Advanced Algorithms Course. Lecture Notes. Part 5

Bipartite Matching

Here is one of the simplest but also most important examples of a reduction of another graph problem to Maximum Flow.

A **bipartite graph** is a graph $G = (X, Y, E)$ whose node set $X \cup Y$ is split in two sets X, Y, and where edges exist only between X and Y. A **matching** in a graph is a set of pairwise node-disjoint edges. The Bipartite Matching problem asks to find a matching of maximum size in a given bipartite graph. Typical applications include job assignment problems: Nodes in X are jobs to be done, nodes in Y are workers or machines, and an edge indicates that the worker/machine is able to do the job. A matching is then a set of jobs that can be executed in parallel.

Bipartite Matching is reduced to Maximum Flow as follows: Add a source s and a sink t to the graph, insert edges from s to all nodes in X, and from all nodes in Y to t, orient the edges of E from X to Y, and set all edge capacities 1.

We claim that the size of a maximum matching equals the value of a maximum flow. This equation looks simple, but it has some issues and needs a proof, as dicussed below.

Let k be the value of a maximum flow. As we know, there exists a maximum flow such that the flow values on all edges are integer. Since all capacities are 1, the only possible flow values on the edges are 0 and 1. Now, due to the conservation constraints, the edges in E carrying flow 1 build a matching M. Morover, their number equals k.

Note carefully that this is not yet sufficient to conclude that M is a matching with maximum size! (Actually, this point often causes confusion.) Right, we started from a maximum flow. However, we are not obliged to use flows at all, in order to solve the Bipartite Matching problem. This is just one approach, and maybe some other approach yields an even larger matching. In order to exclude this hypothetical possibility, we must also prove the (simpler) opposite direction:

Let k be the size of a maxumum matching M. Then we can obviously construct a flow of value k: Let the flow value be 1 on all edges of M and on all edges connecting M with s and t. – Ironically, it is easy to miss this reverse direction in the equivalence proof *because* it is so simple!

Due to this reduction, the time to solve the Bipartite Matching problem is $O(mC) = O(mn)$; remember what C was. One can also find maximum matchings in general graphs in polynomial time, but this is much more tricky, and we will not address that problem.

Disjoint Paths versus Disconnecting Edge Sets

This section deals with two problems motivated by, e.g., network reliability.

In a directed graph with a source s and a sink t we want to find a maximum number of mutually edge-disjoint directed paths from s to t. That is, these paths are allowed to share nodes, but not edges.

We can solve this problem as follows. Give all edges the capacity 1. If k disjoint $s-t$ paths exist then, obviously, they build together is a flow with $val(f) = k$. The converse is also true: Once we have computed a flow with $val(f) = k$ and integer flow values on the edges (which can be only 0 or 1), we can successively extract k different $s-t$ paths consisting of "1-edges", thanks to the conservation constraints. Note that the extraction is necessary to solve the problem – a flow is not yet a family of paths!

The extraction in more detail: Starting in s we simply follow any path of edges that carry flow 1, until we reach t, and we delete every traversed edge immediately from the graph. Again, due to the conservation constraints, we can never get stuck, and it remains a flow of value $k-1$. Thus we can repeat this procedure inductively. We stress again: Although the extraction phase is simple, the mentioned arguments are needed to show that it really works.

The time is again $O(mC) = O(mn)$, including the time for the extraction phase.

Next we consider a dual problem: Given a directed graph with source s and sink t, we want to remove a set F of edges, with minimum size $|F|$, such that s and t are disconnected, that means, all directed $s-t$ paths are interrupted.

Not surprisingly, the smallest possible size of F equals the maximum number k of edge-disjoint $s-t$ path. This statement is known as **Menger's Theorem**.

For the proof, consider any edge set F whose removal disconnects s and t. Since F must contain some edge from every $s-t$ path, we have $|F| \geq k$.

