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.

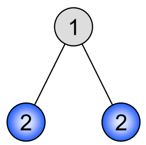

Figure 5.1: Example graph with 1) a maximal independent set (MIS) and 2) a maximum independent set (MaxIS).

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 e.g. on a star graph the MIS is $\Theta(n)$ smaller than the MaxIS (cf. Figure 5.1).
- 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 23 Slow MIS

Require: Node IDs

Every node v executes the following code:

if all neighbors of v with larger identifiers have decided not to join the MIS **then**

  v decides to join the MIS

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 $\mathcal{O}(n)$ and a message complexity of $\mathcal{O}(m)$.*

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 $\mathcal{O}(\log^* n)$.

- With a lower bound argument one can show that this 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.

5.2 Original Fast MIS

Algorithm 24 Fast MIS

The algorithm operates in synchronous rounds, grouped into phases.
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 $p \geq \frac{1}{4d(v)}$.*

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 the random choices of v and nodes in $H(v)$ in Step 1 we get

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

Then

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

□

Lemma 5.5 (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 5.4 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 $\sum_{w \in N(v)} \frac{1}{2d(w)} \geq \frac{1}{6}$ there is a subset of neighbors $S \subseteq N(v)$ such that $\frac{1}{6} \leq \sum_{w \in S} \frac{1}{2d(w)} \leq \frac{1}{3}$

We can now bound the probability that node v will be removed. Let therefore R 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

$$\begin{aligned}
P[R] &\geq P[\exists u \in S, u \in \mathrm{MIS}] \\
&\geq \sum_{u \in S} P[u \in \mathrm{MIS}] - \sum_{u,w \in S; u \neq w} P[u \in \mathrm{MIS} \textbf{ and } w \in \mathrm{MIS}].
\end{aligned}$$

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 $P[u \in M] \geq P[u \in \mathrm{MIS}]$ we get

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

□

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 work-around.

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 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

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

which means 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 $\mathcal{O}(\log n)$.*

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

$$\begin{aligned} \mathbb{E}[R] = \sum_r P[R = r] \cdot r &\leq P[R \leq \mathbb{E}[R]/2] \cdot \mathbb{E}[R]/2 + P[R > \mathbb{E}[R]/2] \cdot m \\ &= p \cdot \mathbb{E}[R]/2 + (1-p) \cdot m. \end{aligned}$$

Solving for p we get

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

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

Remarks:

- With a bit of more math one can even show that Algorithm 24 terminates in time $\mathcal{O}(\log n)$ "with high probability".

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) If $r(v) < r(w)$ for all neighbors $w \in N(v)$, node v enters the MIS and informs its neighbors.
3) If v or a neighbor of v entered the MIS, v terminates (v 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, hence there is progress.
- 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*

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

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

$$\begin{aligned} P[(X,Y) = (x,y)] &= P[X = x] \cdot P[Y = y | X = x] \\ &= P[Y = y] \cdot P[X = x | Y = y] \end{aligned}$$

we get that

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

□

Remarks:

- How can we prove that the algorithm only needs $\mathcal{O}(\log n)$ phases in expectation? It would be great if this algorithm managed to remove a constant fraction of nodes in each phase. Unfortunately, it does not.
- Instead we will prove that the number of *edges* decreases quickly. Again, it would be great if any single edge was removed with constant probability in Step 3. But again, unfortunately, this is not the case.
- Maybe we can argue about the expected number of edges to be removed in one single phase? Let's see: A node v enters the MIS with probability $1/(d(v)+1)$, where $d(v)$ is the degree of node v. By doing so, not only are v's edges removed, but indeed all the edges of v's neighbors as well – generally these are much more than $d(v)$ edges. So there is hope, but we need to be careful: If we do this the most naive way, we will count the same edge many times.
- How can we fix this? The nice observation is that it is enough to count just some of the removed edges. Given a new MIS node v and a neighbor $w \in N(v)$, we count the edges only if $r(v) < r(x)$ for all $x \in N(w)$. This looks promising. In a star graph, for instance, only the smallest random value can be accounted for removing all the edges of the star.

Lemma 5.10 (Edge Removal)**.** *In a single phase, we remove at least half of the edges in expectation.*

Proof. To simplify the notation, at the start of our phase, the graph is simply $G=(V,E)$. In addition, to ease presentation, we replace each undirected edge $\{v,w\}$ by the two directed edges (v,w) and (w,v).

Suppose that a node v joins the MIS in this phase, i.e., $r(v) < r(w)$ for all neighbors $w \in N(v)$. If in addition we have $r(v) < r(x)$ for all neighbors x of a neighbor w of v, we call this event $(v \to w)$. The probability of event $(v \to w)$ is at least $1/(d(v)+d(w))$, since $d(v)+d(w)$ is the maximum number of nodes adjacent to v or w (or both). As v joins the MIS, all (directed) edges (w,x) with $x \in N(w)$ will be removed; there are $d(w)$ of these edges.

