

---

## Lower Bounds

Zinaida Benenson

### 7.1 Introduction

This book presents efficient algorithms for a large amount of important problems in ad hoc and sensor networks. Consider a problem  $\mathcal{P}$  and the most efficient algorithm  $\mathcal{A}$  (known so far) solving it with running time  $O(g(n))$ , where  $n$  is the number of nodes in the network.  $O(g(n))$  is called *upper bound* on the running time for  $\mathcal{P}$ . After the problem is solved, the next step is to ask: Can the problem  $\mathcal{P}$  be solved more efficiently? Is the running time of the algorithm  $\mathcal{A}$  *optimal*?

If somebody devises a faster algorithm for  $\mathcal{P}$ , then the answers to both questions are obvious. However, it can also happen that no faster algorithm is suggested for some time. Then the only way to answer these questions is to establish a *lower bound*  $\Omega(f(n))$  on the running time of the algorithms for the problem  $\mathcal{P}$ . This task is by no means trivial, as it involves showing that even algorithms which have not been invented yet cannot be faster than  $\Omega(f(n))$ .

For example, many network coordination problems, such as partitioning a network into clusters (see Chapter 3), or assigning time slots in a TDMA MAC protocol (Chapter 4), can be solved using algorithms for graph coloring and maximal independent set (MIS).

Consider Algorithm 13 from Chapter 4 for finding a  $(\Delta + 1)$ -coloring of arbitrary graphs with running time  $O(\Delta^2 + \log^* n)$ <sup>1</sup>, where  $\Delta$  is the maximal node degree of the graph, and  $n$  is the number of nodes. It is also possible to color an arbitrary graph with  $O(\Delta^2)$  colors in time  $O(\log^* n)$ . Using a  $c$ -coloring algorithm which runs in  $t$  rounds, MIS can be computed in  $t + c$  rounds, and therefore, MIS can be computed in time  $O(\Delta^2 + \log^* n)$  in arbitrary graphs.

---

<sup>1</sup> We denote the  $k$  times iterated logarithm by  $\log^k x$ , i.e.,  $\log^1 x := \log x$ , and  $\log^k x := \log(\log^{k-1} x)$ . The least integer  $k$  for which  $\log^k x \leq 2$  is denoted by  $\log^* x$ .

Moreover, faster algorithms are possible for unit disk graphs (UDGs). If the nodes know the distances to their neighbors,  $(\Delta + 1)$ -coloring and MIS can be computed in time  $O(\log^* n)$ . Even without distance information, computing MIS for UDGs is possible in time  $O(\log \Delta \cdot \log^* n)$ . For graphs of bounded degree, where  $\Delta$  is a constant independent of  $n$ , computing  $(\Delta + 1)$ -coloring and MIS is possible in time  $O(\log^* n)$ .

It stands out that running time of all these algorithms depends on  $n$  only slightly, as  $\log^* n$  is a very slowly growing function<sup>2</sup>. Therefore, running time  $O(\log^* n)$  can be considered constant for all practical purposes. Nevertheless, an intriguing question arises: Is it possible to design *constant time* algorithms for graph coloring and MIS?

N. Linial answered this question in negative in his seminal paper “Locality in distributed graph algorithms” [260], where he established a *lower bound* on time needed to compute MIS and coloring. He showed that, even on such simple graphs as rings, any algorithm for 3-coloring and MIS *requires* time  $\Omega(\log^* n)$ . This result applies to all above graph classes, as rings are certainly graphs of bounded degree, and as a UDG also can form a ring.

The above discussion exemplifies the value of the results on lower bounds:

- Lower bounds help to establish *asymptotic optimality* of an algorithm for a specific problem, e.g., one cannot color a graph faster than in  $\Omega(\log^* n)$  time. This way we know when to stop looking for a better solution, as such a solution does not exist.
- Lower bounds also show *room for improvement* of the existing solutions. For example, is it possible to compute MIS on UDGs faster than  $O(\log \Delta \cdot \log^* n)$ ?

We proceed with a formal definition of lower bound on time complexity.

**Definition 7.1.1 (lower bound on time complexity).**  $\Omega(f(n))$  is called a lower bound on time complexity of the problem  $\mathcal{P}$  for the class of  $n$ -vertex graphs  $\mathcal{G}_n$  if there is a constant  $c \in \mathbb{N}^+$  such that for all  $n$  there exists a  $n$ -vertex graph  $G_n \in \mathcal{G}_n$  such that the worst-case running time of any algorithm for  $\mathcal{P}$  on  $G_n$  is at least  $cf(n)$ .

The class  $\mathcal{G}_n$  can consist of arbitrary graphs, or unit disk graphs (UDG), or, e.g., all graphs with diameter  $\Omega(\log n)$ . According to the definition, to show a lower bound of  $\Omega(f(n))$  for problem  $\mathcal{P}$ , one has to construct for any  $n$  a graph  $G_n \in \mathcal{G}_n$  such that the running time of any algorithm for  $\mathcal{P}$  cannot be less than  $cf(n)$  for some  $c \in \mathbb{N}^+$ .

