Lecture 5: Applications of Network Flows, Global Min cut

1 Matchings in bipartite graphs

**Definition:** A graph $G(V, E)$ is bipartite if we can partition $V$ into two sets $A$ and $B$ such that $\forall e = (i, j) \in E$, $\{i, j\} \not\subseteq A$ and $\{i, j\} \not\subseteq B$. In other words, the two endpoints of an edge do not belong to the same set. We usually represent a bipartite graph by $G(A, B, E)$.

**Definition:** A matching $M$ is a set of edges such that every vertex is incident to at most one edge in $M$. In other words, it is a set of “independent” edges that share no endpoints in common; $M$ is a perfect matching iff $|A| = |B| = |M|$; $M$ is a **maximal** matching if it is not a subset of any other matching. In other words, $\forall e \in E \setminus M$, $M \cup \{e\}$ is not a matching; $M$ is a **maximum** matching if there are no possible matchings of a larger size.

It is not hard to see that the problem of finding a maximum matching in a bipartite graph $G$ can be reduced to finding maximum flow in a network. Beginning with a graph $G$ in an instance of maximum matching problem, we construct a directed graph $G'$ by directing all the edges in $G$ from $A$ to $B$. We add a node $s$, and an edge $(s, a)$ for every $a \in A$. We also add a node $t$, and an edge $(b, t)$ from each node in $B$ to $t$. We give all edges capacity of 1, and set the maximum $s - t$ flow. Because the capacities are integral, the maximum flow from $s$ to $t$ will be integral. Now it is easy to see that the set of edges with flow 1 in $G'$ form a maximum matching in $G$.

**Question:** Under what conditions does a bipartite graph $G(A, B, E)$ have a perfect matching?

**Theorem 1** Hall’s Marriage Theorem (1935)

Let $G(A, B, E)$ be a bipartite graph such that $|A| = |B| = n$. $G$ has a perfect matching iff

$$\forall S \subseteq A, \ |S| \leq |N(S)|$$

where $N(S)$ is the neighborhood of $S$, i.e. $N(S) = \{v \in B \mid \exists u \in S : (u, v) \in E\}$.

The above theorem can be proved using the maximum-flow minimum-cut theorem. You can see page 372-373 of Kleinberg and Tardos to see that proof in details. Instead, here we present a simpler and self-contained proof.

**Proof:**

“⇒” It’s easy to see that if $\exists S \subseteq A$ such that $|S| > |N(S)|$, $G$ cannot have a perfect matching.

“⇐” We prove this by strong induction on the size of $A$. The case $n = 1$ is trivial. We split the induction step into two cases:

**Case 1:** $\forall S \subseteq A, |S| < |N(S)|$

Pick an arbitrary vertex $u \in A$ and match it to one of its neighbors $v$. Let $A_1 = A \setminus \{u\}$, $B_1 = B \setminus \{v\}$, and $G_1$ be the bipartite graph induced by $A_1$ and $B_1$. $\forall S \in A_1, |N_{G_1}(S)| \geq |N_{G}(S)| - 1 \geq |S|$, thus by assumption $G_1$ has a perfect matching $M_1$. It is easy to see that $M_1 \cup \{(u, v)\}$ is a perfect matching of $G$.

**Case 2:** $\exists S \subseteq A, |S| = |N(S)|$

Let $A_1 = S$ and $B_1 = N(A_1)$; define $A_2 = A \setminus A_1$ and $B_2 = B \setminus B_1$. $E$ can be partitioned into three subsets
\[ E_1, E_2, \text{ and } E_*: \]

\[ E_1 = \{(a, b), a \in A_1, b \in B_1 \} \]
\[ E_2 = \{(a, b), a \in A_2, b \in B_2 \} \]
\[ E_* = \{(a, b), a \in A_2, b \in B_1 \} \]

Note that by definition there is no edge between \( A_1 \) and \( B_2 \). Consider graph \( G_1(A_1, B_1, E_1), |A_1| < |A| \) and \( \forall S \subseteq A_1, N_{G_1}(S) = N_G(S) \geq |S| \), thus by assumption \( G_1 \) has a perfect matching \( M_1 \).

Next consider \( G_2(A_2, B_2, E_2) \), since \( N_{G_2}(S) \leq N_G(S) \) we cannot directly use the induction assumption. However, if we show that

\[ \forall S' \subseteq A_2, |N_{G_2}(S')| \geq |S'|, \]  

then by assumption \( G_2 \) has a perfect matching \( M_2 \) and thus \( M_1 \cup M_2 \) is a perfect matching of \( G \).

Assume that the above condition does not hold, i.e., \( \exists S' \subseteq A_2, \) such that \( |S'| > |N_{G_2}(S')| \); consider the set \( A_1 \cup S' \) in \( G \), \( N_G(A_1 \cup S') = |B_1| + |N_{G_2}(S')| \) or equivalently \( |A_1 \cup S'| = |A_1| + |S'| > |B_1| + |N_{G_2}(S')| = N_G(A_1 \cup S') \) which is a contradiction.

\[ \square \]

2 Graph Connectivity\(^1\)

An amateur graph theorist, in his scribblings, might invent the following two definitions of \( k \)-edge connectivity. \( G \) is \( k \)-edge connected if:

1. \( G \) remains connected after removing any \((k-1)\) edges.

OR

2. There are at least \( k \) edge-disjoint paths between every pair of vertices in \( G \).

Clearly if \( G \) satisfies definition 2 then it also satisfies definition 1. How about the other way around?

**Theorem 2** If a graph \( G(V, E) \) remains connected after removing any \((k-1)\) edges then there are at least \( k \) edge-disjoint paths between every pair of vertices in \( G \).

