Principles of Distributed Computing

Lecture 04

Lecturer: Mohsen Ghaffari *Scribe:*

1 Maximal Independent Set

The Maximal Independent Set (MIS) problem is a central problem in the area of distributed algorithms for local graph problems. One partial reason for this central role is that many other problems, including graph coloring and maximal matching, reduce to MIS, as we soon see.

1.1 Definition and Reductions

Let us start with recalling the definition of MIS:

Definition 1. *Given a graph $G = (V, E)$, a set of vertices $S \subseteq V$ is called a* Maximal Independent Set (MIS) *if it satisfies the following two properties:*

(1) the set S is an independent *set meaning that no two vertices $v, u \in S$ are adjacent,*

(2) the set S is maximal *— with regard to independence — meaning that we cannot add any node $v \notin S$ to the set S, i.e., there exists a neighbor u of v such that $u \in S$.*

The *independence* condition ensures that we choose nodes that are not neighboring each other. For instance, this is what we desire when we want neighboring nodes to avoid performing their operations simultaneously, e.g., when accessing a shared resource, such as a common channel in wireless networks or the same memory location in multi-processor computers. The maximality condition then ensures that we cannot add any more nodes to the solution, without destroying the independence property. In some sense, the nodes in an MIS can be seen as centers for clustering: each node will be either itself such a center or will have a neighbor as a center, while the centers are not adjacent.

More concretely, besides the above intuitive description of applications, algorithms for MIS can be used to solve a number of other graph problems. Below, we see a simple and beautiful reduction that allows us to solve $\Delta + 1$ coloring using an MIS algorithm, without any significant overhead in the round complexity.

Lemma 2. *Given a* LOCAL *algorithm $\mathcal{A}$ that computes an MIS on any N-node graph in $T(N)$ rounds, there is a* LOCAL *algorithm $\mathcal{B}$ that computes a $\Delta+1$ coloring of any n-node graph with maximum degree Δ in $T(n(\Delta + 1))$ rounds.*

In particular, we will soon see an $O(\log n)$ round randomized algorithm for computing an MIS on n-node graphs, which by this lemma, implies an $O(\log n)$ round randomized algorithm for $(\Delta+1)$ coloring.

Proof of Lemma 2. Let G be an arbitrary n-node graph with maximum degree Δ, for which we would like to compute a $(\Delta + 1)$-coloring.

Let $H = G \times K_{\Delta+1}$ be an $n(\Delta+1)$-vertex graph generated by taking $\Delta+1$ copies of G. See Figure 1 for an example illustration. Hence each node $v \in G$ has $\Delta + 1$ copies in H, which we will refer to as v_1, v_2, $\ldots$, $v_{\Delta+1}$. Then, add additional edges between all copies of each node $v \in G$, that is, each two copy vertices v_i and v_j are connected in H.

Run the algorithm $\mathcal{A}$ on H. The resulting MIS produces a maximal independent set S. For each node $v \in G$, the color of v will be the number i such that the $v_i \in S$. Clearly, each node receives at most one color, as at most one copy v_i of $v \in G$ can be in S. However, one can see that each node $v \in G$ receives actually exactly one color. The reason is that each neighboring node $u \in G$ can have at most one of its copies in S. Node v has at most Δ neighbors u and $\Delta+1$ copies v_i. Hence, there is at least one copy v_i of v for which no adjacent copy u_i of neighboring vertices $u \in G$ is in the set S. By maximality of S, we must have $v_i \in S$. □