Algorithms for dynamic networks with large number of nodes, such as ad hoc and sensor networks, are considered the more efficient the less their running time depends on the number of nodes in the network. *Locality* as a design principle for distributed systems means developing algorithms for solving global tasks such that the nodes of the system need to gather more

---

<sup>2</sup>  $\log^* n \leq 4$  for  $n$  being the estimated number of atoms in the observable universe.

information from their neighborhood than from the rest of the network. Locality helps to limit state which the nodes accumulate during the protocol, and also improves fault-tolerance, as failures at some part of the network do not propagate. Thus, finding out whether a particular problem can be solved using locality-sensitive algorithms is a very important question. The above discussion on time complexity of graph coloring and MIS showed how establishing lower bounds can help here.

Following [390], we distinguish between *localized* and *local* algorithms.

**Definition 7.1.2 (localized algorithms).** *In a  $k$ -localized algorithm, for some parameter  $k$ , each node is allowed to communicate at most  $k$  times with its neighbors. A node can decide to retard its right to communicate; for example, a node can wait to send messages until all its neighbors having larger identifiers have reached a certain state of their execution.*

Localized algorithms can have worst-case time complexity of  $\Theta(n)$ , because some nodes may have to wait for other nodes before they start algorithm execution. In the worst case, this may result in only one node being active at any unit of time. On the other hand, local algorithms below do not permit linear running time.

**Definition 7.1.3 (local algorithms).** *In a  $k$ -local algorithm, for some parameter  $k$ , each node can communicate at most  $k$  times with its neighbors. In contrast to  $k$ -localized algorithms nodes cannot delay the algorithm execution. In particular, all nodes process  $k$  synchronized phases, and a node's operations in phase  $i$  may only depend on the information received during phases 1 to  $i$ .*

In the following, this chapter discusses fundamental limitations on design of local algorithms. In Section 7.2, lower bounds on running time of distributed algorithms for graph coloring and MIS are shown in detail. Section 7.3 presents advanced results on the locality of some problems similar to coloring and MIS. Section 7.4 discusses lower bounds for computing minimum-weight spanning tree (MST).

## 7.2 A Lower Bound on 3-Coloring a Ring

In this section, a lower bound  $\Omega(\log^* n)$  on the time required for coloring an  $n$ -vertex ring with 3 colors and for computing MIS on an  $n$ -vertex ring is shown. This means that even in a ring, constructing a constant time algorithm for  $(\Delta + 1)$ -coloring is impossible. This result establishes optimality of various coloring and MIS algorithms, as discussed in the previous section.

For clarity, we repeat the definitions of  $c$ -coloring and MIS problems here.

**Definition 7.2.1 ( $c$ -coloring).** *A  $c$ -coloring of a graph  $G = (V, E)$  for  $c \in \mathbb{N}$  is an assignment of a color from the set  $\{1, \dots, c\}$  to each vertex  $v \in V$  such that no two adjacent vertices have the same color.*

**Definition 7.2.2 (maximal independent set (MIS)).** An independent set for a graph  $G = (V, E)$  is a set of vertices  $U \subseteq V$ , no two of which are adjacent. An independent set is maximal if no vertex can be added to it without violating its independence. The MIS problem is the problem of computing a maximal independent set for a given graph.

To show lower bounds, it is sensible to assume a powerful system model. In this case, the established lower bounds will also apply to less powerful systems. Thus, we assume synchronous and reliable communication between nodes. At each time unit, any node can send messages of *unbounded* size to all of its neighbors.

The vertices of the graph  $G = (V, E)$  are labeled by unique identifiers from the set  $\{1, 2, \dots, |V|\}$ . It is assumed that at time zero, i.e., before the beginning of the computation, each node knows only its own identifier.

For the graph  $G$ ,  $\chi(G)$  denotes the *chromatic number* of  $G$ , which is the least number of colors that suffice to color the vertices of  $G$  such that no two adjacent vertices have the same color.

**Theorem 7.2.3.** Any deterministic distributed algorithm which colors the  $n$ -ring with 3 colors requires time at least  $\frac{1}{2}(\log^* n - 1)$ .

This lower bound can be proven for randomized algorithms as well. However, in this section, only the proof for deterministic algorithms is presented.

This section is organized as follows. Firstly, we give the proof outline in Section 7.2.1. The proof of the main Theorem 7.2.3 is given in Section 7.2.2. Proof of Lemma 7.2.6, which is rather long, is shown in Section 7.2.3 for clarity reasons. In Section 7.2.4, two interesting additional facts are shown: the  $\Omega(n)$  lower bound on 2-coloring a  $2n$ -ring, and the  $\Omega(\log^* n)$  lower bound on computing maximal independent set (MIS) on a ring.

