Flows in Graphs | Instructor: Padraic Bartlett

Lecture 2: Combinatorial Theorems via Flows

Week 2 | *Mathcamp 2011*

Last class, we proved the Ford-Fulkerson Min-Flow Max-Cut theorem, which said the following:

Theorem 1 *Suppose that G is a graph with source and sink nodes s, t, and a rational capacity function c. Then the maximum value of a flow is equal to the minimum value of a cut.*

Furthermore, we noted that if our capacity function was integer-valued, we could find integral values for our maximal flows. In today's lecture, we will show how this theorem allows us to prove a number of classical theorems in combinatorics with almost no effort at all!

1 Hall's Marriage Theorem

To open up, we present a proof of Hall's marriage theorem, one of the best-known results in combinatorics, using the max-flow min-cut theorem:

Theorem 2 *Suppose that G is a bipartite graph (V_1, V_2, E), with $|V_1| = |V_2|$. Then G has a perfect matching*[1] *iff the following condition holds:*

$$\forall S \subseteq V_1, |S| \leq |N(S)|.$$

Proof. We first notice that the condition above is trivially necessary for a perfect matching to exist; indeed, if we had a subset $S \subseteq V_1$ with $|S| > |N(S)|$, then there is no way to match up all of the $|S|$ elements of S along edges with elements in V_2 without using some elements more than once.

We now prove that our condition is sufficient, via the Max-Flow Min-Cut theorem. Take our graph G, orient all of its edges so that they go from V_1 to V_2, add a source vertex s, a sink vertex t, edges from s to all of V_1, from all of V_2 to t, and let c be a capacity function that's identically 1 on all of the edges $s \to V_1, t \to V_2$, and ∞ on all of the original edges in G. By Ford-Fulkerson, there is a minimal cut on this graph: call it S. We trivially know that $c(S, \overline{S}) \leq n$, as there is a n-cut given by simply setting $S = \{s\}$; we seek to show that $c(S, \overline{S})$ is in fact equal to n.

Let $X = S \cap V_1$. Because the capacity of all of the edges originally in G is infinite, we know that any minimal cut cannot contain half of any such edges; therefore, we have $N(X) \subseteq S \cap V_2$.

[1] A **perfect matching** in a bipartite graph G is a collection of disjoint edges M that contains all of the vertices in G. In a sense, it is a way to "match" all of the vertices in V_1 to the vertices in V_2.

But this means that

$$\begin{aligned} c(S,\overline{S}) &= \sum_{x\in S, y\in\overline{S}} c(x,y) \\ &= \sum_{x\in(S\cap\{s\}),y\in(\overline{S}\cap V_1)} c(x,y) + \sum_{x\in(S\cap V_2),y\in(\overline{S}\cap\{t\})} c(x,y) \\ &\leq n - |X| + |N(X)| \\ &\leq n - |X| + |X| \\ &= n. \end{aligned}$$

So any minimal cut has capacity n; therefore, there is a flow with value n. Such a flow sends one unit to each vertex in V_1, and sends one unit from each vertex in V_2 to t; therefore, by Kirchoff's law, the edges marked 1 in G by such a flow form a perfect matching of G's vertices.

2 Menger's Theorem

We now continue with a classical theorem of Menger:

Theorem 3 *Let s, t be a pair of distinct vertices in an undirected graph G. Then, the minimal number of edges needed to separate*[2] *s from t is equal to the maximal number of edge-disjoint paths connecting s to t.*

Proof. Let s, t be the source and sink nodes of G, respectively, replace each of G's undirected edges $\{x, y\}$ with pairs of edges $(x, y), (y, x)$, and let c be a capacity function on G that assigns 1 to every edge. Let f be a maximal flow on G, and S be a corresponding minimal cut.

Our flow f clearly defines the maximal number of edge-disjoint paths from s to t, by identifying edges as being in a path iff f is 1 on them. Similarly, S clearly corresponds to a minimal separating set of edges; therefore, because the value of any maximum flow is the value of a minimal cut, we've completed our proof.

3 Dilworth's Theorem

Finally, we close with a third example of how flows can make difficult combinatorial theorems easier, in the form of Dilworth's theorem! We state this theorem here:

[2] A set K of edges **separates** two vertices s, t, iff there are no paths from s to t that do not use elements in K.

Theorem 4 *Suppose that a every antichain*[3] *in a partially ordered set*[4] P *has less than* m *elements. Show that* P *can be written as the union of* m *chains*[5].

Unlike our earlier theorems, this one does not seem to be quite as amenable to a flow-based attack. In particular, our language of "maximal flows" seems to be good at giving us **edge-disjoint** paths: however, in Dilworth's theorem, we actually want **vertex**-disjoint paths, because we're trying to decompose P into chains (which are made out of vertices, not the inequalities linking them.)

