Chapter 5

Maximal Independent Set

In this chapter we present a highlight of this course, a fast maximal independent set (MIS) algorithm. The algorithm is the first randomized algorithm that we study in this class. In distributed computing, randomization is a powerful and therefore omnipresent concept, as it allows for relatively simple yet efficient algorithms. As such the studied algorithm is archetypal.

A MIS is a basic building block in distributed computing, some other problems pretty much follow directly from the MIS problem. At the end of this chapter, we will give two examples: matching and vertex coloring (see Chapter 1).

5.1 MIS

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. An independent set of maximum cardinality is called maximum.

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

Algorithm 23 Slow MIS

Require: Node IDs

Every node \( v \) executes the following code:

1. if all neighbors of \( v \) with larger identifiers have decided not to join the MIS then
2. \( v \) decides to join the MIS
3. end if

Remarks:

- Not surprisingly the slow algorithm is not better than the sequential algorithm in the worst case, because there might be one single point of activity at any time. Formally:

Theorem 5.2 (Analysis of Algorithm 23). Algorithm 23 features a time complexity of \( O(n) \) and a message complexity of \( O(m) \).

Remarks:

- This is not very exciting.

There is a relation between independent sets and node coloring (Chapter 1), since each color class is an independent set, however, not necessarily a MIS. Still, starting with a coloring, one can easily derive a MIS algorithm: We first choose all nodes of the first color. Then, for each additional color we add "in parallel" (without conflict) as many nodes as possible. Thus the following corollary holds:

Corollary 5.3. Given a coloring algorithm that needs \( C \) colors and runs in time \( T \), we can construct a MIS in time \( C + T \).

Remarks:

- Using Theorem 1.17 and Corollary 5.3 we get a distributed deterministic MIS algorithm for trees (and for bounded degree graphs) with time complexity \( O(\log^* n) \).
5.2 Original Fast MIS

Algorithm 24 Fast MIS

A single phase is as follows:
1) Each node $v$ marks itself with probability $\frac{1}{2d(v)}$, where $d(v)$ is the current degree of $v$.
2) If no higher degree neighbor of $v$ is also marked, node $v$ joins the MIS. If a higher degree neighbor of $v$ is marked, node $v$ unmarks itself again. (If the neighbors have the same degree, ties are broken arbitrarily, e.g., by identifier).
3) Delete all nodes that joined the MIS and their neighbors, as they cannot join the MIS anymore.

Remarks:
- Correctness in the sense that the algorithm produces an independent set is relatively simple. Steps 1 and 2 make sure that if a node $v$ joins the MIS, then $v$'s neighbors do not join the MIS at the same time. Step 3 makes sure that $v$'s neighbors will never join the MIS.
- Likewise the algorithm eventually produces a MIS, because the node with the highest degree will mark itself at some point in Step 1.
- So the only remaining question is how fast the algorithm terminates. To understand this, we need to dig a bit deeper.

Lemma 5.4 (Joining MIS). A node $v$ joins the MIS in Step 2 with probability $P \geq \frac{1}{2d(v)}$.

Proof: Let $M$ be the set of marked nodes in Step 1. Let $H(v)$ be the set of neighbors of $v$ with higher degree, or same degree and higher identifier. Using independence of the random choices of $v$ and nodes in $H(v)$ in Step 1 we get

$$P[v \notin MIS | v \in M] = P[w \notin H(v), w \in M | v \in M] = P[w \notin H(v), w \in M]$$

$$\leq \sum_{w \in H(v)} P[w \in M] = \sum_{w \in H(v)} \frac{1}{2d(w)}$$

$$\leq \sum_{w \in d(v)} \frac{1}{2d(v)} \leq \frac{1}{2d(v)}$$

Then

$$P[v \in MIS] = P[v \in MIS | v \in M] \cdot P[v \in M] \geq \frac{1}{2} \cdot \frac{1}{2d(v)}$$
5.2. ORIGINAL FAST MIS

Remarks:
- We would be almost finished if we could prove that many nodes are good in each phase. Unfortunately this is not the case: In a star-graph, for instance, only a single node is good! We need to find a workaround.

Lemma 5.6 (Good Edges). An edge \( e = (u, v) \) is called bad if both \( u \) and \( v \) are bad, else the edge is called good. The following holds: At any time at least half of the edges are good.

Proof: For the proof we construct a directed auxiliary graph: Direct each edge towards the higher degree node (if both nodes have the same degree direct it towards the higher identifier). Now we need a little helper lemma before we can continue with the proof.

Lemma 5.7. A bad node has outdegree (number of edges pointing away from bad node) at least twice its indegree (number of edges pointing towards bad node).

