COM2031 Advanced Algorithms, Autumn Semester 2019

Lab 9: Reductions: More Applications of Network Flow

Purpose of the lab

This week's lab is to explore two problems covered in Topic 4 (Network Flow Applications): **Project Selection**, and **Survey Design**.

The sheet refers back to slide numbers from the material covered in the lecture. The slideset is the one entitled **COM2031-04-Network-Flow-Applications** (NOT the solutions one).

Each topic on the sheet starts with a worked example for you to follow through to step through the approach to solving the problem. It then asks some further questions for you to solve. These questions in the Project Selection questions follow on from one another, so you can edit your graph from one question to solve the next question, rather than having to start from a blank slate again.

For each of the topics, you will reduce instances of these problems to instances of a Max-Flow/Min-Cut problem. That problem can then be solved with the Ford-Fulkerson algorithm. You will then need to translate the solution back into a solution to the original problem.

In order to best achieve an understanding of what is going on, you should use the tool at:

https://www-m9.ma.tum.de/graph-algorithms/flow-ford-fulkerson/index_en.html

This allows you to build flow networks, and to apply Ford Fulkerson to find the Max Flow / Min Cut. You will translate the problem given into the appropriate flow network, run the algorithm, and in that way obtain the solution.

If you wish to run code to find the solutions then you may also do this, but this is an optional extra. The key element of the lab is to use the visualisation to understand the reduction of the other problems to Max Flow/Min Cut. An implementation of the Ford Fulkerson algorithm using adjacency matrices is provided in the solutions of lab 7 in SurreyLearn. Or you can use your own one that you developed in Lab 7 if you prefer.

1. Project Selection (slides 23-32)

The project selection problem is as follows (from Lectures) :

There is a set P of possible projects. Each project v has associated revenue p_v.
Some projects generate money, in which case ($p_v > 0$). Others cost money ($p_v < 0$)
Each project v has a set of prerequisites – other projects that must be done in order to do project v.
A subset of projects $A \subseteq P$ is feasible if the prerequisite of every project in A also belongs to A.

Project selection: Find a maximal project selection that satisfies all dependency requirements and maximises net total profit.

So a Project Selection problem gives a list of projects, and for each project there is a profit or cost, and a set of other projects that it is dependent on.

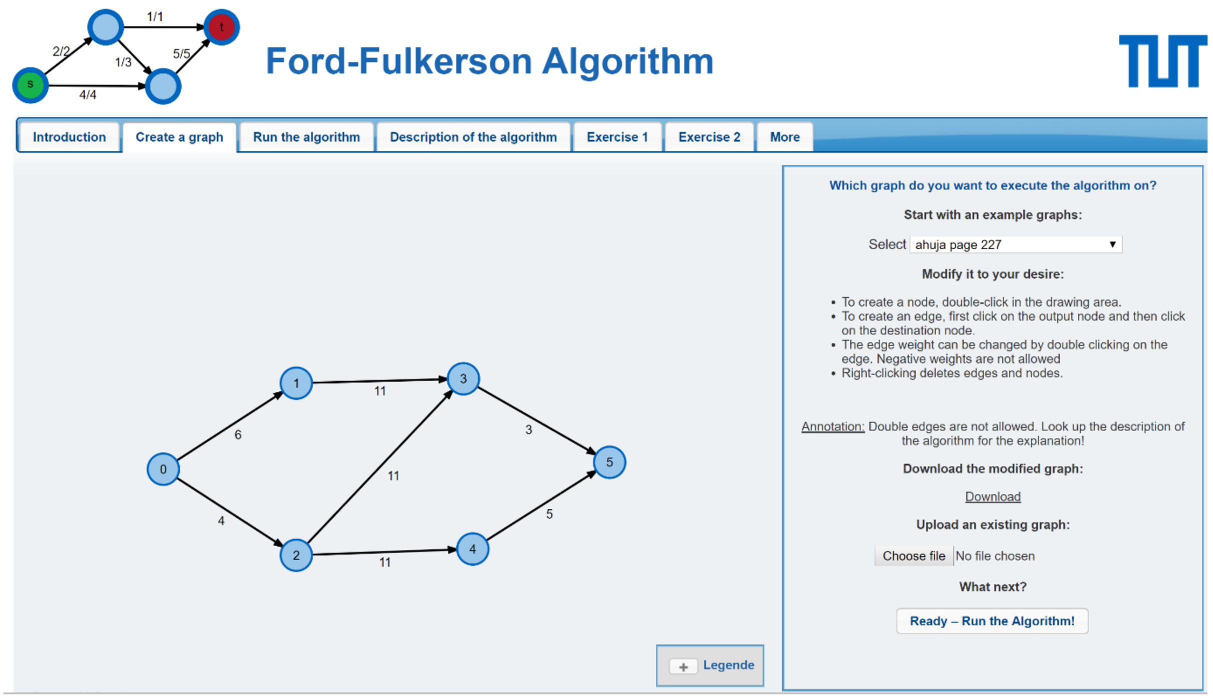
Exercise 1a: Here is a set of 4 projects. Project A gives a profit of 6 and is dependent on project C; project B gives a profit of 4 and is dependent on both C and D. Project C has a cost of 3, and project D has a cost of 5.

