ECS-189A Special Topics in TCS April 25, 2023

Lecture 5

Lecturer: Slobodan Mitrović

1 Introduction

The next two lectures will investigate the computation of approximate maximum matching in MPC. Similar to how we approached CC, we will first see an algorithm for computing maximal matching in the super-linear regime, and then provide an approach that is applicable to the sub-linear regime as well. This lecture is in large based on [LMSV11].

2 Preliminaries

2.1 Notation

We will use $G = (V, E)$ to denote a graph on the vertex set V and edge set E. If not stated otherwise, we will use n to denote $|V|$ and m to denote $|E|$. We will use $N_G(v)$ to denote the neighborhood of v in G; this neighborhood excludes v itself. When it is clear from the context, we will omit the subscript G and use $N(v)$ instead. We use $d(v)$ to denote the degree of vertex v in G, i.e., $d(v) = |N(v)|$.

We will use the notation $\tilde{O}(f)$ to hide poly-logarithmic factors in f, i.e., $\tilde{O}(f) = O(f \cdot \text{poly} \log f)$.

By saying that an event $\mathcal{E}$ happens *with high probability* (whp), we mean that $\Pr[\mathcal{E}] \geq 1 - n^{-c}$ for some constant $c \geq 1$.

2.2 Matchings

A *matching* M in a graph $G = (V, E)$, is defined to be a set of *pairwise* non-adjacent edges. An edge in a graph G is said to be a matched edge if it is in M, and a vertex in a graph is said to be "matched" if it is an endpoint of any matched edge. Otherwise, the vertex is said to be "unmatched".

In Figure 1, $M = \{1, 4, 5, 9\}$ forms a matching of the given graph. Note that if we take any pair of edges in M, then they do not share any endpoints

2.2.1 Maximal Matching

A matching M in a graph $G = (V, E)$ is a *maximal* matching if every edge in G has a non-empty intersection with at least one edge in M. In other words, a maximal matching is one which is not a proper subset of any other matching, i.e., if $\tilde{M}$ is the set of all matchings of a graph G, then M is a maximal matching if $\forall M' \in \tilde{M}$, it holds that $M \not\subset M'$.

Note that in Figure 2, there can be two possible matching sets, $M_1 = \{1, 3\}$ and $M_2 = \{2\}$. It can be discerned that $M_1 \not\subset M_2$ and $M_2 \not\subset M_1$. Hence, both M_1 and M_2 are maximal matchings, but we have only marked M_2 in the aforementioned figure.

2.2.2 Maximum Matching

A matching $M^\star$ in a graph $G = (V, E)$ is a *maximum* matching if it contains the largest possible number of edges. In other words, it has the maximum cardinality amongst all maximal matchings in a graph G. In a maximum matching problem, we want to find the matching with the largest cardinality.

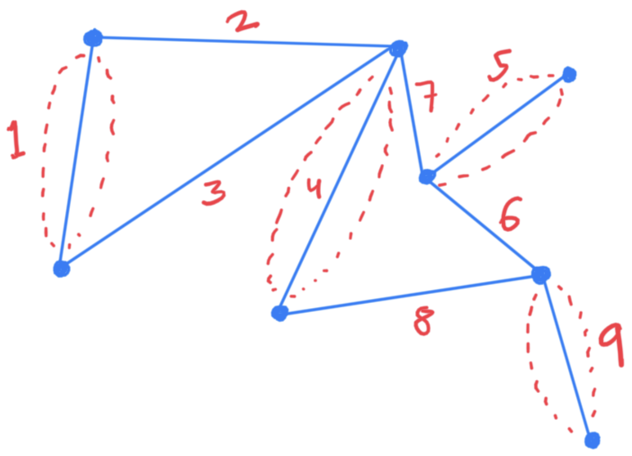

Figure 1: Edge set which forms a matching are circled in red.

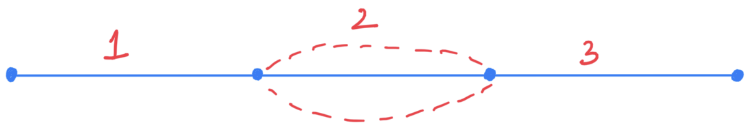

Figure 2: The edge 2 is a maximal matching.

As explained in Section 2.2.1, both M_1 and M_2 are maximal matchings. But since $|M_1| = \max(|M_1|, |M_2|)$, thus, $M^\star = M_1$ is a maximum matching.

