Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
H. Ghodselahi, Y. Maus January 18, 2017

Algorithm Theory, Winter Term 2016/17
Problem Set 5
Sample Solution

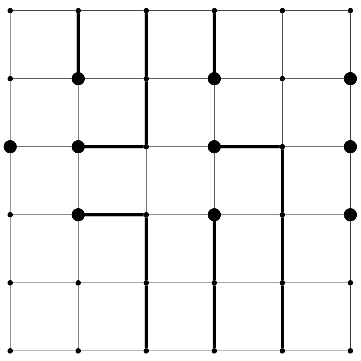
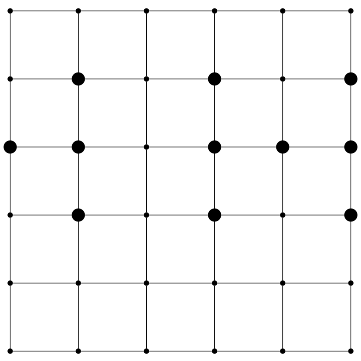
Exercise 1: Circuit Board, Conducting Paths (14 points)

A circuit board with a rectangle grid is given. Some designated points of the grid are connectors and need to be connected with ports via conducting paths. The ports can be anywhere on the rim of the grid. The conducting paths need to run along the grid lines and conducting paths are not allowed to cross.

Design an algorithm which finds a solution for a given grid with designated points or indicates that there is no solution. Your algorithm should run in polynomial time.

Remark 1: If you use any flow network, describe it explicitly.

Remark 2: There is a solution in the left hand side example but no solution in the right hand side example.

Solution*

Solution:

The problem is reduced to a max flow problem with vertex disjoint paths (note that any two vertex disjoint paths are also edge disjoint).

Let $A = \{a_1, \ldots, a_k\}$ be the connectors and $B = \{b_1, \ldots, b_l\}$ the ports on the rim of the graph. Construct a flow network as follows:

1) Start with the grid as a graph, where each grid line is substituted with two directed edges.

2) **Every edge in the flow network will have capacity** 1.

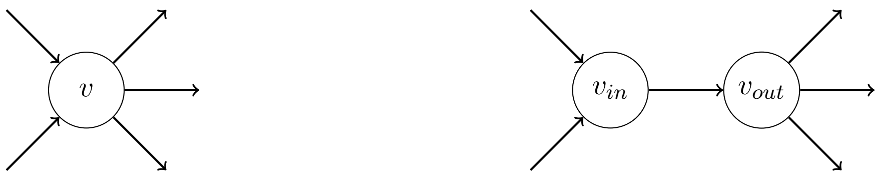
3) Add a source s and a sink t.

4) Connect the source with every connector node, that is, introduce an edge (s, a_i) for $i = 1, \ldots, k$.

5) Furthermore connect every port with the sink t, that is, introduce an edge (b_i, t) for $i = 1, \ldots, l$.

Now a maximum flow of value k (if it exists) in the above network leads a solution to the given problem neglecting that conductor paths are not allowed to cross. We overcome this by ensuring that the flow through every node is at most 1 (vertex disjoint paths).

6) This can be achieved by substituting each node v with two nodes v_{in} and v_{out} and redirecting its edges. Every edge into v is substituted with a corresponding edge into v_{in} and every edge leaving v is substituted with an edge from v_{out}. Furthermore we add an edge (v_{in}, v_{out}) (e.g., see the following example).



The total capacity of all edges leaving s is k, so the value of a max flow will be smaller or equal to k. If there is a max flow with value k there will be vertex disjoint (and path disjoint) paths from the connectors to the ports (lecture).

The max flow problem can be solved in polynomial time.

Exercise 2: *Smallest* Minimum Cut (14 points)

Suppose that you wish to find, among all minimum cuts in a flow network G with integral capacities, one that contains the smallest number of edges. Show how to modify the capacities of G to create a new flow network G' in which any minimum cut in G' is a minimum cut with the smallest number of edges in G.