How can we do this? Well: what we **really** want to do is have a flow that, instead of having some sort of maximum upper bound, has a **minimal** lower bound: i.e. that has to send at least one unit of flow into every vertex! Our desire to do this motivates the following extension of the Max-Flow Min-Cut theorem, which we put on the HW:

Theorem 5 *Suppose that* G *is a directed graph with source and sink nodes* s, t*. Suppose further that* G *comes with a pair of rational capacity functions* $l, u : E(G) \to \mathbb{Q} \cup \{\infty\}$ *such that* $l(e) \leq u(e)$*, and a* ***feasible*** *flow* f_0 *(i.e a flow such that* $l(e) \leq f(e) \leq u(e)$*.) Then there is a feasible flow* f *on* G *and cut* S *on* G *such that*

- *for any* $x \in S, y \notin S$*, we have* $f(x, y) = u(x, y)$*, and*
- *for any* $x \notin S, y \in S$*, we have* $f(x, y) = l(x, y)$*.*

(In a sense, this flow realizes the "maximal" value of any cut, if we regard the capacity of a cut S *here as* $\sum_{x \in S, y \notin S} u(x, y) - \sum_{x \notin S, y \in S} l(x, y)$*.)*

Proof. HW!

This stated, we now turn to our proof of Dilworth's theorem.

Proof. Let $P = \{X, <\}$ be a partially ordered set. We turn P into a directed graph G, as follows:

1. For each element $x \in X$, create a pair of vertices x_1, x_2 in our graph.
2. Create an edge (x_1, x_2) between each such pair of vertices.
3. Whenever $x < y$ in our partially ordered set, add the edge (x_2, y_1) to our graph.
4. Let $A = \{a \in X : \nexists\ y \text{ s.t. } a < y\}$, and $B = \{b \in X : \nexists\ y \text{ s.t. } y < b\}$. Create two new vertices s, t, and add the directed edges (s, b_1) and (a_2, t) for all of the elements in A, B respectively.

[3] An **antichain** is a set $S \subset P$ such that no two elements in S are comparable: i.e. if $x < y$, then either x or y (or both!) are not in S.

[4] A **partially ordered set** $P = (X, <)$ is a collection of vertices $\{x_1, \ldots x_n\}$ that satisfies the following two properties:

- **Antisymmetry**: if $x < y$, we do not have $y < x$.
- **Transitivity**: if $x < y$ and $y < z$, we have $x < z$.

[5] A **chain** in a partially ordered set P is a sequence of elements $x_1 < \ldots < x_n$ ordered by P.

5. Define an upper capacity bound u on G's edges by having $u(x_1, x_2) = -1$ and $u(e) = 0$ for all other edges; define a lower capacity bound u on G's edges by setting it to be $-\infty$ everywhere.

To generate a starting flow on G, simply start at G and repeatedly take paths from s to t that involve some new edges every time, until you have a path covering of G: with this done, simply set $f(e) = -1 \cdot$ (the number of times e is used in these paths).

Apply the Max-Flow Min-Cut theorem here to get a maximal flow f and cut S with respect to our boundary conditions. Now, notice the following: by definition, we have

- for any $x \in S, y \notin S$, we have $f(x, y) = u(x, y)$, and
- for any $x \notin S, y \in S$, we have $f(x, y) = l(x, y)$.

In specific, we have that because $l(x, y) = -\infty$, there are no edges (x, y) where $x \notin S, y \in S$, as this would mean that our cut would have capacity $-\infty$ (and there are clearly cuts with noninfinite capacity, namely the cut $S = \{s\}$.) As a consequence, we can see that any path starting in S and going to $\overline{S}$ has exactly one edge (x, y) with $x \in S, y \notin S$: this is because having two such edges would require a trip from $\overline{S}$ to S, which we just said was impossible.

Why do we care? Well: this gives us a really useful way to count the number of paths in our graph. Specifically, let C be the collection of paths induced by our flow f (i.e. take the collection of edges $s \to t$ marked by f with the multiplicity given by $-f$, and decompose these into $|f|$-many paths from s to t.) How many paths in C are there – i.e. what is $|f|$?

Well: as we noted earlier each path has precisely one "crossing" edge from S to $\overline{S}$. So there are precisely

$$\sum_{x \in S, y \notin S} -f(x, y)$$

many paths in C.

But we know that because f is maximal, we have $f(x, y) = u(x, y)$ on all of these crossing edges. So, notice that u is only -1 on edges of the form (x_1, x_2) and 0 otherwise. Let Q be the collection of all of the edges in C that are crossing edges, and Q' be the collection of corresponding elements in X to these edges. Notice that because

$$\sum_{x \in S, y \notin S} -f(x, y) = \sum_{x \in Q} -f(x_1, x_2) = \sum_{x \in Q} 1,$$

the number of elements in Q is the number of paths.

As it turns out, Q is also an antichain; this is because any path in our graph through a node $q \in Q$ contains the crossing edge (q_1, q_2). If such a path were to contain two such elements of Q, it would necessarily contain an edge from $\overline{S}$ to S, which we said cannot exist: therefore, no two elements of Q lie in the same chain, and thus they form an antichain.

So we have a path decomposition of P with as many paths as elements in an antichain! This tells us that, in specific, there is a path decomposition of P with as many paths as a maximal antichain, which is what we sought to prove.