Lecture 5

Maximal Independent Set

5.1 The Problem

Definition 5.1 (Independent Set)**.** *Given an undirected Graph $G = (V, E)$, an* independent set *is a subset of nodes $U \subseteq V$, such that no two nodes in U are adjacent. An independent set is* maximal *if no node can be added without violating independence, and* maximum *if it maximizes $|U|$.*

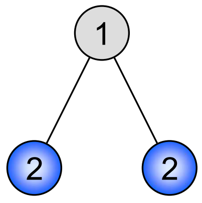

Figure 5.1: Example graph with 1) a maximal independent set and 2) a maximum independent set.

From a centralized perspective, an MIS is an utterly boring structure: Just pass over all nodes and greedily select them, i.e., pick every node without a previously selected neighbor. Voilà, you've got an MIS! However, in the distributed setting, things get more interesting. As you saw in the first exercise, finding an MIS on a list is essentially the same as finding a 3-coloring, and if we don't care about message size, being able to find an MIS on general graphs in T rounds can be used to determine a $(\Delta + 1)$-coloring in T rounds. This means that despite it being easy to *verify* that a set is an MIS locally (i.e., some node will notice if there's a problem just by looking at its immediate neighborhood), it takes $\Omega(\log^* n)$ rounds to compute an MIS!

Today, we're going to study this problem in the synchronous message passing model (without faults), for an arbitrary simple graph $G = (V, E)$. Recall that we can use synchronizers to transform any algorithm in this model into an asynchronous algorithm, so the result will make a very versatile addition to our toolbox! Like last week, we're going to allow for randomness. This can be done in the same way, by assuming that each node has an infinite (read: sufficiently large) supply of unbiased random bits.

Remarks:

- Note that an MIS can be very small compared to a maximum independent set. The most extreme discrepancy occurs in a star graph, where the two possible MIS are of size 1 and $n-1$ (cf. Figure 5.1)!

- One can apply a coloring algorithm first and then, iterating over colors, concurrently add all uncovered nodes of the current color to the independent set. Once this is complete, all nodes are covered, i.e., we have an MIS. However, this can be quite slow, as the number of colors can be large.

5.2 Fast MIS Construction

Algorithm 12 MIS construction algorithm. Of course we cannot implement an algorithm that operates with real values. We'll fix this later.

```
// each iteration of the while-loop is called a *phase*
  **while** true **do**
    choose a uniformly random value $r(v) \in [0,1]$ and send it to all neighbors
    **if** $r(v) < r(w)$ for each $r(w)$ received from some neighbor $w$ **then**
      notify neighbors that $v$ joins the independent set
      return(1) and terminate
    **end if**
    **if** a neighbor joined the independent set **then**
      return(0) and terminate
    **end if**
  **end while**
```

Lemma 5.2. *Algorithm 12 computes an MIS. It terminates with probability 1.*

Proof. We claim that at the beginning of each phase, the set of nodes that joined the set and terminated is an independent set, and the set of all nodes that terminated is the union of their (inclusive) neighborhoods; we show this by induction. Trivially, this holds initially. In each phase, it cannot happen that two adjacent non-terminated nodes enter the set. By the induction hypothesis, no active (i.e., not terminated) node has a neighbor in the independent set. Together this implies that at the end of the phase, the set of nodes that output 1 is still an independent set. As the active neighbors of joining nodes output 0 and terminate, the induction step succeeds and the claim holds true. We conclude that the algorithm computes an independent set.

To see that the independent set is maximal, observe that a node can only terminate if it enters the set or has a neighbor in the set. Thus, once all nodes have terminated, no further nodes could be added to the set without violating independence.

Finally, note that the probability that two random real numbers from $[0, 1]$ are identical is 0. By the union bound, this yields that with probability 1, in a given phase all random numbers are different. This entails that the non-terminated node with the smallest number joins the set. This implies that within finitely many phases in which the numbers differ, all nodes terminate. Using the union bound once more, it follows that the algorithm terminates with probability 1. □

Remarks:

- Simple stuff, but demonstrating how to reason about such algorithms.
- The *union bound* states that the probability of one (or more) of several (more precisely: countably many) events happening is at most the sum of their individual probabilities. It is tight if the events are disjoint.
- The union bound can't even be called a theorem. It's obvious for discrete random variables, and for continuous random variables it's just paraphrasing the fact that the total volume of the union of countably many sets is bounded by the sum of their individual volumes, a property that any measure (in particular a probability measure) must satisfy.
- We'll frequently use the union bound implicitly in the future.
- That's enough with the continuous stuff for today, now we're returning to "proper" probabilities.
- Note that the algorithm can be viewed as selecting in each phase an independent set in the subgraph induced by the still active nodes. This means that all we need to understand is a single phase of the algorithm on an arbitrary graph!