2.2.3 Maximum vs. Maximal matching

As mentioned in Sections 2.2.1 and 2.2.2, Figure 2 has two possible maximal matchings M_1 and M_2. However, only M_1 is a maximum matching, as demonstrated in Figure 3. Hence, a maximal matching with the maximum number of edges is a maximum matching. **Note:** Every maximum matching is a maximal matching, but not vice versa.

Lemma 1. *Let $G = (V, E)$ be a graph. If M is an arbitrary maximal matching, and $M^\star$ is a maximum matching, then it holds that $|M| \leq |M^\star| \leq 2|M|$.*

Proof. Note that since M is a maximal matching and $M^\star$ is a maximum matching. So, by definition, the left inequality $|M| \leq |M^\star|$ holds.

Now, we want to prove $|M^\star| \leq 2|M|$. Our proof goes by, in a sense, a counting argument. For a moment, assume that the choice of M is the worst possible, i.e., by choosing an edge in M we are ruling out choosing more than 1 edge in a matching. To illustrate this, consider Figure 4. Consider edge by edge $\tilde{e}_i \in M$, and delete the red edges $e_j \in M^\star$ in which $\tilde{e}_i$ and e_j share a common vertex. Note that for each edge in M, either this edge exists in $M^\star$, or it has *at most two* incident edges in $M^\star$. (As a side remark, it cannot be the case that $\tilde{e}_i \notin M^\star$ does not intersect $M^\star$ because in that case, we can add $\tilde{e}_i \in M$ to $M^\star$ and that contradicts with the definition of a maximum matching.)

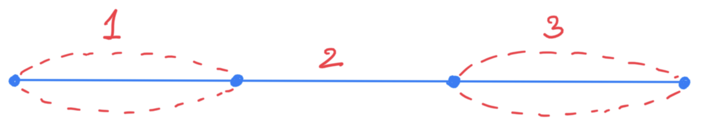

Figure 3: Edge set $M = \{1, 3\}$ forms a maximum matching in this graph.

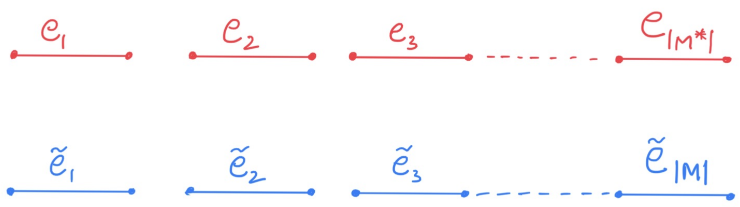

Figure 4: Red edge set is a maximum matching $M^\star$ and Blue edge set is a maximal matching M. We use this figure in the proof of Lemma 1.

Also, for any edge in M existence of more than two incident edges in $M^\star$ is not possible because that results in having two edges sharing a common vertex. Accordingly, every edge in M removes at most two edges in $M^\star$. If a red edge does not end up being removed, that would contradict the definition of a maximal matching M – at least one of those leftover edges can be added to M. Eventually, all edges in $M^\star$ are removed, and it is evident that the cardinality of the maximum matching must be bounded by twice the cardinality of a maximal matching. Hence, it holds that $|M^\star| \le 2|M|$. □

2.2.4 Approximate maximum matching

Given a matching M, if it holds that $|M^\star| \le c|M|$ we say that M is a c-approximate maximum matching. In particular, in light of Lemma 1, a maximal matching is a 2-approximate maximum one.

2.3 Probability tools

Theorem 2 (Markov's inequality)**.** *If X is a nonnegative random variable and $a > 0$, then*

$$\Pr[X \ge a] \le \frac{\mathbb{E}[X]}{a}.$$

Theorem 3 (Chernoff bound)**.** *Let $X_1, \ldots, X_k$ be independent random variables taking values in $[0, 1]$. Let $X \overset{\text{def}}{=} \sum_{i=1}^{k} X_i$ and $\mu \overset{\text{def}}{=} \mathbb{E}[X]$. Then,*

(A) For any $\delta \in [0, 1]$ it holds $\Pr[|X - \mu| \ge \delta\mu] \le 2\exp\left(-\delta^2\mu/3\right)$.

(B) For any $\delta \in [0, 1]$ it holds $\Pr[X \le (1-\delta)\mu] \le \exp\left(-\delta^2\mu/2\right)$.

(C) For any $\delta \in [0, 1]$ it holds $\Pr[X \ge (1+\delta)\mu] \le \exp\left(-\delta^2\mu/3\right)$.

