Chapter 8

Local Graph Algorithms

Chapter by M. Ghaffari

Keywords: local graph algorithms, locality, coloring, maximal independent set, maximal matching, graph decomposition, graph shattering, approximation algorithms, chromatic number, girth, Sperner families, cover free families, neighborhood graphs, Ramsey theory.

Local Graph Algorithms are, in a rough sense, algorithms where each part of the output is determined as a function of only a *local area* in the graph, that is, a small neighborhood. These algorithms capture fundamental aspects of distributed systems and especially computer networks. Thus, they have been studied extensively since the 1980s. Furthermore, over the last decade, these local techniques have turned out to be a key ingredient for processing big graphs and performing graph computations in times that are sublinear in the graph size, or even just a constant.

Throughout this chapter, we present an introduction to the basics of local graph algorithms, in two parts:

- Much of our focus will be on *Distributed Local Algorithms*. These algorithm work in a setting where there is one process on each node of the graph and these processes compute the solution via local communications. A good example to keep in mind is a computer network, e.g., the Internet. The setting and its mathematical computational model will be made more concrete and formal soon.
- Towards the end of the chapter, we discuss *Centralized Local Algorithms*, which use methods of the same flavor and solve graph

problems in sublinear-time, most often being poly-logarithmic time or even just a constant.

8.1 The Distributed LOCAL Model

We work with the LOCAL model, which was first formalized by Linial [Lin87, Lin92]. The model definition is as follows.

Definition 8.1. *(The LOCAL model) We consider an arbitrary n-node graph $G = (V, E)$ where $V = \{1, 2, \ldots, n\}$, which abstracts the communication network. Unless noted otherwise, G is a simple, undirected, and unweighted graph. There is one process on each node $v \in V$ of the network. At the beginning, the processes do not know the graph G, except for knowing[1] n, and their own unique identifier in $\{1, 2, \ldots, n\}$. The algorithms work in synchronous rounds. Per round, each node/process performs some computation based on its own knowledge, and then sends a message to all of its neighbors, and then receives the messages sent to it by its neighbors in that round. In each graph problem in this model, we require that each node learns its own part of the output, e.g., its own color in a graph coloring.*

Comment: We stress that the model does not assume any limitation on the size of the messages, or on the computational power of the processes. Because of this, it is not hard to see that, any t-round algorithm in the LOCAL model induces a function which maps the t-hop neighborhood of each node to its output (why?). For instance, a t-round algorithm for graph coloring maps the topology induced by vertices within distance t of a vertex v to the coloring of vertex v. The converse of this statement is also true, meaning that if for a given graph problem, such a function exists, then there is also a t-round algorithm for solving that problem. Hence, one can say that the LOCAL model captures the *locality* of graph problems in a mathematical sense.

Observation 8.2. *Any graph problem on any n-node graph G can be solved in $O(n)$ rounds. In fact, using D to denote the diameter of the graph, any problem can be solved in $O(D)$ rounds.*

[1] Most often, the algorithms will use only the assumption that nodes know an upper bound N on n such that $N \in [n, n^c]$ for a small constant $c \geq 1$.

8.2 Coloring a Rooted Tree

We start by examining the graph coloring problem, and in a very special case, coloring rooted trees. This basic-looking problem already turns out to have some quite interesting depth, as we see in this section.

The setting is as follows. We consider an arbitrary rooted tree $T = (V, E)$, such that $V = \{1, 2, \ldots, n\}$, and where each node v knows its parent $p(v)$ in T. The objective is to find a proper coloring of T, that is, a color assignment $\phi : V \to \{1, 2, \ldots, q\}$ such that there does not exist any node v with $\phi(v) = \phi(p(v))$. Of course, we are interested in using a small number of colors q, and we seek fast algorithms for computing such a coloring, that is, algorithms that use a small number of rounds.

Clearly, each tree can be colored using just 2 colors. However, computing a 2-coloring in the LOCAL model is not such an interesting problem, due to the following simple observation:

Observation 8.3. *Any* LOCAL *algorithm for 2-coloring an n-node directed path requires at least $\Omega(n)$ rounds.*

In contrast, 3-coloring has no such unfortunate lower bound, and in fact, entails something quite non-trivial: it has a tight round complexity of $\frac{1}{2}\log^* n \pm O(1)$. Recall the definition of the log-Star function:

$$\log^*(x) = \begin{cases} 0 & \text{if } x \leq 1 \\ 1 + \log^*(\log x) & \text{if } x > 1 \end{cases}$$

To prove this tight $\frac{1}{2}\log^* n \pm O(1)$ round complexity, in the next two subsections, we explain the following two directions of this result:

- First, in Section 8.2.1, we explain a $\log^* n + O(1)$ round algorithm for 3-coloring rooted trees. The upper bound can actually be improved to $\frac{1}{2}\log^* n + O(1)$ rounds [SV93], and even to exactly $\frac{1}{2}\log^* n$ rounds [RS14], but we do not cover those refinements. There are four known methods for obtaining $O(\log^* n)$-round algorithms [CV86, SV93, NS93, FHK16]. The algorithm we describe is based on an idea of [NS95] and some extra step from [GPS87]. The approach of [CV86] will be covered in Exercise 8.1.

- Then, in Section 8.2.2, we prove the above bound to be essentially optimal by showing that any deterministic algorithm for 3-coloring

rooted trees requires at least $\frac{1}{2}\log^* n - O(1)$ rounds. This result was first proved by [Lin87, Lin92]. We explain a somewhat streamlined proof, based on [LS14]. The lower bound holds also for randomized algorithms [Nao91], but we will not cover that generalization, for the sake of simplicity. Furthermore, essentially the same lower bound can be obtained as a direct corollary of *Ramsey Theory*. We will have a brief explanation about that, at the end of this subsection.

8.2.1 3-Coloring Rooted Trees in $\log^* n + O(1)$ Rounds

Theorem 8.4. *Any n-node rooted-tree can be colored with 3 colors, in $\log^* n + O(1)$ rounds.*

Notice that the initial numbering $1, 2, \ldots, n$ of the vertices is already a coloring with n colors. We explain a method for gradually improving this coloring, by iteratively reducing the number of colors. They key ingredient is a single-round color-reduction method, based on *Sperner families*, which achieves the following:

Lemma 8.5. *Given a k-coloring ϕ_{old} of a rooted tree where $k \geq C_0$ for a constant[2] C_0, in a single round, we can compute a k'-coloring ϕ_{new}, for $k' = \log k + \log\log k/2 + 1$.*

Proof. Let each node u send its color $\phi_{old}(u)$ to its children. We now describe a method which allows each node v to compute its new coloring $\phi_{new}(v)$, based on $\phi_{old}(v)$ and $\phi_{old}(u)$ where u is the parent of v, with no further communication.

Consider an arbitrary one-to-one mapping $M : \{1, 2, \ldots, k\} \to \mathcal{F}_{k'}$, fixed a priori, where $\mathcal{F}_{k'}$ denotes the set of all the subset of size $k'/2$ of the set $\{1, 2, \ldots, k'\}$. Notice that such a one-to-one mapping exists, because

$$|\mathcal{F}_{k'}| = \binom{k'}{k'/2} \geq 2^{k'}/\sqrt{2k'} \geq k.$$

For each node v, we compute the new color $\phi_{new}(v) \in \{1, 2, \ldots, k'\}$ of v as follows. Let u be the parent of v. Since both $M(\phi_{old}(v))$ and $M(\phi_{old}(u))$ are subsets of size $k'/2$, and because $\phi_{old}(v) \neq \phi_{old}(v)$ and M is a one-to-one mapping, we know that $M(\phi_{old}(v)) \setminus M(\phi_{old}(u)) \neq \emptyset$. Let $\phi_{new}(v)$

[2] We assume this constant lower bound C_0 mainly to simplify our job and let us not worry about the rounding issues.

be any arbitrary color in $M(\phi_{old}(v)) \setminus M(\phi_{old}(u))$. Since each node v gets a color $\phi_{new}(v) \in M(\phi_{old}(v)) \setminus M(\phi_{old}(u))$ that is not in the color set $M(\phi_{old}(u))$ of its parent, $\phi_{new}(v) \neq \phi_{new}(u) \in M(\phi_{old}(u))$. Hence, ϕ_{new} is a proper coloring. □

Remark 8.1. *Notice that in the above proof, the main property that we used is that none of the color-sets $M(i) \in \mathcal{F}_{k'}$ is contained in another $M(j) \in \mathcal{F}_{k'}$, for $i \neq j$. Generally, a family of sets such that none of them is contained in another is called a* Sperner Family. *In particular, ℓ-element subsets of a k'-element set form a Sperner family, the size of which is maximized by setting $\ell = \lfloor k'/2 \rfloor$, as we did above. More generally, Sperner's theorem shows that any Sperner family on a ground set of size k' has size at most $\binom{k'}{\lfloor k'/2 \rfloor}$ [Spe28]. See [Lub66] for a short and cute proof of Sperner's theorem, via a simple double counting.*

We can now iteratively apply the above method, abstracted in the statement of Lemma 8.5, to reduce the number of colors. After one round, we go from an initial n-coloring to a $(\log n + \log\log n/2 + 1)$-coloring. After one more round, we get to a coloring which has no more than $\log\log n + O(\log\log\log n)$ colors. More generally, after at most $\log^* n + O(1)$ repetitions, we get to a coloring with no more than C_0 colors, for a constant C_0. At this point, we cannot apply the above routine anymore. However, we can use an easier method that repeatedly uses two rounds to shave off one color, until arriving at a 3-coloring. We explain this next. Overall, we use $\log^* n + O(1)$ rounds to get a 3-coloring.

Lemma 8.6. *Given a k-coloring ϕ_{old} of a rooted tree where $k \geq 4$, in two rounds, we can compute a $(k-1)$-coloring ϕ_{new}.*

Proof. First, use one round where each node u sends its color $\phi_{old}(u)$ to its children. Then, let each node v set its temporary coloring $\phi'_{old}(v) = \phi_{old}(u)$, where u is the parent of v. For the root node r, this rule is not well-defined. But that is easy to fix. Define $\phi'_{old}(r) \in \{1,2,3\} \setminus \phi_{old}(r)$. Observe that ϕ'_{old} is a proper k-coloring, with the following nice additional property: for each node u, all of its children have the same color $\phi'_{old}(u)$.

