CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2024**

Lecture 23

November 28, 2024

Instructor: Sepehr Assadi

Topics of this Lecture

1 A Quick Recap of Random Walks

In the last lecture, we looked at Markov chains and stationary distributions. A Markov chain consists of n states and a probability matrix—called the *transition matrix*—$P \in \mathbb{R}^{n \times n}$, where P_{ij} denotes the probability that the next state of the random walk is j assuming the current state is i. Recall the following definitions:

- For any states i and j, the **hitting time** $h_{i,j}$ is the expected length of the random walk starting from i and ending in j. We refer to $h_{i,i}$ as the **return time**.
- The **stationary distribution** is a distribution π over the states which remains unchanged after taking another step of the random walk, i.e., $\pi = \pi \cdot P$.
- An **irreducible** Markov chain is a one where the underlying directed graph of transition matrix (having any edge (i,j) whenever $P_{ij} > 0$), is strongly connected.

We will be using the following special case of the fundamental theorem of Markov Chains in this lecture.

Theorem 1. *In any irreducible Markov chain, for any state i,*

$$h_{i,i} = \frac{1}{\pi_i};$$

namely, the return time of a state is equal to the inverse of its stationary probability.

We now use this theorem to design a brilliant and yet extremely simple algorithm for finding perfect matching in *regular* bipartite graphs, due to Goel, Kapralov, and Khanna [GKK13].

2 Perfect Matching in Regular Bipartite Graphs

In this section, we will use random walks to develop an algorithm which finds a perfect matching in d-regular bipartite graphs in $O(n \log n)$ time. Notice that since the input is of size $\Theta(nd)$, whenever $d = \omega(\log n)$, this is even faster than reading the input once! Let us first show that regular bipartite graphs *always* have a perfect matching to begin with.

Proposition 2. *For any integers $n, d \geqslant 1$, any d-regular bipartite graph $G = (L, R, E)$ with $n := |L| = |R|$ has a perfect matching.*

Proof. This can be proven easily using Hall's theorem or alternatively via fractional matchings.

Proof via Hall's theorem. Recall that Hall's (Marriage) Theorem states:

> *Let $G = (L \cup R, E)$ be a bipartite graph. G has a matching that matches all vertices in L iff for every subset $A \subseteq L$, we have $|A| \leqslant |N(A)|$ in G.*

Now consider our d-regular graph G and fix any set $A \subseteq L$. Let $m(A, N(A))$ denote the number of edges between A and $N(A)$. We have,

$$|A| \cdot d = m(A, N(A)) \leqslant |N(A)| \cdot d;$$

the left equality holds because G is d-regular and all edges in A go to $N(A)$, and the right inequality holds because the edges between A and $N(A)$ is a subset of all edges incident on $N(A)$, which are $|N(A)| \cdot d$ many by d-regularity. This implies $|A| \leqslant |N(A)|$. Thus G has a matching that matches all of L by Hall's theorem. But since $|L| = |R|$, this means G has a perfect matching.

Proof of via fractional matchings. Recall that a fractional matching is an assignment $x \in \mathbb{R}^E$ satisfying

$$\forall v \in V: \quad \sum_{e \ni v} x_e \leqslant 1 \qquad \text{and} \qquad \forall e \in E: \quad x_e \geqslant 0.$$

We proved earlier in the course (Lecture 8) that any fractional matching in bipartite graphs can be turned into an integral (standard) matching with at least $\sum_{e \in E} x_e$ many edges. In other words, the integrality gap of the linear program for bipartite matching is 1.

Now consider our d-regular graph G and assign $x_e = 1/d$ to every edge. This is a fractional matching:

$$\forall v \in V: \quad \sum_{e \ni v} x_e = \deg(v) \cdot \frac{1}{d} = 1,$$

and we have

$$\sum_{e \in E} x_e = n \cdot d \cdot \frac{1}{d} = n;$$

hence, there is an integral matching of size n also in G, namely a perfect matching. □

Now that we established that there is a perfect matching, let us see how we can find it.

Theorem 3 ([GKK13]). *There is a randomized algorithm that for every integer $n, d \geqslant 1$, given a bipartite d-regular graph $G = (L \cup R, E)$ with $n = |L| = |R| = n$, outputs a perfect matching of G in $O(n \log n)$ expected time*[1].

