

# Flows and cuts

Éric Schost

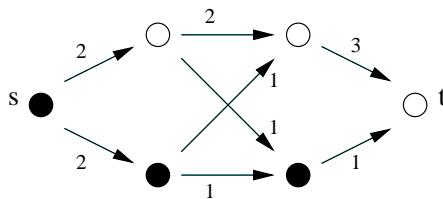
September 13, 2007

## 1 What to read

We mainly follow Chapter 7 of [2], which gives a very good presentation of maximum flows and minimal cut problems. This question is also discussed in Chapter 26 of [1], with slightly different conventions. The former reference gives nice and detailed applications; the latter is slightly more complete regarding improved flow algorithms.

## 2 Problems

1. Give the capacity of the cut in the following graph, where  $A$  is in black and  $B$  in white.



2. Run Ford and Fulkerson's algorithm on the previous graph and give a minimal cut (at all steps of the algorithm, give the flow, the residual graph and the augmenting path you choose).
3. Write a *clean* and *complete* proof of the correctness of the improvement step in Ford and Fulkerson's algorithm: assuming that  $f$  is a flow with integer values, prove that after the improvement,  $f'$  is still a flow with integer values, and that  $\text{Val}(f') \geq \text{Val}(f) + 1$ .
- 4: **Breadth-first search.** Consider an oriented graph  $G$  with  $n$  vertices and  $m$  edges, given by adjacency lists, that is as an array `Edge` of  $n$  lists, where `Edge[i]` contains the indices  $j$  such that  $(v_i, v_j)$  is an edge of  $G$ .

We suppose that we have a data structure *set*, in which we can add elements in time 1.

We want to compute a shortest path from the source  $v_0$  in  $G$  to all other vertices. Our algorithm is the following:

- Initialize an array  $L$  of size  $n$  (indices start at 0). The entries of  $L$  will be sets, initially empty, except for  $L[0] = \{0\}$ .
- Initially an array  $D$  of size  $n$ . The entries of  $D$  will be integers, initially  $\infty$  except for  $D[0] = 0$ .
- For  $i = 1, \dots, n - 1$  do
  - For  $j$  in  $L[i - 1]$  do
    - For  $k$  in  $\text{Edge}(j)$  do
      - If  $D[k] = \infty$  then
        1.  $D[k] = i$
        2.  $L[i] = L[i] \cup \{k\}$

Draw the graph corresponding to  $n = 5$  and

$\text{Edge}[0] = [1, 4]$ ,  $\text{Edge}[1] = [0, 3, 4]$ ,  $\text{Edge}[2] = [1, 2, 3]$ ,  $\text{Edge}[3] = [3]$ ,  $\text{Edge}[4] = []$ .

Give the steps of the algorithm on this example.

Prove that at the end of the algorithm,  $k$  is in  $L[i]$  if and only if  $D[k] = i$  (for  $D[k] \neq \infty$ ).

Prove that at the end of the algorithm,  $D[k]$  is the distance (the length of the shortest path) between  $v_0$  and  $v_k$ . *This is not easy to do properly, so you can get full marks without completing this question. But give it your best try!*

Show that the complexity is  $O(n + m)$ .

Finally, how can you modify the algorithm to find a path  $v_0 \rightarrow v_k$  as well?

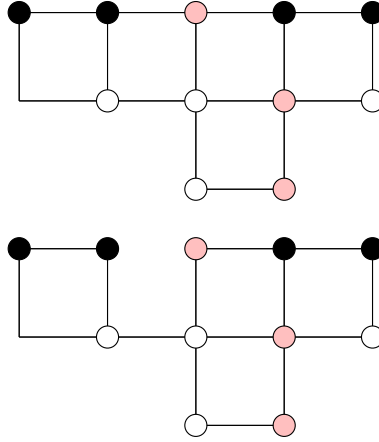
**5: Constrained flows.** A constrained flow problem is a maximal flow problem, together with extra constraints at the vertices.

For each vertex  $v$ , we are given an integer value  $s(v)$ ; we want to find a maximal flow with the constraint that for any  $v$ , the incoming flow at  $v$  is at most  $s(v)$ .