Now, use another round where each node u sends its color $\phi'_{old}(u)$ to its children. Then, define the new color $\phi_{new}(v)$ as follows. For each node v such that $\phi'_{old}(v) \neq k$, let $\phi_{new}(v) = \phi'_{old}(v)$. For each node v such that $\phi'_{old}(v) = k$, let $\phi_{new}(v)$ be a color in $\{1,2,3\} \setminus \{\phi'_{old}(u), \phi_{old}(v)\}$.

Notice that since only nodes of color k are changing their color, these nodes are non-adjacent. Each of them switches to a color that is different than what is held by its parent and its children. Hence, the new coloring $\phi_{new}(v)$ is proper. □

Proof of Theorem 8.4. The proof follows by applying Lemma 8.5 for $\log^* n + O(1)$ iterations, until getting to a coloring with no more than $C_0 = O(1)$ colors, and then applying the method of Lemma 8.6 for $C_0 - 3 = O(1)$ iterations, until getting to a 3-coloring. □

8.2.2 3-Coloring Directed Paths Needs $\frac{1}{2}\log^* n - O(1)$ Rounds

Theorem 8.7. *Any deterministic algorithm for 3-coloring n-node directed paths needs at least $\frac{\log^* n}{2} - 2$ rounds.*

For the sake of contradiction, suppose that there is an algorithm $\mathcal{A}$ that computes a 3-coloring of any n-node directed path in t rounds for $t < \frac{\log^* n}{2} - 2$. When running this algorithm for t rounds, any node v can see at most the k-neighborhood around itself for $k = 2t + 1$, that is, the vector of identifiers for the nodes up to t hops before itself and up to t hops after itself. Hence, if the algorithm $\mathcal{A}$ exists, there is a mapping from each such neighborhood to a color in $\{1, 2, 3\}$ such that neighborhoods that can be conceivably adjacent are mapped to different colors.

We next make this formal by a simple and abstract definition. For simplicity, we will consider only a restricted case of the problem where the identifiers are set monotonically increasing along the path. Notice this restriction only strengthens the lower bound, as it shows that even for this restricted case, there is no t-round algorithm for $t < \frac{\log^* n}{2} - 2$.

Definition 8.8. *We say B is a k-ary q-coloring if for any set of identifiers $1 \leq a_1 < a_2 < \cdots < a_k < a_{k+1} \leq n$, we have the following two properties:*

P1: $B(a_1, a_2, \ldots, a_k) \in \{1, 2, \ldots, q\}$,

P2: $B(a_1, a_2, \ldots, a_k) \neq B(a_2, \ldots, a_{k+1})$.

Observation 8.9. *If there exists a deterministic algorithm $\mathcal{A}$ for 3-coloring n-node directed paths in $t < \frac{\log^* n}{2} - 2$ rounds, then there exists a k-ary 3-coloring B, where $k = 2t + 1 < \log^* n - 3$.*

Proof. Suppose that such an algorithm $\mathcal{A}$ exists. We then produce a k-ary 3-coloring B by examining $\mathcal{A}$. For any set of identifiers $1 \leq a_1 < a_2 < \cdots < a_k \leq n$, define $B(a_1, a_2, \ldots, a_k)$ as follows. Simulate algorithm $\mathcal{A}$ on an imaginary directed path where a consecutive portion of the identifiers on the path are set equal to $a_1, a_2, \ldots, a_k$. Then, let $B(a_1, a_2, \ldots, a_k)$ be equal to the color in $\{1, 2, 3\}$ that the node a_{t+1} receives in this simulation.

We now argue that B as defined above is a k-ary 3-coloring. Property P1 holds trivially. We now argue that property P2 also holds. For the sake of contradiction, suppose that it does not, meaning that there exist a set of identifiers $1 \leq a_1 < a_2 < \cdots < a_k < a_{k+1} \leq n$ such that $B(a_1, a_2, \ldots, a_k) = B(a_2, \ldots, a_{k+1})$. Then, imagine running algorithm $\mathcal{A}$ on an imaginary directed path where a consecutive portion of identifiers are set equal to $a_1, a_2, \ldots, a_{2t+2}$. Then, since $B(a_1, a_2, \ldots, a_k) = B(a_2, \ldots, a_{k+1})$, the algorithm $\mathcal{A}$ assigns the same color to a_{t+1} and a_{t+2}. This is in contradiction with $\mathcal{A}$ being a 3-coloring algorithm. □

To prove Theorem 8.7, we show that a k-ary 3-coloring B where $k < \log^* n - 3$ cannot exist. The proof is based on the following two lemmas:

Lemma 8.10. *There is no* 1*-ary* q*-coloring with* $q < n$.

Proof. A 1-ary q-coloring requires that $B(a_1) \neq B(a_2)$, for any two identifiers $1 \leq a_1 < a_2 \leq n$. By the Pigeonhole principle, this needs $q \geq n$. □

Lemma 8.11. *If there is a* k*-ary* q*-coloring* B*, then there exists a* $(k-1)$*-ary* 2^q*-coloring* B'.

Proof. For any set of identifiers $1 \leq a_1 < a_2 < \cdots < a_{k-1} \leq n$, define $B'(a_1, a_2, \ldots, a_{k-1})$ to be the set of all possible colors $i \in \{1, \ldots, q\}$ for which $\exists a_k > a_{k-1}$ *such that* $B(a_1, a_2, \ldots, a_{k-1}, a_k) = i$.

Notice that B' is a subset of $\{1, \ldots, q\}$. Hence, it has 2^q possibilities, which means that B' has property P1 and it assigns each set of identifiers $1 \leq a_1 < a_2 < \cdots < a_{k-1} \leq n$ to a number in 2^q. Now we argue that B' also satisfies property P2.

For the sake of contradiction, suppose that there exist identifiers $1 \leq a_1 < a_2 < \cdots < a_k \leq n$ such that $B'(a_1, a_2, \ldots, a_{k-1}) = B'(a_2, a_3, \ldots, a_k)$. Let $q^* = B(a_1, a_2, \ldots, a_k) \in B'(a_1, a_2, \ldots, a_{k-1})$. Then, we must have $q^* \in B'(a_2, a_3, \ldots, a_k)$. Thus, $\exists a_{k+1} > a_k$ such that $q^* = B(a_2, a_3, \ldots, a_k, a_{k+1})$. But, this means $B(a_1, a_2, \ldots, a_k) = q^* = B(a_2, a_3, \ldots, a_k, a_{k+1})$, which is

in contradiction with B being a k-ary q-coloring. Having reached at a contradiction by assuming that B' does not satisfy P2, we conclude that it actually does satisfy P2. Hence, B' is a $(k-1)$-ary 2^q-coloring. □

Proof of Theorem 8.7. For the sake of contradiction, suppose that there is an algorithm $\mathcal{A}$ that computes a 3-coloring of any n-node directed path in t rounds for $t \leq \frac{\log^* n}{2} - 2$. As stated in Observation 8.9, if there exists an algorithm $\mathcal{A}$ that computes a 3-coloring of any n-node directed path in t rounds for $t \leq \frac{\log^* n}{2} - 2$, then there exists a k-ary 3-coloring B, where $k = 2t + 1 < \log^* n - 3$. Using one iteration of Lemma 8.11, we would get that there exists a $(k-1)$-ary 8-coloring. Another iteration would imply that there exists a $(k-2)$-ary 2^8-coloring. Repeating this, after $k < \log^* n - 3$ iterations, we would get a 1-ary coloring with less than n colors. However, this is in contradiction with Lemma 8.10. Hence, such an algorithm $\mathcal{A}$ cannot exist. □

An Alternative Lower Bound Proof Via Ramsey Theory:
Let us first briefly recall the basics of Ramsey Theory. The simplest case of Ramsey's theorem says that for any ℓ, there exists a number $R(\ell)$ such that for any $n \geq R(\ell)$, if we color the edges of the n-node complete graph K_n with two colors, there exists a monochromatic clique of size ℓ in it, that is, a set of ℓ vertices such that all of the edges between them have the same color. A simple example is that among any group of at least $6 = R(3)$ people, there are either at least 3 of them which are friends, or at least 3 of them no two of which are friends.

A similar statement is true in hypergraphs. Of particular interest for our case is coloring hyperedges of a complete n-vertex hypergraph of rank k, that is, the hypergraph where every subset of size k of the vertices defines one hyperedge. By Ramsey theory, it is known that there exists an n_0 such that, if $n \geq n_0$, for any way of coloring hyperedges of the complete n-vertex hypergraph of rank k with 3 colors, there would be a monochromatic clique of size $k+1$. That is, there would be a set of $k+1$ vertices $a_1, \ldots, a_k$ in $\{1, \ldots, n\}$ such that all of their $\binom{k+1}{k} = k$ subsets with cardinality k have the same color.

In particular, consider an arbitrary k-ary coloring B, and let B define the colors of the hyperedges $\{a_1, \ldots, a_k\}$ when $1 \leq a_1 < a_2 < \cdots < a_k \leq n$. For the remaining hyperedges, color them arbitrarily. By Ramsey's theorem, we would get the following: there exist vertices $1 \leq a_1 < a_2 < \cdots < a_k <$

$a_{k+1} \leq n$ such that B assigns the same color to hyperedges $\{a_1, \ldots, a_k\}$ and $\{a_2, \ldots, a_{k+1}\}$. But this is in contradiction with the property P2 of B being a k-ary coloring. The value of n_0 that follows from Ramsey theory is such that $k = O(\log^* n_0)$. In other words, Ramsey's theorem rules out $o(\log^* n)$-round 3-coloring algorithms for directed paths. See [CFS10] for more on hypergraph Ramsey numbers.