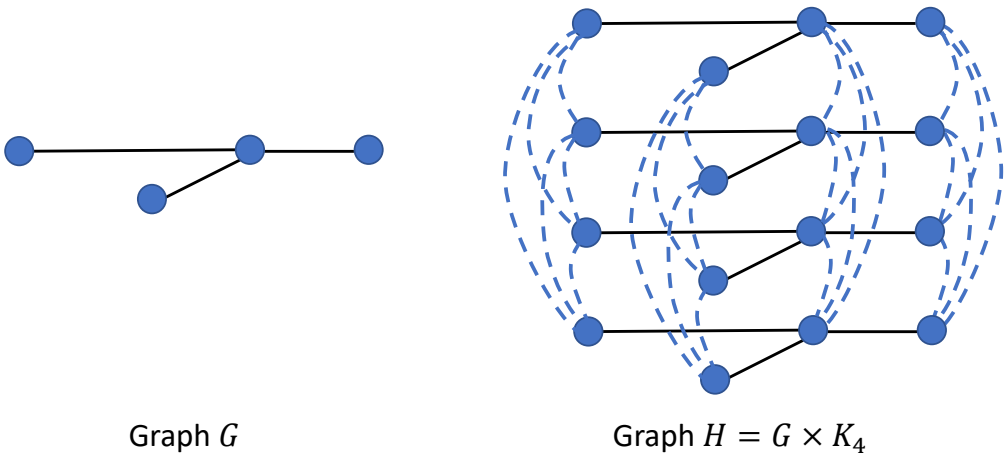

Figure 1: A simple graph G and its transformed version $H = G \times K_{\Delta+1}$.

1.2 Luby's MIS Algorithm

As we see next, there is an extremely simple and elegant randomized algorithm that computes an MIS in merely $O(\log n)$ rounds, with high probability (probability at least $1 - 1/n$). This is a celebrated result of Luby [Lub85] and Alon, Babai, and Itai [ABI86] from the 1980s, for which they received the 2016 Dijkstra award.

Luby's Algorithm: The algorithm is made of iterations, each of which has two rounds, as follows:

- In the first round, each node v picks a random real number $r_v \in [0, 1]$ and sends it to its neighbors[1]. Then, node v joins the (eventual) MIS set S if and only if node v has a strict local maxima, that is, if $r_v > r_u$ for all neighbors u of v.
- In the second round, if a node v joined the MIS, then it informs its neighbors and then, node v and all of its neighbors get removed from the problem. That is, they will not participate in the future iterations.

Analysis: It is easy to see that the algorithm always produces an independent set, and eventually, this set is maximal. The main question is, how long does it take for the algorithm to reach a maximal independent set?

Theorem 3. *Luby's Algorithm computes a maximal independent set in $O(\log n)$ rounds, with high probability.*

Proof. Let us first give a brief proof outline. Consider an arbitrary iteration i and suppose that the graph induced on the remaining vertices is G_i, which has m_i edges. In the following, and as the main ingredient of the proof, we show that the graph G_{i+1} which will remain for the next iteration has in expectation at most $m_i/2$ edges. Then, at the end, we use a repeated application of this to conclude that after $4 \log n$ iterations, the leftover graph is empty with high probability and thus the algorithm terminates.

As outlined above, the main step is to consider the graph G_i remaining for step i, and to show that

$$\mathbb{E}[m_{i+1} \,|\, G_i \text{ is any graph with } m_i \text{ edges}] \leq m_i/2.$$

In the above, m_{i+1} denotes the number of edges in the graph that remains at the end of iteration i. To prove this inequality, let us consider an edge $e = \{u, v\}$ and a neighbor w of u, as depicted in Figure 2. If w has the maximum number among $N(w)$, the set of neighbors of w in the remaining graph, then w joins the MIS and hence nodes w and u and thus also the edge $e = \{u, v\}$ get removed. In this case, we say node w killed edge $e = \{u, v\}$. The probability of w having the maximum number among its neighbors is exactly $1/(d(w)+1)$, where $d(w)$ denotes the degree of node w. But, we are interested in the

[1] One can easily see that having real numbers is unnecessary and values with $O(\log n)$-bit precision suffice.

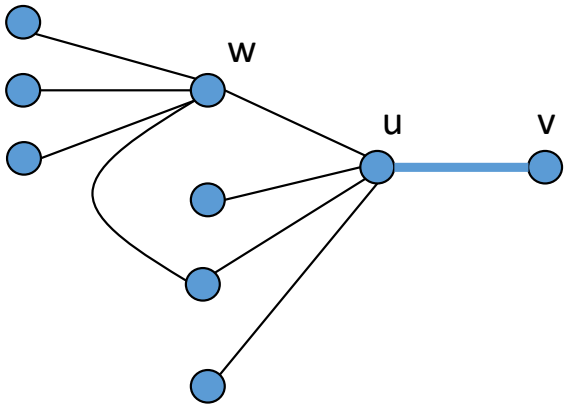

Figure 2: Node w killing edge $e = \{u, v\}$

probability of edge e being killed by any such neighbor w. We cannot easily sum over the probabilities of the neighbors w_1, w_2, ..., killing edge e, as those events are not necessarily disjoint — several of them might happen at the same time, in which case summing the probabilities would be over counting.