**Proof:** Suppose that \( G \) satisfies definition 1 above, and let \( s, t \in V \). Consider the directed version \( G' \) of \( G \) formed by replacing all edges \( \{u, v\} \) with the two edges \( (u, v) \) and \( (v, u) \). Now let \( N \) be the network with \( G' \) as its underlying graph, \( s \) and \( t \) as source and sink, and all edges with capacity 1.

We claim that if we can show there are \( k \) edge-disjoint paths from \( s \) to \( t \) in \( G' \), then the same holds in \( G \). To see this, note that we may turn all directed paths into undirected paths as long as two directed paths \( P_1 \) and \( P_2 \) don’t use both directions \((u, v)\) and \((v, u)\) of an undirected edge \( \{u, v\} \). If this is the case, however, we may simply have the two paths swap endings: define \( \bar{P}_1 \) to be the path following \( P_1 \) from \( s \) to \( u \) and then \( P_2 \) from \( u \) to \( t \), define \( \bar{P}_2 \) to be the path following \( P_2 \) from \( s \) to \( v \) and then \( P_1 \) from \( v \) to \( t \). Then if we consider \( \bar{P}_1 \) and \( \bar{P}_2 \) undirected, neither path uses \( \{u, v\} \). All such conflicts can be inductively removed, proving the claim.

Now in order to find \( k \) edge-disjoint paths from \( s \) to \( t \) in \( G' \), we compute the maximum flow in \( N \). The fact that \( G \) remains connected after removing any \((k-1)\) edges implies that the minimum cut in \( N \) has value at

\(^1\)See Kleinberg and Tardos Section 7.6
least \( k \); thus the max-flow has value at least \( k \). Since there must be an integral maximum flow in which the flow on every edge of \( N \) is \((0,1)\)-valued, we may exhibit \( k \) edge-disjoint paths from \( s \) to \( t \) in \( G' \) by greedily tracing paths from \( s \) to \( t \) along edges with flow 1.

\[ \square \]

### 3 A Randomized Algorithm for Global Min-cut

In the global min-cut problem, we wish to find a minimum set of edges such that a graph becomes disconnected. Formally,

**Definition:** Given an undirected graph \( G(V,E) \), a global min-cut is a partition of \( V \) into two subsets \((A,B)\) such that the number of edges between \( A \) and \( B \) is minimized.

To derive an algorithm for this, note that the global min-cut is the minimum over all possible \( s-t \) cuts.

**Solution 1:** Compute the min \( s-t \) cut for all pairs \( s,t \in V \).

This algorithm requires \( O(n^2) \) calls to a min \( s-t \) cut solver. We can reduce this by noting that any node \( s \) must appear in one of \( A \) or \( B \), meaning we can reduce the number of min \( s-t \) cut solves by a factor of \( n \).

**Solution 2:** Fix \( s \) and find min \( s-t \) cut for all \( t \in V \).

For a long time, the intuition was that global min-cut was harder than \( s-t \) cut. David Karger was able to show that this is not the case with a randomized algorithm.

**Algorithm 1** Karger’s randomized global min-cut

```
repeat
    Choose an edge \( \{u,v\} \) uniformly at random from \( E \).
    Contract the vertices \( u \) and \( v \) to a super-vector \( w \).
    Keep parallel edges but remove self-loops.
until \( G \) has only 2 vertices.
Report the corresponding cut.
```

**Theorem 3** The probability that the algorithm finds the minimum cut in \( G \) is at least \( 2/n^2 \).

**Proof:** Let \( F \) be the set of edges in a global min-cut and supposes \( |F| = k \). Let \( E \) be the event that the algorithm does not contract an edge from \( F \) in step 1. Since the minimum cut has value \( k \), the degree of each vertex must be at least \( k \), which implies the following claim:

**Claim 1** \( |E| \geq k|V|/2 = \frac{kn}{2} \).

Denote by \( E_i \) the event that the \( i \)th edge picked belongs to the min-cut. The probability that the first edge picked belongs to the min-cut is therefore:

\[
Pr(E_1) = 1 - \frac{k}{|E|} \geq 1 - \frac{2}{n}
\]
Similarly,

\[ \Pr(E_2|E_1) \geq 1 - \frac{2}{n-1} \]
\[ \Pr(E_i|E_1 \cap E_2 \cap \ldots) \geq 1 - \frac{2}{n-i+1} \]

Combining these to get the total probability of success gives

\[ \Pr(\text{success}) = \Pr(E_1 \cap E_2 \cap \ldots \cap E_{n-2}) \]
\[ \geq \Pr(E_1)\Pr(E_2|E_1)\ldots\Pr(E_{n-2}|E_1 \cap \ldots E_{n-3}) \]
\[ \geq (1 - \frac{2}{n})(1 - \frac{2}{n-1})\ldots(1 - \frac{2}{3}) \]
\[ \geq 2\frac{(n-2)!}{n!} \geq \frac{2}{n^2}. \]

So the algorithm will succeed at with probability at least \(2/n^2\).

The probability of finding a min-cut seems rather low, and goes to zero as \(n \to \infty\). However, what happens to the probability of success if we run the algorithm \(t\) times? The probability of failure will be at most

\[ \left(1 - \frac{2}{n^2}\right)^t. \]

So if we set \(t = cn^2\) for some constant \(c\), the probability of failure will be at most \((1 - \frac{2}{n^2})^{cn^2} \leq e^{-2c}\). To make the probability of failure as small as, say \(e^{-14}\), it is only necessary to run the algorithm \(7n^2\) times.