8.3 Deterministic Coloring of General Graphs

8.3.1 Take 1: Linial's Coloring Algorithm

In the previous section, we discussed distributed LOCAL algorithms for coloring oriented trees. In this section, we start the study of LOCAL coloring algorithms for general graphs. Throughout, the ultimate goal would be to obtain $(\Delta+1)$-coloring of the graphs — that is, an assignment of colors $\{1,2,\ldots,\Delta+1\}$ to vertices such that no two adjacent vertices receive the same color — where Δ denotes the maximum degree. Notice that by a simple greedy argument, each graph with maximum degree at most Δ has a $(\Delta+1)$-coloring: color vertices one by one, each time picking a color which is not chosen by the already-colored neighbors. However, this greedy argument does not lead to an efficient LOCALprocedure for finding such a coloring[3].

In this section, we start with presenting an $O(\log^* n)$-round algorithm that computes a $O(\Delta^2)$ coloring. This algorithm is known as Linial's coloring algorithm [Lin87, Lin92]. In the next section, we see how to transform this coloring into a $(\Delta+1)$-coloring.

Theorem 8.12. *There is a deterministic distributed algorithm in the* LOCAL *model that colors any n-node graph G with maximum degree Δ using $O(\Delta^2)$ colors, in $O(\log^* n)$ rounds.*

Conceptually, the algorithm can be viewed as a more general variant of the algorithm we discussed in Section 8.2.1. In particular, the core piece is a single-round color reduction method, conceptually similar to that of Lemma 8.5. However, here, each node has to ensure that the color it picks is different than all of its neighbors, and not just its parents.

For that purpose, we will work with a generalization of Sperner families (used in Lemma 8.5), which are called *cover free families*:

Definition 8.13. *Given a ground set $\{1,2,\ldots,k'\}$, a family of sets S_1, S_2, $\ldots$, $S_k \subseteq \{1,2,\ldots,k'\}$ is called a Δ-cover free family if for each set of indices $i_0, i_1, i_2, \ldots, i_\Delta \in \{1,2,\ldots,k\}$, we have $S_{i_0} \setminus \left(\cup_{j=1}^{\Delta} S_{i_j}\right) \neq \emptyset$. That is, if no set in the family is a subset of the union of Δ other sets.*

[3] The straightforward transformation of this greedy approach to the LOCALmodel would be an algorithm that may need $\Omega(n)$ rounds.

Notice that in particular, a Sperner family is simply a 1-cover free family, i.e., no set is a subset of any other set.

Given this definition, and the single-round color reduction method that what we saw in Lemma 8.5 for rooted trees using Sperner families, the reader can already sees the analogous single-round color reduction method for general graphs using cover free families. This new single-color reduction will allow us to transform any k-coloring to a k'-coloring. This will be elaborated later on in Lemma 8.16. We would like to have k' be as small as possible, as a function of k and Δ. In the following, we prove the existence of a Δ-cover free families with a small enough ground set size k'. In particular, Lemma 8.14 achieves $k' = O(\Delta^2 \log k)$ and Lemma 8.15 shows that this bound can be improved to $k' = O(\Delta^2)$, if $k \leq \Delta^3$.

Lemma 8.14. *For any k and Δ, there exists a Δ-cover free family of size k on a ground set of size $k' = O(\Delta^2 \log k)$.*

Proof. We use the probabilistic method [AS04] to argue that there exists a Δ-cover free family of size k on a ground set of size $k' = O(\Delta^2 \log k)$. Let $k' = C\Delta^2 \log k$ for a sufficiently large constant $C \geq 2$. For each $i \in \{1, 2, \ldots, k\}$, define each set $S_i \subset \{1, 2, \ldots, k'\}$ randomly by including each element $q \in \{1, 2, \ldots, k'\}$ in S_i with probability $p = 1/\Delta$. We argue that this random construction is indeed a Δ-cover free family, with high probability, and therefore, such a cover free family exists.

First, consider an arbitrary set of indices $i_0, i_1, i_2, \ldots, i_\Delta \in \{1, 2, \ldots, k\}$. We would like to argue that $S_{i_0} \setminus \left(\cup_{j=1}^{\Delta} S_{i_j} \right) \neq \emptyset$. For each element $q \in \{1, 2, \ldots, k'\}$, the probability that $q \in S_{i_0} \setminus \left(\cup_{j=1}^{\Delta} S_{i_j} \right)$ is at exactly $\frac{1}{\Delta}(1 - \frac{1}{\Delta})^{\Delta} \geq \frac{1}{4\Delta}$. Hence, the probability that there is no such element q that is in $S_{i_0} \setminus \left(\cup_{j=1}^{\Delta} S_{i_j} \right)$ is at most $(1 - \frac{1}{4\Delta})^{k'} \leq exp(-C\Delta \log k/4)$. This is an upper bound on the probability that for a given set of indices set of indices $i_0, i_1, i_2, \ldots, i_\Delta \in \{1, 2, \ldots, k\}$, the respective sets violate the cover-freeness property that $S_{i_0} \setminus \left(\cup_{j=1}^{\Delta} S_{i_j} \right) \neq \emptyset$.

There are $k\binom{k-1}{\Delta}$ way to choose such a set of indices $i_0, i_1, i_2, \ldots, i_\Delta \in \{1, 2, \ldots, k\}$, k ways for choosing the central index i_0 and $\binom{k-1}{\Delta}$ ways for choosing the indices $i_1, i_2, \ldots, i_\Delta$. Hence, by a union bound over all these

choices, the probability that the construction fails is at most

$$\begin{aligned} k\binom{k-1}{\Delta+1} \cdot \exp(-C\Delta \log k/4) &\leq k(\frac{e(k-1)}{\Delta+1})^{\Delta+1} \cdot \exp(-C\Delta \log k/4) \\ &\leq \exp\Big(\log k + (\Delta+1)(\log k+1) - C\Delta \log k/4\Big) \\ &\leq \exp(-C\Delta \log k/8) \ll 1, \end{aligned}$$

for a sufficiently large constant C. That is, the random construction succeeds to provide us with a valid Δ-cover free family with a positive probability, and in fact with a probability close to 1. Hence, such a Δ-cover free family exists. □

Lemma 8.15. *For any k and $\Delta \geq k^{1/3}$, there exists a Δ-cover free family of size k on a ground set of size $k' = O(\Delta^2)$.*

Remark 8.2. *One can easily generalize this lemma's construction, by taking higher-degree polynomials, to a ground set of size $k' = O(\Delta^2 \log^2_{k'} k)$, where no assumption on the relation between k and Δ would be needed.*

Proof of Lemma 8.15. Here, we use an algebraic proof based on low-degree polynomials. Let q be a prime number that is in $[3\Delta, 6\Delta]$. Notice that such a prime number exists by Bertrand's postulate (also known as Bertrand-Chebyshev Theorem). Let $\mathbb{F}_q$ denote a field of characteristic q. For each $i \in \{1, 2, \ldots, k\}$, associate with set S_i — to be constructed — a distinct degree $d = 2$ polynomial $g_i : \mathbb{F}_q \to \mathbb{F}_q$ over $\mathbb{F}_q$. Notice that there are $q^{d+1} > \Delta^3 \geq k$ such polynomials and hence such an association is possible. Let S_i be the set of all evaluation points of g_i, that is, let $S_i = \{(a, g_i(a)) \mid a \in \mathbb{F}_q\}$. These are subsets of the $k' = q^2$ cardinality set $\mathbb{F}_q \times \mathbb{F}_q$. Notice two key properties:

(A) for each $i \in \{1, 2, \ldots, k\}$, we have $|S_i| = q$.

(B) for each $i, i' \in \{1, 2, \ldots, k\}$ such that $i \neq i'$, we have $|S_i \cap S_{i'}| \leq d$.

The latter property holds because, in every intersection point, the degree d polynomial $g_i - g_{i'}$ evaluates to zero, and each degree d polynomial has at most d zeros. Now, the Δ cover-freeness property follows trivially from (A) and (B), because for any set of indices $i_0, i_1, i_2, \ldots, i_\Delta \in \{1, 2, \ldots, k\}$, we have

$$\begin{aligned} |S_{i_0} \setminus \left(\cup_{j=1}^{\Delta} S_{i_j} \right)| \geq |S_{i_0}| &- \sum_{j=1}^{\Delta} |S_{i_0} \cap S_{i_j}| \\ \geq q &- \Delta \cdot d \quad = q - 2\Delta \geq \Delta \geq 1. \end{aligned}$$

□

Lemma 8.16. *Given a k-coloring ϕ_{old} of a graph with maximum degree Δ, in a single round, we can compute a k'-coloring ϕ_{new}, for $k' = O(\Delta^2 \log k)$. Furthermore, if $k \leq \Delta^3$, then the bound can be improved to $k' = O(\Delta^2)$.*

Proof Sketch. Follows from the existence of cover free families as proven in Lemma 8.14 and Lemma 8.15. Namely, each node v of old color $\phi_{old}(v) = q$ for $q \in \{1, dots, k\}$ will use the set $S_q \subseteq \{1, \ldots, k'\}$ in the cover free family as its *color-set*. Then, it sets its new color $\phi_{new}(v) = q'$ for a $q' \in S_q$ such that q' is not in the color-set of any of the neighbors. □

Proof of Theorem 8.12. The proof will be via iterative applications of Lemma 8.16. We start with the initial numbering of the vertices as a straightforward n-coloring. With one application of Lemma 8.16, we transform this into a $O(\Delta^2 \log n)$ coloring. With another application, we get a coloring with $O(\Delta^2(\log \Delta + \log\log n))$ colors. With another application, we get a coloring with $O(\Delta^2(\log \Delta + \log\log\log n))$ colors. After $O(\log^* n)$ applications, we get a coloring with $O(\Delta^2 \log \Delta)$ colors. At this point, we use one extra iteration, based on the second part of Lemma 8.16, which gets us to an $O(\Delta^2)$-coloring. □

8.3.2 Take 2: Kuhn-Wattenhofer Coloring Algorithm

In the previous section, we saw an $O(\log^* n)$-round algorithm for computing a $O(\Delta^2)$-coloring. In this section, we explain how to transform this into a $(\Delta+1)$-coloring. We will first see a very basic algorithm that performs this transformation in $O(\Delta^2)$ rounds. Then, we see how with the addition of a small but clever idea of [KW06], this transformation can be performed in $O(\Delta \log \Delta)$ rounds. As the end result, we get an $O(\Delta \log \Delta + \log^* n)$-round algorithm for computing a $(\Delta + 1)$-coloring.