Show that it is possible to solve constrained flow problems on a graph  $G$  by applying classical flow algorithms on a new graph  $G'$ , that can be constructed from  $G$ .

**6: Escape plan.** Consider a *symmetric* graph  $G$  and two disjoint subsets  $Y$  and  $Z$  of its vertices: the vertices in  $Y$  are the rooms or offices (in a building, ...) and those in  $Z$  are the exits.

An escape plan is a set of paths in the graph, such that none of these paths have a common vertex, and all these paths connect a place to an exit. Indicate which of the following graphs admit an escape plan of size 4 ( $Y$  is in white,  $Z$  in black and the other vertices in pinkish).



Show how to use (constrained) flow algorithms to find an escape plan with a maximal number of paths (it may be useful to prove that the maximal flow you consider has values 0's and 1's only). Which does better, Ford-Fulkerson or Edmonds-Karp?

**7. An optimization problem.** *In this problem, we will use flows and cuts on graphs having some infinite capacities. This is not an issue; we will admit that all results from the lecture are still OK in this context.*

The NASA is planning a flight of its space shuttle. Industrial partners are willing to pay for the results of some experiments  $E = \{E_1, \dots, E_m\}$ ; the revenue generated by experiment  $E_i$  is  $p_i$  dollars.

Doing the experiments requires some material. The overall set of instruments is  $I = \{I_1, \dots, I_n\}$ ; each experiment  $E_i$  requires a subset  $R_i$  of  $I$ , and transporting the instrument  $I_j$  costs  $c_j$  dollars.

We want to determine what experiments to schedule to maximize profit. To do so, we consider a graph  $G$  whose vertices are  $s, t, I_1, \dots, I_n, E_1, \dots, E_m$ . The edges of  $G$  are the following:

- For each  $I_k$ , there is an edge  $(s, I_k)$  of capacity  $c_k$ .
- For each  $E_j$ , there is an edge  $(E_j, t)$  of capacity  $p_j$ .
- For each  $I_k, E_j$ , if  $I_k$  is needed for experiment  $E_j$ , there is an edge  $(I_k, E_j)$  of infinite capacity.

1. Show that the minimal cut has a finite capacity.
2. Show that if  $(A, B)$  is a cut of finite capacity and the experiment  $E_k$  is in  $B$ , then all instruments needed for  $E_k$  are in  $B$  as well.

Let  $E' \subset E$  be a set of experiments and let  $I' \subset I$  be a set of instruments. The pair  $(E', I')$  is *coherent* if all instruments needed by  $E'$  are in  $I'$ .

Then, the *profit* of a coherent pair is the sum of the revenues generated by the experiments in  $E'$ , minus the sum of the cost of the instruments in  $I'$ .

3. Show that you can associate to any cut  $(A, B)$  of finite capacity a coherent pair  $(E', I')$ , and conversely.  
Prove that  $c(A, B) = \sum_{i \leq m} p_i - \text{profit}(E', I')$ .
4. Deduce an algorithm for finding the optimal schedule, that will maximize profit.  
What is its complexity?

8. Give algorithms for the following problems:

- Finding if an integer  $x$  belongs to a linked list of integers
- Given two linked list of integers,  $L$  and  $M$ , sorted in increasing order, compute a list that contains all integers that belong to  $L$  and  $M$  (the output does not have to be sorted).
- Same question, when  $L$  and  $M$  are not sorted.

In all cases, give the complexity of your algorithm. The faster the better.

9. Solve the bipartite matching problem for internships in the case where students also have constraints.

Precisely, each professor  $P_i$  has a list  $L_i = [\ell_{i,1}, \dots, \ell_{i,e_i}]$  of students he would agree to take as intern and each student  $S_j$  has a list  $M_j = [m_{j,1}, \dots, m_{j,f_j}]$  of professors with whom he would want to be an intern.

Give the complexity of your algorithm in terms of the number  $p$  of professors, the number  $s$  of students, the total number of elements in the lists  $L_1, \dots, L_p$  and  $M_1, \dots, M_s$ .

## References

- [1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (Second Edition)*. MIT Press, 2001.
- [2] J. Kleinberg and E. Tardos. *Algorithm design*. Addison Wesley, 2005.