Networks, Flow and the Maximum Flow Problem

Martin Murphy

December 11, 2014

1 A Quick Introduction to Networks

A network is defined in the dictionary as "an arrangement of intersecting horizontal or vertical lines" or "a group or system of interconnected people or things". [2]

In (discrete) mathematics, a network is the term used to describe a directed graph with weighted arcs.

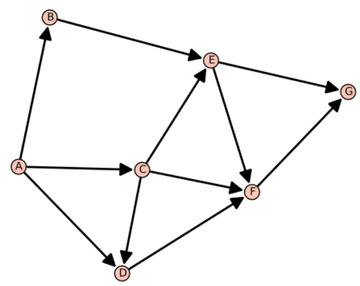


Figure 1: This is an example of a network.

1.1 Terminology [1]

A graph G is made up of a finite set of vertices V and a set of edges E made up of two subsets of V. We usually write G = (V, E) and say that V is the vertex-set and E is the edge-set

(Note that graphs, in theory, do not need to be made up of finite sets but the restriction has been made in this case for the convenience of computing things later).

Vertices - The vertices of a graph are specific points, represented by their position in the set V. Imagining a map of towns and cities, in graph form, each town or city would be represented by a vertex.

Edges - The edges are the lines connecting the vertices, represented by the initial vertex and the vertex that is being connected by the edge E = (A, B) for A, B in V. Following the above example, the roads connecting towns and cities would be represented by the edges in the graph.

Path - A path is a route from a start vertex to an end vertex if there is no arc directly from one to the other.

Digraph - A digraph is a graph where edges have a specified direction. A directed edge can only be followed in one direction.

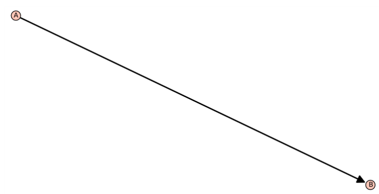
Weight - The weight of an edge is the "capacity" of the edge. If a bridge could only hold five vehicles at a time, the edge representing this bridge would have a weight of five.

2 The Maximum Flow Problem

In a basic sense, the maximum flow problem comes about when one is presented with a network and is required to find the maximum flow from one vertex (the source) to another (the sink).

The maximum flow is found when every available route from the source to the sink is at full capacity. E.g. If a bridge linking one town (A) to another (B) can only carry five vehicles at a time in the direction of B, the maximum flow from A to B would be five. (The maximum flow from B to A would be zero as the bridge is directed from A to B)

This would be shown by the network below:



2.1 A pen and paper method to solve the problem (In simple terms)

Step 1: Find a path from the source to the sink and calculate the maximum flow for this path (the lowest weight of the edges in the path). Do not examine edges whose weight is zero.

Step 2: Flood this path by sending the maximum flow through it. Subtract the value of the flow from the weights of all of the edges in the path. Make a note of the path in terms of the edges used and the flow along this path.

The network now has edges/an edge that have a weight of zero. These edges can now not be used.

Step 4: If there are still paths with a flow greater than zero available, return to step 1. Otherwise, examine the total flow leaving the source and the total flow entering the sink. These values should be the same (as flow should not appear or disappear along the network) and should be the same as the sum of all of the flows of the paths from Step 2.

2.2 Using Sage to solve the problem

I had been working on writing my own piece of code that would find the maximum flow of a network but after doing some online reading [3], I found that sage has a lot of functions already built in!

With my new found knowledge I used the following code to build the network shown in figure 1 and find the maximum flow (from A to G).

```
sage: G = DiGraph([("A", "B", 7), ("A", "C", 6), ("A", "D", 2), ("B", "E", 7), ("C", "E", 2),
("C", "F", 3), ("C", "D", 1), ("D", "F", 3), ("E", "G", 5), ("E", "F", 4), ("F", "G", 10)])

sage: G.flow("A", "G")
15
```

Where the pen and paper method would take a few minutes to find the maximum flow, sage takes a matter of seconds.

The only issue with using this code is that it only returns the maximum flow and you can't examine which paths have been used.

2.3 Further benefits of using code

Consider the following network:

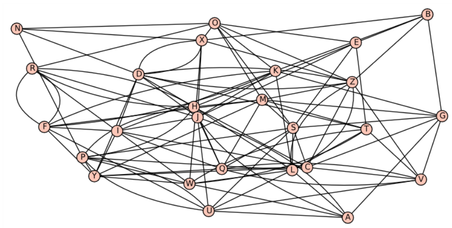


Figure 2: To make it easier to see the network, I have not given a directed graph here.

To find the maximum flow on this network by hand would take a lot of time and effort. Sage however can find a maximum flow in a matter of seconds.

Remember also that graphs can be of any size.

Code is therefore hugely important in tackling larger networks as it would take a human a much greater amount of time to solve even the more basic of problems.

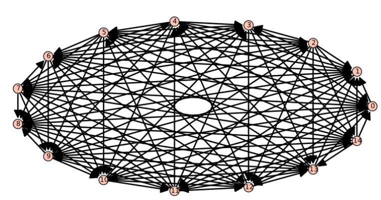
3 Another interesting application:

3.1 Tournaments

The way a tournament is modelled can also be produced in sage.

If there are 15 people participating in a tournament and each person needs to compete against everyone else. To minimise the time taken, it is best to have matches occuring at the same time. Rather than working out a program for each individual, Sage (and I am sure many other systems) can give an "easy" solution:

```
k = digraphs.RandomTournament(15)
plot(k)
```



References

[1] N. L. Biggs (1989). *Discrete Mathematics (Revised Edition)*. Oxford University Press Inc.

[2] Oxford University Press [Online]. Definition of network in english. *http://www.oxforddictionaries.com/definition/english/network*, 2014.

[3] The Sage Development Team [Online]. Sage.graphs.genericgraph.flow. *http://mvngu.googlecode.com/hg/onepage/sage/graphs/generic*$_g$*raph/sage.graphs.generic*$_g$*raph.GenericGraph.flow.htm*