Warm up: One-By-One color Reduction

Lemma 8.17. *Given a k-coloring ϕ_{old} of a graph with maximum degree Δ where $k \geq \Delta + 2$, in a single round, we can compute a $(k-1)$-coloring ϕ_{new}.*

Proof. For each node v such that $\phi_{old}(v) \neq k$, set $\phi_{new}(v) = \phi_{old}(v)$. For each node v such that $\phi_{old}(v) = k$, let node v set its new color $\phi_{new}(v)$ to be a color $q \in \{1, 2, \ldots, \Delta + 1\}$ such that q is not taken by any of the neighbors of u. Such a color q exists, because $v$has at most Δ neighbors. The resulting new coloring ϕ_{new} is a proper coloring. □

Theorem 8.18. *There is a deterministic distributed algorithm in the* LOCAL *model that colors any n-node graph G with maximum degree Δ using $\Delta + 1$ colors, in $O(\Delta^2 + \log^* n)$ rounds.*

Proof. First, compute an $O(\Delta^2)$-coloring in $O(\log^* n)$ rounds using the algorithm of Theorem 8.12. Then, apply the one-by-one color reduction of Lemma 8.17 for $O(\Delta^2)$ rounds, until getting to a $(\Delta + 1)$-coloring. □

Parallelized Color Reduction

Lemma 8.19. *Given a k-coloring ϕ_{old} of a graph with maximum degree Δ where $k \geq \Delta + 2$, in $O(\Delta\lceil\log(\frac{k}{\Delta+1})\rceil)$ rounds, we can compute a $(\Delta + 1)$-coloring ϕ_{new}.*

Proof. If $k \leq 2\Delta+1$, the lemma follows immediately from applying the one-by-one color reduction of Lemma 8.17 for $k - (\Delta + 1)$ iterations. Suppose that $k \geq 2\Delta + 2$. Bucketize the colors $\{1, 2, \ldots, k\}$ into $\lfloor\frac{k}{2\Delta+2}\rfloor$ buckets, each of size exactly $2\Delta + 2$, except for one last bucket which may have size between $2\Delta + 2$ to $4\Delta + 3$. We can perform color reductions in all buckets in parallel (why?). In particular, using at most $3\Delta + 2$ iterations of one-by-one color reduction of Lemma 8.17, we can recolor nodes of each bucket using at most $\Delta + 1$ colors. Considering all buckets, we now have at most $(\Delta + 1)\lfloor\frac{k}{2\Delta+2}\rfloor \leq k/2$ colors. Hence, we managed to reduce the number of colors by a 2 factor, in just $O(\Delta)$ rounds. Repeating this procedure for $\lceil\log(\frac{k}{\Delta+1})\rceil$ iterations gets us to a coloring with $\Delta + 1$ colors. The round complexity of this method is $O(\Delta\lceil\log(\frac{k}{\Delta+1})\rceil)$, because we have $\lceil\log(\frac{k}{\Delta+1})\rceil$ iterations and each iteration takes $O(\Delta)$ rounds. □

Theorem 8.20. *There is a deterministic distributed algorithm in the* LOCAL *model that colors any n-node graph G with maximum degree Δ using $\Delta + 1$ colors, in $O(\Delta \log \Delta + \log^* n)$ rounds.*

Proof. First, compute an $O(\Delta^2)$-coloring in $O(\log^* n)$ rounds using the algorithm of Theorem 8.12. Then, apply the parallelized color reduction

of Lemma 8.19 to transform this into a $(\Delta+1)$-coloring, in $O(\Delta \log \Delta)$ additional rounds. □

8.3.3 Take 3: Kuhn's Algorithm via Defective Coloring

In the previous section, we saw an algorithm that computes a $(\Delta+1)$-coloring in $O(\Delta \log \Delta + \log^* n)$ rounds. We now present an algorithm that improves this round complexity to $O(\Delta + \log^* n)$ rounds, based on a *Defective Coloring* method of Kuhn [Kuh09].

Theorem 8.21. *There is a deterministic distributed algorithm in the* LOCAL *model that colors any n-node graph G with maximum degree Δ using $\Delta+1$ colors, in $O(\Delta + \log^* n)$ rounds.*

It is worth noting that this linear-in-Δ round complexity remained as the state of the art, and it looked as if it might be the best possible for deterministic algorithms[4], until 2015. But then came a breakthrough of Barenboim [Bar15] which computed a $(\Delta+1)$-coloring in $O(\Delta^{3/4} \log \Delta + \log^* n)$ rounds. This was followed by a beautiful and very recent work of Fraigniaud, Heinrich, and Kosowski [FHK16], which improved the round complexity of $(\Delta+1)$-coloring further to $O(\Delta^{1/2} \log \Delta^{2.5} + \log^* n)$ rounds. We will not cover these recent advances in our lectures, but the papers should be already accessible and easy to follow, given what we have covered so far. What is the optimal round complexity for deterministic $(\Delta+1)$-coloring algorithms remains an intriguing and long-standing open problem – an ultimate goal would be to deterministically compute a $(\Delta+1)$ coloring in $\text{poly}\log(n)$ rounds.

Definition 8.22. *For a graph $G = (V, E)$, a color assignment $\phi : V \to \{1, 2, \ldots, k\}$ is called a d-defective k-coloring if the following property is satisfied: for each color $q \in \{1, 2, \ldots, k\}$, the subgraph of G induced by vertices of color q has maximum degree at most d. In other words, in a d-defective coloring, each node v has at most d neighbors that have the same color as v.*

Notice that a standard proper k-coloring — where no two adjacent nodes have the same color — is simply a 0-defective k-coloring.

[4] Randomized algorithms have a very different story: We will see a simple $O(\log n)$-round randomized $\Delta+1$ coloring algorithm in the next sections, and we will also touch upon further improvements on the randomized track.

Lemma 8.23. *Given a d-defective k-coloring ϕ_{old} of a graph with maximum degree Δ, in a single round, we can compute a d'-defective k'-coloring ϕ_{new}, for $k' = O\left(\left(\frac{\Delta-d}{d'-d+1}\right)^2 \log k\right)$.*

Proof. Proof to be added here. See pages 10 to 13 of this handwritten lecture note[5], for now. □

In a sense, this color reduction reduces the number of colors significantly, while increasing the defect only slightly.

Proof of Theorem 8.21. First, compute an 0-defective $C\Delta^2$-coloring, in $O(\log^* n)$-rounds, using the algorithm of Theorem 8.12. Here, C is a sufficiently large constant, as needed in Theorem 8.12. We will now see how to improve this to $\Delta + 1$ colors.

The method is recursive. Let $T(\Delta)$ denote the complexity of $(\Delta+1)$-coloring graphs with maximum degree Δ, given the initial $O(\Delta^2)$-coloring. to perform a recursive method, we would like to decompose the graph into a few subgraphs of degree at most $\Delta/2$ and proceed recursively. In the following, we explain how to do this, using defective coloring as a tool.

We start with an $(C\Delta^2)$-coloring, as computed before, for a large enough constant $C > 0$. Then, we use one iteration of Lemma 8.23 to transform this into a $(\frac{\Delta}{\log \Delta})$-defective $O(\log^3 \Delta)$-coloring. Then, use another iteration of Lemma 8.23 to transform this into a $(\frac{\Delta}{\log\log \Delta})$-defective $O(\log^3 \log \Delta)$-coloring. One more iteration gets us to a $(\frac{\Delta}{\log\log\log \Delta})$-defective $O(\log^3 \log\log \Delta)$-coloring. After $O(\log^* \Delta)$ iterations, we get a $(\frac{\Delta}{2})$-defective k''-coloring for $k'' = O(1)$.

Now, each of these k'' color classes induces a subgraph with maximum degree $\Delta/2$. That is, we have decomposed the graph G into $O(1)$ disjoint subgraphs $G_1, G_2, \ldots, G_{O(1)}$, each with maximum degree at most $\Delta/2$. Hence, by recursion, we can color each of them using $\Delta/2+1$ colors, all in parallel, in $T(\Delta/2)$ rounds. Formally, to be able to invoke the recursion, we should provide to each G_i an initial coloring with $C(\Delta/2)^2$ coloring. Notice that this can be computes easily in at most $O(\log^* n)$ time, using Linial's recoloring method as covered in Theorem 8.12. This allows us to invoke the recursive coloring procedure, and get a $\Delta/2+1$ coloring for each G_i. When paired with the corresponding subgraph G_i index i, these color form an

[5] http://people.csail.mit.edu/ghaffari/DGA14/Notes/L02.pdf

$\Delta/2 \cdot O(1) = O(\Delta)$ coloring of the whole graph. This can be transformed into a $\Delta + 1$ coloring, in $O(\Delta)$ extra rounds, using the one-by-one color reduction method of Lemma 8.17.

As a result, we get the recursion

$$T(\Delta) = O(\log^* \Delta) + T(\Delta/2) + O(\Delta).$$

Recalling the Master theorem for recursions [CLRS01], we easily see that the answer of this recursion is $T(\Delta) = O(\Delta)$. Hence, including the initial $O(\log^* n)$-rounds, this is an overall round complexity of $O(\Delta + \log^* n)$. □

8.4 Network Decomposition

In the previous sections, we zoomed in on one particular problem, graph coloring, and we discussed a number of algorithms for it. In this section and the next two, we will discuss a method that is far more general and can be used for a wide range of *local* problems. The key concept in our discussion will be *network decompositions* first introduced by [ALGP89], also known as *low-diameter graph decomposition*[LS91].

8.4.1 Definition and Applications

Let us start with defining this concept.