For the reverse inequality we can use the Max-Flow Min-Cut Theorem: We take our graph and assign to every edge the capacity 1. As seen above, there exists a maximum flow f such that $k = val(f)$. Hence there exists a cut (A, B) with capacity k. Now let F_0 be the set of directed edges from A to B. Clearly, $|F_0| = k$, and the removal of F_0 disconnects s and t. Hence, there exists some solution F (namely F_0) with $|F| \leq k$.

Besides the mere equation, this also shows that we can *construct* F, using the previous algorithms for flows and cuts.

In undirected graphs we can state the same problems and solve them in the same way. We only need a preprocessing step where we replace every undirected edge with two opposite directed edges. It is easy to show that any maximum flow uses at most one of the opposite edges, and correctness follows.

Interestingly, the problem to connect k different source-sink pairs by edge-disjoint paths is NP-complete, but it becomes polynomial-time solvable if all sources and all sinks, respectively, are identical, as we have just shown.

Circulations with Demands and Lower Capacity Bounds

In a colloquial sense, the word *circulation* refers to any steady-state movement of things through a system. Circulations are ubiquitous in nature and society. In the language of graphs, a circulation can be viewed as a flow without source and sink, that is, a flow where all nodes satisfy the conservation constraints.

Now we introduce a notion of circulations that subsumes both flows and circulations (in the colloquial sense). This will also equip us with useful variants of flow problems which can make it easier to reduce other problems to (at last) Maximum Flow, as they can serve as intermediate problems in reductions.

Let $G = (V, E)$ be a directed graph where every edge $e \in E$ has a lower and upper capacity l_e and c_e, respectively, where $0 \leq l_e \leq c_e$, and every node v has a **demand** $d(v)$. We define a **circulation** to be a function f on the edges such that: $l_e \leq f(e) \leq c_e$ for all edges e (capacity constraints) and $f^-(v) - f^+(v) = d(v)$ for all nodes v (demand constraints).

Note that a demand can be any real number. If $d(v) < 0$, we also speak of a supply. Sometimes we denote the set of all nodes v with $d(v) < 0$ by S, and the set of all nodes v with $d(v) > 0$ by T. Now we call all nodes in S sources, and all nodes in T sinks. Previously we had only considered the special case with exactly one source and one sink, and with $l_e = 0$ for all edges e.

The Circulation problem is: Given a graph (as specified above), construct some circulation, or report that no circulation exists. Note that, in this formulation, the problem is only concerned with the existence of a feasible solution, but nothing has to be maximized or minimized. We only want the sources (sinks) to deliver (consume) exactly prescribed amounts. As a first application, think of suppliers and customers of some good, connected by a network of transportation channels. We will see later why positive lower capacities on certain edges can make sense.

Let us first consider the case when still $l_e = 0$ for all edges e. From the definitions it follows easily that $\sum_v d(v) = 0$ is a necessary condition for the existence of a circulation. We define $d := \sum_{v \in T} d(v) = -\sum_{u \in S} d(u)$. If $S = T = \emptyset$ (equivalently: $d(v) = 0$ for all v), then the everywhere-zero function f is trivially a circulation. Thus it suffices to study the case when both $S \neq \emptyset$ and $T \neq \emptyset$.

In order to re-establish the situation with exactly one source and sink, we do the following reduction that we denote CF-red: We insert new nodes s and t, and directed edges (s, u) and (v, t) for all $u \in S$ and all $v \in T$. These edges get capacities $-d(u)$ and $+d(v)$, respectively. The idea is simply to let the new source s supply an amount d and let the new sink t demand an amount d, whereas the old sources and sinks in S and T become usual nodes with demand 0.

We claim that a circulation exists if and only if the graph obtained by CF-red has a flow with value d, and in the affirmative case, this flow is also maximum, because it saturates all edges at the new source and sink. (An edge is called saturated if the flow on the edge equals the capacity.) The proof of this equivalance is straightforward, but it is recommended that you think about it. – Thus, CF-red reduces the Circulation problem to Maximum Flow, in the case when $l_e = 0$ holds for all edges e.