## 7.2.1 Proof Outline

The proof consists of the following steps:

1. For the  $n$ -ring  $R_n$ , we define a graph  $B_{t,n}$  (Definition 7.2.4), where  $t$  is the running time for a 3-coloring algorithm, and show that every 3-coloring of  $R_n$  immediately yields a 3-coloring of  $B_{t,n}$ , i.e.,  $\chi(B_{t,n}) \leq 3$  (Lemma 7.2.5).
2. We show that  $\chi(B_{t,n}) \geq \log^{(2t)} n$  (Lemma 7.2.6). This proof runs as follows:
  - a) We define a family of directed graphs  $D_{s,n}$  (Definition 7.2.7) and show that  $\chi(B_{t,n}) \geq \chi(D_{2t+1,n})$  (Lemma 7.2.8).
  - b) Then we define *dilinegraph*  $DL(G)$  for each directed graph  $G$  (Definition 7.2.9), and prove Lemma 7.2.10 on the relation between  $D_{s,n}$  and its dilinegraph.

- c) We prove the Lemma 7.2.11:  $\chi(DL(G)) \geq \log \chi(G)$  for any directed graph  $G$ .
  - d) Using Lemmas 7.2.10 and 7.2.11 it follows that  $\chi(D_{s,n}) \geq \log^{(s-1)} n$  for all  $s > 1$  (Lemma 7.2.12).
  - e) Then, from Lemmas 7.2.8 and 7.2.12 it follows that  $\chi(B_{t,n}) \geq \log^{(2t)} n$ .
3. From Lemma 7.2.6, we then derive the lower bound  $\frac{1}{2}(\log^* n - 1)$  on running time  $t$  for 3-coloring of  $B_{t,n}$ . Then  $t \geq \frac{1}{2}(\log^* n - 1)$  also applies for 3-coloring of  $R_n$  (Lemma 7.2.5).

### 7.2.2 Proof of Theorem 7.2.3

We now proceed to the actual proof of Theorem 7.2.3.

Consider an arbitrary deterministic algorithm for  $c$ -coloring a ring which operates in time  $t$ . At time 0, a node  $v$  of the ring  $R_n$  knows only its own identifier.

At time 1, direct neighbors of  $v$  are able to send to  $v$  all information which they possess, i.e., their own identifiers, and perhaps results of some computations. The important point is that these computations can base only upon the information known to the nodes so far, that is, on their own identifiers, as the algorithm is deterministic. It follows that, upon receiving messages from the first round of communication,  $v$  can *simulate* the computations done by its neighbors in the first round, see Fig. 7.1.

At time 2, information initially known to nodes which are at distance 2 from  $v$ , i. e., their identifiers, and, possibly, some computation results, reaches  $v$ . Again, after having received messages from the second round of communication,  $v$  can simulate all computations done by its 2-hop neighbors so far.

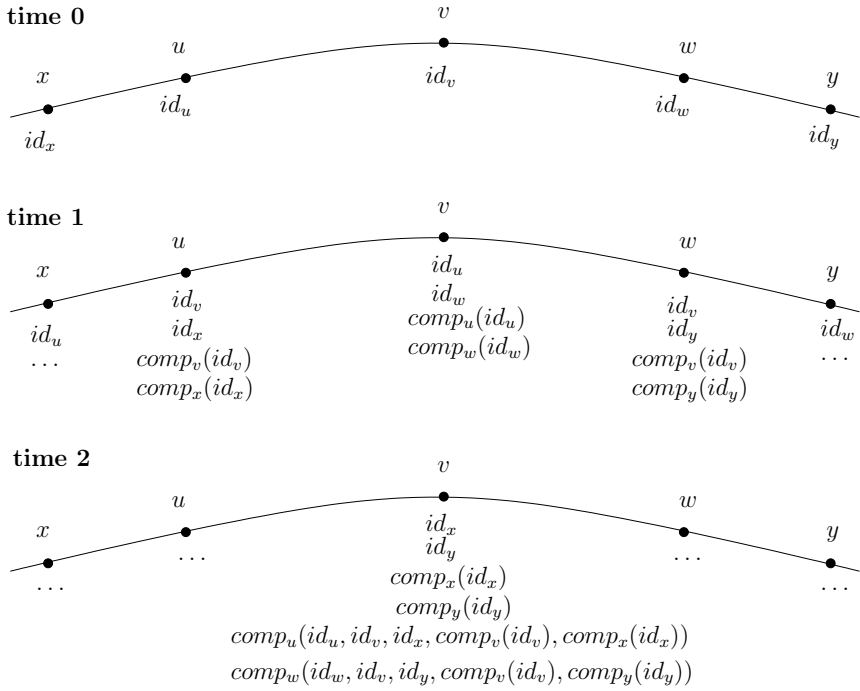
Thus, at time  $t$ , node  $v$  knows identifiers of all  $2t$  nodes which are at distance at most  $t$  from it. No further information can reach  $v$  at time  $t$ , as all possible (deterministic) computations which its  $t$ -hop neighbors might have done in these  $t$  rounds, can be simulated by node  $v$ . Thus, knowledge of any node at time  $t$  can be denoted as a  $(2t + 1)$ -tuple  $(x_1, \dots, x_{2t+1})$ .