Definition 8.24. *(Weak Diameter Network Decomposition) Given a graph* $G = (V, E)$*, a* $(\mathcal{C}, \mathcal{D})$ *weak diameter network decomposition of* G *is a partition of* G *into vertex-disjoint graphs* G_1*,* G_2*, . . . ,* $G_{\mathcal{C}}$ *such that for each* $i \in \{1, 2, \ldots, \mathcal{C}\}$*, we have the following property: the graph* G_i *is made of a number of vertex-disjoint and mutually non-adjacent clusters* X_1*,* X_2*, . . . ,* X_ℓ*, where each two vertices* $v, u \in X_j$ *have distance at most* D *in graph* G*. We note that we do not bound the number* ℓ*. We refer to each subgraph* G_i *as one* block *of this network decomposition.*

Definition 8.25. *(Strong Diameter Network Decomposition) Given a graph* $G = (V, E)$*, a* $(\mathcal{C}, \mathcal{D})$ *strong diameter network decomposition of* G *is a partition of* G *into vertex-disjoint graphs* G_1*,* G_2*, . . . ,* $G_{\mathcal{C}}$ *such that for each* $i \in \{1, 2, \ldots, \mathcal{C}\}$*, we have the following property: each connected component of* G_i *has diameter at most* D*.*

Notice that a strong diameter network decomposition is also a weak diameter network decomposition. For

Network decompositions can be used to solve a wide range of *local* problems. To see the general method in a concrete manner, let us go back to our beloved $(\Delta + 1)$-coloring problem.

Theorem 8.26. *Provided an* $(\mathcal{C}, \mathcal{D})$ *weak-diameter network decomposition of a graph* G*, we can compute a* $\Delta + 1$ *coloring of* G *in* $O(CD)$ *rounds.*

Proof. We will color graphs G_1, G_2, . . . , $G_{\mathcal{C}}$ one by one, each time considering the coloring assigned to the previous subgraphs. Suppose that vertices of graphs graphs G_1, G_2, . . . , G_i are already colored using colors in

$\{1, 2, \ldots, \Delta + 1\}$. We explain how to color G_{i+1} in $O(D)$ rounds. Consider the clusters X_1, X_2, $\ldots$, X_ℓ of G_{i+1} and notice their two properties: (1) they are mutually non-adjacent, (2) for each cluster X_j, its vertices are within distance D of each other (where distances are according to the base graph G). For each cluster X_j, let node $v_j \in X_j$ who has the maximum identifier among nodes of X_j be the leader of X_j. Notice that leaders of clusters X_1, X_2, $\ldots$, X_ℓ can be identified in $O(D)$ rounds (why?). Then, let v_j aggregate the topology of the subgraph induced by X_j as well as the colors assigned to nodes adjacent to X_j in the previous graphs G_1, G_2, $\ldots$, G_i. This again can be done in $O(D)$ rounds, thanks to the fact that all the relevant information is within distance $D + 1$ of v_j. Once this information is gathered, node v_j can compute a $(\Delta + 1)$-coloring for vertices of X_j, while taking into account the colors of neighboring nodes of previous graphs, using a simple greedy procedure. Then, node v_j can report back these colors to nodes of X_j. This will happen for all the clusters X_1, X_2, $\ldots$, X_ℓ in parallel, thanks to the fact that they are non-adjacent and thus, their coloring choices does not interfere with each other. □

8.4.2 Randomized Algorithm for Network Decomposition

Theorem 8.27. *There is a randomized* LOCAL *algorithm that computes a $(\mathcal{C}, \mathcal{D})$ weak-diameter network decomposition of any n-node graph G, for $\mathcal{C} = O(\log n)$ and $\mathcal{D} = O(\log n)$, in $O(\log^2 n)$ rounds, with high probability*[6].

We remark that, as we will see in Exercise 8.8, the round complexity of this construction can be improved to $O(\log n)$ rounds. On the other hand, as we see in Exercise 8.9, the two key parameters $\mathcal{C}$ and $\mathcal{D}$ are nearly optimal and one cannot improve them simultaneously and significantly.

Network Decomposition Algorithm: Suppose that we have already computed subgraphs G_1, $\ldots$, G_i so far. We now explain how to compute a subgraph $G_{i+1} \subseteq G \setminus \left(\cup_{j=1}^{i} G_j\right)$, in $O(\log n)$ rounds, which would satisfy the properties of one block of a weak diameter network decomposition.

Let each node v pick a random radius r_u from an geometric distribution with parameter ε, for a desired (free parameter) constant $\varepsilon \in (0, 1)$. That

[6] Throughout, we will use the phrase *with high probability to indicate that an event happens with probability at least $1 - \frac{1}{n^c}$, for a desirably large but fixed constant $c \geq 2$.*

is, for each integer $y \geq 1$, we have $\Pr[r_u = y] = \varepsilon(1-\varepsilon)^{y-1}$. We will think of the vertices within distance r_u of u as the *ball of node* u. Now for each node v, let $\mathit{Center}(v)$ be the node u^* among nodes u such that $\mathrm{dist}_G(u,v) \leq r_u$ that has the smallest identifier. The is, $\mathit{Center}(v) = u^*$ is the smallest-identifier node whose ball contains v. Define the clusters of G_i by letting all nodes with the same center define one cluster, and then discarding nodes who are at the boundary of their cluster. That is, any node v for which $\mathrm{dist}_G(v,u) = r_u$ where $u = \mathit{Center}(v)$ remains unclustered.

There are two properties to prove: one that the clusters have low diameter, and second, that after $\mathcal{C}$ iterations, all nodes are clustered. In the following two lemmas, we argue that with high probability, each cluster has diameter $O(\log n/\varepsilon)$ and after $\mathcal{C} = O(\log_{1/\varepsilon} n)$ iterations, all nodes are clustered.

Lemma 8.28. *With high probability, the maximum cluster diameter is at most* $O(\log n/\varepsilon)$*. Hence, this clustering can be computed in* $O(\log n/\varepsilon)$ *rounds, with high probability.*

Proof. The proof is simple and is left as an exercise. □

Lemma 8.29. *For each node* v*, the probability that* v *is not clustered — that* v *is on the boundary of its supposed cluster and thus it gets discarded — is at most* ε*.*

Proof. Notice that

$$\begin{aligned} &\Pr\ [v \textit{ is not clustered }] = \\ \sum_{u \in V} &\Pr\ [v \textit{ is not clustered } \mid \mathit{Center}(v) = u] \cdot \Pr[\mathit{Center}(v) = u] \end{aligned}$$

For each vertex u, let $\mathit{before}(u)$ denote the set of all vertices whose identifier is less than that of u. Define the following events

- $\mathcal{E}_1 = \Big(r_u = \mathrm{dist}_G(v,u)\Big)$.
- $\mathcal{E}_2 = \Big(r_u \geq \mathrm{dist}_G(v,u)\Big)$.
- $\mathcal{E}_3 = \Big(\forall u' \in \mathit{before}(u), r_{u'} < \mathrm{dist}_G(v,u')\Big)$.

We have

$$\begin{aligned}
&\Pr\ [v \textit{ is not clustered } | \operatorname{Center}(v) = u] \\
&= \Pr[\mathcal{E}_1 \cap \mathcal{E}_3 \,|\, \mathcal{E}_2 \cap \mathcal{E}_3] \\
&= \frac{\Pr[\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3]}{\Pr[\mathcal{E}_2 \cap \mathcal{E}_3]} \\
&= \frac{\Pr[\mathcal{E}_1 \cap \mathcal{E}_3]}{\Pr[\mathcal{E}_2 \cap \mathcal{E}_3]} \\
&= \frac{\Pr[\mathcal{E}_3] \cdot \Pr[\mathcal{E}_1|\mathcal{E}_3]}{\Pr[\mathcal{E}_3] \cdot \Pr[\mathcal{E}_2|\mathcal{E}_3]} \\
&= \frac{\Pr[\mathcal{E}_1]}{\Pr[\mathcal{E}_2]} = \varepsilon,
\end{aligned}$$

where in the penultimate equality, we used the property that the event $\mathcal{E}_3$ is independent of events $\mathcal{E}_1$ and $\mathcal{E}_2$, and the last equality follows from the probability distribution function of the exponential distribution (recall that this is exactly the *memoryless property* of the exponential distribution). Hence, we can now go back and say that

$$\begin{aligned}
&\Pr\ [v \textit{ is not clustered }] \\
&= \sum_{u \in V} \Pr[v \textit{ is not clustered } | \operatorname{Center}(v) = u] \cdot \Pr[\operatorname{Center}(v) = u] \\
&= \sum_{u \in V} \varepsilon \cdot \Pr[\operatorname{Center}(v) = u] = \varepsilon.
\end{aligned}$$

□

Corollary 8.30. *After $C = O(\log_{1/\varepsilon} n)$ iterations, all nodes are clustered, with high probability.*

8.4.3 Deterministic Algorithm for Network Decomposition

In this section, we explain a deterministic LOCAL algorithm that achieves the following network decomposition.

Theorem 8.31. *There is a deterministic* LOCAL *algorithm that computes a (C, D) strong-diameter network decomposition of any n-node graph G, for $C = 2^{O(\sqrt{\log n})}$ and $D = 2^{O(\sqrt{\log n})}$, in $2^{O(\sqrt{\log n})}$ rounds.*

In the exercises, we will see how to use this algorithm to compute an $(O(\log n), O(\log n))$ strong-diameter network decomposition, still in the same running time of $2^{O(\sqrt{\log n})}$.

The result of Theorem 8.31 is due to [PS92]. Here, we will present a slightly weaker but simple result, due to [ALGP89], that computes the following:

Theorem 8.32. *There is a deterministic* LOCAL *algorithm that computes a $(\mathcal{C}, \mathcal{D})$ strong-diameter network decomposition of any n-node graph G, for $\mathcal{C} = 2^{O(\sqrt{\log n \log\log n})}$ and $\mathcal{D} = 2^{O(\sqrt{\log n \log\log n})}$, in $2^{O(\sqrt{\log n \log\log n})}$ rounds.*

Towards this goal, we first need to introduce a helper tool, *ruling sets*, and present an efficient algorithm for computing them.