Next we come to the general case where the lower capacities l_e can be any non-negative numbers. We give a reduction named LZ-red that reduces this general Circulation problem to the case when all lower capacities are zero: Take any edge $e = (u, v)$ with $l_e > 0$. Since at least an amount of l_e must flow on the edge e, we can assign this amount immediately. Thus we set the lower capacity of e to 0, and the upper capacity of e to $c_e - l_e$. Clearly, we must also adjust the demands: Since we have increased the outgoing flow from u by l_e, its demand becomes $d(u) + l_e$, and since we have increased the incoming flow into v by l_e, its demand becomes $d(v) - l_e$. These simple operations are done for all edges, in any ordering. Of course, alternatively we may process all edges simultaneously and describe the changed demand of every node v more compactly as $d(v) + \sum_{e=(v,u)} l_e - \sum_{e=(u,v)} l_e$.

That is, in order to solve the general Circulation problem, we may first apply LZ-red to get a graph with $l_e = 0$ for all edges e, then apply CF-red, and finally solve Maximum Flow in the resulting graph. The reductions cost only $O(m)$ time in a graph with m edges.

Equipped with these tools we will now go through several applications where one might not even expect flows and cuts at first glance. For each problem we will only give the idea and sketch the network construction, but we omit formal equivalence proofs. We also stress that only one possible reduction is given for each problem – this does not exclude the possibility of doing many details differently.

Planning for Data Mining: Survey Design

A company sells several products, and customers shall be asked about their satisfaction. Each customer i gets questions about some products (s)he has purchased. The number of questions to customer i shall be between c_i and c'_i. Moreover, between p_j and p'_j customers shall be asked about each product j. (The survey must generate enough data to ensure statistical significance, but it should not be too large, tiresome, and costly.) All the mentioned numbers are given. The problem is: Does there exist a survey with the given constraints, and if so, how can we construct one?

We represent customers and products by nodes of a bipartite graph. A directed edge (i, j) is inserted if customer i has purchased product j. We add nodes s and t, edges from s to all customers, and from all products to t. We also insert an edge from t back to s, with lower capacity 0 and a huge upper capacity. All demands are set to 0. The lower and upper capacity bounds are fixed as follows: c_i, c'_i for every edge from s to the respective customer node, $0, 1$ for customer-product edges, and p_j, p'_j for every edge from the respective product node to t.

We claim that the feasible surveys correspond to circulations in this graph (with integer values on the edges), where the survey questions are the edges (i, j) with flow value 1.

The idea here is that the flow units model "question tickets". Every customer and every product must get a number of tickets in the desired range, customer i is asked about product j if some question ticket moves from i to j, and all tickets are eventually collected in t and returned to s.

Note that we have turned the required minimum numbers of questions immediately into lower capacities. A "direct" reduction to Maximum Flow without this nice tool would be technical and cumbersome, and perhaps hard to follow.

Appendix

An interesting example that can counteract some frequent misconceptions about Bipartite Matching is the following bipartite graph with $3 + 3$ nodes:

It has the node set $X \cup Y$ where $X = \{x_0, x_1, x_2\}$ and $Y = \{y_0, y_1, y_2\}$, and one fixed node on each side is adjacent to all nodes on the other side. In other words, the edges are: (x_0, y_1), (x_0, y_2), (x_0, y_0), (x_1, y_0), (x_2, y_0).

One might naively think that every connected bipartite graph has a matching of size $\min\{|X|, |Y|\}$. But this graph is a counterexample. Its maximum matching has only 2 edges.

Hence the network constructed from the graph, in the standard reduction from Maximum Matching to Maximum Flow, should possess some cut (A, B) of capacity only 2. Maybe this cut is not so obvious, but we can get it from the algorithms. Here is the result: $A = \{s, x_1, x_2, y_0\}$ and $B = \{x_0, y_1, y_2, t\}$, In fact, the only cut edges are (s, x_0) and (y_0, t), whereas (x_0, y_0) goes from B to A and does not contribute to the cut capacity.