It follows that, without loss of generality, we can assume that any deterministic  $c$ -coloring algorithm for an  $n$ -ring  $R_n$  works as follows:

1. For  $t$  rounds, each node collects and relays information about identifiers. Each node  $v$  ends up with a tuple  $(x_1, \dots, x_{2t+1})$  where all  $x_i$  are pairwise distinct numbers from  $\{1, \dots, n\}$ . We denote the set of all such tuples  $V_{t,n}$ .
2. Each node  $v$  stores the color it should choose for each of  $n(n-1) \dots (n-2t)$  possible tuples of identifiers. After time  $t$ , the node chooses its color according to the gathered information.

Thus, any deterministic  $c$ -coloring algorithm defines a mapping  $\Psi_c$  from  $V_{t,n}$  into the set of colors  $\{1, \dots, c\}$ .

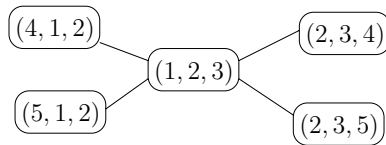
To formalize the above reasoning, we now define an auxiliary graph with the set of vertices  $V_{t,n}$  where two vertices are adjacent if and only if vertices with identifiers  $x_t$  and  $x_{t+1}$  are neighbors in  $R_n$ .



**Fig. 7.1.** New information known by node  $v$  and its neighbors at time 0, 1, and 2;  $comp_u(\dots)$  denotes computation done by node  $u$ .

**Definition 7.2.4** ( $B_{t,n}$ ). Graph  $B_{t,n}$  with the set of vertices  $V_{t,n}$  is defined as follows: any two vertices  $(x_1, \dots, x_{2t+1})$  and  $(y, x_1, \dots, x_{2t})$  with  $y \neq x_{2t+1}$  are joined by an edge, and no other edges exist in the graph.

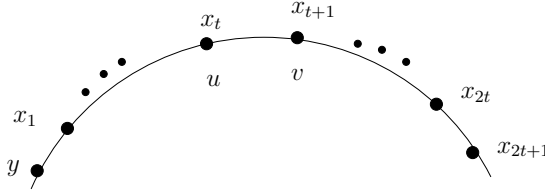
Graph  $B_{t,n}$  is regular of degree  $2(n - 2t - 1)$ , as any node  $(x_1, \dots, x_{2t+1})$  is adjacent to  $(n - 2t - 1)$  nodes of form  $(y, x_1, \dots, x_{2t})$  with  $y \neq x_{2t+1}$ , and to  $(n - 2t - 1)$  nodes of form  $(x_2, \dots, x_{2t+1}, z)$  with  $z \neq x_1$ , see Figure 7.2.



**Fig. 7.2.** Node  $(1, 2, 3)$  in  $B_{1,5}$

**Lemma 7.2.5.** If  $R_n$  can be  $c$ -colored in  $t$  rounds, then  $\chi(B_{t,n}) \leq c$ .

*Proof.* Let  $\Psi_c : V_{t,n} \rightarrow \{1, \dots, c\}$  be a  $c$ -coloring of  $R_n$ . Then  $\Psi_c$  assigns different colors to neighboring nodes. Thus, if node  $v$  gathered a tuple of indices  $(x_1, \dots, x_{2t+1})$  ( $x_{t+1}$  being the identifier of  $v$ ), and its neighbor  $u$  with identifier  $x_t$  gathered the tuple  $(y, x_1, \dots, x_{2t})$  (see Figure 7.3), then  $\Psi_c$  should satisfy:  $\Psi_c((x_1, \dots, x_{2t+1})) \neq \Psi_c((y, x_1, \dots, x_{2t}))$ . Then,  $\Psi_c$  is also a  $c$ -coloring for  $B_{t,n}$ , and therefore,  $\chi(B_{t,n}) \leq c$ .  $\square$



**Fig. 7.3.** After time  $t$ ,  $v$  knows the tuple  $(x_1, \dots, x_{2t+1})$ , and  $u$  knows the tuple  $(y, x_1, \dots, x_{2t})$ .

We now consider how  $\chi(B_{t,n})$  depends on  $t$  and  $n$ :

**Lemma 7.2.6.**  $\chi(B_{t,n}) \geq \log^{(2t)} n$

Proof of Lemma 7.2.6 is given in Section 7.2.3.

