Minimum-cost Flow

Assignment 1

Advanced Algorithms and Data Structures, block 1, 2015

Christian Wulff-Nilsen

This is the first of two mandatory assignments for the course Advanced Algorithms and Data Structures, block 1, 2015. The assignment should be solved in groups of 2 or 3 students. Permission for individual assignments is only granted in special cases. The assignment is due September 12 at 22:00. To pass the assignment the group must have completed most of the questions satisfactorily. To be allowed to resubmit the group must have made a reasonable attempt at answering most of the questions. The resubmission is due September 24 at 22:00. Completed assignments must be uploaded to Absalon. All descriptions and arguments should be kept concise while still containing relevant points.

Exercise 1: *b*-flow

Let $G = (V, E)$ be a directed graph. For each vertex $v \in V$, let $\delta^+(v)$ be the set of outgoing edges from v and $\delta^-(v)$ be the set of incoming edges to v.

Each edge $e \in E$ has a *capacity* $u_e \in \mathbb{R} \cup \{\infty\}$, and each vertex $v \in V$ has a so called *demand* $b_v \in \mathbb{R}$.

Our goal (if possible) is to find a *b-flow*, which is an assignment of real numbers x_e to each edge $e \in E$, such that

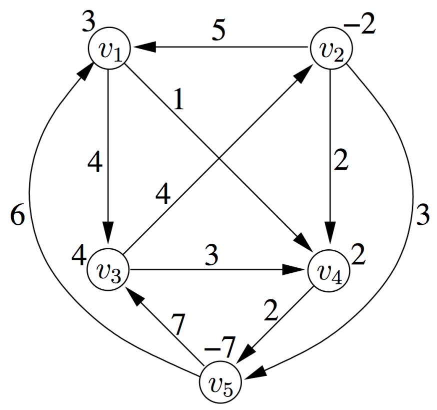
$$\sum_{e\in\delta^-(v)} x_e - \sum_{e\in\delta^+(v)} x_e = b_v, \forall v \in V,$$

$$0 \le x_e \le u_e, \forall e \in E.$$

In addition to capacities and demands, we also have a cost $c_e \in \mathbb{R}$ for each edge $e \in E$. Given a b-flow $\{x_e | e \in E\}$ in G, we define its cost as $\sum_{e\in E} c_e x_e$.

In the *minimum-cost flow-problem (MCFP)* we are looking for a b-flow in G of minimum cost. In this first exercise, we shall ignore edge costs.

Find, for each of the two graphs in Figure 1, a b-flow for the graph or argue that the graph has no b-flow.



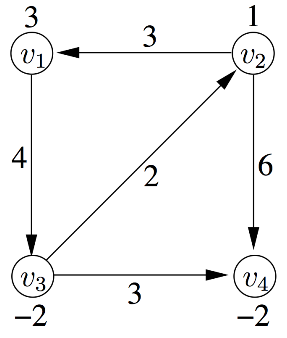


Figure 1: Graphs with capacities and demands (for instance vertex v_2 has demand -2 and the edge (v_1, v_3) has capacity 4 in (a)).

Exercise 2: An application of MCFP: rectilinear planar embedding

To solve this exercise, you are referred to pages 843–847 in CLRS, where the terms linear constraint, objective function, and linear program are defined.

We will now look at an application of MCFP to a geometric problem in the plane. In the following we make the following assumptions:

- $G = (V, E)$ is an undirected graph.
- For technical reasons, we assume that G is connected, consists of at least three vertices, and has no cut vertices, i.e., vertices whose removal yields a disconnected graph (you may simply ignore these details).
- No vertices in G have degree greater than 4.

We define a *planar embedding* of G as a drawing of G in the plane such that no two edges intersect (except possibly at their endpoints). An example is given in Figure 2. If in addition the graph is drawn such that each edge consists exclusively of vertical and horizontal line segments, we say that the embedding is *rectilinear* (see Figure 3). For such an embedding we define *breakpoints* to be the points where the horizontal and vertical line segments from the same edge meet.