The algorithm proceeds by finding augmenting paths in the graph using random walks. First, we establish what augmenting paths are and their role in increasing the size of a matching. Then, we will see how we can construct these augmenting paths.

[1]The algorithm of [GKK13] can find a perfect matching with probability at $1 - 2^{-\Theta(n)}$, namely, with an exponentially high probability; however, we do not prove this latter part in this lecture.

2.1 Role of Augmenting Paths

We define augmenting paths formally.

Definition 4. Given a graph $G = (V, E)$ and a matching $M \subseteq E$, a path $P = (u_1, u_2, \ldots, u_{2k})$ in G is said to be an **augmenting path** for M if,

- Edge (u_{2i-1}, u_{2i}) is not present in M for all $i \in [k]$.
- Edge (u_{2i}, u_{2i+1}) is present in M for all $i \in [k-1]$.

That is, path P starts and ends at unmatched vertices, and alternates between edges from M and $E \setminus M$.

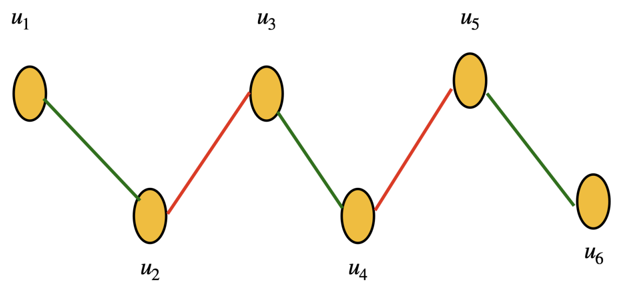

Figure 1: An augmenting path of length 5. The red edges are matching edges, and green edges are non-matching edges. Vertices u_1, u_2 are unmatched. Replacing red edges with green edges in M gives us a larger matching M'.

It is easy to see that whenever we find an augmenting path for a matching M, we can update M easily to increase its size by one.

Claim 5. *Given a matching M of graph $G = (L \cup R, E)$ and an augmenting path P of M in graph G, there is a matching M' of G such that $|M'| = |M| + 1$.*

Proof. Let path P be $(u_1, u_2, \ldots, u_k)$ for some even k. Let $E(P)$ be the set of all edges which are present in path P. Then we construct M' as,

$$M' = (M \setminus E(P)) \cup (E(P) \setminus M).$$

In other words, we replace all the edges in path P which are in M by all the edges in path P which are not in M. Refer to Figure 1 for an illustration.

To see that M' remains a matching, note that no edge which is not present in path P is affected, so the status of all vertices which are not in P is unchanged. For vertices in P, both u_1, u_k are unmatched, and we add one edge incident to each of them to M'. For all the other vertices u_i for $1 < i < k$, we remove edge (u_i, u_{i+1}) and add edge (u_{i-1}, u_i) or vice-versa. M' never has two edges incident to one vertex.

Finally, the size of M' is larger than that of M by one, because we remove $(k/2) - 1$ edges from M, and add $(k/2)$ edges. □

Another important observation is that as long as a matching M is not maximum, it admits at least one augmenting path.

Claim 6. *Given a matching M of graph $G = (L \cup R, E)$ such that M is <u>not</u> a maximum matching in G, there always exist at least one augmenting path for M in G.*

Proof. Let M^* be a maximum matching of G. Consider the subgraph $M \triangle M^* := (M \cup M^*) \setminus (M \cap M^*)$. Notice that this subgraph is a bipartite graph (this is true even if G was not bipartite to begin with), and it has at least one more edge of $M^* \setminus M$ compared to $M \setminus M^*$. Moreover, maximum degree of this subgraph is two and thus it is a collection of vertex-disjoint even cycles and paths. Every even cycle has the same number of edges from M and M^* thus, there should be at least one path with strictly more edges from $M^* \setminus M$ than $M \setminus M^*$. That path forms an augmenting path for M in G. □

Claim 5 and Claim 6 suggest a natural strategy for finding maximum matchings in any bipartite graph: start from an empty matching and as long as it updates an augmenting path, find one, and augment the matching by one using this path. The only question is how we can find the augmenting path efficiently. Classical algorithms for this problem do this using a simple graph search in $O(m)$ time (as an exercise, find this simple algorithm). This leads to an $O(mn)$ time algorithm for bipartite matching in any arbitrary bipartite graph (we have seen this algorithm in Lecture 11).