5.3 Bounding the Running Time of the Algorithm

Before we can do this, we need a (very basic) probabilistic tool: linearity of expectation.

Theorem 5.3 (Linearity of Expectation)**.** *Let X_i, $i = 1, \dots, k$ denote random variables, then*

$$\mathbb{E}\left[\sum_i X_i\right] = \sum_i \mathbb{E}\left[X_i\right].$$

Proof. It is sufficient to prove $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ for two random variables X and Y, because then the statement follows by induction. We'll do the proof for a discrete random variable; for a continuous one, simply replace the sums by integrals. We compute

$$\begin{aligned}
\mathbb{E}[X+Y] &= \sum_{(X,Y)=(x,y)} P[(X,Y)=(x,y)]\cdot(x+y)\\
&= \sum_{X=x}\sum_{Y=y} P[(X,Y)=(x,y)]\cdot x\\
&\quad + \sum_{Y=y}\sum_{X=x} P[(X,Y)=(x,y)]\cdot y\\
&= \sum_{X=x}\sum_{Y=y} P[X=x]\cdot P[Y=y \mid X=x]\cdot x\\
&\quad + \sum_{Y=y}\sum_{X=x} P[Y=y]\cdot P[X=x \mid Y=y]\cdot y\\
&= \sum_{X=x} P[X=x]\cdot x + \sum_{Y=y} P[Y=y]\cdot y\\
&= \mathbb{E}[X]+\mathbb{E}[Y],
\end{aligned}$$

where in the second last step we used that

$$\sum_{X=x} P[X=x \mid Y=y] = \sum_{Y=y} P[Y=y \mid X=x] = 1,$$

as X and Y are random variables (i.e., *something* happens with probability 1, even when conditioning on another variable's outcome). □

Denote by $N_v := \{w \in V \mid \{v,w\}\} \in E$ the neighborhood of $v \in V$ and by $\delta_v := |N_v|$ its *degree*. Now it's alluring to reason as follows. Since $\delta_v + 1$ nodes are removed "because of v" if v joins the set, by linearity of expectation we have that

$$\sum_{v\in V} P[v \text{ joins the set}]\cdot(\delta_v+1) = \sum_{v\in V}\frac{\delta_v+1}{\delta_v+1} = |V| \qquad \text{wrong!!}$$

nodes are removed in expectation. This is utter nonsense, as it would automatically mean that all nodes are eliminated in a single step (otherwise the expectation must be smaller than $|V|$), which clearly is false!

The mistake here is that we counted eliminated nodes multiple times: it's possible that *several* neighbors of a node join the set. In fact, there are graphs in which only a small fraction of the nodes gets eliminated in a phase, see Figure 5.2.

In summary, we cannot hope for many nodes being eliminated in each phase. We might be able to reason more carefully, over multiple phases, but how? It turns out that the easiest route actually goes through figuring out the expected number of *edges* that are deleted (i.e., one of their endpoints is deleted) from the graph in a given phase. Still, we need to be careful about not counting deleted edges repeatedly!

Lemma 5.4. *In a single phase, we remove at least half of the edges in expectation.*