Proof: For the sake of contradiction, assume that a bad node \( v \) has outdegree at least twice its indegree. In other words, at least one third of the \( d(v) \) is good, a contradiction.

Continuing the proof of Lemma 5.6: According to Lemma 5.7 the number of edges directed into bad nodes is at most half the number of edges directed out of bad nodes. Thus, the number of edges directed into bad nodes is at most half the number of edges. Thus, at least half of the edges are directed into good nodes. Since these edges are not bad, they must be good.

Theorem 5.8 (Analysis of Algorithm 24). Algorithm 24 terminates in expected time \( O(\log n) \).

Proof. With Lemma 5.5 a good node (and therefore a good edge!) will be deleted with constant probability. Since at least half of the edges are good (Lemma 5.6) a constant fraction of edges will be deleted in each phase.

More formally: With Lemmas 5.5 and 5.6 we know that at least half of the edges will be removed with probability at least \( 1/36 \). Let \( R \) be the number of edges to be removed. Using linearity of expectation (cf. Theorem 5.9) we know that \( \mathbb{E}[R] \geq m/72 \), \( m \) being the total number of edges at the start of a phase. Now let \( p = P[R \leq \mathbb{E}[R]/2] \). Bounding the expectation yields

\[
\mathbb{E}[R] = \sum_i P[R = i] \cdot i \leq P[R \leq \mathbb{E}[R]/2] \cdot \mathbb{E}[R]/2 + P[R > \mathbb{E}[R]/2] \cdot m
\]

\[
= p \cdot \mathbb{E}[R]/2 + (1 - p) \cdot m
\]

Solving for \( p \) we get

\[
p \leq \frac{m - \mathbb{E}[R]}{m - \mathbb{E}[R]/2} \leq \frac{m - \mathbb{E}[R]/2}{m} \leq 1 - 1/144.
\]

In other words, with probability at least \( 1/144 \) at least \( m/144 \) edges are removed in a phase. After expected \( O(\log m) \) phases all edges are deleted. Since \( m \leq n^2 \) and thus \( O(\log m) = O(\log n) \) the Theorem follows.

5.3. Fast MIS v2

Algorithm 25 Fast MIS 2

The algorithm operates in synchronous rounds, grouped into phases.

A single phase is as follows:

1. Each node \( v \) chooses a random value \( r(v) \in [0,1] \) and sends it to its neighbors.
2. \( v \)'s neighbors do not join the MIS at the same time. Step 3 makes sure that whose neighbors do not join the MIS at the same time.
3. If \( v \) or a neighbor of \( v \) entered the MIS, \( v \) terminates (and all edges adjacent to \( v \) are removed from the graph), otherwise \( v \) enters the next phase.

Remarks:
- Correctness in the sense that the algorithm produces an independent set is simple: Steps 1 and 2 make sure that if a node \( v \) joins the MIS, then \( v \)'s neighbors do not join the MIS at the same time. Step 3 makes sure that \( v \)'s neighbors will never join the MIS.
- Likewise, the algorithm eventually produces a MIS, because the node with the globally smallest value will always join the MIS at the same time.
- So the only remaining question is how fast the algorithm terminates. To understand this, we need to dig a bit deeper.
- Our proof will rest on a simple, yet powerful observation about expected values of random variables that may not be independent.

Theorem 5.9 (Linearity of Expectation). Let \( X_i \), \( i = 1, \ldots, k \) denote random variables, then

\[
\mathbb{E} \left[ \sum_{i=1}^{k} X_i \right] = \sum_{i=1}^{k} \mathbb{E}[X_i].
\]

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. Since

\[
P(X = x | Y = y) = P(X = x | Y = y | X = x)
\]

and

\[
P(Y = y | X = x) = P(Y = y | X = x)
\]

then

\[
P(X = x \land Y = y) = P(X = x | Y = y) \cdot P(Y = y).
\]

Using the above statements, we get

\[
\begin{align*}
\mathbb{E}[X + Y] &= \sum_{x, y} (x + y) P(X = x \land Y = y) \\
&= \sum_{x, y} x P(X = x | Y = y) \cdot P(Y = y) + \sum_{x, y} y P(X = x | Y = y) \cdot P(Y = y) \\
&= \sum_{x} x \cdot \sum_{y} P(X = x | Y = y) \cdot P(Y = y) + \sum_{y} y \cdot \sum_{x} P(X = x | Y = y) \cdot P(Y = y) \\
&= \sum_{x} x \cdot P(X = x) + \sum_{y} y \cdot P(Y = y) \\
&= \mathbb{E}[X] + \mathbb{E}[Y].
\end{align*}
\]
5.3. Fast MIS V2

In other words, in expectation, each directed edge is removed with probability more than $2/3$, while the algorithm removes at least a quarter of all edges in expectation. This enables us to follow a bound on the expected running time of Algorithm 25 quite easily.

