Chapter 4

Maximal Independent Set

In this chapter we present a first 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 1.1).

4.1 MIS

Definition 4.1 (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.*

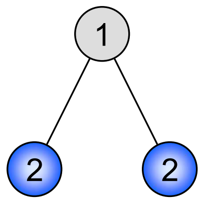

Figure 4.1: Example graph with 1) a maximal independent set (MIS) and 2) a maximum independent set (MaxIS).

Remarks:

- Computing a maximum independent set (MaxIS) is a notoriously difficult problem. It is equivalent to maximum clique on the complementary graph. Both problems are NP-hard, in fact not approximable within $n^{\frac{1}{2}-\epsilon}$.
- In this course we concentrate on the maximal independent set (MIS) problem. Please note that MIS and MaxIS can be quite different, indeed there are graphs where the MIS is $\Theta(n)$ smaller than the MaxIS.
- Computing a MIS sequentially is trivial: Scan the nodes in arbitrary order. If a node u does not violate independence, add u to the MIS. If u violates independence, discard u. So the only question is how to compute a MIS in a distributed way.

Algorithm 16 Slow MIS

```
Require: Node IDs
1: Every node $v$ executes the following code
2: if all neighbors of $v$ with larger identifiers have decided not to join the MIS then
3:   $v$ decides to join the MIS
4: end if
```

Remark:

- 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 4.2 (Analysis of Algorithm 16)**.** *Algorithm 16 features a time complexity of $O(n)$ and a message complexity of $O(m)$.*

Remark:

- This obviously 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 4.3. *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 4.3 we get a distributed deterministic MIS algorithm for trees (and for bounded degree graphs) with time complexity $O(\log^* n)$.

- With a lower bound argument one can show that the deterministic MIS algorithm for rings is asymptotically optimal.
- There have been attempts to extend Algorithm 5 to more general graphs, however, so far without much success. Below we present a radically different approach that uses randomization. Please note that the algorithm and the analysis below is not identical with the algorithm in Peleg's book.

Algorithm 17 Fast MIS

```
The algorithm operates in synchronous rounds, grouped into phases.
A single phase is as follows:
1) 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.
```

Remark:

- 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 4.4 (Joining MIS)**.** *A node v joins the MIS in step 2 with probability $p \geq \frac{1}{4d(v)}$.*

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 v and $H(v)$ in step 1 we get

$$\begin{aligned}
Pr\left[v \notin \text{MIS} | v \in M\right] &= Pr\left[\exists w \in H(v), w \in M | v \in M\right] \\
&= Pr\left[\exists w \in H(v), w \in M\right] \\
&\leq \sum_{w \in H(v)} Pr\left[w \in M\right] = \sum_{w \in H(v)} \frac{1}{2d(w)} \\
&\leq \sum_{w \in H(v)} \frac{1}{2d(v)} \leq \frac{d(v)}{2d(v)} = \frac{1}{2}.
\end{aligned}$$

Then

$$Pr\left[v \in \text{MIS}\right] = Pr\left[v \in \text{MIS} | v \in M\right] \cdot Pr\left[v \in M\right] \geq \frac{1}{2} \cdot \frac{1}{2d(v)}$$

□

Lemma 4.5 (Good Nodes)**.** *A node v is called* good *if*

$$\sum_{w \in N(v)} \frac{1}{2d(w)} \geq \frac{1}{6}.$$

Otherwise we call v a bad *node. A good node will be removed in step 3 with probability $p \geq \frac{1}{36}$.*

Proof: Let node v be good. Intuitively, good nodes have lots of low-degree neighbors, thus chances are high that one of them goes into the independent set, in which case v will be removed in step 3 of the algorithm.

If there is a neighbor $w \in N(v)$ with degree at most 2 we are done: With Lemma 4.4 the probability that node w joins the MIS is at least $\frac{1}{8}$, and our good node will be removed in step 3.

So all we need to worry about is that all neighbors have at least degree 3: For any neighbor w of v we have $\frac{1}{2d(w)} \leq \frac{1}{6}$. Since $\displaystyle\sum_{w \in N(v)} \frac{1}{2d(w)} \geq \frac{1}{6}$ there is a subset of neighbors $X \subseteq N(v)$ such that $\displaystyle\frac{1}{6} \leq \sum_{w \in X} \frac{1}{2d(w)} \leq \frac{1}{3}$

We can now bound the probability that node v will be removed. Let therefore E be the event of v being removed. Again, if a neighbor of v joins the MIS in step 2, node v will be removed in step 3. We have

$$\begin{aligned} Pr\left[E\right] &\geq Pr\left[\exists u \in X, u \in \text{MIS}\right] \\ &\geq \sum_{u \in X} Pr\left[u \in \text{MIS}\right] - \sum_{u,w \in X; u \neq w} Pr\left[u \in \text{MIS } \textbf{and } w \in \text{MIS}\right]. \end{aligned}$$