To circumvent this, we make a slight adjustment: we say that node w single-handedly kills $e = \{u, v\}$ (from the side of u) if r_w is the maximum random number among those of nodes in $N(w) \cup N(u)$. Notice that this limits the number of double-counting of an edge being killed to 2, meaning that at most one node w might single-handedly kill $e = \{u, v\}$ (from the side of u) and at most one node w' might single-handedly kill $e = \{u, v\}$ (from the side of v). Hence, we can lower bound the number of removed edges by bounding the number of single-handedly killed edges and dividing that by 2. More concretely, suppose we use the indicator random variable $X_{w\to e}$ which is equal to 1 if and only if node $w \in N(e)$— where $N(e) = N(u) \cup N(v)$ is the set of all nodes that are adjacent to at least one endpoint of e—single-handedly kills edge $e = \{u, v\}$. If the edge e is removed, then $(\sum_{w\in N(e)} X_{w\to e})/2$ is at least $1/2$ and at most 1. Hence, $(\sum_{w\in N(e)} X_{w\to e})/2$ is a (good) lower bound for the indicator of whether edge e is removed or not. Thus, we can lower bound the expected number of removed edges over the entire graph G_i as follows:

$$\mathbb{E}[m_i - m_{i+1}] \geq \mathbb{E}\left[\sum_{e\in E}(\sum_{w\in N(e)} X_{w\to e})/2\right] = \sum_{e\in E}\sum_{w\in N(e)} \mathbb{E}[X_{w\to e}]/2.$$

To analyze this sum, we rearrange the summations and put the one on nodes w on the outside:

$$\sum_{e\in E}\sum_{w\in N(e)} \mathbb{E}[X_{w\to e}]/2 = \sum_{w\in V}\sum_{e\in E \text{ s.t. } w\in N(e)} \mathbb{E}[X_{w\to e}]/2.$$

Now, we can focus on the contributions of each node w to this summation (grouping the edges e by their other endpoint u), as follows: Consider two neighboring nodes w and u. The probability that w has the maximum among $N(w) \cup N(u)$ is at least $\frac{1}{d(w)+d(u)}$. In that case, node w single-handedly kills $d(u)$ edges incident on u (from the side of u). Similarly, the probability that u has the maximum among $N(w) \cup N(u)$ is at least $\frac{1}{d(w)+d(u)}$, and in that case, u single-handedly kills $d(w)$ edges incident on w (from the side of w). Therefore, we can lower bound the total expected number of removed edges as follows:

$$\begin{aligned}
\mathbb{E}[m_i - m_{i+1}] &\geq \sum_{w\in V}\sum_{e\in E \text{ s.t. } w\in N(e)} \mathbb{E}[X_{w\to e}]/2 \\
&= \sum_{\{w,u\}\in E}\left(\sum_{e\in E \text{ s.t. } u\in N(e)} \mathbb{E}[X_{w\to e}]/2 + \sum_{e\in E \text{ s.t. } w\in N(e)} \mathbb{E}[X_{u\to e}]/2\right) \\
&\geq \sum_{\{w,u\}\in E_i}\left(\frac{d(u)}{d(w)+d(u)} + \frac{d(w)}{d(w)+d(u)}\right)/2 = m_i/2.
\end{aligned}$$

Now we know that we have $\mathbb{E}[m_{i+1} \mid G_i \textit{ is any graph with } m_i \textit{ edges}] \leq m_i/2$, for every iteration i. By a repeated application of this inequality over different iterations, we get that the expected number of the edges in the graph that remains after $4 \log n$ iterations is $\mathbb{E}[m_{4 \log n}] \leq m_0/2^{4 \log n} \leq 1/n^2$ edges. Hence, by Markov's inequality, the probability that the graph $G_{4 \log n}$ has at least 1 edge is at most $1/n^2$. That is, with high probability, $G_{4 \log n}$ has no edge left, and thus, the algorithm finishes in $4 \log n + 1$ iterations, with high probability. □

References

[ABI86] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms*, 7(4):567–583, 1986.

[Lub85] Michael Luby. A simple parallel algorithm for the maximal independent set problem. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 1–10, 1985.