A planar embedding divides the plane into a number of regions called *faces*; see Figure 2(b). Exactly one of these faces is unbounded, and is called

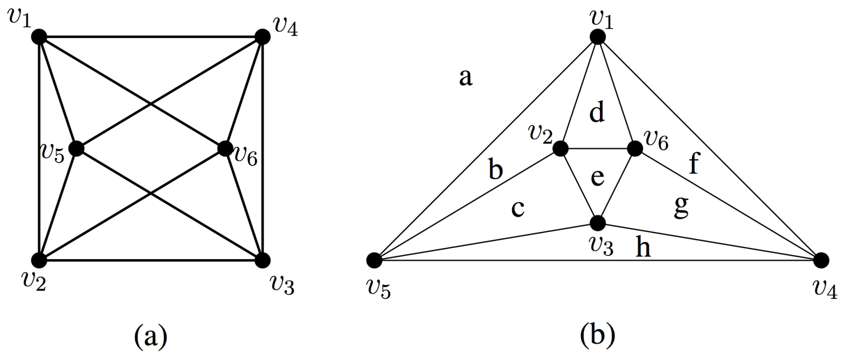


Figure 2: (a): A graph and (b): A planar embedding of this graph. The embedding divides the plane into faces a, b, c, d, e, f, g and h, where a is external.

the *external face*. The vertices and edges of a face form a simple cycle (this will not be shown but follows from one of our assumptions above). We call such a cycle a *boundary cycle*.

Suppose now that we have a planar embedding of G. Then define a *rectilinear layout* of G to be a rectilinear embedding of G, containing the same boundary cycles as the given embedding.

The problem considered in this exercise is to find a rectilinear layout of G with a minimum number of breakpoints. The idea is to formulate the problem as an instance of MCFP by introducing variables that keep track of the number of breakpoints while maintaining the feasibility.

In the following we will consider a rectilinear layout which we do not know explicitly; we only know the associated planar embedding. Given a boundary cycle f of this rectilinear layout and a vertex v in f, we say that f makes an *inner turn* in v if the two edges in f incident to v form an angle of strictly less than 180° viewed from f. If the angle is strictly greater than 180°, f is said to make an outer turn in v. An example is given in Figure 3. We define inner and outer turns of breakpoints similarly. For each boundary cycle f and each vertex v in f we introduce a variable x_{vf} whose value is defined as follows:

$$x_{vf} = \begin{cases} 1 & \text{if } f \text{ makes an inner turn in } v \\ -1 & \text{if } f \text{ makes an outer turn in } v \\ 0 & \text{otherwise.} \end{cases}$$

Let f and g be two boundary cycles and let E_{fg} be the (possibly empty) set of edges shared by f and g. We define the variable z_{fg} to be the number of inner turns that f makes in the breakpoints of edges of E_{fg}. Note that z_{fg} and z_{gf} are two different variables.

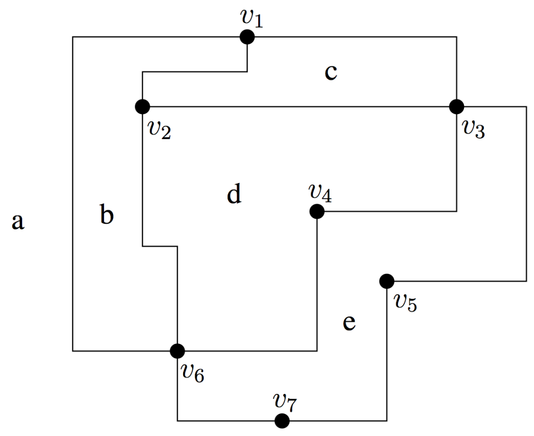


Figure 3: A rectilinear layout of a graph. Example: boundary cycle d makes an inner turn in v_2, v_3 and v_6 and an outer turn in v_4.

Exercise 2.1

Give the values of all variables z_{fg} and x_{vf} for the rectilinear layout in Figure 3. How many breakpoints are there?

Draw a rectilinear layout of the graph from Figure 2(a) given the planar embedding in Figure 2(b).

Exercise 2.2

Let f be a boundary cycle in a rectilinear layout. It can be shown that the number of inner turns minus the number of outer turns that f makes in its vertices and breakpoints is -4 if f corresponds to the external face and 4 otherwise.

State these constraints as linear constraints in the previously introduced variables.

Verify that the constraints hold for boundary cycles a and e in Figure 3.

Exercise 2.3

In the beginning of the exercise, we assumed that no vertex in G has degree greater than 4. Why is this assumption necessary?

It also follows from our assumptions about G that no vertex has degree less than 2 (You are not required to show this). Show that the following holds for each vertex v:

$$\sum_f x_{vf} = \begin{cases} 0 & \text{if } v \text{ has degree 2} \\ 2 & \text{if } v \text{ has degree 3} \\ 4 & \text{if } v \text{ has degree 4,} \end{cases} \tag{1}$$

where the sum is over all boundary cycles f containing v.

