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 $G = (X, Y, E)$ where the node set $X \cup Y$ is split in two sets X, Y, and 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 are job assignment problems: Nodes in X are jobs to be done, nodes in Y are workers or machines, and an edge means 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, 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. Then the size of a maximum matching equals the value of a maximum flow. Although this equation looks simple, it needs a proof:

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 values on the edges are 0 and 1. Now, the edges in E carrying flow 1 build a matching M, and their number is k. This follows from the constraints of a flow.

Note carefully that this is not yet sufficient to conclude that M has maximum size! Right, we started from a maximum flow. However, we are not obliged to use flows at all for the Bipartite Matching problem. This is just one approach, and maybe some other approach yields an even larger matching. In order to exclude that, 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.

Due to this reduction, the time to solve the Bipartite Matching problem is $O(mC) = O(mn)$. One can also find maximum matchings in general graphs in polynomial time, but this is much more tricky.

Disjoint Paths versus Disconnecting Edge Sets

This section deals with two problems motivated by network reliability.

In a directed graph with source s and sink t, we want to find a maximum number of mutually edge-disjoint paths from s to t. Interestingly, the problem to connect k different source-sink pairs by edge-disjoint paths is NP-complete, but it becomes polynomial is polynomial if all sources and all sinks, respectively, are identical.

We can solve it as follows. Give all edges the capacity 1. If k disjoint $s-t$ paths exist then, obviously, they build 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. The time is again $O(mC) = O(mn)$, including the time for this decomposition of the flow into paths. Note that the decomposition phase is needed, because the flow as such is not yet a set of paths.

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 paths from s to t are interrupted.

Not surprisingly, the smallest possible size of F equals the maximum number k of edge-disjoint paths from s to t. This statement is called **Menger's Theorem**.

For the proof, consider any edge set F that disconnects s and t. Since F must contain some edge from every $s-t$ path, we have $|F| \geq k$. On the other hand, we have seen earlier that $k = val(f)$ holds for a maximum flow f (where all edge capacities are 1). By the Max-Flow Min-Cut Theorem 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 F_0 disconnects s and t. Hence, there exists some solution F (namely F_0) with $|F| \leq k$.

Moreover, we can construct F using the previous algorithms. (Think about the details.)

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.

Circulations with Demands and Lower Capacity Bounds

These are useful variants of flow problems which can make it easier to reduce other problems to Maximum Flow.

In a directed graph, let S and T be sets of sources and sinks, with given **supply and demand** values $d(v) < 0$ for $v \in S$, and $d(v) > 0$ for $v \in T$. We have $d(v) = 0$ for all nodes not in $S \cup T$. A **circulation** is a function f on the edges such that: $0 \leq f(e) \leq c_e$ for all edges e (capacity constraints) and $f^-(v) - f^+(v) = d(v)$ for all nodes v (demand constraints).

Circulations generalize the concept of flow. But note that the Circulation problem is only concerned with the *existence* of a feasible solution, that is, we do not maximize or minimize anything. Instead we want to the sources (sinks) to deliver (consume) an exactly prescribed amount; think of suppliers and customers of some goods in a network.

Trivially, $\sum_v d(v) = 0$ is a necessary condition for the existence of a circulation. Once this condition is fulfilled, we can reduce Circulation to the Maximum Flow problem (yes, even though it is not an optimization problem). This works as follows. Insert new nodes s, t, and edges (s, u) and (v, t) for all $u \in S$ and $v \in T$. These edges have capacities $-d(u)$ and $+d(v)$, respectively. The idea of this construction should be obvious, as well as the equivalence: A circulation exists if and only if the extended graph has a flow whose value is the sum of all supplies.

Besides the usual upper bounds c_e on capacities, we may also have capacity constraints involving lower bounds: $l_e \leq f(e) \leq c_e$. That means, at least an amount of l_e must flow on edge e. In this case we can simply adjust the capacities to $c_e - l_e$ and also adjust the demands and supplies accordingly. (Details are straightforward.) Then we are back to the "classical" Maximum Flow problem.

Equipped with these tools we will now go through several applications where one might not even expect flows and cuts at first glance.

Planning for Data Mining: Survey Design

A company sells several products, and customers shall be asked about their satisfaction with the products. Each customer i gets questions about some products, but only about such products (s)he has purchased. The number of questions to customer i shall be between c_i and c_i'. (The survey must be informative enough, but not too long and tiresome.) Moreover, between p_j and p_j' customers shall be asked about each product j. (Reasons are similar: The survey must generate enough data to ensure statistical significance, but it should not be too large and costly.) The obvious 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 to s with huge capacity. All demands are set to 0. The lower and upper capacity bounds are c_i, c_i' for edges starting in s, further $0, 1$ for customer-product edges, and p_j, p_j' for edges ending in the sink 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. Think about the role of every edge in this construction.

The proposed construction is not the only possible one, and one can modify the details, but we must decide on some version.

A Simplified Airline Scheduling Problem

An airline has to operate m flight segments where each segment j is characterized by: origin and destination airport, departure and arrival time. For any two flight segments we define, by ad-hoc criteria, when flight j is *reachable* from flight i (for example: same airport, enough time between the arrival of i and departure of j). This reachability relation defines a directed acyclic graph (DAG) on the set of flight segments. If j is reachable from i, then both flight segments may be performed by the same plane. A fleet of k planes is available. The problem is: Can the given flight schedule be realized using at most k planes?

We model the problem as a circulation problem. Here it is tempting to

represent the airports by nodes, but in order to allow for various reachability criteria we model everything by edges, whereas the nodes are merely the ends of edges but do not correspond to physical objects. Now a possible construction follows. The pairs of numbers are lower and upper bounds on capacities, and the directions of edges should be obvious.

We represent flight segments by mutually disjoint edges with capacities $(1, 1)$, since every flight must be performed. If a flight is reachable from another one, we connect the two edges by another edge with capacity $(0, 1)$, since we may use this connection but we are not obliged to do so. Furthermore, we insert a source s and a sink t. We connect s to every flight, by an edge of capacity $(0, 1)$. Similarly, we connect every flight to t, by an edge of capacity $(0, 1)$. Finally we connect s to t by an edge of capacity $(0, k)$. Nodes s and t have demands $-k$ and k, respectively, and all other nodes have demand 0.

It is not hard to see that the possible schedules correspond to the possible circulations, where the flow on edge (s, t) is the number of unused planes. Each of the other units of flow corresponds to one plane and connects all the flight segments operated by that plane.