ECS-189A Special Topics in TCS April 18, 2023

Lecture 4

Lecturer: Slobodan Mitrović

1 Introduction

We are continuing to study graph algorithms in the context of the MPC model. We are shifting our focus to the maximal independent set problem, which is one of the most fundamental graph algorithmic questions with a long history in distributed and parallel computation. In this lecture, our focus is on the sub-linear MPC memory regime.

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 refer that $\Pr[\mathcal{E}] \geq 1 - n^{-c}$ for some constant $c \geq 1$.

2.2 Independent sets

Definition 1 (Independent set (IS))**.** *Given a graph $G = (V, E)$, a set $I \subseteq V$ is an independent set if no two vertices of I are neighbors in G.*

Definition 2 (Maximal independent set (MIS))**.** *Given a graph $G = (V, E)$, a set $I \subseteq V$ is a maximal independent set if I is an IS and if I can not be augmented by another vertex while remaining IS.*

Definition 3 (Maximum independent set)**.** *Given a graph $G = (V, E)$, a set $I \subseteq V$ is a maximum independent set if I is an MIS and if the cardinality of I is maximum over all MISes.*

Observe that each maximum IS is also a maximal IS, while the converse is not true. To see that, consider a star graph. Then, the center of the star is an MIS but it is not a maximum IS. The only maximum IS in this case is the set of all the leaves.

2.3 Probability tools

Theorem 4 (Markov's inequality)**.** *If X is a nonnegative random variable and $a > 0$, then*

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

3 Computing MIS

3.1 Centralized setting

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

Input : Graph $G = (V, E)$

$I \leftarrow \emptyset$

`/* The vertices of` V `in the loop below can be visited in any order. */`

for $u \in V$ **do**

  $I \leftarrow I \cup \{u\}$

  Remove u and all its neighbors from G.

return I

Algorithm 1: A centralized algorithm for finding an MIS.

3.2 Sub-linear memory regime of MPC

This section presents an algorithm for finding an MIS in the sub-linear MPC memory regime. Although at first it might seem that for finding an MIS it is needed to coordinate decisions of different vertices globally, it turns out that there is a highly local procedure for solving the MIS problem. One such procedure we provide is Algorithm 2, and it corresponds to work [Lub85]. That procedure itself is designed independently of the model of computation and has been applied to various computational settings. In Section 3.2.2 we discuss how to execute it in MPC.

Input : Graph $G = (V, E)$

  Space per machine S

$I \leftarrow \emptyset$

while G *is non-empty* **do**

  Each vertex $u \in V(G)$ in parallel samples an integer i_u from the range $[1, n^4]$ uniformly at random and independently of other vertices.

  For each vertex u whose value i_u is smaller than the values of its neighbors:

  - $I \leftarrow I \cup \{u\}$
  - Remove u and all its neighbors from G.

return I

Algorithm 2: An algorithm for finding an MIS. In Section 3.2.2 we outline its implementation in the sub-linear memory regime of MPC.

3.2.1 Correctness

Algorithm 2 outputs an MIS by design. That is, each time a vertex u is added to I it is ensured that no u's neighbor will be included in I. Also, the algorithm performs computation as long as there is at least one vertex that can be added to I.

3.2.2 Implementation in MPC

We are again not going to dive into MPC implementation of low-level steps; in this case, it includes computing the minimum chosen number in the neighborhood of a vertex and removing all the neighbors of a vertex added to I. We again point out that such operations are easy to implement if one has access to a segment-tree-interval-like structure. Implementation of such structures in $O(1/c)$ rounds is provided in [GSZ11, Section 4]. In particular, the following are two powerful claims shown in [GSZ11].

Theorem 5 ([GSZ11]). *Given a binary search tree $\mathcal{T}$ of size n, we can perform a multi-search of n queries in the sub-linear regime of the MPC model in $O(1/c)$ rounds and $O(n/c)$ total communication with high probability.*

Note that the theorem provides round complexity of $O(1)$ for constant c, as opposed to $O(\log n)$. Also, this theorem supports n searches *at the same time*.

Theorem 6 ([GSZ11]). *A set of n elements can be sorted in the sub-linear regime of the MPC model in $O(1/c)$ rounds and $O(n/c)$ total communication with high probability.*

3.2.3 Round complexity

Theorem 7. *Algorithm 2 runs in $O(\log n)$ rounds whp.*

To show Theorem 7, we first prove the following claim.