2.2 The Algorithm of [GKK13] for Regular Bipartite Graphs

The algorithm in Theorem 3 also works by repeatedly find augmenting paths to increase the size of a maintained matching. The key novelty of this algorithm however is in the way it finds augmenting paths, which is quite efficient on regular graphs (although on arbitrary graphs this approach will be quite inefficient). The following lemma is the heart of the proof.

Lemma 7. *For any $k, n, d \geqslant 1$, given any d-regular bipartite graph $G = (L \cup R, E)$ with $|L| = |R| = n$, and a matching M which leaves $2k$ vertices unmatched from $L \cup R$, there is a randomized algorithm which finds an augmenting path for M in expected $O(n/k)$ time. The algorithm assumes adjacency list access to G as well as being able to query for every vertex $v \in L \cup R$, if it is matched in M or not, and if matched, receives $M(v)$, i.e., the matched pair of v.*

We will prove this lemma in the next subsection. Let us show how this proves Theorem 3.

Proof of Theorem 3. We start with matching $M = \emptyset$. Then we can find an augmenting path in $O(1)$ expected time by Lemma 7, and increase the size of the matching by Claim 5. We continue to do this until M becomes a perfect matching, which is possible by Proposition 2 (which ensures G has a perfect matching) and Claim 6 (which ensures M admits an augmenting path).

The size of M increases from 0 to n with this approach. When the total number of unmatched vertices is $2k$, we know from Lemma 7 that we can find an augmenting path in $O(n/k)$ expected time. Then we use Claim 5 to reduce the number of unmatched vertices by 2. The number of unmatched vertices decreases from $2n$ to 0, so the total expected running time is,

$$2n \cdot \left(\frac{1}{2n} + \frac{1}{2n-2} + \ldots + \frac{1}{4} + \frac{1}{2}\right) \leqslant 2n \cdot \sum_{i=1}^{n} \frac{1}{i} = O(n \log n),$$

where in the last step we used that the harmonic sum up to n terms is bounded by $O(\log n)$.

This proves Theorem 3 modulo the proof of Lemma 7 which we prove in the next subsection. □

2.3 Finding Augmenting Paths

In this section, we will see how to find augmenting paths by using a random walk.

The algorithm simply starts from a random unmatched vertex. Then, it picks a random edge going out of this vertex, go to the matched pair of the vertex, picks another random edge going out, and continues like this until it finds another unmatched vertex and forms an augmenting path. The algorithm is as follows.

Algorithm 1. Algorithm `Aug-Walk` for finding a single augmenting path for a matching M.

```
(i)   Pick an unmatched vertex $u_1 \in L$ at random. Start a walk $W$ from $u_1$, and repeat the following.
(ii)  Let $u \in L$ be the last vertex of the walk $W$. Pick an edge $(u, v)$ with $v \in R$ at random out of
      vertex $u$, and add it to $W$.
(iii) Return the walk if $v$ is unmatched. Otherwise, add the edge $(v, M(v))$ to $W$, and move to the
      vertex $M(v) \in L$. Continue from step $(ii)$.
```

Remark. We emphasize that `Aug-Walk` returns an augmenting *walk* on and not an augmenting path. However, since the graph is bipartite, the only cycles in this walk are even-length cycles and thus removing them forms an augmenting path as desired.

To prove Lemma 7, we need to show that algorithm **`Aug-Walk`** terminates in $O(n/k)$ time when $2k$ vertices are left unmatched by M. The walk W is not an *undirected random walk* on the underlying graph, as all the matching edges are picked deterministically. We will construct a Markov Chain from graph G and matching M that captures this walk.

Definition 8. Given a bipartite d-regular graph $G = (L \cup R, E)$ and a matching M which leaves $2k$ vertices unmatched, we construct the **matching Markov chain** as follows.

(i) Create a state for each vertex $v \in L \cup R$, and add all edges by orienting them from L to R.