*Proof (of Theorem 7.2.3).* Consider an algorithm for 3-coloring of  $R_n$  with running time  $t$ . Then  $\chi(B_{t,n}) \leq 3$  (Lemma 7.2.5). Then from Lemma 7.2.6 it follows that  $3 \geq \log^{(2t)} n$ , and therefore,  $t \geq \frac{1}{2}(\log^* n - 1)$ .  $\square$

### 7.2.3 Proof of Lemma 7.2.6

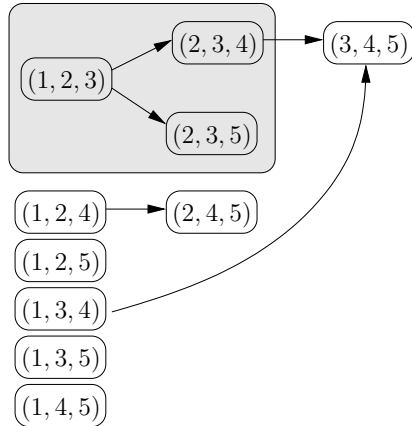
We first define two more auxiliary graphs and prove some of their properties.

**Definition 7.2.7** ( $D_{s,n}$ ). *The vertices of directed graph  $D_{s,n}$  are all sequences  $(a_1, \dots, a_s)$  with  $1 \leq a_1 < \dots < a_s \leq n$ . The outneighbors of  $(a_1, \dots, a_s)$  are all vertices of  $D_{s,n}$  having the form  $(a_2, \dots, a_s, b)$ .*

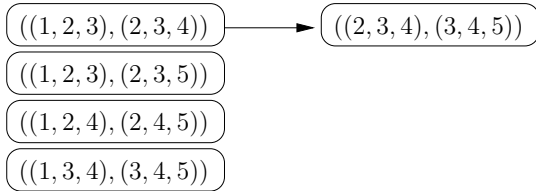
**Lemma 7.2.8.**  $\chi(B_{t,n}) \geq \chi(D_{2t+1,n})$

*Proof.* The Lemma follows immediately from the observation that graph  $B_{t,n}$  contains the underlying (undirected) graph of  $D_{2t+1,n}$  (see also Figure 7.4 for an example).  $\square$

**Definition 7.2.9 (dilinegraph).** *Given a directed graph  $G = (V, E)$ , its dilinegraph  $DL(G)$  is a directed graph whose vertex set is  $E$  with  $((u_1, u_2), (v_1, v_2))$  being an edge if  $u_2 = v_1$ .*



**Fig. 7.4.** Graph  $D_{3,5}$ . The underlying undirected graph is contained in  $B_{1,5}$ . Shaded area can be seen at Figure 7.2.



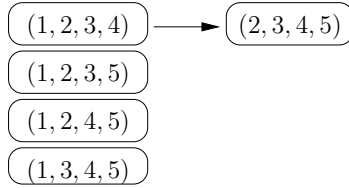
**Fig. 7.5.** Graph  $DL(D_{3,5})$ .

The dilinegraph of  $D_{3,5}$  is shown in Figure 7.5.

**Lemma 7.2.10.** (1)  $D_{1,n}$  is obtained from the complete graph of order  $n$  by replacing each edge by a pair of edges, one in each direction. (2)  $D_{s+1,n} = DL(D_{s,n})$  for all  $s \geq 1$ .

*Proof.* The first claim follows immediately from the definition. For the second claim, set of edges of  $D_{s,n}$  contains all pairs  $((x_1, \dots, x_s), (x_2, \dots, x_s, y))$  with  $x_s < y$ . Then  $DL(D_{s,n})$  consists of nodes of the above form. We associate each node of  $DL(D_{s,n})$  with the node of  $D_{s+1,n}$  of the form  $(x_1, x_2, \dots, x_s, y)$ . This mapping is a bijection. It remains to show that the adjacency relations in both graphs are the same. Indeed, any edge from  $DL(D_{s,n})$  has the form  $((x_1, \dots, x_s), (x_2, \dots, x_s, y)), ((x_2, \dots, x_s, y), (x_3, \dots, x_s, y, z))$  where  $y < z$ , which maps to the edge of  $D_{s+1,n}$  of the form  $((x_1, \dots, x_s, y), (x_2, \dots, x_s, y, z))$  and vice versa. For an example, see Figure 7.6.  $\square$

**Lemma 7.2.11.** For every directed graph  $G$ ,  $\chi(DL(G)) \geq \log(\chi(G))$ .



**Fig. 7.6.** Graph  $D_{4,5}$  is isomorphic to the graph  $DL(D_{3,5})$  (Figure 7.5.).

*Proof.* Let  $k = \chi(DL(G))$ , and consider a  $k$ -coloring  $\hat{c}$  of  $DL(G)$ . Then  $\hat{c}$  is an edge coloring of  $G$  such that if the edge  $e'$  starts in the vertex where the edge  $e$  ends, then  $\hat{c}(e) \neq \hat{c}(e')$ . We then define the coloring  $c$  for  $G$  such that  $c(v) := \{\hat{c}(e) \mid \text{edge } e \text{ ends in the vertex } v\}$ .

The coloring  $c$  uses at most  $2^k$  colors, as each color is a subset of  $\{1, \dots, k\}$ . Consider nodes  $u$  and  $v$  which are neighbors in  $G$ . We show now that  $c(v) \neq c(u)$ , i.e., that  $c(v)$  is a proper coloring for  $G$ .