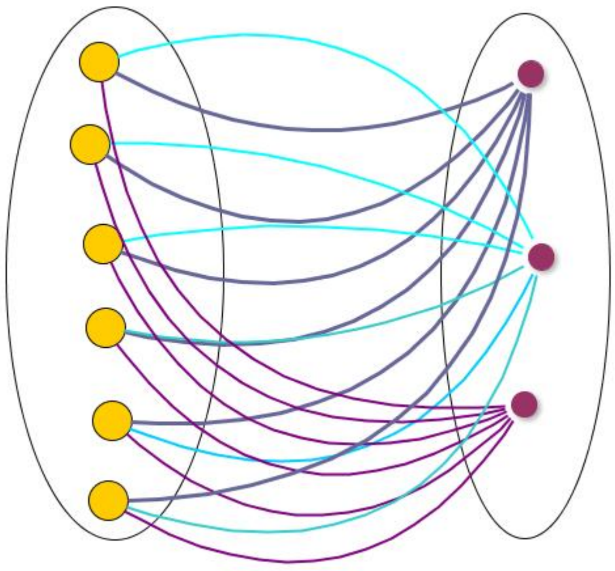

Figure 5.2: A graph where it's highly unlikely that more than a small fraction of the nodes gets deleted in the first phase. The graph is fully bipartite, but with few nodes on the right side. If $R \ll n$ nodes are on the right, the expected number of such nodes that gets selected is $R/(n-R+1) \ll 1$. This means the probability that at least one node on the right is selected is at most $R/(n-R+1) \ll 1$, too. From the left side, in expectation $(n-R)/R$ nodes are selected. If $1 \ll R \ll n$, this means that it's very likely that only nodes on the left side are selected and the selected nodes are much fewer than $n-R$; as $R \ll n$, the fact that all nodes on the right side are deleted does not affect the tally substantially.

Proof. To simplify the notation, at the start of our phase, denote the subgraph induced by the non-terminated nodes by $G = (V, E)$, i.e., ignore all terminated nodes and their incident edges. In addition, think of each undirected edge $\{v, w\}$ as being replaced by the two directed edges (v, w) and (w, v); we make sure that we count each such directed edge at most once when bounding the expected number of removed edges.

Suppose that a node v joins the MIS in this phase, i.e., $r(v) < r(w)$ for all neighbors $w \in N_v$. Consider a fixed $w \in N_v$. If we also have $r(v) < r(x)$ for all $x \in N_w \setminus \{v\}$, we call this event $(v \to w)$. The probability of event $(v \to w)$ is at least $1/(\delta_v + \delta_w)$, since (i) $\delta_v + \delta_w$ is the maximum number of nodes adjacent to v or w (or both) and (ii) ordering by $r(\cdot)$ induces a uniformly random permutation on V. As v joins the MIS, in particular all (directed) edges (w, x) with $x \in N_w$ are removed, which we attribute to event $(v \to w)$; there are δ_w such edges.

We now count the removed (directed) edges. Whether we remove the edges adjacent to w because of event $(v \to w)$ is a random variable $X_{(v\to w)}$. If event $(v \to w)$ occurs, $X_{(v\to w)}$ has the value δ_w, if not it has the value 0. For each undirected edge $\{v, w\}$, we have two such variables, $X_{(v\to w)}$ and $X_{(w\to v)}$. Due to Theorem 5.3, the expected value of the sum X of all these random variables is at least

$$\begin{aligned}\mathbb{E}\,[X] &= \sum_{\{v,w\}\in E} \mathbb{E}[X_{(v\to w)}] + \mathbb{E}[X_{(w\to v)}] \\ &= \sum_{\{v,w\}\in E} P\,[(v\to w)]\cdot\delta_w + P\,[(w\to v)]\cdot\delta_v \\ &\geq \sum_{\{v,w\}\in E} \frac{\delta_w}{\delta_v+\delta_w} + \frac{\delta_v}{\delta_w+\delta_v} \\ &= \sum_{\{v,w\}\in E} 1 = |E|.\end{aligned}$$

In other words, in expectation $|E|$ directed edges are removed in a single phase! Note that we did not count any edge removals twice: we attribute the directed edge (v, w) being removed to an event $(u \to v)$, which inhibits a concurrent event $(u' \to v)$, because then $r(u) < r(u')$ for all $u' \in N_v$. We may have counted an undirected edge at most twice (once in each direction). So, in expectation at least half of the undirected edges are removed. □

This tells us that in expectation we make good progress. But how do we derive some explicit bound on the time until *all* edges are eliminated (and thus all nodes are, too) from this? We need another very basic tool, Markov's inequality.

Theorem 5.5 (Markov's Inequality)**.** *Let X be a non-negative random variable (in fact, $P[X \geq 0] = 1$ suffices). Then, for any $K > 1$,*

$$P[X \geq K\mathbb{E}[X]] \leq \frac{1}{K}.$$

Proof. We'll prove the statement for a discrete random variable with values from $\mathbb{N}_0$; it's straightforward to generalize to arbitrary discrete variables and continuous variables. If $P[X = 0] = 1$, the statement is trivial; hence, assume w.l.o.g. that $P[X = 0] < 1$, implying that $\mathbb{E}[X] > 0$. We bound

$$\begin{aligned}\mathbb{E}[X] &= \sum_{i=0}^{\infty} P[X=i]\cdot i \\ &\geq \sum_{i=\lceil K\mathbb{E}[X]\rceil}^{\infty} P[X=i]\cdot i \\ &\geq K\mathbb{E}[X] \sum_{i=\lceil K\mathbb{E}[X]\rceil}^{\infty} P[X=i] \\ &= K\mathbb{E}[X]P[X\geq K\mathbb{E}[X]].\end{aligned}$$

Dividing by $K\mathbb{E}[X] > 0$ yields the statement of the theorem. □

Using Markov's bound, we can now infer that the bound on the expected number of removed edges in a given phase also implies that a constant fraction of edges must be removed with constant probability.