Solution:

Given the flow network G we will produce a flow network G' such that any minimum cut in G' is a minimum cut in G with the smallest number of edges (note that the minimum cut with the smallest number of edges might not be unique). Then we execute any max-flow algorithm on G', determine any min-cut of G' and return the corresponding cut in G.

Let $G = (V, E)$ be a flow network with integer edge capacities $c_e \geqslant 0, e \in E$. We define the flow network $G' = (V, E)$ with edge capacities $c'_e \geqslant 0, e \in E$ and we set $c'_e := |E| \cdot c_e + 1$.

Claim: Any min-cut in G' is a min-cut of G with a smallest number of edges.

Let $(S, V\backslash S)$ be a min-cut in G' and let F be the set of edges crossing the cut. Then the capacity of the cut in G' is $\sum_{e\in F} c'_e = \sum_{e\in F}(|E|c_e + 1) = |F| + |E| \sum_{e\in F} c_e$ and the capacity of the corresponding cut in G is $C_G(S) := \sum_{e\in F} c_e$.

We prove the claim by contradiction. At first assume that $(S, V\backslash S)$ is not a min-cut in G, i.e., there is a cut $(S', V\backslash S')$ in G with capacity smaller than $C_G(S)$. Let F' be the edges crossing this cut. Then the cut in G' has capacity

$$\begin{aligned}
\sum_{e\in F'} c'_e &= |F'| + |E| \sum_{e\in F'} c_e && (1)\\
&\overset{*}{\leqslant} |F'| + |E|(-1 + \sum_{e\in F} c_e) && (2)\\
&\leqslant |E| + |E|(-1 + \sum_{e\in F} c_e) && (3)\\
&= |E| \sum_{e\in F} c_e < |E| \sum_{e\in F} c_e + |F| = C_{G'}(S). && (4)
\end{aligned}$$

This is a contradiction to $(S, V\backslash S)$ being a minimum cut of G'. At $*$ we used that all capacities are integers and thus the capacity of $(S', V\backslash S)$ is at least one smaller than the capacity of the cut $(S, V\backslash S)$.

Now, we know that $(S, V\backslash S)$ is a minimum cut of G. Now, assume that there is a minimum cut $(S', V\backslash S')$ of G that uses fewer edges than the cut S. Again, let F' be the edges crossing this cut. Then we can perform a similar calculation as above and we obtain that the cut S' in G' has capacity

$$\begin{aligned}
\sum_{e\in F'} c'_e &= |F'| + |E| \sum_{e\in F'} c_e && (5)\\
&\leqslant |F'| + |E| \sum_{e\in F} c_e && (6)\\
&< |F| + |E| \sum_{e\in F} c_e = C_{G'}(S). && (7)
\end{aligned}$$

This is a contradiction to $(S, V\backslash S)$ being a minimum cut of G', which proves the claim.

Exercise 3: Route Planning with an Intermediate Stop (12 points)

You are given an unweighted undirected graph $G = (V, E)$ and three nodes u, v and w. Devise an efficient algorithm which finds a path from u to w that visits v and visits every node of G at most once.

Solution

Crucial in this exercise is the concept of node demands in a flow network (as presented in the lecture) which we use as a blackbox.

Given the graph G we define the following flow network $G' = (V', E')$ where $V' := V$ and E' is obtained from E by substituting every edge by two directed edges. All edge capacities are set to 1. Now we add the demands for the nodes. Node v gets demand -2 and is thus the only source of the network. Nodes u and w both get demand 1 and are the only sinks of the network. Every other node gets node demand 0 and is neither a source nor a sink. Now we furthermore modify the network (to obtain a flow network G'') with the same trick as in Exercise 1, 6) so ensure that every node has at most 1 unit of flow going through it, i.e., we split every node into two nodes and add an edge with capacity one in between.

Now it is easy to see that a maximum flow in G'' which satisfies the demands provides a path from u to w via v that visits every node at most once. In particular there are edge disjoint paths, one from v to u and one from v to w. Due to the modification we performed on the network these edge disjoint paths are also node disjoint (except for the starting node v). As the graph is undirected we can patch both paths together to obtain a path from u to w via v which visits every node at most once.