W.l.o.g.  $(u, v) = e$  is an edge of  $G$ . Then  $\hat{c}(e) \in c(v)$ . However,  $\hat{c}(e) \notin c(u)$ , as otherwise, there would have been an edge  $e' = (w, u)$  for some vertex  $w$  of  $G$  such that  $\hat{c}(e') = \hat{c}(e)$ . This is a violation of the edge coloring of  $H$ . Thus,  $c$  is a proper coloring of  $G$ , and therefore,  $\chi(G) \leq 2^k$ .  $\square$

**Lemma 7.2.12.**  $\chi(D_{s,n}) \geq \log^{(s-1)} n$  for all  $s > 1$ .

*Proof.* Proof is by induction on  $s$ .

For  $s = 2$ , we have to show that  $\chi(D_{2,n}) \geq \log n$ . From Lemma 7.2.10 it follows that  $D_{2,n} = DL(D_{1,n})$ . Then, from Lemma 7.2.11,  $\chi(DL(D_{1,n})) \geq \log(\chi(D_{1,n})) = \log n$ .

For  $s > 2$ , assume that  $\chi(D_{s,n}) \geq \log^{(s-1)} n$ . We show that  $\chi(D_{s+1,n}) \geq \log^{(s)} n$ . From Lemma 7.2.10,  $D_{s+1,n} = DL(D_{s,n})$ , and from Lemma 7.2.11,  $\chi(DL(D_{s,n})) \geq \log(\chi(D_{s,n}))$ . Then, from the induction hypothesis,  $\log(\chi(D_{s,n})) \geq \log^{(s)} n$ .  $\square$

Finally, we prove Lemma 7.2.6.

*Proof (of Lemma 7.2.6).* We have to show that  $\chi(B_{t,n}) \geq \log^{(2t)} n$ . From Lemma 7.2.8, it follows that  $\chi(B_{t,n}) \geq \chi(D_{2t+1,n})$ . Then  $\chi(D_{2t+1,n}) \geq \log^{(2t)} n$  (Lemma 7.2.12).  $\square$

## 7.2.4 Additional Results

Although Theorem 7.2.3 precludes a constant-time algorithm for 3-coloring a ring, nevertheless, an algorithm which runs in time  $\Omega(\log^* n)$  is very efficient, as  $\log^* n$  grows very slowly with  $n$ . In contrast, for a ring with an even number of nodes, finding a 2-coloring requires time  $\Omega(n)$ .

**Theorem 7.2.13.** *Any deterministic distributed algorithm for coloring a  $2n$ -ring with two colors requires at least  $n - 1$  rounds.*

*Proof.* Consider an algorithm which 2-colors  $R_n$  in  $t$  rounds. Then  $\chi(B_{t,2n}) \leq 2$  (Lemma 7.2.5).

If  $B_{t,2n}$  can be colored with two colors, then this graph is bipartite. A graph is bipartite if and only if it contains no odd-length ring. However, for  $t \leq n - 2$ ,  $B_{t,2n}$  contains such a ring (of length  $2t + 3$ ):

$$(1, 2, \dots, 2t + 1), (2, \dots, 2t + 1, 2t + 2), (3, \dots, 2t + 3), (4, \dots, 2t + 3, 1) \\ (5, \dots, 2t + 3, 1, 2), \dots, (2t + 3, 1, \dots, 2t), (1, \dots, 2t + 1) \quad \square$$

The next theorem shows a lower bound for the MIS (maximal independent set) problem on a ring.

**Theorem 7.2.14.** *Any deterministic distributed MIS algorithm for the  $n$ -ring requires at least  $\frac{1}{2}(\log^* n - 3)$  rounds.*

*Proof.* Due to Theorem 7.2.3 it suffices to show that, given a MIS for the  $n$ -ring  $R_n$ , it is possible to 3-color  $R_n$  in one round. The algorithm runs as follows:

- If the vertex  $v$  is in the MIS, then it colors itself 1 and sends message “my color is 1” to its both neighbors.
- If  $v$  is not in MIS, it sends its identifier  $id_v$  to its both neighbors. Note that as  $v$  is not in MIS, then at least one neighbor of  $v$  must be in MIS. Upon receiving the messages from round 1,  $v$  colors itself 2 if both its neighbors are in MIS. Otherwise,  $v$  must have received  $id_u$  from one of its neighbors  $u$ . If  $id_v < id_u$ , then  $v$  colors itself 2, otherwise it colors itself 3.