Project	p_v	Dependencies
A	6	C
B	4	C, D
C	-3	
D	-5	

In order to convert this Project Selection problem into a Network Flow problem we create the flow graph with the following nodes and edges:

- A source node s and a sink node t
- A node for each project
- For each project with cost $p_v < 0$ add an edge with weight $- p_v$ from the associated node to the sink t.
- For each project with positive profit $p_v > 0$ add an edge from the source s to the corresponding node with weight p_v.
- For each dependency between projects add an edge from a project node to any node on which it is dependent. In the lectures these edges are labelled with infinity, but the key point is that they should have so much capacity that they are not full even with the maximum flow. We can see that the sum of the weights from s is a total of 10, so the maximum flow can be no more than 10. Therefore if we assign a weight of 11 to each of these edges then there is no flow in which they will be full

Create the associated graph in the tool. You should end up with something like the following on your screen (though possibly with different numbers on the nodes)



Which graph do you want to execute the algorithm on?

Start with an example graphs:

Select ahuja page 227

Modify it to your desire:

- To create a node, double-click in the drawing area.
- To create an edge, first click on the output node and then click on the destination node.
- The edge weight can be changed by double clicking on the edge. Negative weights are not allowed
- Right-clicking deletes edges and nodes.

Annotation: Double edges are not allowed. Look up the description of the algorithm for the explanation!

Download the modified graph:

Download

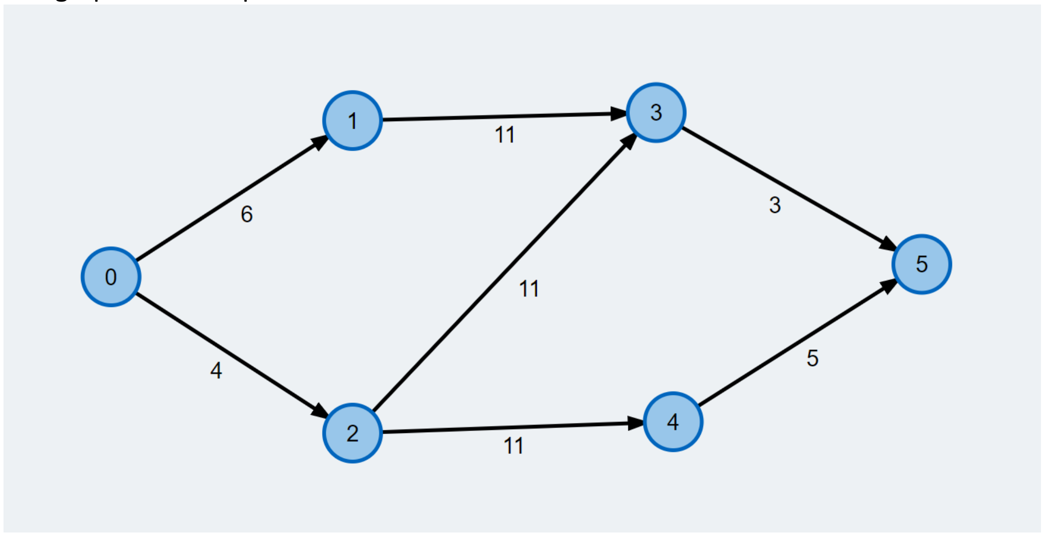
Upload an existing graph:

Choose file No file chosen

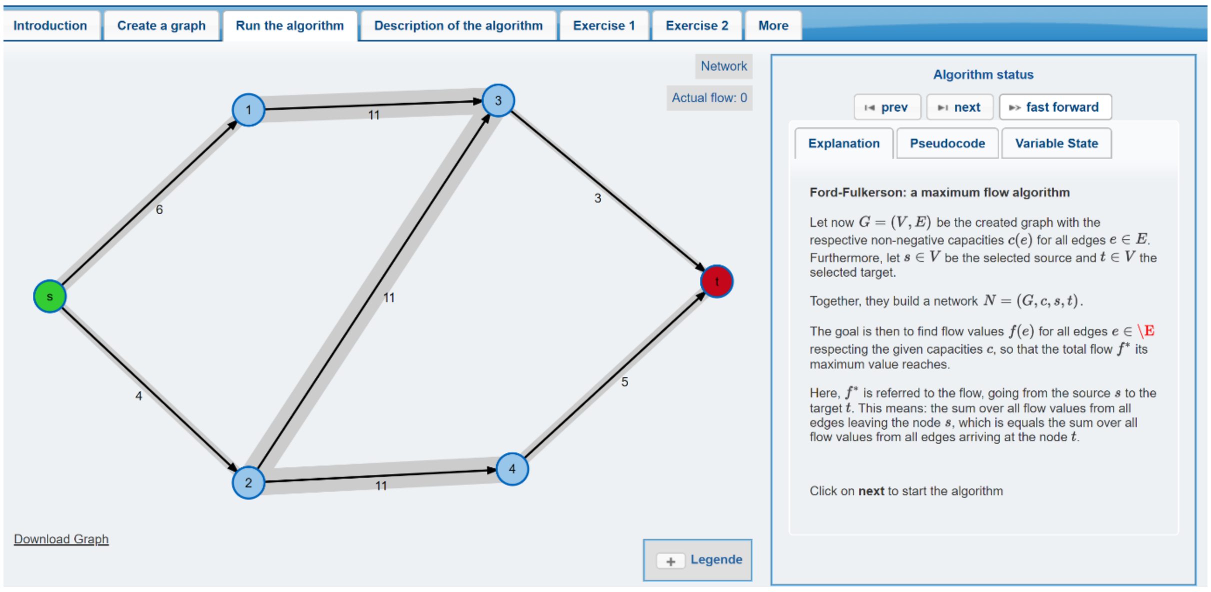
What next?

