ECS-189A Special Topics in TCS May 4, 2023

Lecture 6

Lecturer: Slobodan Mitrović

1 Introduction

In this lecture, we continue studying the computation of approximate maximum matchings in MPC, but this time in the sub-linear memory regime. The algorithm we present is a version of the so-called peeling algorithm that, with particular adaptations, has been used in numerous research articles [II86, IS86, OR10, CŁM+18, GGK+18, GU19]. We will see a simplified version of that approach, where simplification comes at the expense of the approximation factor. Nevertheless, this approximation factor remains constant and, in your homework, you will show that such an algorithm can be in a black-box manner used to construct a $(2+\varepsilon)$ -approximate maximum matching with a relatively small overhead in the round complexity.

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., $\tilde{O}(f) = O(f \cdot \operatorname{poly} \log f)$.

By saying that an event $\mathcal{E}$ happens *with high probability* (whp), we refer 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 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".

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 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 $M \not\subset M'$.

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.

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

Every maximum matching is a maximal matching, but not vice versa. Proof of the following claim is given in the previous lecture.

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 $|M| \leq |M^\star| \leq 2|M|$.*

2.3 Probability tools

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

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}.$$

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

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

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

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

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

3 Computing Approximate Maximum Matchings in the Sublinear Memory Regime

3.1 Maximal matching and MIS

In Lecture 4, we saw an algorithm for computing a maximal independent set of vertices in the sublinear regime. Maximal matching is a maximal independent set of edges in a graph. So, it is natural to wonder whether we can use the algorithm from Lecture 4 to find a maximal matching in MPC – in this version, each edge would select a number uniformly at random. At first sight, it might seem that such an algorithm would require too much memory. Namely, an edge $e = \{u, v\}$ would need to learn the minimum value among all the edges incident to e. If edge e gets information from all its incident edges, that amounts to $O(d(u) + d(v))$ messages. Summing up the number of messages that have to be exchanged between all the edges by using this approach, it amounts to $\sum_{v \in V} d^2(v)$ many of them, which can be significantly larger than $2m = \sum_{v \in V} d(v)$. Also, recall that a standard assumption we are making is that the total memory across all the machines is $O(m \operatorname{poly} \log n)$, which also upper-bounds the total size of messages the machines can exchange.

Fortunately, finding the minimum value among the edges incident to $e = \{u, v\}$ can be done much more efficiently. Namely, each vertex w first computes the minimum value among the edges incident to w; call this value y_w. This computation can be performed using $O(m)$ total communication in the same way as we did in Lecture 4. Then, the minimum value among the edges incident to e equals $\min\{y_u, y_v\}$.

3.2 Our main algorithm

Even though we could re-use the algorithm seen in Lecture 4 to compute a maximal matching in the sublinear regime, we will study another approach, given as Algorithm 1. Variants of that algorithm have applications in many settings, including dynamic, streaming, MPC linear memory regime, and local computation algorithms. Approaches akin to Algorithm 1 are sometimes also called peeling algorithms – Algorithm 1 "peels" the graph step by step by removing the highest degree vertices.

Input :
A graph $G = (V, E)$
Space per machine $S = n^c$
1 $M \leftarrow \emptyset$
2 **for** $i = 1 \dots \log n$ **do**
3 Let A be the set of edges of G such that each edge is sampled with probability $\frac{2^i}{2n}$ and independently of other vertices.
4 Let $\tilde{A}$ be the subset of all edges of A which do not share endpoints with any other edge in A.
5 $M \leftarrow M \cup \tilde{A}$
6 Remove from G all the vertices with endpoints in $\tilde{A}$ and all the vertices whose degree is at least $\frac{n}{2^i}$.
7 **return** M

Algorithm 1: An algorithm for finding an approximate maximum matching.

3.3 Correctness

Algorithm 1 outputs a matching as (1) the set $\tilde{A}$ added to M on Line 5 consists of edges which are a matching themselves, and (2) after an edge is added to M its endpoints are removed from G on Line 6.

3.4 Implementation in MPC

We are again not going to dive into MPC implementation of low-level steps. The non-trivial step is removing vertices, computing the degree of a vertex, and obtaining $\tilde{A}$ from A on Line 4. Some of the prior works explain how to achieve that by using [GSZ11]; some of those publications are [CŁM+18, ASS+18].

3.5 Round complexity

The algorithm uses $O(\log n)$ iterations by design. Each iteration can be implemented in $O(1)$ MPC rounds.

3.6 Approximation guarantee

The most challenging part of this algorithm is showing that it outputs a reasonably good approximation. We will approach that task by showing that, in expectation, the number of vertices removed on Line 6 is a constant factor of the number of vertices in $\tilde{A}$.

For the analysis, we will need the following fact.

Fact 4. *For any* $0 \leq x \leq 1$*, it holds* $1 - x/2 \geq e^{-x}$.

