CSE 525: Randomized Algorithms **Spring 2023**

Lecture 19: Counting Matchings

Lecturer: Shayan Oveis Gharan *May 30th*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

In this lecture we use the Path technology to generate a uniformly random matching in a graph $G = (V, E)$. We will prove the following theorem.

Theorem 19.1 ([?]). *For any graph $G = (V, E)$ and any $\lambda > 0$ there is an algorithm that runs in time polynomial in $|V|, |E|, \lambda$ and returns a an approximate sample of the distribution*

$$\pi(M) = \frac{1}{Z}\lambda^{|M|} \tag{19.1}$$

where Z is the partition function of the distribution.

Note that for $\lambda = 1$ the algorithm generates a uniformly random matching in G. As λ gets bigger it gets more likely to choose a (near) maximum matching of G. Before proving the above theorem we first give several additional examples of the Path technology and we develop a general machinery to employ this method in a generic application of Markov chains.

19.1 Path Technology for the hypercube

First, observe that the technique that we talked about in the last lecture naturally extends to fractional multicommodity flows. Namely, If for each $x, y \in \Omega$ we use a distribution of paths $\mu_{x,y}$ to route $\pi(x)\pi(y)$ commodity from x to y then, the congestion of an edge $e = (u, v)$ is defined as $\max_e \frac{f(e)}{Q(e)}$ where as before, $Q(e) = \pi(u)K(u, v)$, and

$$f(e) = \sum_{x,y} \sum_{P\sim\mu_{x,y}: e\in P} \pi(x)\pi(y)\mu_{x,y}(P).$$

Recall that we showed that inverse of the Poincaré constant is at least $\max_P |P_{x,y}| \cdot \max_e \frac{f(e)}{Q(e)}$. Next, we bound the Poincaré constant of the hypercube.

Lemma 19.2. *The inverse of the Poincaré constant of the lazy random walk on the hypercube $\{0,1\}^n$ is $O(n^2)$.*

We will construct a routing that uses only shortest paths to route from x to y. So, since the diameter of graph is n, it is enough to construct a routing such that the maximum congestion is $O(n)$.

To minimize the congestion it is enough to spread the flow between x and y evenly among all shortest paths from x to y. So, if x, y differ in k bits we consider the 2^k shortest paths from x to y and we send $\pi(x)\pi(y)/2^k$ from of mass along each of these paths.

Let $N = 2^n$. Now, let us bound the congestion. By symmetry of the hypercube the congestion of all edges are equal. Therefore, for any edge e^*,

$$f(e^*) = \frac{\sum_e f(e)}{|E|} = \frac{\frac{1}{N^2}\sum_{x,y}\{\text{length of the shortest path from } x \text{ to } y\}}{Nn/2}.$$

It is not hard to see the average distance between two random vertices in the hypercube is $n/2$. Therefore,

$$f(e^*) = \frac{\frac{1}{N^2} \cdot \frac{n}{2} N^2}{Nn/2} = \frac{1}{N}.$$

Therefore,

$$\frac{f(e^*)}{Q(e^*)} = \frac{1/N}{\frac{1}{N} \cdot \frac{1}{2n}} = 2n.$$

It follows that $\frac{1}{\alpha} \leq O(n^2)$ and

$$\tau_{\text{mix}} \leq \frac{\log \frac{1}{\pi(x)}}{\alpha} = \frac{\log \frac{1}{N}}{\alpha} \leq O(n^3).$$

Recall that using the coupling method we proved the the mixing time of the lazy random walk on a hypercube is $O(n \ln n)$. So the above bound is two order of magnitude worse the right bound. Furthermore, observe that the above bound is essentially the best possible using the path technology, because we equalized the flow on all edges.

This example shows an inherent weakness of the path technology. In the above bound we lose a factor n in bounding the Poincaré constant. In fact it can be shown that the Poincaré constant of the lazy chain on the hypercube is $1/2n$. As we will discuss in future. This loss is always at most logarithmic in $|\Omega|$ for the best multicommodity flow. We lose another factor n because we only use the Poincaré constant in bounding the mixing time. In this case, it turns out that one can use the rest of the eigenvalues of the Markov chain to only incur a loss of $\ln n$. Nonetheless, this loss is also always at most logarithmic in $|\Omega|$.