Ready – Run the Algorithm!

The graph in close up is as follows:



Now select the “run the algorithm” tab, and select the appropriate nodes for s and t:



Ford-Fulkerson: a maximum flow algorithm

Let now $G = (V, E)$ be the created graph with the respective non-negative capacities $c(e)$ for all edges $e \in E$. Furthermore, let $s \in V$ be the selected source and $t \in V$ the selected target.

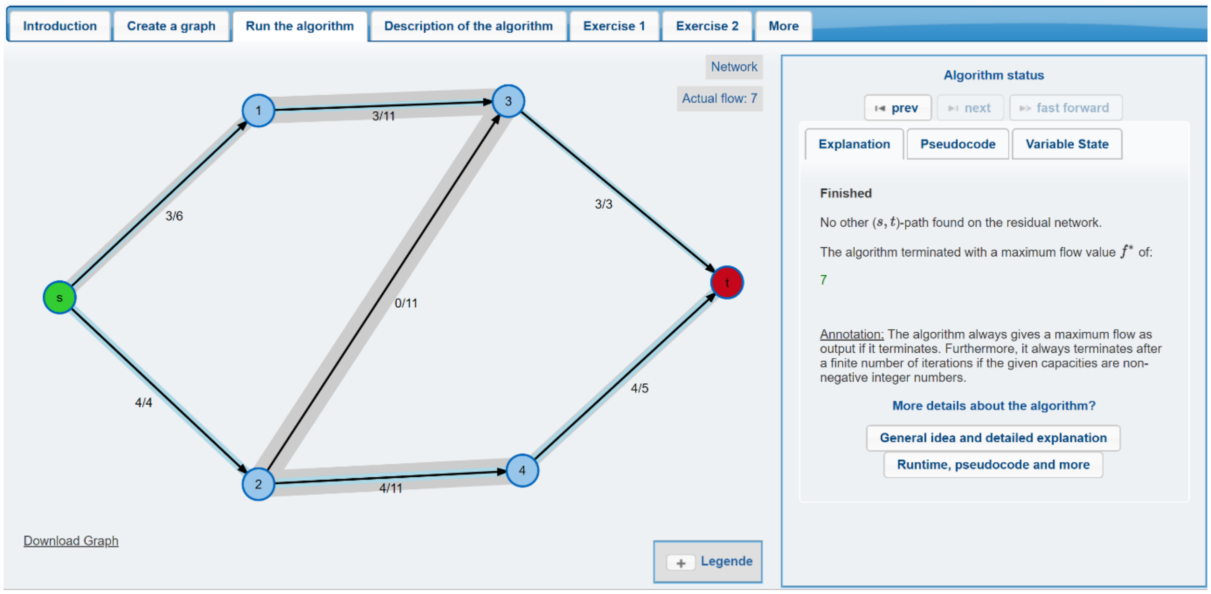
Together, they build a network $N = (G, c, s, t)$.

The goal is then to find flow values $f(e)$ for all edges $e \in \backslash E$ respecting the given capacities c, so that the total flow f^* its maximum value reaches.

Here, f^* is referred to the flow, going from the source s to the target t. This means: the sum over all flow values from all edges leaving the node s, which is equals the sum over all flow values from all edges arriving at the node t.

Click on **next** to start the algorithm

Run the algorithm through to the end. You can select "fast forward" or you can step through the algorithm to watch it build up the maximum flow. After you have run the algorithm you should reach the following, which gives the maximum flow.



Finished

No other (s, t)-path found on the residual network.

The algorithm terminated with a maximum flow value f^* of:

7

Annotation: The algorithm always gives a maximum flow as output if it terminates. Furthermore, it always terminates after a finite number of iterations if the given capacities are non-negative integer numbers.