**Theorem 5.11** (Expected running time of Algorithm 25) Algorithm 25 terminates after at most $3 \log_3(n + 1) = O(\log n)$ phases in expectation.

**Proof:** The probability that in a single phase at least a quarter of all edges are removed is at least $\frac{1}{3}$. For the sake of contradiction, assume not. Then the expected fraction of removed edges is strictly less than $1/3$. However, at least every third phase is “good,” and removes at least a quarter of the edges. To get rid of all but two edges we need $\log_3 3$ good phases in expectation. Therefore $\frac{3 \log_3(n + 1)}{3} = \log_3(n + 1)$ phases are enough in expectation.

**Remarks:**

- One can prove that the algorithm only needs $O(\log n)$ phases in expectation. We would like to prove the algorithm managed to remove a constant fraction of nodes in each phase. Unfortunately, it does not.

- Instead we will prove that the number of removed edges is at least a fraction of the total number of edges. To get rid of all but two edges we need $\log_3 3$ good phases in expectation.

- We now count the removed edges. Whether we remove the edges adjacent to a node $v$ or to a node $w$, we call this event $(v \rightarrow w)$, and $(w \rightarrow v)$ respectively. The probability that event $(v \rightarrow w)$ occurs is $d(v) + d(w)$ (the maximum number of edges incident to a node $v$ or $w$ is $m$). The probability that event $(w \rightarrow v)$ occurs is $d(w) + d(v)$. Since $d(v) = d(w)$, we have $P((v \rightarrow w)) + P((w \rightarrow v)) = 1$. Hence, at least every third phase is “good,” and removes at least a quarter of the edges. To get rid of all but two edges we need $\log_3 3$ good phases in expectation. Therefore $\frac{3 \log_3(n + 1)}{3} = \log_3(n + 1)$ phases are enough in expectation.

**Lemma 5.10** (Erdős-Rényi) In a single phase, if we replace each undirected edge $\{u, v\}$ by the two directed edges $(u \rightarrow v)$ and $(v \rightarrow u)$, then we have expected at least $3 \log_3(n + 1)$ of the undirected edges are removed.

**Proof:**

If $u$ is a new MIS node added in the current phase, we denote by $d(u)$ the number of edges incident to $u$. If $u$ joins the MIS, all (directed) edges $(v \rightarrow u)$ for all $v \in N(u)$ will be removed. If $u$ does not join the MIS, there are at most $d(u) + 1$ edges $(v \rightarrow u)$ for all $v \in N(u)$. For $u \notin N(v)$, we call this event $(v \rightarrow u)$, and $(w \rightarrow v)$ respectively. The probability that event $(v \rightarrow u)$ occurs is $d(v) + d(u)$ (the maximum number of edges incident to a node $v$ or $w$ is $m$). The probability that event $(w \rightarrow v)$ occurs is $d(w) + d(v)$. Since $d(v) = d(w)$, we have $P((v \rightarrow u)) + P((w \rightarrow v)) = 1$. Hence, at least every third phase is “good,” and removes at least a quarter of the edges. To get rid of all but two edges we need $\log_3 3$ good phases in expectation. Therefore $\frac{3 \log_3(n + 1)}{3} = \log_3(n + 1)$ phases are enough in expectation.
for any choice of $c \geq 1$. Here $c$ may affect the constants in the Big-O notation because it is considered a “tunable constant” and usually kept small.

**Definition 5.13 (Chernoff’s Bound).** Let $X = \sum_{i=1}^{n} X_i$ be the sum of $k$ independent $0–1$ random variables. Then Chernoff’s bound states that w.h.p.

$$|X - \mathbb{E}[X]| \leq O(\sqrt{\mathbb{E}[X] \log n})$$

**Corollary 5.14 (Running Time of Algorithm 25).** Algorithm 25 terminates w.h.p. in $O(\log n)$ time.

**Proof:** In Theorem 5.11 we used that independently of everything that happened before, in each phase we have a constant probability $p$ that a quarter of the edges are removed. Call such a phase good. For some constants $C_1$ and $C_2$, let us check after $C_1 \log n + C_2 \in O(\log n)$ phases, in how many phases at least a quarter of the edges have been removed. In expectation, these are at least $p(\log \log n + C_2)$ many. Now we look at the random variable $X = \sum_{i=1}^{C_1 \log n + C_2} X_i$, where the $X_i$ are independent $0–1$ variables being one with exactly probability $p$. Certainly, if $X$ is at least $x$ with some probability, then the probability that we have $x$ good phases can only be larger (if no edges are left, certainly “all” of the remaining edges are removed). To $X$ we can apply Chernoff’s bound. If $C_1$ and $C_2$ are chosen large enough, they will overcome the constants in the Big-O from Chernoff’s bound, i.e., w.h.p. it holds that $|X - \mathbb{E}[X]| \leq \mathbb{E}[X]/2$, implying $X \geq \mathbb{E}[X]/2$. Choosing $C_1$ large enough, we will have w.h.p. sufficiently many good phases, i.e., the algorithm terminates w.h.p. in $O(\log n)$ phases.