Corollary 5.6. *In a single phase, we remove at least a third of the edges with probability at least* $1/4$.

Proof. Fix a phase. For simplicity, denote the remaining graph by $G = (V, E)$. Denote by X the random variable counting the number of removed edges and by $\bar{X}$ the random variable counting the number of surviving edges, i.e., $|E| = X + \bar{X}$. By Lemma 5.4,

$$\mathbb{E}[\bar{X}] = |E| - \mathbb{E}[X] \leq \frac{|E|}{2}.$$

By Markov's inequality (Theorem 5.5, for $K = 4/3$),

$$P\left[X \leq \frac{|E|}{3}\right] = P\left[\bar{X} \geq \frac{2|E|}{3}\right] \leq P\left[\bar{X} \geq \frac{4\mathbb{E}[\bar{X}]}{3}\right] \leq \frac{3}{4}.$$

Hence,

$$P\left[X > \frac{|E|}{3}\right] = 1 - P\left[X \leq \frac{|E|}{3}\right] \geq \frac{1}{4}. \qquad \square$$

In other words, we translated the bound on the expected number of eliminated edges into a bound on the probability that a constant fraction of the remaining edges gets eliminated. This now can happen only $\mathcal{O}(\log n)$ times before we run out of edges!

Theorem 5.7. *Algorithm 12 computes an MIS in* $\mathcal{O}(\log n)$ *rounds in expectation.*

Proof. Irrespectively of how the remaining graph looks like, Corollary 5.6 tells us that with probability at least $1/4$, at least a fraction of $1/3$ of the remaining edges is removed in each phase. Let's call a phase in which this happens *good*. After

$$\log_{3/2} |E| < \frac{\log n^2}{\log(3/2)} < 4 \log n$$

good phases, we can be sure that there is at most a single remaining edge; in the following phase the algorithm will terminate. As each phase requires 2 rounds, it thus suffices to show that the expected number of phases until $4 \log n$ good phases occurred is in $\mathcal{O}(\log n)$.

Computing this expectation precisely is tedious. However, consider, say, $40 \log n$ phases. The expected number of good phases in $40 \log n$ phases is, by Corollary 5.6, at least $10 \log n$. Hence, the expected number of phases that are not good is at most $30 \log n$, and by Markov's inequality the probability that more than $36 \log n$ phases are bad is at most $30/36 = 5/6$.

In fact, this holds for *any* set of $40 \log n$ phases. So we can upper bound the expected number of phases by considering the random experiment in which we flip for each $40 \log n$ phases a coin showing heads with probability $1/6$, determine the expected number of trials until we see heads for the first time, and multiply by $40 \log n = 80 \log n$. That is, the expected number of phases until at least

$4 \log n$ good phases occurred is bounded by

$$
\begin{aligned}
40 \log n \sum_{i=0}^{\infty} \left(\frac{5}{6}\right)^i \cdot \frac{1}{6} \cdot (i+1) &= \frac{20 \log n}{3} \left(\sum_{i=0}^{\infty} i \left(\frac{5}{6}\right)^i + \sum_{i=0}^{\infty} \left(\frac{5}{6}\right)^i \right) \\
&= \frac{20 \log n}{3} \left(\frac{5}{6} \cdot 6^2 + 6 \right) \\
&= 240 \log n.
\end{aligned}
$$

We conclude that the expected running time of the algorithm is bounded by $2 \cdot 240 \log n \in \mathcal{O}(\log n)$ rounds. □

Remarks:

- This analysis is somewhat heavy-handed with respect to constants. I'd guess that the probability to eliminate at least half of the edges is at least $1/2$, and that the algorithm terminates in expected time smaller than $4 \log n$. Proving this, however, might be quite difficult!

- The analysis is also tight up to constants. If one samples uniformly at random from the family of graphs of uniform degree Δ for, say, $\Delta = n^{1/100}$, there will be extremely few short cycles. In such a graph, it's easy to show that the degree of surviving nodes falls almost exactly by a constant factor in each phase (until degrees become fairly small).