(D) For any $\delta \ge 1$ it holds $\Pr[X \ge (1+\delta)\mu] \le \exp(-\delta\mu/3)$.

3 Computing Maximal Matchings

3.1 Centralized setting

We first recall a well-known centralized algorithm for computing a maximal matching, it is given as Algorithm 1.

```
   Input : Graph $G = (V, E)$
1  $M \leftarrow \emptyset$
   /* The edges of $E$ in the loop below can be visited in any order.        */
2  for $e = \{u, v\} \in E$ do
3      $M \leftarrow M \cup \{e\}$
4      Remove $u$ and $v$ from $G$.
5  return $M$
```

Algorithm 1: A centralized algorithm for finding a maximal matching.

3.2 Super-linear memory regime of MPC

In this section, we present an algorithm for finding a maximal matching in the super-linear MPC memory regime. Our algorithm is given as Algorithm 2, and it corresponds to work [LMSV11].

```
    Input : Graph $G = (V, E)$
            Space per machine $S = n^{1+c}$
1   Let $M \leftarrow \emptyset$
2   Let $E_0 \leftarrow E$
3   Let $i \leftarrow 0$
4   while $E_i \neq \emptyset$ do
5       Let $E'$ be a set of edges sampled from $E_i$ such that each edge is sampled uniformly at
          random with probability $p_i = \frac{S}{10|E_i|}$. Gather these edges on Machine 1.
6       if $|E'| > S$ then
7           The algorithm fails.
8       else
9           Let $M'$ be a maximal matching computed on $E'$.
10          $M \leftarrow M \cup M'$
11          Let $V_i$ be the set of unmatched vertices in $G$.
12          $E_{i+1} = E(G[V_i])$
13      $i \leftarrow i + 1$
14  return $M$
```

Algorithm 2: An algorithm for finding a maximal matching in the super-linear memory regime of MPC.

3.3 Correctness

If Algorithm 2 does not fail on Line 7, it outputs a maximal matching by design. We now analyze the failure probability.

Lemma 4. *Algorithm 2 does not fail with high probability.*

We will not cover this in class.

Proof. We now compute $\mathbb{E}[|E'|]$ in the i-th round. For an edge $e \in E_i$, let X_e be an indicator variable

$$X_e = \begin{cases} 1, & \text{if } e \text{ is sampled} \\ 0, & \text{otherwise} \end{cases}$$

Then, we have $\mathbb{E}[X_e] = p_i$. Also, it holds that $|E'| = \sum_{e \in E_i} X_e$, and hence

$$\mathbb{E}[|E'|] = \mathbb{E}\left[\sum_{e \in E_i} X_e\right] = \sum_{e \in E_i} \mathbb{E}[X_e] = |E_i| \cdot p_i = \frac{S}{10}.$$

Note that all the edges are sampled independently, and hence X_e random variables are independent. Hence, we can apply the Chernoff bound, Theorem 3, to argue about the concentration of $|E'|$. Specifically, we have

$$\Pr[|E'| \geq 2\mathbb{E}[|E'|]] = \Pr\left[|E'| \geq \frac{S}{5}\right] \leq \exp\left(-\frac{\mathbb{E}[|E'|]}{3}\right) \leq 2^{-n}.$$

Hence, with an exponentially small probability in n the algorithm fails in a single round. As we will show, the algorithm takes $O(1/c)$ rounds. Taking union bound over all the rounds, the algorithm fails with probability at most $O(2^{-n}/c)$. □

3.4 Implementation in MPC

We, again, are not going to dive into MPC implementation of low-level steps. The non-trivial step is ignoring edges that are incident to matched vertices; this is needed for Line 11. Some of the prior works explain how to achieve that by using [GSZ11]; one such publication is [CŁM$^+$18].

Obtaining set E' can be done by simply sampling edges locally, and then sending all sampled ones to machine 1.

3.5 Round complexity

Theorem 5. *Algorithm 2 runs in $O(1/c)$ rounds and outputs a maximal matching whp.*

The failure probability of Algorithm 2 is analyzed by Lemma 4. If the algorithm does not fail, then it outputs a maximal matching.

Our proof of Theorem 5 consists of two parts. First, we will argue about a certain measure of progress the algorithm makes in each round/iteration. Second, we will derive from that claim the round complexity.