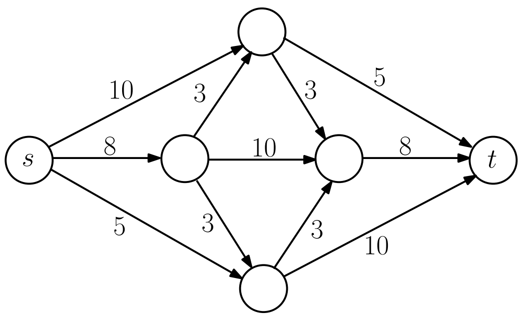
Furthermore any such path provides two non conflicting augmenting paths in the flow network G'' such that a flow satisfying the demands can be found.

Exercise 4*: Ford Fulkerson Algorithm (10^* points)

Remark: This exercise is optional and it does not increase the threshold for exam admittance. However, your points in this exercise contribute to reach the 50% threshold to be admitted for the exam.

This exercise will not be presented in the tutorial session. However, a sample solution will be provided.

Consider the following flow network:



a) **(7 points)** Solve the maximum flow problem on the above network by using the Ford Fulkerson algorithm. Give all intermediate results.

b) **(3 points)** Give a minimum capacity s-t cut of the given network. Describe how you can get the cut from the maximum flow computed in a).

The Ford-Fulkerson algorithm starts by looking for an augmenting $s-t$-path in the residual graph, and after finding such a path it updates the flow and residual capacity values of the edges in the residual graph. This procedure is repeated until no more $s-t$-path is found in the residual graph. At the beginning of the algorithm, the flow is zero, so we initialize all backward edges in the residual graph with capacity 0. Next we search for an $s-t$-path in the residual graph, and update the flow values of all edges. For convenience, when we look for a path, we choose the (graphically) top free edge in the network. On each iteration of the algorithm, we search for the minimum capacity value on the path, update the flow and change the capacities of the residual graph. The set of figures 1-3 show the resulting graph step-by-step. The selected path in each step is shown in red.

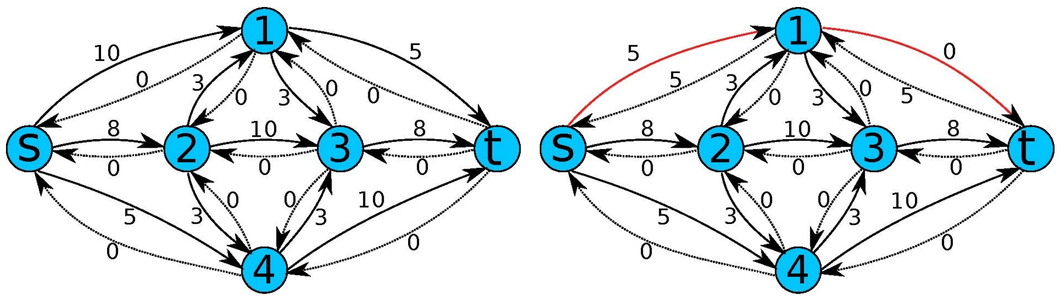


Figure 1: Residual graph at the beginning of the algorithm and after the first step. Currently considered path is $p_{s,1,t}$

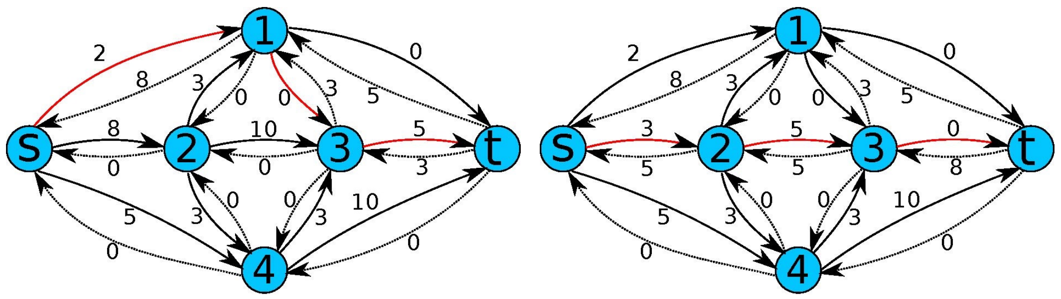


Figure 2: Residual graph at steps 2 and 3 of the algorithm. Considered paths are $p_{s,1,3,t}$ and $p_{s,2,3,t}$