- No algorithm that has expected running time $o(\log n)$ on all graphs is known to date. Plenty of research has been done and better bounds for many restricted graphs classes (like, e.g., trees) are known, but nothing that *always* works.

- The strongest known lower bound on the number of rounds to compute an MIS is $\Omega(\sqrt{\log n / \log \log n})$ or $\Omega(\log \Delta / \log \log \Delta)$ (depending on whether one parameterizes by the number of nodes or the maximum degree). It holds also for randomized algorithms; closing this gap is one of the major open problems in distributed computing.

- Embarrassingly, the best known upper bound on the time to compute an MIS deterministically is $2^{\mathcal{O}(\sqrt{\log n})}$, i.e., there's an exponential gap to the lower bound! Even worse, the respective algorithm may end up collecting the topology of the entire graph locally, i.e., using huge messages!

5.4 Exploiting Concentration

We have bounded the expected running time of the algorithm, but often that is not too useful. If we need to know when we can start the next task and this is not controlled by termination of the algorithm (i.e., we need to be ready for something), we need to know that all nodes are essentially *certain* to be finished! Also, there are quite a few situations in which we don't have such a trivial-to-check local termination condition. Yet, we would like to have an easy way of deciding when to stop!

Definition 5.8 (With high probability (w.h.p.)). *We say that an event occurs* with high probability (w.h.p.), *if it does so with probability at least* $1 - 1/n^c$ *for any (fixed) choice of* $c \geq 1$. *Here* c *may affect the constants in the* $\mathcal{O}$*-notation because it is considered a "tunable constant" and usually kept small.*

This weird definition asks for some explanation. The reason why the probability bound depends on n is that it makes applying the union bound *extremely* convenient. Think for instance that you showed that each node terminates w.h.p. within $\mathcal{O}(\log n)$ rounds (the constant c being absorbed by the $\mathcal{O}$-notation). Then you pick $c' := c + 1$, apply the union bound over all nodes and conclude that *everyone* terminates with probability at least $1 - 1/n^{c'} \cdot n = 1 - 1/n^c$, i.e., w.h.p.!

The cool thing here is that this works for any polynomial number of events, as c is "tunable." For instance, if something holds for each edge w.h.p., it holds for all edges w.h.p. Even better, we can condition on events that happen w.h.p. and basically pretend that they occur deterministically. The probability that they do not happen is so small that any dependencies that might exist have negligible effects!

After this sales pitch, the obvious question is where we can get such strong probability bounds from. Chernoff's bound comes to the rescue! It holds for sums of *independent* random variables.

Definition 5.9 (Independence of random variables). *A set of random variables* $X_1, \ldots, X_k$ *is* independent, *if for all* i *and* $(x_1, \ldots, x_k)$ *it holds that*

$$\begin{aligned} & P[X_i = x_i] \\ = \; & P[X_i = x_i \,|\, (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k) = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)]. \end{aligned}$$

In words, the probability that $X_i = x_i$ *is independent of what happens to the other variables.*

Theorem 5.10 (Chernoff's Bound). *Let* $X = \sum_{i=1}^{k} X_i$ *be the sum of* k *independent Bernoulli (i.e., 0-1) variables. Then, for* $0 < \delta \leq 1$,

$$\begin{aligned} P\big[X \geq (1+\delta)\mathbb{E}[X]\big] &\leq e^{-\delta^2 \mathbb{E}[X]/3} \\ P\big[X \leq (1-\delta)\mathbb{E}[X]\big] &\leq e^{-\delta^2 \mathbb{E}[X]/2}. \end{aligned}$$

Let's see Chernoff's bound in action.

Corollary 5.11. *Algorithm 12 terminates w.h.p. in* $\mathcal{O}(\log n)$ *rounds.*

Proof. By Corollary 5.6, with probability at least 1/4, a third of the remaining edges is removed from the graph in a given phase. This bound holds *independently* of anything that happened before! We reason as in the proof of Theorem 5.7, but bound the number of phases until $4 \log n$ phases are good using Chernoff's bound.

