CS 388R: Randomized Algorithms, Fall 2019 October 10

Lecture 13: Bipartite Matching on regular graphs

Prof. Eric Price *Scribe: George Lu, Shailesh Mani Pandey*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Notation and Definitions

Definition 1 (Matching)**.** *Given an undirected graph $G = (V, E)$, we say a subset of edges $M \subseteq E$ is a matching if every vertex in V has at most degree 1 in M - in other words, no two edges share an endpoint.*

Definition 2 (Maximum Matching)**.** *Given an undirected graph $G = (V, E)$, a maximum matching M is a matching of maximum size. Thus for any other matching M', $|M| > |M'|$.*

Definition 3 (Perfect Matching)**.** *A matching is perfect if every vertex has degree exactly 1 in M.*

Definition 4 (d-regular Graph)**.** *We say a graph is d-regular if every vertex has degree d*

Definition 5 (Bipartite Graph)**.** *We say a graph is bipartite if there is a partitioning of vertices of a graph, V, into disjoint subsets A, B such that $A \cup B = V$ and all edges $(u, v) \in E$ have exactly 1 endpoint in A and 1 in B. We will notate such a bipartite graph as $(A + B, E)$.*

Definition 6 (Neighborhood)**.** *For any vertex, u, in a bipartite graph, the neighborhood of u, $N(u)$ is the set of all vertices v such that $(u, v) \in E$. For a set of vertices U, $N(U) = \bigcup_{u \in U} N(u)$.*

In this lecture, we'll be primarily concerned with the case of finding perfect matchings in d-regular bipartite graphs.

Definition 7 (s-t Flow)**.** *We say a mapping from $f : E \to \mathbb{R}$ is an $s-t$ flow on a directed weighted graph G iff*

- $\forall v \in V \backslash \{s, t\} \sum_{(u,v)\in E} f((u, v)) = \sum_{(v,w)\in E} f((v, w))$
- $\forall e \in E 0 \leq f(E) \leq w(E)$
- $0 \leq \sum_{(s,w)\in E} f((s, w)) - \sum_{(u,s)\in E} f((u, s)) = \sum_{(u,t)\in E} f((u, t)) - \sum_{(t,w)\in E} f((t, w))$

This last quantity is the flow of the graph.

Definition 8 ($s-t$ Cut)**.** *An s-t cut of a directed, weighted graph is a partition S, T of the vertices such that $s \in S$ and $t \in T$. We say the size of cut is*

$$\sum_{(u,v)\in E\cap(S\times T)} w(u, v)$$

2 Theory

Theorem 9 (Min-Cut Max-Flow)**.** *The minimum $s-t$ cut of a graph is equal to the maximum $s-t$ flow.*

Proof. Check your favorite undergrad algorithms textbook ■

Theorem 10 (Hall's Marriage Theorem)**.** *A bipartite graph $G = (A+B, E)$ has a perfect matching iff $\forall S \subseteq A, |S| \leq |N(S)|$.*

Proof. If there is a perfect matching, then clearly $\forall S \subseteq A\ |S| \leq |N(S)|$, as the edges matched to S are disjoint and a subset of $N(S)$.

To complete the proof, we will show the inverse of the above statement - If there is no perfect matching, then $\exists S \subseteq A$ such that $|S| > |N(S)|$. Let's consider the following graph G' on $\{s,t\} \cup A + B$. There are direct weight 1 edges from s to every element of A, and from every element of B to T. Finally, there is a directed edge of infinite (or arbitrarily large) capacity from $a \in A$ to $b \in B$ iff (a,b) was an edge in G.

Lemma 11. *The maximum $s-t$ flow of G' is equal to the maximum matching on G*

Proof. Consider any matching M on G. We can define a flow, f, of size $|M|$ on G' as $f((s,a)) = 1$, $f((a,b)) = 1$, and $f((b,t)) = 1$ iff $\exists (a,b) \in M$. f is 0 otherwise. It is easy to verify that this is indeed size $|M|$ and satisfies flow conservation

Conversely, we consider a max flow of G of size F. First, we note the existence of integral max flows when G' has integral edges (the proof of this is through a simple examination of the execution and correctness of the Ford-Fulkerson algorithm), so we will restrict our consideration to such flows. We consider the set of edges S going between A and B which have positive flow. Since S's endpoint in A and B has capacity 1, these edges can only exactly have 1 flow through them. Similarly, by flow conservation on their endpoints, no two saturated edges can share an endpoint without violating flow conservation, so we realize S is a matching of size F. ■

Thus, by Min-Cut Max-Flow, we conclude that since there is no perfect matching, there must be an $s-t$ cut (S,T) of G' of size $\leq |M| < |A|$. We can observe that the size of any $s-t$ cut in this graph is equal to $|T \cap A| + |S \cap B| + \infty \cdot |N(S \cap A) \cap (T \cap B)|$. Clearly we can't cut any infinite capacity edges, so this cut is actually equal to $|T \cap A| + |S \cap B| = |B \cap S| + (|A| - |A \cap S|)$. However, since we know that $N(S \cap A) \cap (T \cap B) = \emptyset$, that tells us $N(S \cap A) = N(S \cap A) \cap S \cap B \subseteq S \cap B$, so $|N(S \cap A)| \leq |S \cap B|$. We also know that the cut has size less than $|A|$, which means $|B \cap S| < |S \cap A|$. Combining these two inequalities, we obtain that $|N(S \cap A)| < |S \cap A|$, thus providing a witness for Hall's marriage theorem.