We now count the removed 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 $d(w)$, 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 5.9, the expected value of the sum X of all these random variables is at least

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

In other words, in expectation $|E|$ directed edges are removed in a single phase! Note that we did not double count any edge removals, as a directed edge (v,w) can only be removed by an event $(u \to v)$. The event $(u \to v)$ inhibits a concurrent event $(u' \to v)$ since $r(u) < r(u')$ 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. □

Remarks:

- 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 $3\log_{4/3} m + 1 \in O(\log n)$ phases in expectation.*

Proof: The probability that in a single phase at least a quarter of all edges are removed is at least 1/3. For the sake of contradiction, assume not. Then with probability less than 1/3 we may be lucky and many (potentially all) edges are removed. With probability more than 2/3 less than 1/4 of the edges are removed. Hence the expected fraction of removed edges is strictly less than $1/3\cdot 1 + 2/3\cdot 1/4 = 1/2$. This contradicts Lemma 5.10.

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 $\log_{4/3} m$ good phases in expectation. The last two edges will certainly be removed in the next phase. Hence a total of $3\log_{4/3} m + 1$ phases are enough in expectation.

Remarks:

- Sometimes one expects a bit more of an algorithm: Not only should the expected time to terminate be good, but the algorithm should *always* terminate quickly. As this is impossible in randomized algorithms (after all, the random choices may be "unlucky" all the time!), researchers often settle for a compromise, and just demand that the probability that the algorithm does not terminate in the specified time can be made absurdly small. For our algorithm, this can be deduced from Lemma 5.10 and another standard tool, namely Chernoff's Bound.

Definition 5.12 (W.h.p.)**.** *We say that an algorithm terminates* w.h.p. *(with high probability) within $\mathcal{O}(t)$ time if it does so with probability at least $1-1/n^c$*

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 $X = \sum_{i=1}^{k} X_i$ be the sum of k independent $0-1$ random variables. Then* Chernoff's bound *states that w.h.p.*

$$|X - \mathbb{E}[X]| \in O\left(\log n + \sqrt{\mathbb{E}[X]\log n}\right).$$

Corollary 5.14 (Running Time of Algorithm 25). *Algorithm 25 terminates w.h.p. in $\mathcal{O}(\log n)$ 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 $p(C_1 \log n + C_2)$ 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 $\mathcal{O}(\log n)$ 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 $\mathcal{O}(\log n)$ 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., 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, 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 25 directly produces a $\mathcal{O}(\log n)$ bound for maximal matching.

- More importantly, our MIS algorithm can also be used for vertex coloring (Problem 1.1):

Definition 5.16. *An approximation algorithm $\mathcal{A}$ for a maximization problem Π has an approximation factor of r if the following condition holds for all instances $I \in \Pi$:*

$$\frac{OPT(I)}{\mathcal{A}(I)} \leq r.$$

Algorithm 26 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 25 on G'.
7: If node v_i is in the MIS in G', then node v gets color i.

Theorem 5.17 (Analysis of Algorithm 26). *Algorithm 26 $(\Delta + 1)$-colors an arbitrary graph in $\mathcal{O}(\log n)$ 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 G' has $\mathcal{O}\left(n^2\right)$ 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 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 $\mathcal{O}(\log n)$ 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, $|I| \in O(|M|)$. Therefore, we can compute a MIS with Algorithm 25 and output it as the dominating set, which takes $\mathcal{O}(\log n)$ rounds w.h.p.

Chapter Notes

The fast MIS algorithm is 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 [ABI86], or by Israeli and Itai [II86]. 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 [Pel00] by David Peleg is different, and only achieves $\mathcal{O}(\log^2 n)$ 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 $\mathcal{O}(\log n)$ bits per node, which is asymptotically optimum, even on unoriented trees [KSOS06]. However, the distributed time complexity for MIS is still somewhat open, as the strongest lower bounds are $\Omega(\sqrt{\log n})$ or $\Omega(\log \Delta)$ [KMW04]. Recent research regarding the MIS problem focused on improving the $\mathcal{O}(\log n)$ time complexity for special graph classes, for instances 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 $2^{O(\sqrt{\log n})}$ time [PS96]. The maximum matching algorithm mentioned in the remarks is the blossom algorithm by Jack Edmonds.

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.

[BE09] Leonid Barenboim and Michael Elkin. Distributed (delta+1)-coloring in linear (in delta) time. In *41st ACM Symposium On Theory of Computing (STOC)*, 2009.

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

[KMW04] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot Be Computed Locally! In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing (PODC)*, July 2004.

[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.

[Kuh09] Fabian Kuhn. Weak graph colorings: distributed algorithms and applications. In *21st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2009.

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

[LW11] Christoph Lenzen and Roger Wattenhofer. MIS on trees. In *PODC*, pages 41–48, 2011.

[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.

[Pel00] David Peleg. *Distributed computing: a locality-sensitive approach.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

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

[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.

[WW04] Mirjam Wattenhofer and Roger Wattenhofer. Distributed Weighted Matching. In *18th Annual Conference on Distributed Computing (DISC), Amsterdam, Netherlands*, October 2004.