CS 7880: Algorithms for Big Data (Fall'22)	Northeastern University
Lecture 9	
October 11, 2022	
Instructor: Soheil Behnezhad	*Scribe: Omer Wasim*

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

1 Sublinear Algorithms for Maximum Matching (Part II)

In the last lecture, we gave a proof of the following theorem which appeared in a seminal paper by Yoshida, Yamamoto and Ito [1].

Theorem 1. *Let $G = (V, E)$ be a graph of maximum degree Δ and let $\mu(G)$ denote the size of the maximum matching of G. For any $\epsilon > 0$, there is an $\tilde{O}(\Delta^3)$ time algorithm to compute a $(\frac{1}{2}, \epsilon)$-approximation of $\mu(G)$ w.h.p in the adjacency list model. More precisely, the output $\tilde{u}$ of the algorithm satisfies,*

$$\frac{1}{2}\mu(G) - \epsilon n \leq \tilde{\mu} \leq \mu(G)$$

The high level idea was to use the `IsInMIS` subroutine on the line graph $L(G)$ of G, and since a MIS on $L(G)$ corresponds to a maximal matching of G, we obtained an upper bound on the number of queries to determine whether an edge in G was contained in a maximal matching or not. Then, to estimate the size of the maximum matching, we sampled $O(\frac{\log n}{\epsilon^2})$ vertices (in the line graph) and called the subroutine `IsInMM` to determine whether a vertex is matched or not. By applying the Chernoff bound, we obtained concentration.

> **Remark.** We also noted in the previous lecture that with a slightly more efficient implehmentation of the oracles, the running time can be improved to $\tilde{O}(\Delta^2)$.

In this lecture, we will sketch the main ideas towards obtaining an $\tilde{O}(\bar{d})$ running time where $\bar{d}$ denotes the average degree. This gives a truly sublinear algorithm for all graphs since always $\bar{d} = O(m/n) \ll m$. The near-tight analysis of the average query complexity for greedy maximal matching [1] which results in the improved bound is due to Behnezhad [2].

2 An improved bound

We give the following procedures.

`Vertex-Oracle`(v, π)**:**
Input: Vertex v and a permutation π over edges.
Output: Returns True if v is matched.

1. Let $(v, u_1), (v, u_2), ..., (v, u_d)$ be the edges incident to v such that $\pi(v, u_1) < \pi(v, u_2) < ... < \pi(v, u_d)$.
2. For $i = 1$ to d:
   - **If** `Edge-Oracle` $(e_i, u_i, \pi) =$ True:
     return True.
3. **return** False.

`Edge-Oracle`(e, u, π):
Input: Edge e, endpoint u of e and a permutation π over edges.
Output: Returns True if e is in the matching.

If `Edge-Oracle`(e, u, π) has already been computed before, return the value. **Else:**

1. Let $e_1 = (u, w_1), e_2 = (u, w_2), ..., e_d = (u, w_d)$ be the edges incident to u such that $\pi(e_1) < \pi(e_2) < ... < \pi(e_d)$.
2. **For** $i = 1$ to d:
   - **If** `Edge-Oracle` (e_i, w_i, π) = True:
     return False.
3. **return** True.

Exercise. Prove the correctness of `Vertex-Oracle`.

The rest of the lecture will be devoted towards giving a proof sketch of the following theorem.

Theorem 2. *[2] For a random vertex v and a random permutation π, `Vertex-Oracle`(v, π) calls the procedure `Edge-Oracle` $O(\bar{d} \log n)$ times.*

We note that the theorem immediately yields the following corollary.

Corollary 3. *For any $\epsilon > 0$, there is an $O(\Delta \bar{d} \log n)$ time algorithm to compute a $(\frac{1}{2}, \epsilon)$-approximation of $\mu(G)$ w.h.p in the adjacency list model.*

With a careful implementation of the oracles, we can improve the running time to $\tilde{O}(\bar{d})$. However, we only give a sketch of Theorem 2 below.

We define a query path as follows.