Exercise 2.4

We want to minimize the number of breakpoints. Formulate an objective function whose minimization ensures this. The objective function must be expressed in the previously introduced variables.

Write a linear program with this objective function and the following constraints:

1. The constraints from Exercise 2.2.

2. The constraints from Exercise 2.3.

3. All variables z_{fg} are non-negative.

The following theorem can be used without proof: *An optimal solution to the given linear program has integer values for the variables x_{vf} and z_{fg}.*

Exercise 2.5

Rewrite the linear program from Exercise 2.4 to an instance of MCFP. It is important to define the constraints such that they are on the form specified in Exercise 1.

Describe which vertices and edges belong to the associated graph, and give the demands for the vertices and for the capacities and costs of the edges.

Next, write the instance of MCFP corresponding to the embedding in Figure 2(b).

Hint: make an appropriate substitution of certain variables in order to ensure that they are non-negative.

Comments on Exercise 2

It can be shown that an optimal solution to the above instance of MCFP corresponds to a rectilinear layout, with a minimum number of breakpoints. In addition, it can be shown that it is relatively easy to find this layout, when the optimal solution for the MCFP-instance is found. To limit the scope of the assignment we will not focus on this.

Exercise 3: Reduction to MCFP

In Exercise 2, we had to modify certain variables in order to ensure that they were non-negative, as required in MCFP.

In this exercise, we consider a (seemingly) more general problem than MCFP. The variant allows both the upper and lower limits of the capacities to be arbitrary constants. In particular negative variable values are allowed.

Consider a graph G as in Exercise 2 and suppose further that for each edge $e \in E$ there is an associated *lower* capacity limit $l_e \in \mathbb{R} \cup \{-\infty\}$.

The problem of general minimum-cost flow problem (GMCFP) is to assign values x_e for each edge e with minimal cost $\sum_{e\in E} c_e x_e$, such that

$$\sum_{e\in\delta^-(v)} x_e - \sum_{e\in\delta^+(v)} x_e = b_v, \forall v \in V,$$
$$l_e \le x_e \le u_e, \forall e \in E.$$

Note that any instance of MCFP is an instance of GMCFP. Thus an algorithm that solves GMCFP also solve MCFP.

In this exercise, we wish to show that in a certain sense, it is also applicable the other way, that is, an algorithm for MCFP can be used to solve GMCFP. We will do so by performing what is known as a reduction from GMCFP to MCFP.

We define such a *reduction* as a mapping f from the set of GMCFP-instances to the set of MCFP-instances, such that:

1) if I_{GMCFP} is an instance of GMCFP with the vertex set V and edge-set E, then, $f(I_{GMCFP})$ is an instance of MCFP with $O(|V|+|E|)$ vertices and edges,

2) given an optimal solution to $f(I_{GMCFP})$, it can be converted to an optimal solution to I_{GMCFP}.

These requirements will ensure that if we have an algorithm $\mathcal{A}$ for MCFP then we get an algorithm for GMCFP by first reducing an instance of GMCFP to an instance of MCFP and then using $\mathcal{A}$ on this new instance. The first requirement limits the size of the new instance. The second requirement ensures that the found optimal solution (if one exists) can be converted into an optimal solution to GMCFP.

In Exercises 3.1 to 3.3, we will modify the general instance of GMCFP to an instance of MCFP in three steps. In Exercise 3.4, we bound the size of the transformed instance. It is important that these steps constitute a reduction, so for instance in Exercise 3.1, you should explain how a solution to I_0 can be obtained from a solution to I_1.

Assume we have an instance I_0 of GMCFP, i.e., we are given a directed graph $G = (V, E)$ with values of the variables l_e, u_e, b_v and c_e.

Exercise 3.1

Modify I_0 to an instance I_1 of GMCFP, where for each edge e at least one of the variables l_e and u_e is finite.

Exercise 3.2

Modify I_1 to an instance I_2 of GMCFP where for each edge e, l_e is finite.

Exercise 3.3

Modify I_2 to an instance I_3 of GMCFP where for edge e, $l_e = 0$.

Exercise 3.4

Note that instance I_3 in Exercise 3.3 is an instance of MCFP. Show that it has $O(V + E)$ vertices and edges.