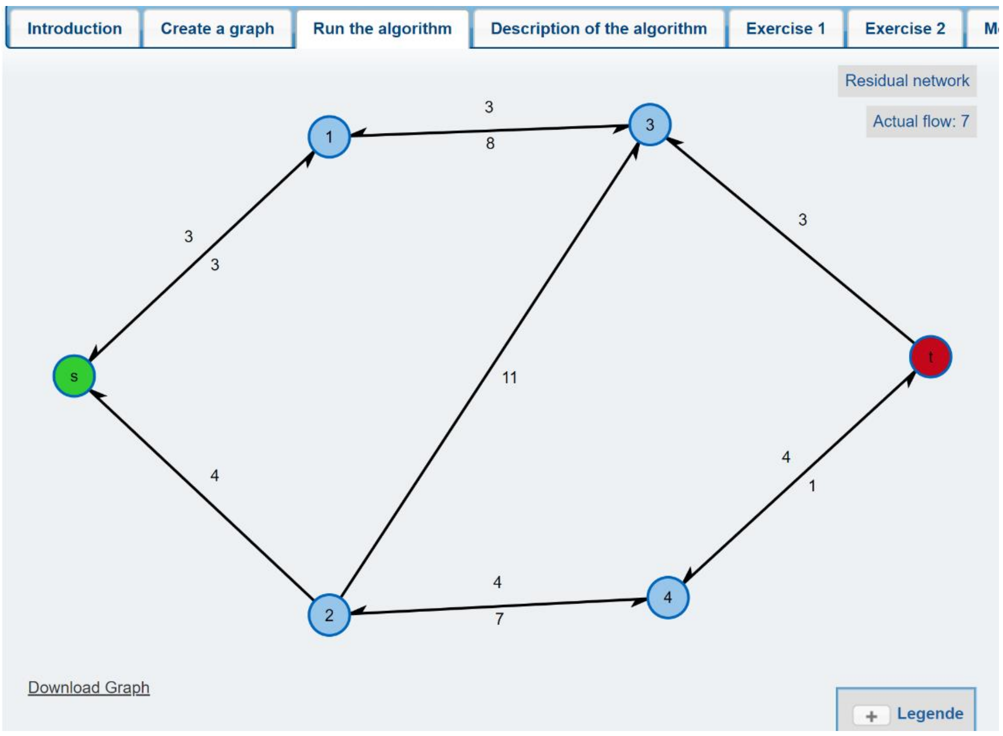
More details about the algorithm?