Although the Path technology has above weaknesses compared the coupling technique, It is a more general tool and it can be used in many cases that coupling technique fails completely. Although the above bound is not tight, we proved a mixing time which is polynomial in $\log |\Omega|$ using this technology. This is a remarkable fact. For a concrete example, it was shown that no Markovian coupling argument can prove rapid mixing of the Jerrum-Sinclair Markov chain, [**?**], for sampling almost uniformly from the set of perfect and near perfect matchings of a given graph[**?**]. See below for the Jerrum-Sinclair chain.

19.2 Flow Encoding

The flow that we used for the hypercube example proved rapid mixing of the chain but we used several symmetry property of the cube to analyze the maximum congestion of edges. These global symmetry properties are very hard to prove for a generic Markov chain. Instead we would like a local property that if satisfied it directly implies a bound on maximum congestion.

Let us run a though experiment. Say we are running a Markov chain with $|\Omega| = 2^n$ states on a ground set of n elements. Suppose that we are considering a uniform distribution over Ω. Then, assuming that each state x has a polynomial in n neighborhood size, e.g., in the Metropolis chain, one would have $Q(e) = \frac{\text{poly}(n)}{|\Omega|}$. Therefore, in order to prove a poly(n) bound on the inverse Poincaré constant we need to show

$$f(e) \leq \text{poly}(n)|\Omega|,$$

for all edge e. Suppose in our multicommodity flow we use a single path $P_{x,y}$ for each pair x, y. It follows that from $|\Omega| \times |\Omega|$ paths that we need to construct only poly$(n)|\Omega|$ of them can go through a fixed edge e, so

$$|\operatorname{paths}(e)| \leq \text{poly}(n)|\Omega|,$$

where paths(e) denotes the set of paths the go through e.

To prove the above inequality we need to devise a "local" technique to compare $|\operatorname{paths}(e)|$ with the size of Ω. The problem is $|\Omega|$ is usually the question of interest and it seems out of reach. The following machinery of Jerrum and Sinclair, called *injective mapping*, is designed to relate these two quantities.

Definition 19.3 (Encoding)**.** *An encoding for a flow f (that uses only single paths $P_{x,y}$ for each x, y) is a set of functions $\eta_e : \operatorname{paths}(e) \to \Omega$ (one for each edge e) such that*

i) η_e is injective,

ii) For all $e = (z, z')$ and for all x, y where $P_{x,y} \in \operatorname{paths}(e)$, we have

$$\pi(x)\pi(y) \leq \beta \cdot \pi(z)\pi(\eta_e(x, y)).$$

Note that property (ii) of Definition 19.3 is automatically satisfied when *pi* is the uniform distribution. So, it is mainly used to charge the weights correctly when the underlying distribution is not uniform. Property (i) is exactly the local property which relates $|\operatorname{paths}(e)|$ and $|\Omega|$. Note that in some applications η_e may be approximately injective. Meaning that we may use some additional $O(\log n)$ bits of information to invert η_e. The following lemma is immediate:

Lemma 19.4. *If there is an encoding for a flow f with parameter β, then*

$$\max_e \frac{f(e)}{Q(e)} \leq \beta \max_{u,v:K(u,v)>0} \frac{1}{K(u, v)}.$$

Proof. Consider an arbitrary edge $e = (u, v)$. Then,

$$f(e) = \sum_{x,y:P_{x,y} \operatorname{paths}(e)} \pi(x)\pi(y) \leq \beta\pi(u) \sum_{x,y:P_{x,y} \in \operatorname{paths}(e)} \pi(\eta_e(x, y)) \leq \beta\pi(u).$$

Here, the first inequality uses property (ii) and the second inequality uses property (i). Therefore,

$$\frac{f(e)}{Q(e)} \leq \frac{\beta\pi(u)}{\pi(u)K(u, v)} = \frac{\beta}{K(u, v)}.$$

□

19.3 Flow Encoding for the Hypercube

Before discussing the application of flow encoding in counting matchings let us use it to bound the mixing time of the lazy random walk on the hypercube. For any pair of states x, y, we use the "left-right bit-fixing path". That is we go from x to y by fixing the bits of x from left to right. Obviously, this follows a shortest path from x to y. So the maximum length of the paths that we construct is n.