Ruling Sets

Definition 8.33. *Given a graph $G = (V, E)$ and a set $W \subseteq V$, an (α, β)-ruling set of W in G is a subset $S \subseteq W$ such that the following two properties are satisfied:*

(A) For each two vertices $v, u \in S$, we have $\text{dist}_G(v, u) \geq \alpha$

(B) For each vertex $v \in W$, there exists a vertex $u \in S$ such that $\text{dist}_G(v, u) \leq \beta$.

For instance, if $W = V$, a maximal independent set $S \subset V$ is simply a $(2, 1)$-ruling set. Moreover, letting G^k be the supergraph of G where each two vertices with distance at most k are connected, a set $S \subset V$ that is a maximal independent set in G^k is actually a $(k+1, k)$-ruling set in G.

Lemma 8.34. *Given a graph $G = (V, E)$ and a set $W \subseteq V$, there is a deterministic* LOCAL *algorithm that computes a $(k, k \log n)$-ruling set of W in G, in $O(k \log n)$ rounds.*

Proof. The algorithm is recursive. Let W_0 be the set of vertices whose identifier ends in a 0 bit, and let W_1 be the set of vertices whose identifier ends in a 1 bit. Recursively compute $(k, k(\log n - 1))$-ruling sets S_0 and S_1 of W_0 and W_1, respectively, in $O(k(\log n - 1))$ rounds. Notice that the parameter of the recursion is the length of the binary representation of the

identifiers. Now, let $S = S_0 \cup S_1^*$ where S_1^* is the set of all vertices in S_1 who do not have any S_0-vertex within their distance k. Note that S can be computed from S_0 and S_1, in k rounds. One can see that S is a (k, β)-ruling set of W for $\beta = k(\log n - 1) + k = k \log n$ (why?). □

Constructing the Network Decomposition

Here, we describe the network decomposition algorithm that establishes Theorem 8.32. The construction uses a free parameter d, which we will set later on, in order to optimize some trade off.

The construction is iterative, and works in $\log_d n$ similar iterations. Let us start with the description of the first iteration.

Partition vertices into two classes, high-degree vertices H whose degree is at least d, and low-degree vertices L whose degree is at most $d - 1$. Compute a $(3, O(\log n))$-ruling set S of the H vertices in G, in $O(\log n)$ rounds, using Lemma 8.34. Now for the set of all vertices in V who have at least one S-vertex within their distance $O(\log n)$, bundle them around the closest S-vertex, breaking ties based on the identifiers. Hence, each bundle induces a subgraph of radius at most $O(\log n)$ and has at least $d+1$ vertices (why?). Furthermore, the set of nodes that remain unbundled induces a graph with maximum degree at most $d - 1$.

Compute a d-coloring of the subgraph induced by unbundled vertices in $O(d + \log^* n)$ rounds, using the algorithm we discussed in Theorem 8.21. Each of these d colors will be one of the subgraphs G_i in our network decomposition's partition, and each vertex of each color class is simply its own cluster. Note that clearly the clusters of the same class are non-adjacent.

We are now essentially done with the first iteration. To start the second iteration, we will switch to a new graph $\mathcal{G}_2$, defined as follows. We will imagine contracting each bundle into a single new node in $\mathcal{G}_2$. We call these the level-2 nodes. In $\mathcal{G}_2$, two level-2 nodes are connected if their bundles include adjacent vertices in $G = \mathcal{G}_1$. Notice that one round of communication on $\mathcal{G}_2$ can be performed in $O(\log n)$ rounds of communication on the base graph $G = \mathcal{G}_1$, simply because each of the bundles has diameter $O(\log n)$.

Now the construction for level-2 works similar to that of level-1, but on the graph $\mathcal{G}_2$. Again, we partition level-2 vertices into high and low-degree,

by comparing to threshold d. Then, we compute a $(3, O(\log n))$-ruling set of high-degree level-2 vertices, with respect to the graph $\mathcal{G}_2$. We then bundle nodes around these ruling set vertices into bundles of diameter $O(\log n)$ in $\mathcal{G}_2$. Afterward, we color the level-2 vertices that remain unbundled using d colors. Notice that each level-2 vertex is actually one of the bundles in level-1, and hence, by this coloring, we color all the vertices of that bundle using the same color. Therefore, this is not a proper color of $\mathcal{G}_1$, but each color of this level does induce a block satisfying the conditions of network decomposition (why?).

Afterward, we repeat to level 3, by contracting the bundles of level-2, producing the graph $\mathcal{G}_3$ as a result, and continuing the process on $\mathcal{G}_3$.

Lemma 8.35. *Each level i bundle has at least $(d+1)^{i-1}$ vertices. Therefore, $\log_d n$ levels of iteration suffice.*

Proof. Follows by a simple induction and observing that each level i bundle includes at least $d+1$ level $(i-1)$ bundles. □

Lemma 8.36. *Each level i bundle has diameter $O(\log n)^i$ in G. Hence, the algorithm for level i can be performed in $O(\log n)^i \cdot O(d + \log^* n)$ rounds.*

Proof. Follows by a simple induction and observing that each level i bundle is a graph of diameter $O(\log n)$ on the level $(i-1)$ graph $\mathcal{G}_{i-1}$. □

Corollary 8.37. *Overall the algorithm takes at most $d \cdot O(\log n)^{\log_d n}$ rounds. At the end of all levels, we get a $(\mathcal{C}, \mathcal{D})$ strong-diameter network decomposition for $\mathcal{C} = d \log_d n$ and $\mathcal{D} = O(\log n)^{\log_d n}$. Setting $d = 2^{O(\sqrt{\log n \log\log n})}$, we get round complexity of $d \cdot O(\log n)^{\log_d n} = 2^{O(\sqrt{\log n \log\log n})}$ and $\mathcal{C} = d \log_d n = 2^{O(\sqrt{\log n \log\log n})}$ and $\mathcal{D} = O(\log n)^{\log_d n} = 2^{O(\sqrt{\log n \log\log n})}$.*

8.5 Maximal Independent Set

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

8.5.1 Definition and Reductions

Let us start with recalling the definition of MIS:

Definition 8.38. *Given a graph* $G = (V, E)$*, a set of vertices* $S \subseteq V$ *is called a* Maximal Independent Set (MIS) *if it satisfies the following two properties:*

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

(2) the set S *is* maximal *— with regard to independence — meaning that for each node* $v \notin S$*, there exists a neighbor* u *of* v *such that* $u \in S$*.*

Lemma 8.39. *Given a* LOCAL *algorithm* $\mathcal{A}$ *that computes an MIS on any* N*-node graph in* $T(N)$ *rounds, there is a* LOCAL *algorithm* $\mathcal{B}$ *that computes a* $\Delta + 1$ *coloring of any* n*-node graph with maximum degree* Δ *in* $T(n(\Delta + 1))$ *rounds.*

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

Proof of Lemma 8.39. Let G be an arbitrary n-node graph with maximum degree Δ, for which we would like to compute a $(\Delta + 1)$-coloring.

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

Run the algorithm $\mathcal{A}$ on H. The resulting MIS produces a maximal independent set S. For each node $v \in G$, the color of v will the the number

i such that the $v_i \in S$. Clearly, each node receives at most one color, as at most one copy v_i of $v \in G$ can be in S. However, one can see that each node $v \in G$ receives actually exactly one color. The reason is that each neighboring node $u \in G$ can have at most one of its copies in S. Node v has at most Δ neighbors u and $\Delta + 1$ copies v_i. Hence, there is at least one copy v_i of v for which no adjacent copy u_i of neighboring vertices $u \in G$ is in the set S. By maximality of S, we must have $v_i \in S$. □

Unfortunately, the best deterministic algorithms for computing an MIS remain slow: An $O(\Delta + \log^* n)$-round algorithm follows by computing a $\Delta + 1$ coloring and then iterating through the colors and greedily adding vertices to the MIS. A $2^{O(\sqrt{\log n})}$-round algorithm follows from the network decomposition method, by using the decomposition of Theorem 8.31 along with the general method of Theorem 8.26. Improving these bounds remains a long-standing open problem. In contrast, there is an extremely simple randomized algorithm that computes an MIS in merely $O(\log n)$ rounds, as we see next.

8.5.2 Luby's MIS Algorithm

We now present the so called Luby's algorithm — presented essentially simultaneously and independently by Luby [Lub85] and Alon, Babai, and Itai [ABI86] — that computes an MIS in $O(\log n)$ rounds, with high probability.

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

- In the first round, each node v picks a random real number $r_v \in [0, 1]$ and sends it to its neighbors[7]. Then, node v joins the (eventual) MIS set S if and only if node v has a strict local maxima, that is, if $r_v > r_u$ for all neighbors u of v.
- In the second round, if a node v joined the MIS, then it informs its neighbors and then, node v and all of its neighbors get removed from the problem. That is, they will not participate in the future iterations.

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

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

Theorem 8.40. *Luby's Algorithm computes a maximal independent set in $O(\log n)$ rounds, with high probability.*

Proof. Consider an arbitrary iteration i and suppose that the graph induced on the remaining vertices is G_i, which has m_i edges. In the following, we will argue that the graph G_{i+1} which will remain for the next iteration has in expectation at most $m_i/2$ edges. By a repeated application of this, we get that the graph that remains after $4 \log n$ iterations has at most $m_0/2^{4 \log n} \leq 1/n^2$ edges. Hence, by Markov's inequality, the probability that the graph $G_{4 \log n}$ has at least 1 edge is at most $1/n^2$. That is, with high probability, $G_{4 \log n}$ has no edge left, and thus, the algorithm finishes in $4 \log n + 1$ iterations, with high probability.

Given the above outline, what remains is to prove that

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

For that, let us consider an edge $e = \{u, v\}$ and a neighbor w of u, as depicted in Figure 8.1. If w has the maximum number among $N(w)$, the set of neighbors of w in the remaining graph, then w joins the MIS and hence nodes w and u and thus also the edge $e = \{u, v\}$ get removed. In this case, we say node w killed edge $e = \{u, v\}$. However, unfortunately, there is a possible double counting in that an edge e might be killed by many neighbors w_1, w_2, w_3, etc, and thus we cannot lower bound the number of removed edges easily by counting how many edges are killed in this manner.