General idea and detailed explanation

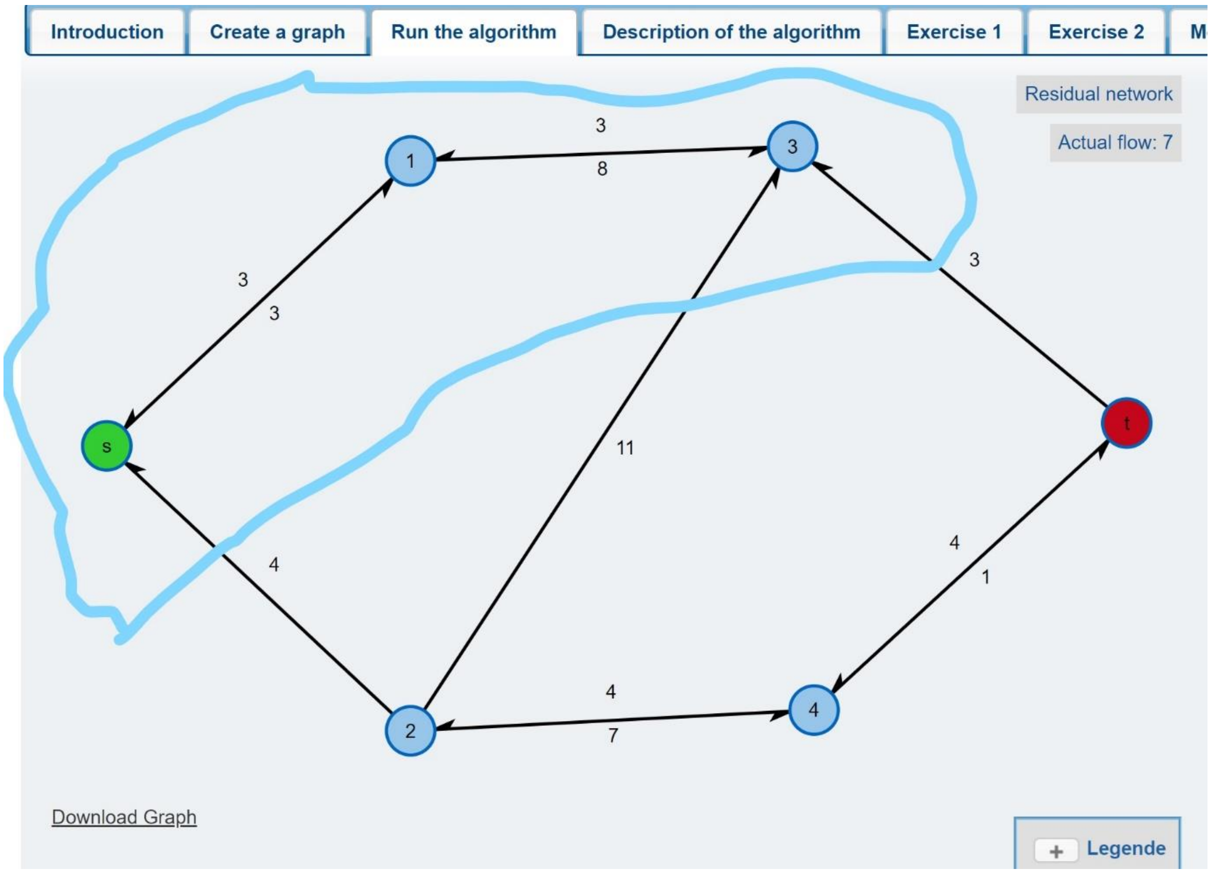
Runtime, pseudocode and more

We need to find the minimum cut corresponding to the maximum flow. The set of projects to select will be those that are connected to the source s in the minimum cut.

To find the minimum cut we need to find the residual graph for this maximal flow. To do this you should select the "prev" tab twice to go back two steps in the algorithm. This yields the following residual graph:



Many of the edges have capacity in both directions but some do not. Here there is an edge from s to A (node 1), and from A to C (node 3). However no other nodes are reachable from s, 1 or 3. Therefore this is the min cut set:



The projects in the min cut connected to s are the projects A and C, so the selection of projects giving the maximum profit is {A, C}.

The next three exercises can be approa ched in each case by editing the graph you are already using, so you do not need to create a new graph from scratch in each case.

Exercise 1.b:

Now work out the best set of projects to select for the following set of projects and dependencies:

Project	p_v	Dependencies
A	6	C
B	4	C, D
C	-3	D
D	-5	

Exercise 1.c:

Project	p_v	Dependencies
A	6	C
B	4	C, D
C	-7	
D	-2	

Exercise 1.d:

Project	p_v	Dependencies
A	6	C
B	4	C, D
C	-7	
D	-5	

Exercise 1e:
Now consider a larger set of projects, with more dependencies as follows:

Project	Profit/cost	Dependencies
A	2	F, H
B	4	G, I
C	6	H, J
D	8	I, K
E	10	F, J
F	-1	
G	-3	
H	-5	
I	-7	
J	-9	
K	-11	

What is the set of projects with the maximum profit?

The min cut given by the residual graph is as follows:

Exercise 1f:

Project	Profit/cost	Dependencies
A	5	F, H, I
B	4	F, G, J
C	6	I, J, K
D	8	J
E	10	F, I, K
F	-4	
G	-3	
H	-5	
I	-10	
J	-6	
K	-5	

Exercise 1g: The same projects and dependencies as exercise 1f except remove the dependency from E to F.

Exercise 1h: Find one dependency to remove from the problem of 1g so that the optimal solution has all of the profit making projects (i.e. contains all of A, B, C, D, E). Verify that your answer is correct by computing the min cut.

2. Survey Design (slides 64 – 68)

The Survey Design problem. We have a bipartite graph with a number of links from one set L to another set R: Each node in L and each node in R has a constraint on the number of links to be selected: a range, from a lower bound, to an upper bound. We need to select a collection links so that each node has a number of edges within its range.

One example is links between customers and products thjat they have purchased. In that case the problem is formulated as follows:

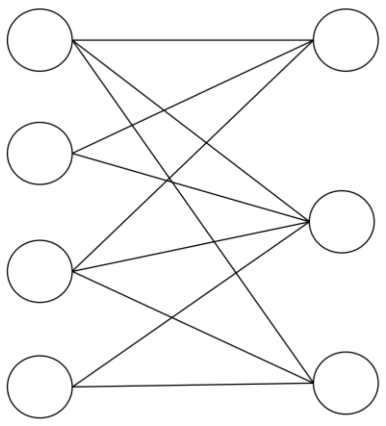
We want to select a collection of edges so that

- Design survey asking n_1 consumers about n_2 products.
- Can only survey consumer i about a product j if they own it.
- Ask consumer i between c_i and c_i' questions.
- Ask between p_j and p_j' consumers about product j.

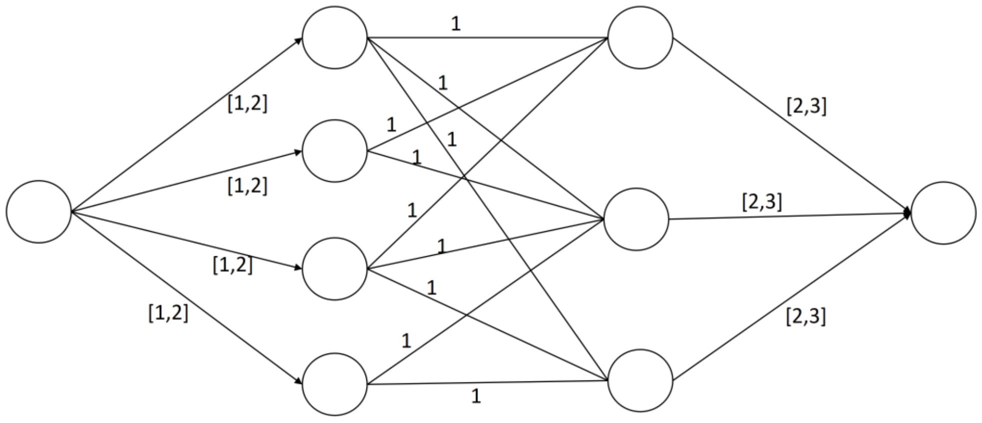
Goal. Design a survey that meets these specs, if possible.