For $c \geq 1$, the probability that we need more than $k := 32\lceil c \log n \rceil$ phases for $4 \log n$ of them to be good is bounded by the probability that the sum $X := \sum_{i=1}^{k} X_i$ of independent Bernoulli variables X_i with $P[X_i = 1] = 1/4$ is smaller than $4 \log n$. We have that $\mathbb{E}[X] = 8\lceil c \log n \rceil$. Hence, by Chernoff's bound for $\delta = 1/2$,

$$P[X < 4 \log n] \leq P\left[X < \frac{\mathbb{E}[X]}{2}\right] \leq e^{-\mathbb{E}[X]/8} \leq e^{-c \log n} < n^{-c}.$$

Thus, the probability that the algorithm does *not* terminate within $2(k+1) \in \mathcal{O}(\log n)$ rounds is at most $1/n^c$. Since $c \geq 1$ was arbitrary, this proves the claim. □

Remarks:

- Chernoff's bound is *exponentially* stronger than Markov's bound. However, it requires independence!
- Surprisingly, in fact Chernoff's bound is just a very clever application of Markov's bound (see exercises)!
- **Careful:** Pairwise independence of random variables $X_1, \dots, X_k$ does *not* imply that they are independent! A counterexample are two independent Bernoulli variables X_1 and X_2 with $P[X_1 = 1] = P[X_2 = 1] = 1/2$ (i.e., unbiased coin flips), and $X_3 := X_1$ XOR X_2. If one fixes *either* X_1 or X_2, X_3 is determined by the respective other (independent) coin flip, and hence remains independent. If one fixes both X_1 and X_2, X_3 is already determined!

5.5 Bit Complexity of the Algorithm

We still need to fix the issue of the algorithm using random real numbers. We don't want to communicate an infinite number of bits! To resolve this, recall the relation between uniformly random real numbers from $[0,1]$ and infinite strings of unbiased random bits: The latter is a binary encoding of the former. Next, note that in order to decide whether $r_v > r_w$ or $r_v < r_w$ for neighbors $v, w \in V$ ($r_v = r_w$ has probability 0 and thus does not concern us here), it is sufficient to communicate only the leading bits of both strings, until the first differing bit is found! What is the expected number of sent bits? Easy:

$$\sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i \cdot i = 2.$$

Inconveniently, the number of bits that need to be exchanged between each pair of nodes among three nodes that form a triangle are not independent. This is a slightly more elaborate example showing that pairwise independence does not imply "collective" independence.

Our way out is using that the probability to exchange many bits is so small that the dependence does not matter, as it also must become very small. The probability that a pair of nodes needs to exchange more than $1 + (c+3)\log n$ bits in a given phase is

$$2^{-(c+3)\log n} = n^{-(c+3)}.$$

By the union bound, the probability that *any* pair of nodes needs to exchange more than this many bits is thus no larger than $n^{-(c+1)}$ (there are fewer than n^2 edges). Applying the union bound over all rounds (fewer than n, as at least one node is eliminated per iteration), we can conclude it suffices for each node to broadcast $\mathcal{O}(\log n)$ bits per round. But we can do better!

Corollary 5.12. *If n is known, Algorithm 12 can be modified so that it sends 1-bit messages and terminates within $\mathcal{O}(\log n)$ rounds w.h.p.*

Proof. Before running Algorithm 12, each pair of neighbors v, w determines for each of the w.h.p. $\mathcal{O}(\log n)$ phases of the algorithm whether $r_v < r_w$ or vice versa (we don't know how many phases are needed, but we'll just use the $\mathcal{O}(\log n)$ upper bound). This is done by v sending the leading bits of r_v (and receiving the ones from w) until the first difference is noted. Then the nodes move on to exchanging the leading bits for the next phase, and so on. The total number of bits that needs to be exchanged is in expectation $\mathcal{O}(\log n)$ (2 times the number of phases). By Chernoff's bound, for any fixed $K \geq 2$ the probability that the sum X of $K \log n$ independent unbiased coin flips is smaller than $K \log n/2$ is in $n^{-\Omega(K)}$. Note that the probability that the exchange for a given phase ends is indeed $1/2$ and independent of earlier such events. Choosing K suitably, we conclude that w.h.p. v and w exchange at most $\mathcal{O}(\log n)$ bits in total.

Applying the union bound, we conclude that w.h.p. *no* pair of nodes exchanges more than $\mathcal{O}(\log n)$ bits. Knowing n, we can determine the respective number, run the exchange algorithm for the respective number of rounds, and then run Algorithm 12 without having to communicate the values r_v, $v \in V$, at all. $\square$

Remarks:

- Using this trick means that nodes need to communicate *different* bits to different neighbors.
- It's possible to get rid of needing to know n.[1]
- It's not hard to see that at least some nodes will have to send $\Omega(\log n)$ bits across an edge: in a graph consisting of $n/2$ pairs of nodes connected by an edge, the expected number of edges for which $\log(n/2)$ random bits are required to break symmetry is 1.