Lemma 8. *In a given iteration of the while-loop of Algorithm 2, values i_u across all the vertices are distinct with probability $1 - n^{-2}$ at least.*

Proof. For two fixed vertices u and v we have $\Pr[i_v = i_u] = n^{-4}$. Since there are $\binom{n}{2}$ pairs of vertices, we have that any two of them have the same i values with probability at most $\binom{n}{2} \cdot n^{-4} \leq n^{-2}$. □

Lemma 9. *Consider a single iteration of the while-loop of Algorithm 2. Assume that the values i_u across all vertices u are distinct. Let z be the number of edges at the beginning of the loop and $\tilde{z}$ the number of edges in G at the end of that loop. Then,*

$$\mathbb{E}[\tilde{z}] \leq \frac{1}{2} z.$$

Proof. In this proof, if we say (u, v) then we refer to an arc, but if we say $\{u, v\}$ then we refer to an edge.

In this analysis, perhaps the most challenging part is defining appropriate random variables. For every arc (u, v), we define a random variable $X_{u \to v}$ that means

$$X_{u \to v} = \begin{cases} 1 & \text{if } \forall w \in N(u) \cup N(v) \text{ it holds } i_u \leq i_w \\ 0 & \text{otherwise} \end{cases}$$

Observe that if $X_{u \to v} = 1$, then $e = \{u, v\}$ gets removed from the graph as: $X_{u \to v} = 1$ implies u is added to I and hence v is removed from G.

Let Y be a random variable denoting the number of edges removed from G, i.e., $Y = z - \tilde{z}$. In addition, let $Y_{u \to v}$ be an indicator random variable which equals 1 only if *the arc* (u, v) is removed from G, i.e., $2Y = \sum_{(u,v) \in E} Y_{u \to v}$. We switched from considering arcs as opposed to edges since X random variables are also defined in a directed way and defining $Y_{u \to v}$ random variables in the same way will simplify calculation.

We would like now to tie random variables X and Y. To that end, observe that for a fixed v, across all $u \in N(v)$ at most one of $X_{u \to v}$ equals 1, i.e., $\sum_{u \in N(v)} X_{u \to v} \leq 1$. This holds by definition.

On the other hand, if $X_{u \to v} = 1$, then all the edges $\{w, v\}$, for all $w \in N(v)$, get removed from G. So, this implies

$$\sum_{(u,v) \in E} X_{u \to v} \cdot d(v) \leq \sum_{(u,v) \in E} Y_{u \to v} = 2Y.$$

This further implies

$$\mathbb{E}[2Y] \geq \mathbb{E}\left[\sum_{(u,v) \in E} X_{u \to v} \cdot d(v)\right] = \sum_{(u,v) \in E} \mathbb{E}[X_{u \to v}] \cdot d(v). \tag{1}$$

Recall that $X_{u,v} = 1$ iff i_u is the minimum value among at most $d(u) + d(v)$ values, each chosen uniformly at random from the same range and with the guarantee that no two of them are the same. Hence, $\Pr[X_{u\to v} = 1] \geq \frac{1}{d(u)+d(v)}$ and $\mathbb{E}[X_{u\to v}] \geq \frac{1}{d(u)+d(v)}$. **Note:** $|N(v) \cup N(u)|$ might be smaller than $d(v) + d(u)$ as u and v might share neighbors.

From Eq. (1) we have

$$\begin{aligned}
\mathbb{E}[2Y] &\geq \sum_{(u,v)\in E} \mathbb{E}[X_{u\to v}] \cdot d(v) \\
&= \sum_{(u,v)\in E \,:\, u<v} (\mathbb{E}[X_{u\to v}] \cdot d(v) + \mathbb{E}[X_{v\to u}] \cdot d(u)) \\
&\geq \sum_{(u,v)\in E \,:\, u<v} \left(\frac{1}{d(u)+d(v)} \cdot d(v) + \frac{1}{d(u)+d(v)} \cdot d(u) \right) \\
&= \sum_{(u,v)\in E \,:\, u<v} 1 \\
&= |E(G)| \\
&= z.
\end{aligned}$$

This now implies that $\mathbb{E}[Y] \geq z/2$ and hence $\mathbb{E}[\tilde{z}] \leq z/2$. □

Application of Lemma 9 in proving Theorem 7 is left as homework. Note that Lemma 9 assumes that all the values i_v are distinct, so in the proof of Theorem 7 you probably want to use Lemma 8 as well.

References

[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.

[Lub85] Michael Luby. A simple parallel algorithm for the maximal independent set problem. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 1–10, 1985.