Theorem 6 (Union Bound)**.** *Boole's Inequality or Union Bound states that for at most countable set of events, the probability of one of the events happening is no greater than the sum of the probabilities of each individual event. Mathematically, if $E_1, E_2, E_3 \cdots$ are the events, then it holds that*

$$\Pr\left[\bigcup_{j=1}^{\infty} E_i\right] \leq \sum_{j=1}^{\infty} \Pr[E_i].$$

Lemma 7. *Let $E' \subseteq E$ be a set of edges chosen independently with probability p. Then with probability at least $1 - e^{-n}$, for all $I \subseteq V$, either $|E[I]| < 2n/p$ or $E[I] \cap E' \neq \emptyset$.*[1]

[1]For the sake of brevity, we use $E[I]$ to denote $E(G[I])$, i.e., $E[I]$ refers to the edges in the subgraph induced on a vertex subset I.

Proof. Consider one such subgraph, $G[I] = (I, E[I])$ with $|E[I]| \geq 2n/p$. The probability that none of the edges in $E[I]$ were chosen to be in E' equals

$$\Pr[E[I] \cap E' = \emptyset] = \prod_{e \in E[I]} \Pr[e \notin E'] = (1-p)^{|E[I]|} \leq (1-p)^{2n/p}. \tag{1}$$

Next, from Taylor's expansion it follows $e^{-1} \geq (1-x)^{1/x}$ for $|x| < 1$. Applying this to Eq. (1) yields

$$\Pr[E[I] \cap E' = \emptyset] \leq (1-p)^{2n/p} \leq e^{-2n}.$$

Since there are at most 2^n total induced subgraphs $G[I]$, the probability that there exists one such that $|E[I]| \geq 2n/p$ and $G[I]$ does not have an edge in E' is at most $2^n e^{-2n}$; this probability is calculated by using Union bound (Theorem 6) over all the subgraphs. Note that $2^n e^{-2n} = \left(\frac{2}{e}\right)^n e^{-n} \leq 1 \cdot e^{-n}$. □

Next, we employ Lemma 7 to conclude the proof of Theorem 5. Consider iteration i. Lemma 7 essentially claims that if an induced subgraph $G[I]$ has at least $2n/p_i$ edges, then whp it intersects E'. If $G[I]$ intersects E', e.g., $e = \{u, v\} \in E(G[I]) \cap E'$, and since Algorithm 2 finds a maximal matching in E', that maximal matching contains at least one of u and v. It in turn means that whp only a proper subset of I will remain among the unmatched vertices. That is, whp it holds that $|E(G[V_i])| < 2n/p_i$. From this and from the definition of p_i given on Line 5, we further derive that whp it holds

$$|E_{i+1}| = |E(G[V_i])| \leq \frac{2n}{p_i} = \frac{2n \cdot 10|E_i|}{S} = \frac{20|E_i|}{n^c}.$$

This implies that the number of edges among unmatched vertices from iteration to iteration whp drops by a factor of $n^c/20$ at least; we are assuming that our graph is large and hence $n^c \gg 20$. Therefore, the number of edges between unmatched vertices whp becomes at most n^{1+c} within $O(1/c)$ many rounds.

This completes the analysis.

3.6 Further discussion

This discussion is inspired by an excellent question posed by your peer, Gaash David.

You might wonder why we could not find a maximal matching the same way as we found connected components in Lecture 2. That is, what goes wrong if we perform the following: each machine among its n^{1+c} edges finds a maximal matching; all these maximal matchings are sent to n^c fewer machines; we repeat this?

Let X be the number of machines. To see what can go wrong, assume that machine $k = 1 \ldots X$ has edges: $\{c, a_k\}$ and $\{a_k, b_k\}$. Moreover, assume that machine k outputs maximal matching $\{c, a_k\}$ and removes $\{a_k, b_k\}$ from further computation. This scenario is bad because among the edges $\{c, a_k\}$, for all $k = 1 \ldots X$, a maximum matching has size 1. On the other hand, a maximum matching among the edges $\{c, a_k\}$ and $\{a_k, b_k\}$ for all $k = 1 \ldots X$ has size X. In other words, if we adopt the algorithm from the paragraph above, in the process we might significantly reduce the size of the final matching. This example is just to illustrate that matchings do not have the properties we relied on while designing our algorithm for connectivity.

This example can be generalized in a way so that the underlying graph is much more dense.

References

[CŁM$^+$18] Artur Czumaj, Jakub Łącki, Aleksander Mądry, Slobodan Mitrović, Krzysztof Onak, and Piotr Sankowski. Round compression for parallel matching algorithms. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 471–484, 2018.

[GSZ11] Michael T Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In *ISAAC*, volume 7074, pages 374–383. Springer, 2011.

[LMSV11] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 85–94, 2011.