Now, let us bound the congestion. By Lemma 19.4 we need to define an encoding. Consider an arbitrary edge $e = (u, v)$ and suppose u and v different in the i-th bith. Now, consider a pair x, y where $P_{x,y}$ goes through edge e. Observe that the first $i - 1$ bits of u (or v) must match y and the last $n - i$ bits of u must match x, i.e., we can write

$$u = y_1 y_2 \ldots y_{i-1} x_i, x_{i+1}, \ldots, x_n, v = y_1, \ldots, y_{i-1}, y_i, x_{i+1}, \ldots, x_n.$$

We define

$$\eta_e(x, y) = x_1, \ldots, x_i, y_{i+1}, \ldots, y_n.$$

To prove that this is an encoding we need to show that η_e is injective. So, it is enough to show that we can uniquely recover x, y from e and $\eta_e(x, y)$. But this is true by construction. Note that since the stationary distribution is uniform we have $\beta = 1$. It follows by Lemma 19.4 that the maximum congestion is at most

$$\max_{K(u,v)>0} \frac{1}{K(u,v)} = 2n.$$

Therefore, the chain mixes in $O(n^3)$ steps.

Observe that we proved a bound almost similar to the bound in the proof of Lemma 19.2 (up to constants). But, here we did not use any symmetry property of the cube and also we did not use any information on $|\Omega|$. Next, we use this idea to analyze a Markov chain on sampling matchings.

19.4 Counting Matchings

In this section we prove Theorem 19.1. We note that the problem of counting matchings in a given graph is a #P-complete problem for any fixed $\lambda > 0$. We start by defining the Markov chain.

Markov Chain. Let Ω be the set of all matchings of G. We use the Metropolis rule on three set of actions: Edge deletion, Edge addition, and Edge exchange. Given a matching M we use the following rules to choose a neighbor of M.

- With probability $1/2$ stay at M (This ensures laziness).
- Otherwise, we choose an edge $e = (u, v) \in E$ uniformly at random and we do the following

  Edge Addition: If none of the endpoints of e are saturated in M we go to $M + e$ with probability $\min\{1, \lambda\}$.

  Edge Deletion: If $e \in M$ we delete M we go to $M - e$ with probability $\min\{1, 1/\lambda\}$.

  Edge Exhange: If exactly one of the endpoints of e, say u are saturated in M, let $e' \in M$ be the unique edge incident to u. We go to $M - e' + e$.

Observe that this chain is obviously following the Metropolis rule. So it is reversible and the corresponding stationary distribution is (19.1). Furthermore, it is not hard to see that the Markov chain is irreducible because we can go from any matching M to the empty state and go to any arbitrary matching M'.

Flow: Next, we define a flow for the above Markov chain. Consider any two matchings x, y. We want to construct a unique path $P_{x,y}$ that moves x to y following the actions of the chain. Consider the graph $x \oplus y$, that is the symmetric different of x, y. It follows that $x \oplus y$ is a union of paths an even cycles. Let us color the edges of x by red and the edges of y by blue. So, we want to construct a path to turn the red edges to blue.

In order to construct $P_{x,y}$ first we define an ordering on all cycles and all paths of G. Note that there may be exponentially many such objects but we need this ordering only for the sake of the analysis. Furthermore, for every cycle and every path we define a vertex as the “start vertex”. We make sure that the start vertex of a path is one of its two endpoints. Note that this global ordering also induces an ordering on cycles/paths of $x \oplus y$. That is the ordering we use to construct $P_{x,y}$.