■

Lemma 12. *$\forall d > 0$, d-regular bipartite graphs have perfect matchings.*

Proof. There are total $d \cdot |S|$ edges from any set $S \subseteq A$ to $N(S)$. Also, total number of edges into $N(S)$ is $d \cdot |N(S)|$.

$\implies d|N(S)| \geq d|S| \implies |N(S)| \geq |S|$ ■

3 Algorithms

3.1 Previous Results

- Ford Fulkerson - $O(mF)$ to find max flow. $m \in O(nd)$, $F \in O(n)$, so $O(n^2 d)$
- Hungarian Algorithm - $O(n^3)$, works for weighted graphs.
- Hopcroft-Karp $O(m\sqrt{n}) = O(n^{1.5}d)$

The algorithms listed above work in general bipartite graphs, but we will show below that performance can be significantly improved when the graphs are d-regular.

3.2 Gabow-Kariv '82

This algorithm only works when d is a power of 2, but runs in $O(m) = O(nd)$ time. Idea - we find an Eulerian tour of G in $O(m)$ time, then remove $m/2$ edges which go from B to A in the tour, producing a $\frac{d}{2}$-regular graph. The running time per recursive call is geometric and converges.

3.3 Goel-Kapralov-Khanna '06

This is a randomized algorithm that finds a perfect matching for all $d > 0$, and runs in expected $O(n \log n)$ time.

This is essentially a randomized version of the Ford Fulkerson algorithm applied to the flow construction described in our proof of Hall's Marriage Theorem.

To be more specific, we repeatedly find a path from the source, s, to the sink, t, by a random walk and augment along that path. In finding a random path, we never go back to s and always go to t whenever possible.

3.3.1 Runtime Analysis

Lemma 13. *For d-regular graphs, if k vertices are still unmatched, then $\mathbb{E}$[time to match next vertex] $\lesssim \frac{n}{k}$.*

Proof. For graph G' on $\{s, t\} \cup X + Y$, let $X_m \subseteq X$, $Y_m \subseteq Y$ be the set of matched vertices, and $X_u \subseteq X$, $Y_u \subseteq Y$ be the set of unmatched vertices. We write $M(x) = y$ if x matches y.

$\forall v \in G'$, we define $b(v) =$ expected number of back edges in the random walk from v to sink, t.

$$b(y) = \begin{cases} 0 & y \in Y_u \\ 1 + b(M(y)) & y \in Y_m \end{cases}$$

$$\begin{aligned}
\forall x \in X_u, d \cdot b(x) &= \sum_{y \in N(x)} b(y) \\
\forall x \in X_m, (d-1) \cdot b(x) &= \sum_{y \in N(x) - \{M(x)\}} b(y) \\
&= \sum_{y \in N(x)} b(y) - b(M(x)) \\
&= \sum_{y \in N(x)} b(y) - (1 + b(x)) \\
\implies d \cdot b(x) &= \sum_{y \in N(x)} b(y) - 1
\end{aligned}$$

$$\begin{aligned}
\implies \sum_{\forall x \in X} d \cdot b(x) &= \sum_{y \in Y} d \cdot b(y) - |M| \\
&= d \cdot \left(|M| + \sum_{x \in X_m} b(x) \right) - |M| \\
\implies \sum_{\forall x \in X_u} d \cdot b(x) &= (d-1) \cdot |M|
\end{aligned}$$

We can now write $b(s)$ as:

$$\begin{aligned}
b(s) = \frac{1}{k} \sum_{x \in X_u} b(x) &= \frac{(d-1)}{d \cdot k} |M| \\
&= \frac{(d-1)}{d} \cdot \frac{(n-k)}{k} \leq \frac{n}{k}
\end{aligned}$$

Hence, $\mathbb{E}[\text{time to match next vertex}] = \mathbb{E}[\text{length of path from } s \text{ to } t] \leq 3 + 2 \cdot b(s) \lesssim \frac{n}{k}$. ■

Using this lemma, $\mathbb{E}[\text{time to match all vertices}] = \sum_{i=n}^{1} \frac{n}{i} = n \cdot H_n \approx n \log n$.

4 Intro to Online Bipartite Matching

The graph is not known in advance and vertices appear one at a time. A matching can be chosen for a vertex as it appears, and that matching can not be revoked. The resultant may not be regular. One scenario where this occurs is matching users to different advertisers on a website.

The greedy algorithm gets "maximal" matching, which is at least half the size of the maximum matching. As it turns out, any deterministic algorithm or even choosing a random edge for every vertex doesn't help much. However, there exists a randomized algorithm by Karp-Vazirani-Vazirani (KVV '90) that gives a matching which is approximately $\left(1 - \frac{1}{e}\right)$ times as good as the maximum matching.