Definition 4. A query path P at any given point in time during the execution of `Vertex-Oracle`(v, π) is a path in the graph G corresponding to the stack of recursive calls to the procedure `Edge-Oracle`.

Lemma 5. *For a random permutation π, the longest query path is of length $O(\log n)$ with probability at least $1 - \frac{1}{n^2}$.*

> **Remark.** A proof of Lemma 5 is given in [2] via a reduction to the parallel depth of randomized greedy MIS and a nice result of Fischer and Noever [3] which bounds it by $O(\log n)$. There is a much simpler proof, yielding a bound of $O(\log^2 n)$ (see Homework 1). Using the latter bound only increases the claimed running time by an extra $\log n$ factor to $O(\bar{d} \log^2 n)$.

For an edge $e = (u, v)$, we let $P(e, \pi)$ denote the number of times that `Edge-Oracle`$(e, \cdot, \cdot)$ is called if `Vertex-Oracle`(w, π) is called for all $w \in V$.

Claim 6. *For every edge e, it holds that $\mathbb{E}_\pi[P(e, \pi)] = O(\log n)$.*

Before giving a proof sketch for Claim 6, we give a proof of Theorem 2 via Claim 6.

Proof of Theorem 2. We let $Q(v, \pi)$ denote the *total* number of calls to `Edge-Oracle` when we call `Vertex-Oracle`(v, π).

Note that this is exactly what we want to bound for the statement of Theorem 2. We have that,

$$\begin{aligned}\sum_{v \in V} \mathbb{E}_{\pi}[Q(v,\pi)] &= \sum_{e \in E} \mathbb{E}_{\pi}[P(e,\pi)] \\ &= \sum_{e \in E} O(\log n) \\ &= O(m \log n)\end{aligned}$$

where the first inequality follows from the definition of $P(e,\pi)$ and the second inequality follows from Claim 6. This implies that the number of times `Vertex-Oracle`(v,π) calls `Edge-Oracle` for a random vertex v and random permutation π is,

$$\mathbb{E}_{v,\pi}[Q(v,\pi)] = \frac{1}{n} \sum_{v \in V} \mathbb{E}_{\pi}[Q(v,\pi)] = O(\frac{m}{n} \log n) = O(\bar{d} \log n)$$

which completes the proof. □

2.1 Proof Sketch of Claim 6

The idea is similar to before: we blame $P(e,\pi)$ other permutations. Such permutations π' differ only on a subset of edges-in particular edges on the query path. Take a query path $P = (w, ..., u, v)$ which ends in edge (u,v). Intuitively, we want to bound the number of queries *into* e if `Vertex-Oracle`(x,π) was called on all $x \in V$. We let $BL(\pi, P)$ be the blamed permutation obtained by rotating the ranks on the path P by 1 (See Figures 1-3). In particular, the rest of the permutation is unchanged. We will illustrate the idea with an example (which can be formalized similarly to the previous lecture).

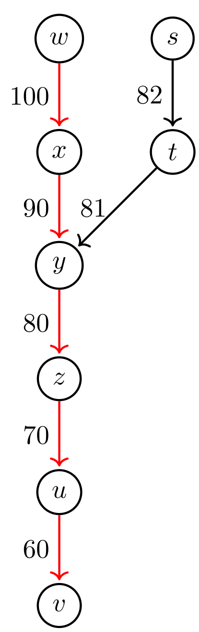

Figure 1: The path P shown in red.

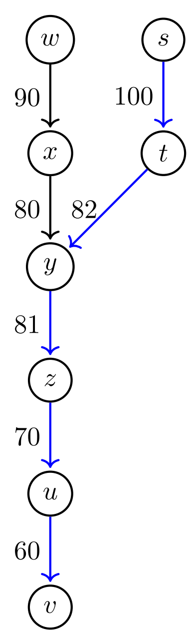

Figure 2: The path P' shown in blue.