This can be reduced to a Circulation with Demand problem, which can then be reduced to a Network Flow problem and fed into the tool, as described in the lectures. The first exercise will take you through the series of steps. The following exercise will just pose the question for you to solve and you have to follow the same steps yourself.

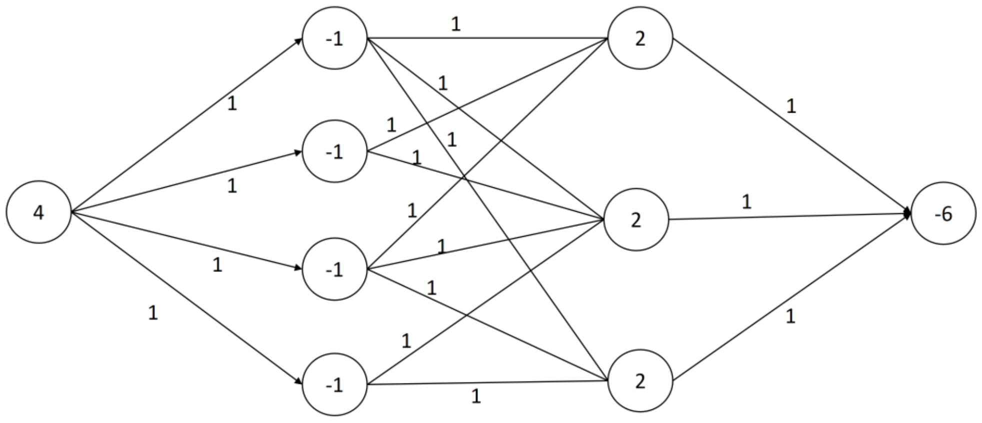
Here we have a bipartite graph:



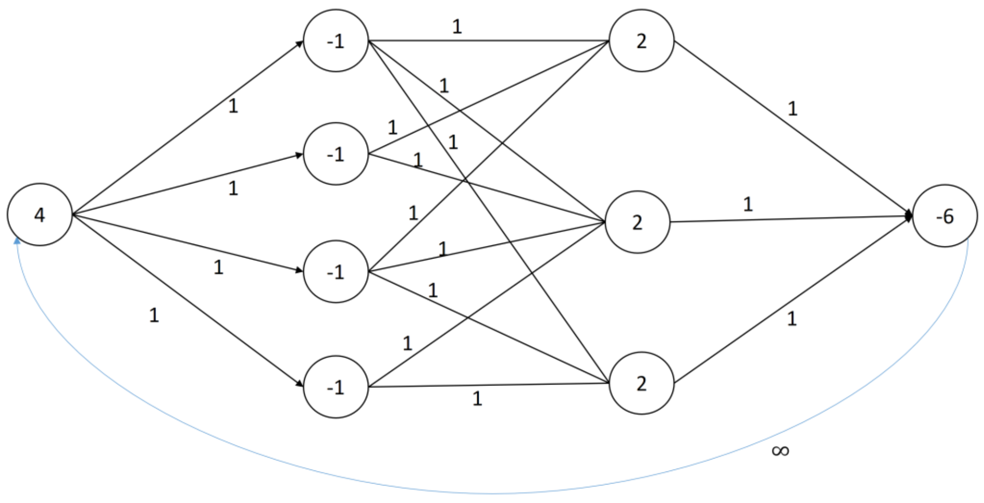
We will introduce the requirement that every node on the left must have between 1 and 2 links, and every node on the right must have between 2 and 3 links. These are incorporated by introducing s and t with the ranges labelling the edges. The original edges are all labelled with capacity 1:



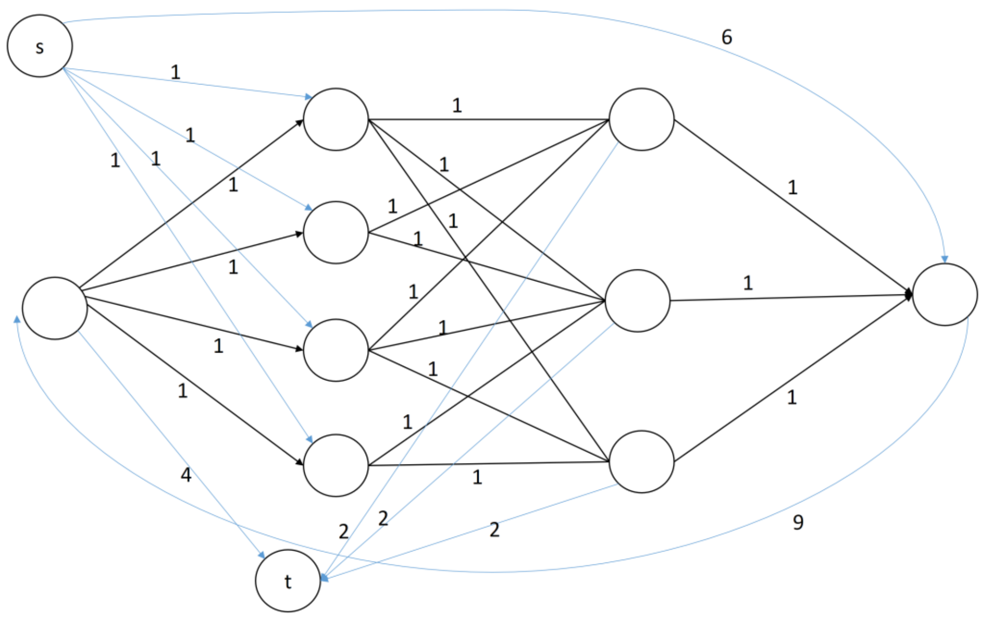
This can now be reduced to a Circulation with Demands problem (see slides 58-59) by initially labelling each node with demand 0, and then adjusting the demands according to the lower bound on the edge, as explained in slide 59. This results in the following graph:



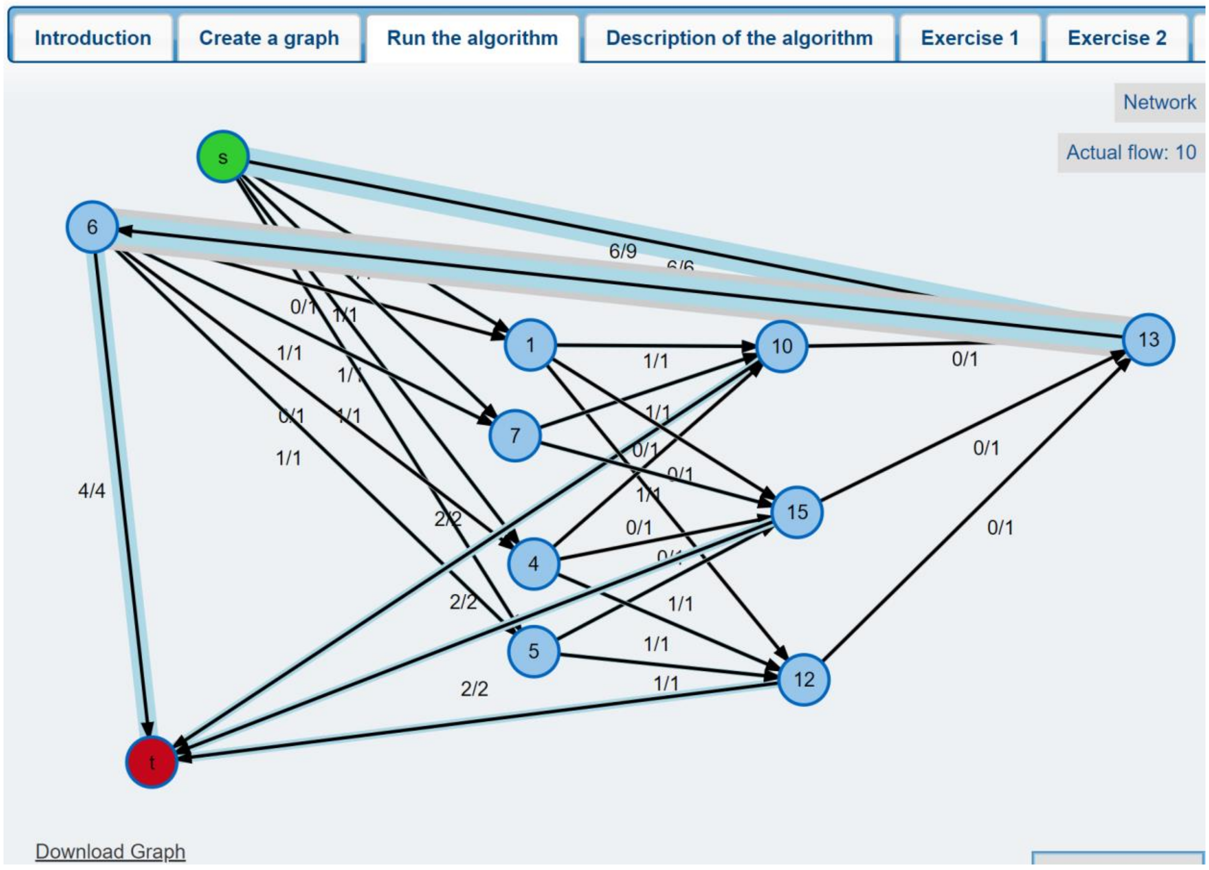
For technical reasons we need to connect the sink t to the source s (with infinite capacity) to complete the circulation:



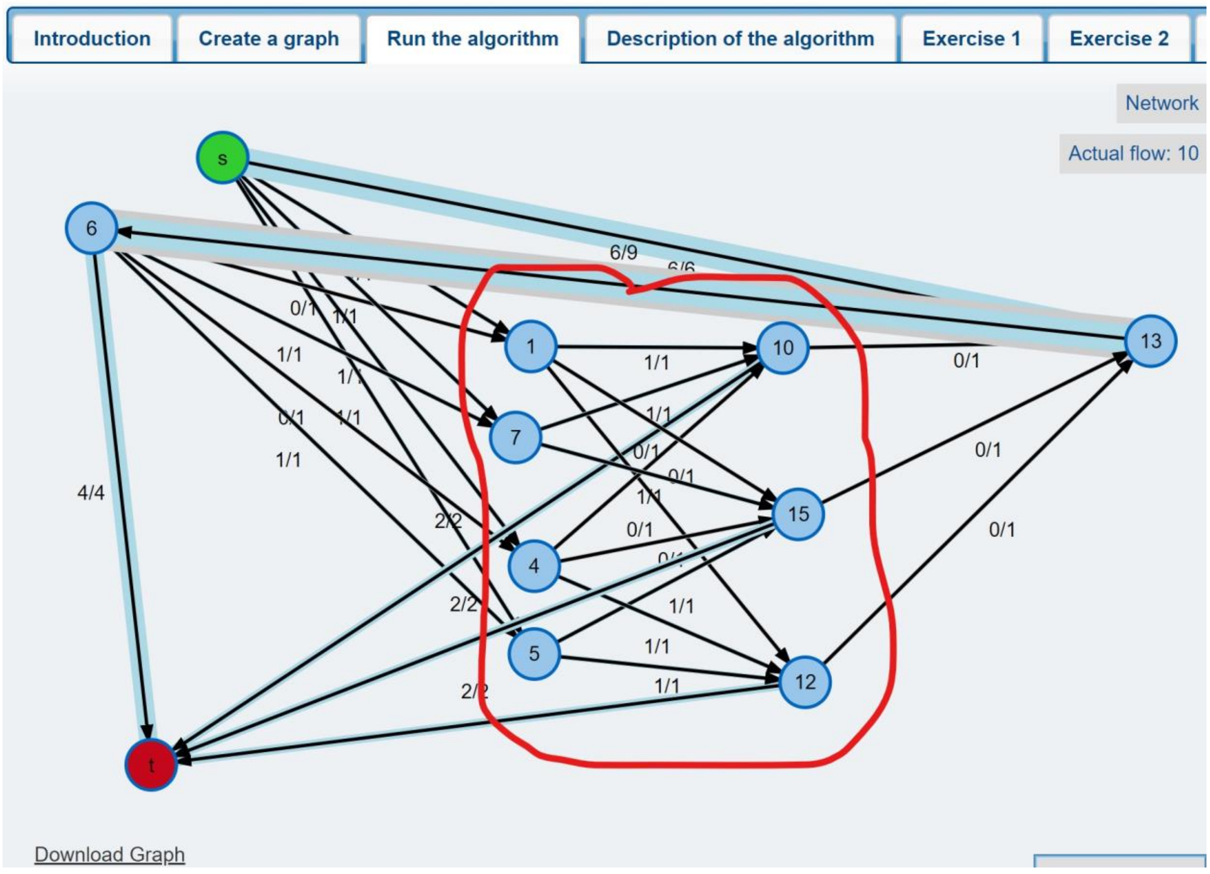
This is now a circulation with demands problem, so it can be reduced to a network flow problem by introducing a new source and sink and connecting the source to nodes with supply (negative values), and the sink with nodes with demand (positive values), as explained in slides 46-49:



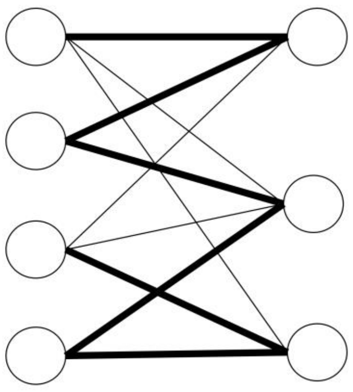
This can now be run through the tool to find the maximum flow through the network. This yields the following:



The flows along the edges between the nodes in L and the nodes in R give the result to the original survey design question: where there is a flow of 1 then the edge should be included, and where there is a flow of 0 the edge should not be included. This is the region to look at:



Hence the solution to the original Survey Design problem is as follows



The original constraints are met: for the set of selected edges, each node on the left has 1 or 2 edges, and each node on the right has between 2 and 3 edges.

Exercise 2b: Now you need to work out your own one:

Here is a bipartite graph. We wish to select a set of edges so that every node on the left has between 1 and 3 edges (range [1,3]); and every node on the right has between 3 and 4 edges (range [3,4]).

Apply the method of the previous question to find such a selection.