To circumvent this, we make a slight adjustment: we say that node w single-handedly kills $e = \{u, v\}$ (from the side of u) if r_w is the maximum random number among those of nodes in $N(w) \cup N(u)$. Notice that this limits the number of double-counting of an edge being killed to 2, meaning that at most one node w might single-handedly kill $e = \{u, v\}$ (from the side of u) and at most one node w' might single-handedly kill $e = \{u, v\}$ (from the side of v). Hence, we can lower bound the number of removed edges by estimating the number of single-handedly killed edges and dividing that by 2. We next make this concrete.

Consider two neighboring nodes w and u. The probability that w has the maximum among $N(w) \cup N(u)$ is at least $\frac{1}{d(w)+d(u)}$. In that case, node

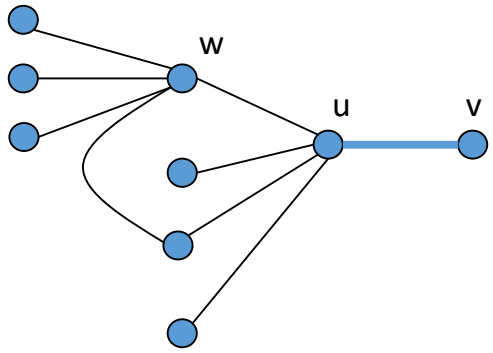

Figure 8.1: Node w killing edge $e = \{u, v\}$

w single-handedly kills $d(u)$ edges incident on u (from the side of u). Similarly, the probability that u has the maximum among $N(w) \cup N(u)$ is at least $\frac{1}{d(w)+d(u)}$, and in that case, u single-handedly kills $d(w)$ edges incident on w (from the side of w). Therefore, by linearity of expectation, and given that we are counting each edge killed at most twice (once from each endpoint), we cay say the following: the total expected number of removed edges is at least

$$\mathbb{E}[m_i - m_{i+1}] \geq \sum_{\{w,u\} \in E_i} \left(\frac{d(u)}{d(w) + d(u)} + \frac{d(w)}{d(w) + d(u)} \right) / 2 = m_i / 2.$$

□

8.6 Sublogarithmic-Time Randomized Coloring

Here, we explain a randomized algorithm that achieves the following:

Theorem 8.41. *There is a randomized* LOCAL *algorithm that computes a* $\Delta(1+\varepsilon)$*-coloring in* $O(\sqrt{\log n})$ *rounds*[8]*, for any constant* $\varepsilon > 0$*, with high probability.*

[8] We remark that a much faster algorithm, with a round complexity of $2^{O(\sqrt{\log \log n})}$, is known. We will cover that result in the next sections.

We present the algorithm in two parts: we first explain the algorithm assuming that $\Delta \leq 2^{\sqrt{\log n}}$, and then we extend the algorithm to larger Δ using a small additional step.

8.6.1 The algorithm for low-degree graphs

Theorem 8.42. *There is a randomized algorithm that computes a $\Delta(1+\varepsilon)$ coloring of the graph in $\Theta(\sqrt{\log n}/\varepsilon)$ rounds, for any constant $\varepsilon > 0$, assuming[9] that $\Delta \leq 2^{\sqrt{\log n}}$.*

Suppose that each vertex has degree at most $2^{\sqrt{\log n}}$. Consider running the following algorithm for $\Theta(\sqrt{\log n}/\varepsilon)$ iterations. In each iteration, each node v picks a random color among the colors not previously taken by any of its neighbors. If no neighbor of v picked the same color, then v takes this color as its permanent coloring, and gets removed from the problem. The neighbors update their palette of remaining colors, by removing the colors taken by the colored neighbors.

In Lemma 8.43 below, we show that after $\Theta(\sqrt{\log n}/\varepsilon)$ iterations for any $\varepsilon \in (0, 1]$, with high probability, each connected component of the remaining graph has at diameter at most $\Theta(\sqrt{\log n}/\varepsilon)$. Hence, each of these components can be colored in $\Theta(\sqrt{\log n}/\varepsilon)$ additional steps, deterministically. Therefore, once we prove Lemma 8.43, we have essentially completed the proof of Theorem 8.42.

Lemma 8.43. *After $\Theta(\sqrt{\log n}/\varepsilon)$ iterations, with high probability, each connected component in the subgraph induced by the remaining nodes has at diameter at most $\Theta(\sqrt{\log n}/\varepsilon)$.*

Proof. We show that with high probability, no path of length $C\sqrt{\log n/\varepsilon}$ can have all of its vertices remain, for a large enough constant $C > 3$.

We start with some simple observations. Notice that in every iteration, the palette size of each node is at least a $(1 + \varepsilon)$ factor larger than its number of remaining neighbors, i.e., its degree in the remaining graph. Hence, in each iteration, each node gets colored with probability at least $\varepsilon/(1 + \varepsilon) \geq \varepsilon/2$, even independent of its neighbors (why?).

Consider an arbitrary path $P = v_0, v_1, \ldots, v_\ell$ where $\ell = \Theta(\sqrt{\log n}/\varepsilon)$. In each iteration, each of these vertices gets removed with probability at least

[9] As mentioned before, we will soon see how to remove this assumption.

ε, regardless of the coloring of the other vertices. Hence, the probability that all these nodes remain for $k = C\sqrt{\log n}/\varepsilon$ iterations is at most

$$(1-\varepsilon/2)^{\ell k} \leq \exp(-\ell k\varepsilon/2) \leq \exp(-C^2 \log n/(2\varepsilon)).$$

Now, there are at most $n\Delta^\ell$ ways for choosing the path P, because there are n choices for the starting point and then Δ choices for each next hop. Hence, by a union bound, the probability that any such path remains is at most

$$\begin{aligned} & n \cdot \Delta^\ell \cdot \exp(-C \log n/(2\varepsilon)) \\ = \; & \exp(\log n + \ell \log \Delta - C^2 \log n/(2\varepsilon)) \\ \leq \; & \exp(\log n + C\sqrt{\log n}/\varepsilon \cdot \sqrt{\log n} - C^2 \log n/(2\varepsilon)) \\ = \; & \exp(\log n + C \log n/\varepsilon - C^2 \log n/(2\varepsilon)) \leq 1/n^C. \end{aligned}$$

□

8.6.2 The extension to high-degree graphs

We now see how to extend Theorem 8.42 to higher degree graphs, using one extremely simple step.

Theorem 8.44. *There is a randomized algorithm that computes a $\Delta(1+\varepsilon)$ coloring of the graph in $\Theta(\sqrt{\log n}/\varepsilon)$ rounds, for any constant $\varepsilon > 0$.*

Suppose that $\Delta \geq 2^{\sqrt{\log n}}$, as otherwise Theorem 8.42 suffices. Partition G into $k = \alpha\varepsilon^2\Delta/\log n$ vertex-disjoint subgraphs $G_1, G_2, G_3, \ldots, G_k$ by putting each vertex in one of these subgraphs at random. Here, α is a sufficiently small positive constant $\alpha > 0$. In Lemma 8.45, we argue that, with high probability, each subgraph has degree at most $\frac{\Delta}{k}(1+\varepsilon/3) = O(\log n)$. This will be by a simple application of the Chernoff bound. Hence, it can be colored using $\frac{\Delta}{k}(1+\varepsilon/3)(1+\varepsilon/3) \leq \frac{\Delta}{k}(1+\varepsilon)$ colors, via the method of Theorem 8.42, in $\Theta(\sqrt{\log n}/\varepsilon)$ rounds. We use different colors for different subgraphs, and color them all in parallel. Hence, overall, we get a coloring with $k \cdot \frac{\Delta}{k}(1+\varepsilon) = \Delta(1+\varepsilon)$ colors, in $\Theta(\sqrt{\log n}/\varepsilon)$ rounds.

Lemma 8.45. *With high probability, each subgraph G_i has maximum degree at most $\frac{\Delta}{k}(1+\varepsilon/3)$.*

The lemma follows in a straightforward manner from the Chernoff bound, and a union bound. The Chernoff bound is among the most basic concentration of measure tools. In a rough sense, it shows that the sum of independent (indicator) random variables has a distribution well-concentrated around its expected value. More concretely, the probability that this sum deviates significantly from its expected value is exponentially small in the expected value. The mathematical statement is as follows:

Theorem 8.46. *(Chernoff Bound) Suppose* $X_1, X_2, \ldots, X_n$ *are independent random variables taking values in* $\{0,1\}$. *Let* $X = \sum_{i=1}^{\ell} X_i$ *denote their sum and let* $\mu = \mathbb{E}[X]$ *denote the sum's expected value. For any* $\delta > 0$, *we have*

$$\Pr[X \notin [\mu(1-\delta), \mu(1+\delta)] \leq 2e^{-\delta^2\mu/3}.$$

Proof of Lemma 8.45. Consider each node $v \in G$. Let $X_1, X_2, \ldots X_\Delta$ be the indicator random variables of whether the i^{th} neighbor of v picks the same subgraph as v does. Let $X = \sum_{i=1}^{\ell} X_i$ and $\mu = \mathbb{E}[X]$. Notice that $\mu = \Delta/k$. Given that $k = \alpha\varepsilon^2\Delta/\log n$, we get that $\mu \geq \frac{\log n}{\alpha\varepsilon^2}$. Hence, by Chernoff bound, we have

$$\Pr[X \geq \mu(1+\varepsilon/3)] \leq 2e^{-\varepsilon^2\mu/18} \leq e^{1-\log n/(18\alpha)} \leq 1/n^3,$$

where the last inequality holds for small enough α, e.g., $\alpha = 0.01$.