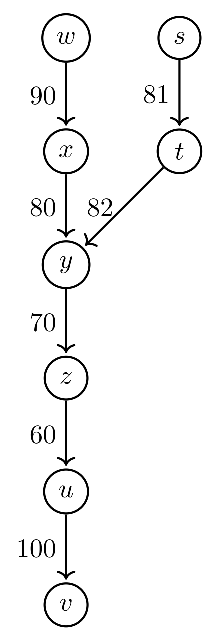

Figure 3: The blame permutation $BL(\pi, P)$ showing ranks rotated by 1.

Figure 1 and Figure 2 show two different query paths P and P' ending in the same edge $e = (u,v)$. The claim is that if $BL(\pi,P) = BL(\pi',P')$, then one of P and P' is a subpath of the other. We prove this by contradiction. First note that by definition of the blame permutation, π and π' are the same for every edge not shown in the figures-each such edge is assigned a rank from $1, ..., 79$. Let $e' = (t,y), e'' = (x,y)$ and

$f = (y, z)$ be the edges. Since e'' queries the edge f which has rank 80 in π (and 81 in π') implies that e'' has no edge incident to it with rank less than 80 in the greedy maximal matching $GMM(\pi)$. Since π and π' are the same on ranks $1, ..., 79$ this means that e'' in π' has no edge incident to it with rank less than 80; this implies that $e'' \in GMM(\pi')$. However, if this is the case then the edge e' in π' would first query e'' and immediately terminate. This means that P' is not a valid query path, which is a contradiction.

Moving on, let $\mathbb{P}_m$ denote the set of all permutations on m elements. Let π_L denote the set of 'likely' permutations where all query paths in the graph have length $O(\log n)$ and $\pi_U = \mathbb{P}_m \backslash \pi_L$ denote the set of all other 'unlikely' permutations. Consider the bipartite (blame) graph G_n on vertex sets L and R, where $|L| = |R| = |\mathbb{P}_m|$ and each vertex of L and R corresponds to a permutation of π. The edge set of G_n consists of all edges of the form (ℓ, r) where $\ell \in L$, and $r \in R$, such that r is blamed by ℓ. Now for any vertex $\ell \in L \cap \pi_L$ its degree is at most $O(\log n)$ since for likely permutations, the query path length is only $O(\log n)$ by definition, and thus only $O(\log n)$ paths ending in (u, v) (corresponding to permutations in R) can be blamed. If we show that for a random vertex selected from L has degree $O(\log n)$ then we are effectively done since this average degree corresponds to the quantity $\mathbb{E}_\pi[P(e, \pi)]$.

We now observe that $P(e, \pi)$ is bounded by $O(n^2)$. For any vertex w, `Vertex-Oracle` calls `Edge-Oracle` with w as the second argument at most $O(n)$ times. For any edge e. `Edge-Oracle`(e, π) is called at most once by edges incident to it since calls to `Edge-Oracle` are cached when `Vertex-Oracle` is called for any vertex w. Thus, calling `Vertex-Oracle` on all n vertices results in at most $O(n^2)$ calls to `Edge-Oracle`(e, π).

Let us now bound the average degree of any vertex $\pi \in L$. Note that the number of edges incident to π_L are bounded by $O(m! \log n)$ since there are only $m!$ total vertices in L and each vertex in π_L has degree at most $O(\log n)$ as noted earlier. Next, note that the total number of edges incident to all $\ell \in \pi_U$ is given by, $|\pi_U| n^2 = \frac{m!}{n^2} = O(m!)$ where the first inequality follows from Lemma 5 and the n^2 term comes from our observation. For a random permutation, $\pi \in \mathbb{P}_n$, the average degree in graph G_n is then $\frac{O(m! \log n)}{m!} = O(\log n)$.

References

[1] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 225–234, New York, NY, USA, 2009. Association for Computing Machinery. 1

[2] Soheil Behnezhad. Time-optimal sublinear algorithms for matching and vertex cover. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 873–884, 2022. 1, 2

[3] Manuela Fischer and Andreas Noever. Tight analysis of parallel randomized greedy MIS. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2152–2160, 2018. 2