For an example execution of the algorithm, see Figure 7.7. The proof that the above algorithm properly 3-colors  $R_n$  is left as an exercise to the reader.  $\square$

### 7.3 Locally Checkable Labelings

After realizing that neither  $(\Delta + 1)$ -coloring nor MIS can be computed in constant time even on rings, an intriguing question is if there are *any* non-trivial problems from graph theory which *can* be computed locally, i.e., in constant time. Naor and Stockmeyer investigate this problem in [301] for a class of problems they call *locally checkable labelings* (LCL). In an LCL problem, the vertices (or the edges) of the graph must be labeled in accordance to some rules such that the legality of the labeling can be verified locally. Vertex coloring (Def. 7.2.1) is an LCL, as any node only needs to know the colors of its neighbors to check the legitimacy of the coloring. MIS (Def. 7.2.2)

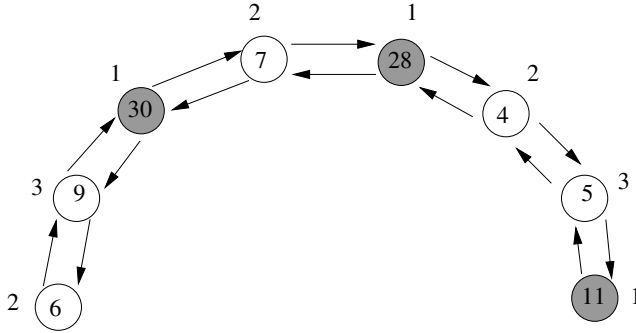


Fig. 7.7. 3-coloring a ring given an MIS (shaded nodes)

is also an LCL, as each node can check the following: If the node is in MIS, then none of its neighbors should be in MIS. If the node does not belong to MIS, then at least one of its neighbors should be in MIS.

It turns out that there is at least one LCL problem which indeed has a local algorithm for a certain non-trivial class of graphs. This problem is called *weak  $c$ -coloring*.

**Definition 7.3.1 (weak  $c$ -coloring).** A weak  $c$ -coloring of a graph  $G = (V, E)$  for  $c \in \mathbb{N}$  is an assignment of a color from the set  $\{1, \dots, c\}$  to each vertex  $v \in V$  such that each non-isolated node has at least one neighbor colored differently.

Weak 2-coloring exists for any graph. To find such a coloring, we can first compute the breadth-first spanning tree of the graph, and then assign one color to even levels, and a different color to odd levels. However, even weak 2-coloring cannot be computed locally for some classes of graphs, as the following theorem (given without proof) shows.

**Theorem 7.3.2.** *Weak coloring has the following locality properties:*

- Consider a class of graphs of maximum degree  $d$  where every vertex has odd degree. There is a constant  $b$  such that, for every  $d$ , there is a local algorithm with running time at most  $\log^* d + b$  that solves the weak 2-coloring problem.
- For graphs of even degree, there is no local algorithm which solves weak  $c$ -coloring for any  $c$ .
- If the definition of weak coloring is changed such that every vertex must have at least two neighbors colored differently, than this LCL problem cannot be solved locally.

Sometimes, if some problem cannot be solved using a deterministic algorithm, it can be solved using a randomized algorithm having a small error probability. However, this does not apply in case of LCL problems.

**Theorem 7.3.3.** *Randomization cannot make an LCL problem local. That is, if an LCL problem has a randomized local algorithm, then it also has a deterministic local algorithm.*

It would be very convenient to have an algorithm which, given any LCL problem, decides if this problem has a local algorithm. Unfortunately, such an algorithm does not exist. More precisely, it can be shown that, if such an algorithm was possible, it could be used to solve the Halting Problem.

**Theorem 7.3.4.** *It is undecidable whether a given LCL problem has a local algorithm. However, there exists an algorithm which, given any LCL problem, decides if this problem has an algorithm which operates in time  $t$ .*

## 7.4 Minimum-Weight Spanning Trees

Algorithms for LCL problems such as coloring and MIS are essentially local, as their running time depends on  $n$  only “slightly”. This fits well with the local nature of these problems: The decision if a node should join the MIS, or which color should the node take, is largely independent on the nodes at the far end of the network.

On the other hand, some problems seem to require global knowledge about the network. For these problems, an algorithm may be considered local if it runs in polylogarithmic time. One of such problems is computing minimum-weight spanning tree (MST) of a given weighted graph. The MST can be used for efficient broadcast, see Chapter 12.

**Definition 7.4.1 (minimum-weight spanning tree (MST)).** *For a weighted graph  $G = (V, E, w)$  with the weight function  $w : E \rightarrow \mathbb{R}^+$ , minimum-weight spanning tree is a spanning tree with the minimal total weight (sum of the weights of all its edges).*

Recall that in Section 7.2, lower bound on 3-coloring a ring was proven for synchronous and reliable communication with unbounded messages. That is, in each round each node can send messages of unbounded size to all its neighbors. In this model, MST can be constructed in time  $O(D)$ , where  $D$  is the graph diameter (the maximum distance in hops between any two vertices). This can be done, e. g., by collecting the entire graph topology at some vertex, computing the MST locally, and then broadcasting the result to all nodes. Moreover, it can be shown that in this model, MST construction requires  $\Omega(D)$  time.

However, solving the MST problem becomes much more involved in the *bounded message model*, where each node can send messages of size at most  $B$  bits to its neighbors. Usually,  $B = \mathcal{O}(\log n)$  is considered. Of course, the lower bound  $\Omega(D)$  also applies in this stronger model. However, more precise lower bounds can be shown here.