To construct $P_{x,y}$ we process the paths and cycles in $x \oplus y$ in the order defined above. To process a path with r edges $e_1, \ldots, e_r$ and e_1 being the edge with the first vertex we do the following

1) If e_1 is red, then remove e_1, exchange e_3 for e_2, exchange e_5 for e_4, and so on. If e_r is blue, we have the additional move of adding e_r at the end.

2) If e_1 is blue, then exchange e_2 for e_1, exchange e_4 for e_3, and so on. If e_r is blue, we have the additional move of adding e_r at the end.

For an even cycle with edges $e_1, \ldots, e_{2r}$ let e_1 be the red edge incident to the start vertex. First, we delete e_1, and then we exchange e_3 for e_2, exchange e_5 for e_4 and so on. Finally, we will add e_{2r}.

Encoding: Consider an edge $e = (u, v)$ we need to define the injective mapping $\eta_e(x, y)$. We simply define

$$\eta_e(x, y) = x \oplus y \oplus (u \cup v).$$

There is a fundamental reason for this definition. In particular, given $\eta_e(x, y)$ and $e = (u, v)$ we can reconstruct $x \oplus y$ simply as $\eta_e(x, y) \oplus (u \cup v) = x \oplus y$. There is a little bit of technicality that we should be aware of.

Firstly, note that u, v always differ in a single edge. The reason we xor $x \oplus y$ with $(u \cup v)$ as opposed to u is to make sure that $\eta_e(x, y)$ is a matching. It turns out that, with the above definition, $\eta_e(x, y)$ is always a matching except in the special case where we go through edge (u, v) when we are processing a cycle. In that case we need to remove the first edge of the cycle, e_1, to make sure that $\eta_e(x, y)$ is a matching, i.e.,

$$\eta_e(x, y) = x \oplus y \oplus (u \cup v) - e_1.$$

This modification of η_e may seem problematic because it seems impossible to recover e_1 from η_e and u, v. First of all, note that even if that was completely impossible it was not a big deal, because we could find e_1 in our decoding step using $O(\log n)$ bit additional information. In other words, we could only guarantee that η_e is approximately injective. Secondly, we know that if we are missing e_1 it has to be an edge completing the path we are processing at the moment into a cycle. So, in fact there are only two possibilities: Wether the path that we are processing when going through (u, v) is a cycle or not.

Lastly, observe that once we have $x \oplus y$ we can easily reconstruct x, y. By looking at u, v we can figure out the path that we are processing at the moment. Then for every path that preceeds the current one, we know that the path agrees with y in u and with x in $\eta_t(x, y)$. For paths following the current one, the parity is reversed.

Bounding the Mixing Time: First observe that in the above construction $u \cup \eta_e(x, y)$ has at most two edges less than $x \cup y$. Therefore,

$$\pi(x)\pi(y) \le \lambda^2 \pi(u)\pi(\eta_e(x, y)).$$

Therefore, we can let $\beta = \lambda^2$ in Definition 19.3. In addition observe that for any edge u, v

$$K(u, v) \ge \frac{1}{\lambda |E|}.$$

Therefore, by Lemma 19.4, we have the maximum congestion is at most $O(\lambda^3 |E|)$. Furthermore, observe that the length of all paths that we construct is $O(n)$ because both x, y have at most $O(n)$ edges. Therefore,

$$\tau_{mix} \le O(\lambda^3 |E| n \log \frac{1}{\pi(x)}).$$

where x is the starting state. Now, if start the process from the maximum matching of G (if $\lambda \ge 1$) and from the empty set otherwise, we have that

$$\pi(x) \ge \frac{1}{|\Omega|}$$

Note that the number of matchings of size n is at most $\binom{|E|}{n}$, and this term dominates the maximum possible number of smaller size matchings. It follows that $\log(\frac{1}{\pi(x)}) \leq O(n \log n)$. Therefore, the chain mixes in $O(\lambda^3 |E| n^2 \log(n))$.

References