Now, we know that the probability of one node v having a degree (in its own subgraph) higher than the desired threshold $\frac{\Delta(1+\varepsilon/3)}{k}$ is at most $1/n^3$. By a union bound over all nodes v, we get that the probability of having such a node is at most $1/n^2$. In other words, with high probability, each node has degree at most $\frac{\Delta(1+\varepsilon/3)}{k}$ in its own subgraph. □

8.7 Exercises

Exercise 8.1. *In Lemma 8.5, we saw a single-round algorithm for reducing the number of colors exponentially. Here, we discuss another such method, which transforms any k-coloring of any rooted-tree to a $2\log k$-coloring, so long as $k \geq C_0$ for a constant C_0.*

The method works as follows. Let each node u send its color $\phi_{old}(u)$ to its children. Now, each node v computes its new color $\phi_{new}(v)$ as follows: Consider the binary representation of $\phi_{old}(v)$ and $\phi_{old}(u)$, where u is the parent of v. Notice that each of these is a $\log_2 k$-bit value. Let i_v be the smallest index i such that the binary representations of $\phi_{old}(v)$ and $\phi_{old}(u)$ differ in the i^{th} bit. Let b_v be the i_v^{th} bit of $\phi_{old}(v)$. Define $\phi_{new}(v) = (i_v, b_v)$. Prove that $\phi_{old}(v)$ is well-defined, and that it is a proper $(2\log k)$-coloring.

Exercise 8.2. *In Theorem 8.4, we saw that on a tree graph where each node knows its parent, a 3-coloring can be computed in $O(\log^* n)$ rounds. This result heavily relies on each node knowing its parent, and no such result is true in unrooted trees, that is, when nodes do not know which neighbor is their parent.*

In particular, it is known that there exists Δ-regular graphs with girth — that is, the length of the shortest cycle — being at least $\Omega(\log_\Delta n)$ and chromatic number at least[10] $\Omega(\Delta/\log\Delta)$ [Bol78]. Use this existence to prove that any deterministic $o(\log_d n)$-round algorithm for coloring trees requires at least $\Omega(\Delta/\log\Delta)$ colors.

Exercise 8.3. *Here, we use the concept of cover free families, as defined in Definition 8.13, to obtain an encoding that allows us to recover information after superimposition. That is, we will be able to decode even if k of the codewords are* superimposed *and we only have the resulting bit-wise OR.*

More concretely, we want a function $Enc : \{0,1\}^{\log n} \to \{0,1\}^m$ — that encodes n possibilities using m-bit strings for $m \geq \log_2 n$ — such that the following property is satisfied: $\forall S, S' \subseteq \{1, ..., n\}$ such that $|S| \leq k$ and $|S'| \leq k$, we have that $\vee_{i\in S} Enc(i) \neq \vee_{i\in S'} Enc(i)$. Here $\vee$ denotes the bit-wise OR operation.

[10] This lower bound on the chromatic number is tight as triangle-free graphs with maximum degree Δ have chromatic number $O(\log\Delta/\log\log\Delta)$ [Kim95, Jam11, PS15].

Present such an encoding function, with a small m, that depends on n and k. What is the best m that you can achieve?

Exercise 8.4. *This exercise has two parts:*

(A) Design a single-round algorithm that transforms any given k-coloring of a graph with maximum degree Δ into a k'-coloring for $k' = k - \lceil \frac{k}{2(\Delta+1)} \rceil$, assuming $k' \geq \Delta + 1$.

(B) Use repetitions of this single-round algorithm, in combination with the $O(\log^* n)$-round $O(\Delta^2)$-coloring of Theorem 8.12, to obtain an $O(\Delta \log \Delta + \log^* n)$-round $(\Delta+1)$-coloring algorithm.

Exercise 8.5. *Here, we see yet another deterministic method for computing a $(\Delta+1)$-coloring in $O(\Delta \log \Delta + \log^* n)$ rounds. First, using Theorem 8.12, we compute an $O(\Delta^2)$-coloring ϕ_{old} in $O(\log^* n)$ rounds. What remains is to transform this into a $(\Delta+1)$-coloring, in $O(\Delta \log \Delta)$ additional rounds.*

The current $O(\Delta^2)$-coloring ϕ_{old} can be written using $C \log \Delta$ bits, assuming a sufficiently large constant C. This bit complexity will be the parameter of our recursion. Partition G into two vertex-disjoint subgraphs G_0 and G_1, based on the most significant bit in the color ϕ_{old}. Notice that each of G_0 and G_1 inherits a coloring with $C \log \Delta - 1$ bits. Solve the $\Delta+1$ coloring problem in each of these independently and recursively. Then, we need to merge these colors, into a $\Delta+1$ coloring for the whole graph.

(A) Explain an $O(\Delta)$-round algorithm, as well its correctness proof, that once the independent $(\Delta+1)$-colorings of G_0 and G_1 are finished, updates only the colors of G_1 vertices to ensure that the overall coloring is a proper $(\Delta+1)$-coloring of $G = G_0 \cup G_1$.

(B) Provide a recursive time-complexity analysis that proves that overall, the recursive method takes $O(\Delta \log \Delta)$ rounds.

Exercise 8.6. *Explain how given a (C, D) network decomposition of graph G, a maximal independent set can be computed in $O(CD)$ rounds.*

Exercise 8.7. *Prove Lemma 8.28.*

Exercise 8.8. *Improve the round complexity of the algorithm stated in Theorem 8.27 to* $O(\log n)$ *rounds.*

Exercise 8.9. *We here see that the network decomposition obtained in Theorem 8.27 has the nearly best possible parameters. In particular, it is known that there are* n*-node graphs that have girth*[11] $\Omega(\log n / \log\log n)$ *and chromatic number* $\Omega(\log n)$*[AS04, Erd59]. Use this fact to argue that on these graphs, an* $(o(\log n/\log\log n), o(\log n))$ *network decomposition does not exist.*

Exercise 8.10. *Given an* n*-node undirected graph* $G = (V, E)$*, a* $d(n)$-diameter ordering *of* G *is a one-to-one labeling* $f : V \to \{1, 2, \ldots, n\}$ *of vertices such that for any path* $P = v_1, v_2, \ldots, v_p$ *on which the labels* $f(v_i)$ *are monotonically increasing, any two nodes* $v_i, v_j \in P$ *have* $\mathrm{dist}_G(v_i, v_j) \leq d(n)$*.*

Use the network decomposition of Theorem 8.27 to argue that each n*-node graph has an* $O(\log^2 n)$*-diameter ordering.*

Exercise 8.11. *Consider the following simple* 1*-round randomized algorithm: each node* v *picks a random real number* $r_v \in [0, 1]$ *and then,* v *joins a set* S *if its random number is a local minima, that is, if* $r_v < r_u$ *for all neighbors* u *of* v*. Prove that, with high probability, the set* S *is a* $(2, O(\log n))$*-ruling set.*

[11] Recall that the girth of a graph is the length of its shortest cycle.

Bibliography

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

[ALGP89] Baruch Awerbuch, M Luby, AV Goldberg, and Serge A Plotkin. Network decomposition and locality in distributed computation. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 364–369. IEEE, 1989.

[AS04] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2004.

[Bar15] Leonid Barenboim. Deterministic ($\delta+1$)-coloring in sublinear (in δ) time in static, dynamic and faulty networks. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 345–354. ACM, 2015.

[Bol78] Béla Bollobás. Chromatic number, girth and maximal degree. *Discrete Mathematics*, 24(3):311–314, 1978.

[CFS10] David Conlon, Jacob Fox, and Benny Sudakov. Hypergraph ramsey numbers. *Journal of the American Mathematical Society*, 23(1):247–266, 2010.

[CLRS01] Thomas H.. Cormen, Charles Eric Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*, volume 6. MIT press Cambridge, 2001.

[CV86] Richard Cole and Uzi Vishkin. Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms. In *Proceedings of the eighteenth annual*

ACM symposium on Theory of computing, pages 206–219. ACM, 1986.

[Erd59] Paul Erdős. Graph theory and probability. *Canada J. Math*, 11:34G38, 1959.

[FHK16] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local conflict coloring. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*, 2016.

[GPS87] Andrew Goldberg, Serge Plotkin, and Gregory Shannon. Parallel symmetry-breaking in sparse graphs. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 315–324. ACM, 1987.

[Jam11] Mohammad Shoaib Jamall. A brooks' theorem for triangle-free graphs. *arXiv preprint arXiv:1106.1958*, 2011.

[Kim95] Jeong Han Kim. On brooks' theorem for sparse graphs. *Combinatorics, Probability and Computing*, 4(02):97–132, 1995.

[Kuh09] Fabian Kuhn. Weak graph colorings: distributed algorithms and applications. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 138–144. ACM, 2009.

[KW06] Fabian Kuhn and Rogert Wattenhofer. On the complexity of distributed graph coloring. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 7–15. ACM, 2006.

[Lin87] Nathan Linial. Distributive graph algorithms global solutions from local data. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*, pages 331–335. IEEE, 1987.

[Lin92] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[LS91] Nathan Linial and Michael Saks. Decomposing graphs into regions of small diameter. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '91, pages 320–330, 1991.

[LS14] Juhana Laurinharju and Jukka Suomela. Brief announcement: Linial's lower bound made easy. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 377–378. ACM, 2014.

[Lub66] David Lubell. A short proof of sperner's lemma. *Journal of Combinatorial Theory*, 1(2):299, 1966.

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

[Nao91] Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991.

[NS93] Moni Naor and Larry Stockmeyer. What can be computed locally? In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 184–193. ACM, 1993.

[NS95] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

[PS92] Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 581–592. ACM, 1992.

[PS15] Seth Pettie and Hsin-Hao Su. Distributed coloring algorithms for triangle-free graphs. *Information and Computation*, 243:263–280, 2015.

[RS14] Joel Rybicki and Jukka Suomela. Exact bounds for distributed graph colouring. In *International Colloquium on Structural Information and Communication Complexity*, pages 46–60. Springer, 2014.

[Spe28] Emanuel Sperner. Ein satz über untermengen einer endlichen menge. *Mathematische Zeitschrift*, 27(1):544–548, 1928.

[SV93] Mário Szegedy and Sundar Vishwanathan. Locality based graph coloring. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 201–207. ACM, 1993.