5.6 Applications

We know from the first exercise that we can use an MIS algorithm to compute a $(\Delta + 1)$-coloring.

Corollary 5.13. *On any graph G, a $(\Delta + 1)$-coloring can be computed in $\mathcal{O}(\log n)$ rounds w.h.p.*

MIS are not only interesting for graph coloring. As we will see in the exercises, they can also be very helpful in finding small *dominating sets*[2] in some graph families. Moreover, an MIS algorithm can be used to compute a *maximal matching.*

[1]This is done by alternating between bit exchange rounds and rounds of Algorithm 12. However, then some nodes may not yet have succeeded in comparing the random values for the "current" phase of Algorithm 12. This can be understood as the exchange of the random numbers happening asynchronously (nodes do not know when they will be ready to compare their value to all neighbors), and thus can be resolved by running the α-synchronizer version of Algorithm 12. Now one needs to avoid that the communication of the synchronizer dominates the complexity. Solution: Exploiting synchrony, we can count "synchronizer rounds" by locally counting how many rounds neighbors completed and just communicating "I advance to the next round" (plus the respective message content) and otherwise remaining silent.

[2]A dominating set $D \subseteq V$ satisfies that each $v \in V \setminus D$ has a neighbor in D.

Definition 5.14 (Matching)**.** *Given a graph $G = (V, E)$, a* matching *is a subset of edges $M \subseteq E$, such that no two edges in M are adjacent (i.e., where no node is incident to 2 edges in the matching). A matching is* maximal *if no edge can be added without violating the above constraint. A matching of maximum cardinality is called* maximum. *A matching is called* perfect *if each node is adjacent to an edge in the matching.*

Corollary 5.15. *On any graph G, a maximal matching can be computed in $\mathcal{O}(\log n)$ rounds w.h.p.*

Proof. For a simple graph $G = (V, E)$, the line graph is (E, L), where $\{e, e'\} \in L$ if and only if $e \cap e' \neq \emptyset$. In words, edges become nodes, and a pair of "new" nodes is connected by an edge exactly if the corresponding edges in the original graph share an endpoint. We simulate an MIS algorithm on the line graph. It's straightforward to show that this can be done at an overhead of factor 2 in time complexity. □

Another use of MIS is in constructing small *vertex covers.*

Definition 5.16 (Vertex cover)**.** *A vertex cover $C \subseteq V$ is a set of nodes covering all edges, i.e., for all $e \in E$, $e \cap C \neq \emptyset$. It is minimal if no node can be removed without violating coverage. It is minimum, if it is of minimum size.*

Corollary 5.17. *On any graph G, a vertex cover that is at most a factor 2 larger than a minimum vertex cover can be computed in $\mathcal{O}(\log n)$ rounds w.h.p.*

Proof. Compute a maximal matching using Corollary 5.15 and output the endpoints of all its edges. Clearly, this is a vertex cover, as an uncovered edge could be added to the matching. Moreover, any vertex cover must contain for each matching edge at least one of its endpoints. Hence the cover is at most a factor of 2 larger than the optimum. □

Finally, we can also use a maximal matching to approximate a maximum matching.

Corollary 5.18. *On any graph G, a matching that is at most a factor 2 smaller than a maximum matching can be computed in $\mathcal{O}(\log n)$ rounds w.h.p.*

Proof. Compute and output a maximal matching using Corollary 5.15. Now consider a maximum matching. A minimum vertex cover must be as least as large as this matching (as no node can cover two matching edges). As the endpoints of the edges of a maximal matching form a vertex cover, they must be at least as many as those of a minimum vertex cover and thus the size of a maximum matching. The number of edges in the maximal matching is half this number. □

Remarks:

- Given the important role of all these graph structures in "traditional" algorithms, it comes hardly as a surprise that being able to construct an MIS fast in distributed systems comes in very handy on many occasions!

What to take Home

- Finding an MIS is a trivial task for centralized algorithms. It's a pure *symmetry breaking* problem.
- An MIS is typically not useful by itself, but MIS algorithms are very helpful in the construction of many other basic graph structures.
- Your basic toolbox for analyzing randomized algorithms:
  - linearity of expectation
  - Markov's bound
  - probabilistic independence (and its convenient properties)
  - probabilistic domination (see below)
  - Chernoff's bound
  - union bound
- Probabilistic domination simply means that you replace a random variable by some other variable "dominating" it. We did this with the probability that a constant fraction of the edges is removed: whether this happens might depend on the graph, so we could not apply Chernoff to collections of such variables. However, we knew that *no matter what*, the respective probability is constantly bounded *independently* of the graph, so we used independent Bernoulli variables to indicate whether a phase was good or not, possibly treating some good phases as "bad," to be on the safe side.
- One particularly useful toolchain is probabilistic domination $\to$ Chernoff $\to$ union bound. It's not uncommon to use the other tools just to feed this chain, like we did today, too!
- That's almost everything you need to analyze the majority of randomized distributed algorithms out there. The difficulty typically lies in how to apply these tools properly, not in finding new ones![3]