**Remarks:**

- The algorithm can be improved a bit more even. Drawing random real numbers in each phase for instance is not necessary. One can achieve the same by sending only a total of $O(\log n)$ random (and as many non-random) bits over each edge.

- One of the main open problems in distributed computing is whether one can beat this logarithmic time, or at least achieve it with a deterministic algorithm.

- Let’s turn our attention to applications of MIS next.

### 5.4 Applications

**Definition 5.15 (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., no node is adjacent to two 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.

**Theorem 5.17 (Analysis of Algorithm 26).** Algorithm 26 $(\Delta + 1)$-colors an arbitrary graph in $O(\log n)$ time, with high probability, $\Delta$ being the largest degree in the graph.

**Proof:** Thanks to the clique among the clones at most one clone is in the MIS. And because of the $d(v)+1$ clones of node $v$ every node will get a free color! The running time remains logarithmic since $G'$ has $O(n^2)$ nodes and the exponent becomes a constant factor when applying the logarithm.

**Remarks:**

- This solves our open problem from Chapter 1.1!

- Together with Corollary 5.3 we get quite close ties between $(\Delta+1)$-coloring and the MIS problem.
Computing a MIS also solves another graph problem on graphs of bounded independence.

Definition 5.18 (Bounded Independence). \( G = (V, E) \) is of bounded independence, if each neighborhood contains at most a constant number of independent (i.e., mutually non-adjacent) nodes.

Definition 5.19 (Minimum Dominating Sets). A dominating set is a subset of the nodes such that each node is in the set or adjacent to a node in the set. A minimum dominating set is a dominating set containing the least possible number of nodes.

Remarks:
- In general, finding a dominating set less than factor \( \log n \) larger than an minimum dominating set is \( \mathcal{NP} \)-hard.
- Any MIS is a dominating set: if a node was not covered, it could join the independent set.
- In general a MIS and a minimum dominating sets have not much in common (think of a star). For graphs of bounded independence, this is different.

Corollary 5.20. On graphs of bounded independence, a constant-factor approximation to a minimum dominating set can be found in time \( O(n \log n) \) w.h.p.

Proof: Denote by \( M \) a minimum dominating set and by \( I \) a MIS. Since \( M \) is a dominating set, each node from \( I \) is in \( M \) or adjacent to a node in \( M \). Since the graph is of bounded independence, no node in \( M \) is adjacent to more than constantly many nodes from \( I \). Thus, \( |I| \leq O(|M|) \). Therefore, we can compute a MIS with Algorithm 25 and output it as the dominating set, which takes \( O(n \log n) \) rounds w.h.p.

Chapter Notes

The first MIS algorithm was a simplified version of an algorithm by Luby [Lub86]. Around the same time there have been a number of other papers dealing with the same or related problems, for instance by Alon, Babai, and Itai [ABB86], or by Israeli and Itai [IIt86]. The analysis presented in Section 5.2 takes elements of all these papers, and from other papers on distributed weighted matching [WW04]. The analysis in the book [Po00] by David Peleg is different, and only achieves \( O(n \log^2 n) \) time. The new MIS variant (with the simpler analysis) of Section 5.3 is by Métivier, Robson, Saheb-Djahromi, and Zemmari [MRSDZ11]. With some adaptations, the algorithms [Lub86, MRSDZ11] only need to exchange a total of \( O(n \log n) \) bits per node, which is asymptotically optimum, even on unoriented trees [KOS06]. However, the distributed time complexity for MIS is still somewhat open, as the strongest lower bounds are \( \Omega(\sqrt{n} \log n) \) or \( \Omega(\log \Delta) \) [KMW04]. Recent research regarding the MIS problem focused on improving the \( O(n \log n) \) time complexity for special graph classes, for instance growth-bounded graphs [SW08] or trees [LW11]. There are also results that depend on the degree of the graph [BE09, Kuh09]. Deterministic MIS algorithms are still far from the lower bounds, as the best deterministic MIS algorithm takes \( 2^{O(n^{1/4})} \) time [PS96]. The maximum matching algorithm mentioned in the remarks is the blossom algorithm by Jack Edmonds.

Bibliography