In the bounded message model, one of the most efficient protocols for distributed MST construction has the time complexity  $O(D + \sqrt{n} \log^* n)$ . Does a faster (e.g., polylogarithmic time) algorithm exist for MST computation? The following theorem shows the asymptotical near-optimality of the above MST construction algorithm.

**Theorem 7.4.2.** *Any distributed algorithm for the distributed MST construction requires  $\Omega(D + \sqrt{n}/\log n)$  time on graphs of diameter  $D = \Omega(\log n)$ .*

This result is shown by construction for each  $n$  a special  $n$ -vertex graph where the MST problem cannot be solved faster than  $\Omega(D + \sqrt{n}/\log n)$ . Interestingly, even for graphs with very low constant diameter, there is no polylogarithmic MST algorithm. The following theorem presents more exact lower bounds on distributed MST construction.

**Theorem 7.4.3.** *Any distributed algorithm for the distributed MST construction requires:*

- $\Omega(\sqrt{n/\log n})$  for graphs of diameter  $n^\delta$ ,  $0 < \delta < 1/2$ ,
- $\Omega(\sqrt{n/\log n})$  for graphs of diameter  $\Theta(\log n)$ ,
- $\Omega(\sqrt[3]{n/\log n})$  for graphs of diameter 4,
- and  $\Omega(\sqrt[4]{n/\log n})$  for graphs of diameter 3.

The above results apply not only for deterministic, but also for randomized algorithms. On the other hand, MST for graphs of diameter 2 can be computed efficiently in time  $O(\log n)$ .

Sometimes, if the desired efficiency results cannot be achieved with exact algorithms, there are approximation algorithms which solve problems less exact, but with better efficiency results. For the MST problem, this means that the resulting  $H$ -approximate MST has the weight at most  $H$  times greater than the exact MST.  $H$  is called *approximation ratio*. Thus, even if one cannot compute exact MST in polylogarithmic time, perhaps it is possible to compute its  $H$ -approximation efficiently? Unfortunately, as the following theorem shows, the MST problem is even hard to approximate.

**Theorem 7.4.4.** *Approximating the MST within an approximation ratio  $H$  requires time  $\Omega\left(\sqrt{\frac{n}{H \cdot \log n}}\right)$ .*

In particular, this means that approximating the MST problem within any constant factor ( $H = O(1)$ ) requires  $\Omega(\sqrt{n/\log n})$  time, which is the same lower bound as for the exact MST construction, cf. Theorem 7.4.3. Moreover, for any  $0 < \epsilon < 1$ , an  $\left(\frac{n}{\log n}\right)^{1-\epsilon}$ -approximation of the MST problem requires  $\Omega\left(\left(\frac{n}{\log n}\right)^{\epsilon/2}\right)$  time, i. e., it cannot be solved locally (in polylogarithmic time).

## 7.5 Chapter Notes

In this chapter, we discussed lower bounds on running time of distributed algorithms for such important problems as graph coloring, MIS, and MST. These results help to establish optimality of existing algorithms, and pose a challenge to find algorithms which meet lower bounds.

The distinction between localized and local algorithms originates from [390].

Lower bounds of  $\Omega(\log^* n)$  on computing 3-coloring and MIS in a ring are shown in [260]. The LCL problems are investigated in [301]. For further results on lower bounds for approximability of MIS and minimum dominating set problems, see also Section 3.4.4. Most recent results on lower bounds for graph coloring, MIS, and related problems can be found in [233] and [239].

Lower bounds on the exact MST construction for the unbounded message model, and for bounded message model for graphs of diameter  $O(\log n)$  originate from [314]. Results for graphs of constant diameter are shown in [264]. In [110], refined lower bounds for the bounded message model are presented. Lower bounds on approximation algorithms for MST are also shown in [110]. An interesting open question is which lower bounds on MST construction apply for UDGs.

As stated in corresponding sections, the above results apply not only for deterministic algorithms, but also for randomized ones.

In this chapter, we also used for comparison results on upper bounds for different problems. For origins of Algorithm 13, see Chapter Notes to Chapter 4. The algorithm for  $O(\Delta^2)$ -coloring of arbitrary graphs with running time  $O(\log^* n)$  is presented in [260]. On the relation of coloring and MIS see, e.g., [313, page 93].

Algorithms with running time  $O(\log^* n)$  for  $(\Delta + 1)$ -coloring and MIS in UDGs with known distances are obtained in [234]. An  $O(\log \Delta \cdot \log^* n)$  MIS algorithm for UDGs without distance information originates from [229].

The protocol for distributed MST construction with running time  $O(D + \sqrt{n} \log^* n)$ , where  $D$  is the diameter of the graph, is given in [246]. Most efficient so far MST algorithm is due to Elkin [109]. A method to compute MST in graph of diameter 2 in time  $O(\log n)$  is shown in [264].