Bibliographic Notes

In the 80s, several groups of researchers came up with more or less the same ideas: Luby [Lub86], Alon, Babai, and Itai [ABI86], and Israeli and Itai [II86]. Essentially, all of these works imply fast randomized distributed algorithms for computing an MIS. The new MIS variant (with a simpler analysis) presented here is by Métivier, Robson, Saheb-Djahromi, and Zemmari [MRSDZ11]. With some adaptations, also the algorithms [Lub86, MRSDZ11] only need to transmit a total of $\mathcal{O}(\log n)$ bits per node, which is asymptotically optimal, even on unoriented trees [KSOS06].

The state-of-the-art running time on general graphs, due to Ghaffari, is $\mathcal{O}(\log \Delta + 2^{\mathcal{O}(\sqrt{\log \log n})})$. However, when the maximum degree $\Delta \in n^{\Omega(1)}$, this

[3]Admittedly, this might be a bad case of "if all you have is a hammer, everything looks like a nail." In a course on probabilistic algorithms I took I drove the TA mad by solving most of the exercises utilizing Chernoff's bound (or trying to), even though the questions were handcrafted to strongly suggest otherwise.

is not asymptotically faster than the presented algorithm. Also, while this nearly matches the lower bound of $\Omega(\min\{\log \Delta / \log\log \Delta, \sqrt{\log n / \log\log n}\})$ [KMW10] with respect to Δ, it does not yield further insight for the range $\Delta \gg 2^{\sqrt{\log n}}$. Deterministic MIS algorithms are embarrassingly far from the lower bounds: the best known upper bound is $2^{\mathcal{O}(\sqrt{\log n})}$ [PS96].

In growth-bounded graphs,[4] an MIS is a constant-factor approximation to a minimum dominating set. In this graph family, an MIS (and thus small dominating set) can be computed in $\mathcal{O}(\log^* n)$ rounds [SW08]; the respective algorithm leverages the Cole-Vishkin technique. On graphs that can be decomposed into a constant number of forests, a constant-factor approximation to a minimum dominating set can be computed with the help of an MIS algorithm, too [LPW13] (see exercises).

Wide parts of the script for this lecture are based on material kindly provided by Roger Wattenhofer. Thanks!

Bibliography

[ABI86] Noga Alon, László Babai, and Alon Itai. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *J. Algorithms*, 7(4):567–583, 1986.

[II86] Amos Israeli and Alon Itai. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Inf. Process. Lett.*, 22(2):77–80, 1986.

[KMW10] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local Computation: Lower and Upper Bounds. *CoRR*, abs/1011.5470, 2010.

[KSOS06] Kishore Kothapalli, Christian Scheideler, Melih Onus, and Christian Schindelhauer. Distributed coloring in $O(\sqrt{\log n})$ Bit Rounds. In *20th international conference on Parallel and Distributed Processing (IPDPS)*, 2006.

[LPW13] Christoph Lenzen, Yvonne-Anne Pignolet, and Roger Wattenhofer. Distributed Minimum Dominating Set Approximations in Restricted Families of Graphs. *Distributed Computing*, 26(2):119–137, 2013.

[Lub86] Michael Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.

[MRSDZ11] Yves Métivier, John Michael Robson, Nasser Saheb-Djahromi, and Akka Zemmari. An optimal bit complexity randomized distributed MIS algorithm. *Distributed Computing*, 23(5-6):331–340, 2011.

[PS96] Alessandro Panconesi and Aravind Srinivasan. On the Complexity of Distributed Network Decomposition. *J. Algorithms*, 20(2):356–374, 1996.

[4]A graph is growth-bounded if the maximal number of independent nodes in r-hop neighborhoods is bounded as a function of r, independently of n or Δ.

[SW08] Johannes Schneider and Roger Wattenhofer. A Log-Star Distributed Maximal Independent Set Algorithm for Growth-Bounded Graphs. In *27th ACM Symposium on Principles of Distributed Computing (PODC), Toronto, Canada*, August 2008.