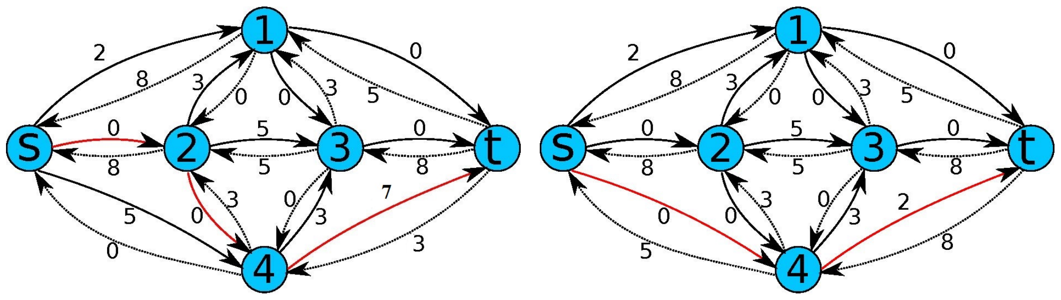


Figure 3: Residual graph at steps 4 and 5 of the algorithm. Considered paths are $p_{s,2,4,t}$ and $p_{s,4,t}$.

Step 1: Path is $p_{s,1,t}$. The smallest capacity on the path is 5, see figure 1.

Step 2: Path is $p_{s,1,3,t}$. The smallest capacity on the path is 3.

Step 3: Path is $p_{s,2,3,t}$. The smallest capacity on the path is 5, see figure 2.

Step 4: Path is $p_{s,1,3,t}$. The smallest capacity on the path is 3.

Step 5: Path is $p_{s,2,3,t}$. The smallest capacity on the path is 5, see figure 3.

After Step 5 there are no more $s-t$-paths left in the graph, see the left graph on figure 4. Edges with 0-capacities have been removed for clarification.

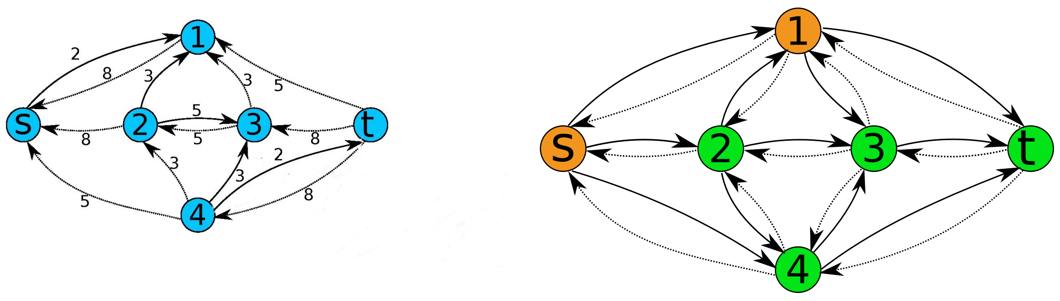


Figure 4: Residual graph at the end of the algorithm (left). Minimum cut on the network(right).

To find the the minimal $s-t$-cut on the graph, we can use the maximum flow computed in the previous part. The set of vertices that are reachable from s is one part of the cut and the rest of vertices will form the other one. These sets form two components, which define a minimal $s-t$-cut on the given network. The result is shown on figure 4 on the right.

Note that in general the computed max-flow/min-cut are not unique, i.e., your solution might differ from this one.