(ii) For each matching edge (u, v) with $u \in L, v \in R$, combine both states u, v into one state. This state has all the edges incoming to v as well as all the edges outgoing from u.

(iii) Create a new state $\star$ which denotes the start and final states. Add edges from $\star$ to all unmatched vertices in L, and add edges from all unmatched vertices in R to $\star$.

A random walk in this Markov chain at each step takes a random outgoing edge of the current state.

The key point is that when we want to pick a matching edge from $v \in R$, there is only one choice for such an edge, and the state of the Markov chain does not change. Let us formally argue why `Aug-Walk` is a random walk on our matching Markov chain.

Claim 9. *Algorithm `Aug-Walk`, for graph G and matching M performs a random walk on the Markov chain from Definition 8.*

Proof. We start by picking an unmatched vertex at random, and moving out of state $\star$ picks such a vertex $u_1 \in L$, by step (iii) of the construction. We pick an edge (u_1, u_2) out of u_1 at random, and all edges out of u are equiprobable, by step (v). If $u_2 \in R$ is unmatched, we move out of u_2 into final state $\star$ by step (iii).

If u_2 is matched, u_2 is fused in state $u_2M(u_2)$ by step (ii). We move out of $M(u_2) \in L$ at random into another vertex from R in the graph G, which may be a matching edge state in Markov chain, or a state for an unmatched vertex in R. We repeat till we reach an unmatched vertex in R, and in one more timestep we reach final state $\star$. The walk starts from $\star$, and ends when we reach $\star$ again. □

Given Claim 9, bounding the runtime of `Aug-Walk` is the same (up to an $O(1)$ factor) as bounding the return time of the state marked by $\star$, i.e., $h_{\star,\star}$ in the matching Markov chain. To do so, we only need to compute the stationary distribution of our Markov chain and we can then apply Theorem 1 to conclude the proof. An important observation is that the Markov chain we created corresponds to a random walk on a *directed* graph, wherein in-degree and out-degree of vertices are equal. The stationary distribution of such a random walk is well-known as captured by the following lemma.

Lemma 10. *Let D be any directed graph such that for any vertex $v \in D$, $\text{in-deg}(v) = \text{out-deg}(v)$. Then, for the stationary distribution π of random walk on D (which at any vertex takes a random outgoing edge), and for any vertex $v \in D$, we have $\pi_v = \text{out-deg}(v)/m$ where m is the number of edges in D.*

Proof. Suppose we sample a vertex u from the stationary distribution and take one step of the random walk. For any vertex v, the probability that we will be at v will be

$$\begin{aligned}
&\sum_{u \in \text{in-neighbors of } v} \Pr(u \text{ was chosen in the sampling}) \cdot \Pr(\text{random walk chooses } (u,v) \text{ edge}) \\
&= \sum_{u \in \text{in-neighbors of } v} \frac{\text{out-deg}(u)}{m} \cdot \frac{1}{\text{out-deg}(u)} \\
&= \frac{\text{in-deg}(v)}{m} = \frac{\text{out-deg}(v)}{m},
\end{aligned}$$

where the final equality is by the lemma assumption. So, after this step, probability of being at v is again π_v for all $v \in D$, hence π is the stationary distribution. □

We are ready to prove Lemma 7, as we know what the stationary distribution is now.

Proof of Lemma 7. We use algorithm `Aug-Walk` to find an augmenting path in graph G for matching M. By Claim 9, we know that this is a random walk on the matching Markov chain from Definition 8.

By Theorem 1, any random walk on the matching Markov chain which starts at state $\star$ reaches $\star$ again in expected $1/\pi_\star$ steps. We know what $\pi_\star$ is from Lemma 10. Therefore, the expected number of steps of the random walk is,

$$h_{\star,\star} = \frac{1}{\pi_\star} = \frac{\text{number of edges in } D}{\text{out-deg}(\star)} = \frac{4k \cdot d + (n-k) \cdot d}{k \cdot d} \leqslant \frac{4n}{k},$$

concluding the proof. □

References

[GKK13] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings in $O(n \log n)$ time in regular bipartite graphs. *SIAM Journal on Computing*, 42(3):1392–1404, 2013. 1, 2, 4