2	2	5

We reject the next link (O,Z) because the two nodes are already in the same component, and instead try the next shortest (J,B), which can be placed. Our final table is

B	O	L	Z	J
1	2	3	4	5
1	2	2	2	5
1	2	2	2	1

Now there are only two components left. The one containing Zlin and the one containing Brno. So we are done.

In retrospect we could have even solved the problem a bit easier if we had assigned the same component number to Zlin and Brno in the beginning without actually linking them. This would have saved us the work of checking whether a candidate link would connect B and Z, as those links would be ruled out automatically.

Perhaps more importantly, being able to solve the spanning tree problem without drawing the network, allows us also to implement the algorithm efficiently on a computer. Recording the component membership in a table is useful for pen and paper solutions because as new rows can easily be added when updates occur. But we are only ever using the most recent row of the table. So when we implement the algorithm on a computer we wont need the whole table. Instead we can just store the current cluster memberships of the nodes in a one dimensional array.