For the last inequality we used the inclusion-exclusion principle truncated after the second order terms. Let M again be the set of marked nodes after step 1. Using $Pr\left[u \in M\right] \geq Pr\left[u \in \text{MIS}\right]$ we get

$$\begin{aligned} Pr\left[E\right] &\geq \sum_{u \in X} Pr\left[u \in \text{MIS}\right] - \sum_{u,w \in X; u \neq w} Pr\left[u \in M \textbf{ and } w \in M\right] \\ &\geq \sum_{u \in X} Pr\left[u \in \text{MIS}\right] - \sum_{u \in X} \sum_{w \in X} Pr\left[u \in M\right] \cdot Pr\left[w \in M\right] \\ &\geq \sum_{u \in X} \frac{1}{4d(u)} - \sum_{u \in X} \sum_{w \in X} \frac{1}{2d(u)} \frac{1}{2d(w)} \\ &\geq \sum_{u \in X} \frac{1}{2d(u)} \left(\frac{1}{2} - \sum_{w \in X} \frac{1}{2d(w)} \right) \geq \frac{1}{6} \left(\frac{1}{2} - \frac{1}{3} \right) = \frac{1}{36} \end{aligned}$$

□

Remark:

- We would be almost finished if we could proof 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 work-around.

Lemma 4.6 (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 4.7. *A bad node has outdegree at least twice its indegree.*

Proof: For the sake of contradiction, assume that a bad node v does not have outdegree at least twice its indegree. In other words, at least one third of the neighbor nodes (let's call them S) have degree at most $d(v)$. But then

$$\sum_{w \in N(v)} \frac{1}{2d(w)} \geq \sum_{w \in S} \frac{1}{2d(w)} \geq \sum_{w \in S} \frac{1}{2d(v)} \geq \frac{d(v)}{3} \frac{1}{2d(w)} = \frac{1}{6}$$

which means v is good, a contradiction. □

Continuing the proof of Lemma 4.6: According to Lemma 4.7 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 4.8 (Analysis of Algorithm 17)**.** *Algorithm 17 terminates in expected time $O(\log n)$.*

Proof: With Lemma 4.5 a good node (and therefore a good edge!) will be deleted with constant probability. Since at least half of the edges are good (Lemma 4.6) a constant number of edges will be deleted in each phase. After $O(\log m)$ phases all edges are deleted. Since $m \leq n^2$ the Theorem follows. □

Remarks:

- The proof of Theorem 4.8 was a bit hasty. To make the log argument work one has to show that with high probability one can remove a constant number of edges in every round. With a bit of more "boring" math the argument can be made formal, in fact one can even show that Algorithm 17 terminates in time $O(\log n)$ "with high probability".

- The presented algorithm is a simplified version of an algorithm by Michael Luby, published 1986 in the SIAM Journal of Computing. 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, or by Israeli and Itai. The analysis presented takes elements of these papers, and from yet other papers on distributed weighted matching. The analysis in the book by David Peleg is different, and only achieves $O(\log^2 n)$ time.

- Surprisingly, much later, there have been half a dozen more papers published, with much worse results!! In 2002, for instance, there was a paper with linear running time, improving on a 1994 paper with cubic running time, restricted to trees!
- Let's turn our attention to applications of MIS next.

4.2 Applications

Definition 4.9 (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 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.*

Remarks:

- In contrast to MaxIS, a maximum matching can be found in polynomial time (Blossom algorithm by Jack Edmonds), and is also easy to approximate (in fact, already any maximal matching is a 2-approximation).
- An independent set algorithm is also a matching algorithm: Let $G = (V, E)$ be the graph for which we want to construct the matching. The auxiliary graph G' is defined as follows: for every edge in G there is a node in G'; two nodes in G' are connected by an edge if their respective edges in G are adjacent. A (maximal) independent set in G' is a (maximal) matching in G, and vice versa. Using Algorithm 17 directly produces a $O(\log n)$ bound for maximal matching.
- More importantly, our MIS algorithm can also be used for vertex coloring (Problem 1.1):

Algorithm 18 General Graph Coloring

```
1: Given a graph $G = (V, E)$ we virtually build a graph $G' = (V', E')$ as follows:
2: Every node $v \in V$ clones itself $d(v)+1$ times ($v_0, \ldots, v_{d(v)} \in V'$), $d(v)$ being the degree of $v$ in $G$.
3: The edge set $E'$ of $G'$ is as follows:
4: First all clones are in a clique: $(v_i, v_j) \in E'$, for all $v \in V$ and all $0 \leq i < j \leq d(v)$
5: Second all $i^{th}$ clones of neighbors in the original graph $G$ are connected: $(u_i, v_i) \in E'$, for all $(u, v) \in E$ and all $0 \leq i \leq \min(d(u), d(v))$.
6: Now we simply run (simulate) the fast MIS Algorithm 17 on $G'$.
7: If node $v_i$ is in the MIS in $G'$, then node $v$ gets color $i$.
```

Theorem 4.10 (Analysis of Algorithm 18)**.** *Algorithm 18 $(\Delta + 1)$-colors an arbitrary graph in $O(\log n)$ time, with high probability, Δ 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 Graph G' can be simulated without overhead.

Remarks:

- This solves our open problem from Chapter 1.1!
- Together with Corollary 4.3 we get quite close ties between (Δ+1)-coloring and the MIS problem.