Proof. This is a calculus exercise that can be proved via Taylor's expansion. Namely,

$$e^{-x} = \sum_{i \geq 0} \frac{x^i}{i!}.$$

The first several terms of this expansion are $1 - x + x^2/2 - x^3/6$, which can be rewritten as $1 - x/2 - x/2 + x^2/2 - x^3/6 \leq 1 - x/2 - x^3/6 \leq 1 - x/2$, where we are using that $x^2 \leq x$ for $x \in [0, 1]$. Our claim follows since $x^3/6$ dominates the remainder of the expansion. □

We first state this invariant that follows from Line 6.

Invariant 1. *At the beginning of iteration i, the maximum vertex degree is $n/2^{i-1}$.*

Lemma 5. *There is an absolute constant c such that, given a graph G, Algorithm 1 outputs a matching M whose size in expectation is at most c smaller than the maximum matching size in G.*

Proof. Fix iteration i of Algorithm 1, assuming that iterations $1 \dots i-1$ are already executed. For an edge $e = \{u, v\}$, let X_e be an indicator random variable which equals 1 if edge e is added to M in iteration i. Note that our definition of X_e conditions on iterations $1 \dots i-1$ already being executed. We have

$$\Pr[X_e = 1] = \Pr[e \text{ is sampled and no other edges incident to } u \text{ and } v \text{ is sampled}].$$

Due to the independence of edge sampling, this implies

$$\begin{aligned}\Pr[X_e = 1] &= \Pr[e \text{ is sampled}] \cdot \Pr[\text{no edge other than } e \text{ incident to } u \text{ and } v \text{ is sampled}] \\ &= \frac{2^i}{2n} \cdot \left(1 - \frac{2^i}{2n}\right)^{2 \cdot \frac{n}{2^{i-1}}} = \frac{2^i}{2n} \cdot \left(1 - \frac{2^i}{2n}\right)^{4 \cdot \frac{n}{2^i}} \overset{\text{by Fact 4}}{\geq} \frac{2^i}{2n} \cdot e^{-4},\end{aligned}$$

where we used Invariant 1 to upper-bound the degree of u and v by $2 \cdot n/2^{i-1}$.

To continue with the analysis, let R_i be the vertices removed on Line 6 whose degree is at least $n/2^i$, and let $\tilde{A}_i$ be the edges in iteration i added to our output matching M. Since we are considering iteration i after iterations $1 \dots i-1$ have happened, note that R_i is not a random variable. We have

$$\begin{aligned}\mathbb{E}\left[|\tilde{A}_i|\right] &= \mathbb{E}\left[\sum_e X_e\right] \\ &\geq \sum_{v \in R_i} \left(\frac{1}{2} \sum_{e=\{u,v\}} \mathbb{E}[X_e]\right) \\ &\geq \sum_{v \in R_i} \left(\frac{1}{2} \sum_{e=\{u,v\}} \frac{2^i}{2n} \cdot e^{-4}\right) \\ &= \sum_{v \in R_i} \left(\frac{1}{2} d(v) \frac{2^i}{2n} \cdot e^{-4}\right) \\ &\geq \sum_{v \in R_i} \left(\frac{1}{2} \frac{n}{2^i} \frac{2^i}{2n} \cdot e^{-4}\right) \\ &= |R_i| \frac{e^{-4}}{4}.\end{aligned}$$

The 1/2 in the analysis comes from the fact that an edge $\{u, v\}$ can have both endpoints in R_i, so X_e might be counted twice in the summation. From this sequence of inequalities, and summing

up over all possible executions of iterations $1 \dots i-1$, we have that the final matching size is in expectation at least $e^{-4}/4$ fraction of the number of vertices removed on Line 6 whose degree is above the corresponding threshold. Observe that after the algorithm is over, G has no edge. Hence, removing M and $\cup_i R_i$ from G the graph becomes empty. So, there is a maximal matching of size *at most* $\mathbb{E}[|M|] + \sum_i \mathbb{E}[|R_i|]$ in G.[1] This implies that, in expectation, $|M|$ is at most $2 \cdot \left(1 + 4e^4\right)$ times smaller than a maximum matching in G. □

Note that Lemma 5 provides an expectation guarantee instead of whp. This guarantee can be turned into a whp one if we execute $O(\log n)$ instances of Algorithm 1 simultaneously and output the largest matching among those $O(\log n)$ executions. We will not dive there.

References

[ASS+18] Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, and Peilin Zhong. Parallel graph connectivity in log diameter rounds. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 674–685. IEEE, 2018.

[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.

[GGK+18] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrović, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 129–138, 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.

[GU19] Mohsen Ghaffari and Jara Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1636–1653. SIAM, 2019.

[II86] Amos Israeli and Alon Itai. A fast and simple randomized parallel algorithm for maximal matching. *Information Processing Letters*, 22(2):77–80, 1986.

[IS86] Amos Israeli and Yossi Shiloach. An improved parallel algorithm for maximal matching. *Information Processing Letters*, 22(2):57–60, 1986.

[OR10] Krzysztof Onak and Ronitt Rubinfeld. Maintaining a large matching and a small vertex cover. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 457–464, 2010.

[1] Typically, this argument is phrased in the language of vertex cover, which is the problem dual to maximum matching. Since we are not covering duality in this class, we are only sticking to an argument based on matchings.