Advanced Distributed Algorithms **Spring 2019**

Lecture 1: March 28

Lecturer: Merav Parter

Distributed Models and Problems

In this course, we consider the two most classical models of distributed graph algorithms:

(1) The LOCAL model [Lin87]: The network is abstracted as an n-vertex graph $G = (V, E)$. There is a processor per node of the graph, which initially knows only the edges incident on that node. Per round each node can send messages to each of its neighbors (possibly of unbounded size). Thus, after r-rounds, each node can learn its entire r-ball in G.

(2) The CONGEST model [Pel00]: The same as the LOCAL model, except for the restriction that the message size is bounded by B bits, where typically $B = O(\log n)$.

In the first half of this course, we will consider the LOCAL model and mainly focus on the four canonical local graph problems: Maximal Independent Set (MIS), $(\Delta + 1)$ Coloring, Maximal Matching and $(2\Delta - 1)$ Edge Coloring. Note that all these problems can be easily solved by a sequential greedy algorithm. Our goal is to study algorithms that efficiently solve these problems in the LOCAL model. We start by presenting a general useful tool for solving local problems deterministically.

Network Decomposition

Given a graph $G = (V, E)$, a (c, d) network decomposition decomposes V into a collection of vertex-disjoint clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ such that:

- $\bigcup_i C_i = V$,
- the *diameter* of each cluster is at most d (as defined next), and
- the cluster graph, obtained by contracting each cluster into a vertex, can be colored with c colors.

There are two ways to define the *diameter* of a cluster: the strong diameter of a cluster is the diameter of the graph $G[C_i]$. In contrast, the weak diameter of C_i is $\max_{u,v \in C_i} \mathtt{dist}(u, v, G)$, i.e., it allows using nodes that are not in C_i. The network decomposition will be denoted as strong or weak depending on whether it provides strong (or weak) diameter. The cluster graph $\widetilde{G}$ of the clusters is formally defined by $\widetilde{G} = (\mathcal{C}, \mathcal{E})$ where every two cluster-nodes $C_i, C_j \in \mathcal{C}$ are neighbors in $\widetilde{G}$ if they are adjacent in G. That is, $\mathcal{E} = \{(C_i, C_j) \in \mathcal{E} \mid \exists u \in C_i, v \in C_j \text{ such that } (u, v) \in G\}$.

Applications of Network Decomposition. Network decompositions provide a generic way for solving a large collection of local problems. Roughly speaking, the small diameter of the clusters allows nodes to collect their cluster within $O(d)$ rounds. The fact that number of colors is small allows having few number of iterations, in each iteration we work in parallel on non-conflicting (i.e., non-neighboring) clusters.

Lemma 1.1 (From ND to MIS) Given *a (d, c) network decomposition, one can compute an MIS in $O(d \cdot c)$ rounds.*

First, we explain what we mean by the word "given". The output format of a distributed network decomposition algorithm specifies for each node the ID of the cluster to which it belongs (e.g., the maximum ID of the vertex in the cluster), along with the color in $[1, c]$ of this cluster in the cluster graph. The computation of MIS proceeds in c iterations. Initially, all vertices are marked and throughout the process nodes will get unmarked, indicating that they are not part of the MIS. In the first iteration, we consider all clusters of color 1, and let their leaders collect the entire cluster C, locally compute MIS in the subgraph $G[C]$, and notify the nodes in C whether or not they should join the MIS. As all clusters that are colored 1 are not adjacent in the cluster graph, they do not have any neighboring vertices in G, and thus this MIS computation can be done in parallel safely (i.e., it will never be the case that two neighboring nodes joined the MIS). The first iteration terminates by letting all nodes in color-1 clusters that joined the MIS send a message to their neighbors. Those neighbors will get unmarked (indicating that they are excluded from being considered to join the MIS in the future iterations). In the second iteration, we do the same only for all clusters of color 2. This time the leaders of each 2-colored cluster C' collect the subgraph $G[C']$, along with the states of the nodes in the cluster (i.e., marked or unmarked). Only the marked nodes will be able to join the MIS. Again, the leaders locally compute an MIS on each subgraph, notify the newly MIS nodes which in turn notify their neighbors. This continues for each color class. Since handling a color class takes a total of $O(d)$ rounds (collecting information of the d-diameter cluster), overall the algorithm takes $O(c \cdot d)$ rounds.

Towards the middle of the course, we will get back again to network decompositions, and establish their centrality as a "complete" problem of the local problems. Now, that we are motivated to study (c, d) network decomposition, lets see what c, d values can be obtained for any n-vertex graphs, and how much time it takes to compute these structures in the distributed setting.

What's Known? Network decomposition were introduced by Awerbuch, Goldberg, Luby and Plotkin [ALGP89] who presented a deterministic construction for a (c, d) network decomposition in t rounds, where $c, d, t = 2^{O(\sqrt{\log n \cdot \log\log n})}$. Panconesi and Srinivasan [PS96] improved the parameters to $c, d, t = 2^{O(\sqrt{\log n})}$. In addition, they showed how to improve this (c, d) decomposition with $c, d = 2^{O(\sqrt{\log n}))}$ into an $(O(\log n), O(\log n))$ decomposition (which still takes $t = 2^{O(\sqrt{\log n})}$ rounds).

In the *randomized* setting[1], the situation is considerably better. Linial and Saks [LS93] showed that every graph admits an $(O(\log n), O(\log n))$ *weak* network decomposition, that can be computed within $O(\log^2 n)$ rounds with high probability[2]. Elkin and Neiman [EN16] were able to "fix" this construction into a *strong* network decomposition within the same number of rounds. Note that if randomness is allowed, we can compute MIS faster than $O(\log^2 n)$ rounds, and thus we usually do not use the randomized constructions of network decomposition to solve local problems. In contrast, in the deterministic setting, to this date, the state of the art of deterministic MIS computation is still $2^{O(\sqrt{\log n})}$ by applying the algorithm for Lemma 1.1 with the deterministic decomposition of [PS96].

In this class, we will present the original deterministic algorithm by Awerbuch, Glodberg, Luby and Plotkin [ALGP89].

Key Tool: Ruling-Sets. A useful tool that is repeatedly applied in this deterministic network decomposition algorithm is that of ruling sets, which can be viewed as a generalization of MIS.

Definition 1.2 (**(s, r) ruling sets**) *Given a graph $G = (V, E)$, and a subset $W \subseteq V$ of nodes, an (s, r) ruling set for W is a subset $R \subseteq W$ satisfying that:*

1. *[Separation]* $\mathtt{dist}(u, v, G) \geq s$ *for every* $u, v \in R$, *and*
2. *[Domination]* $\mathtt{dist}(w, R, G) = \min_{v \in R} \mathtt{dist}(w, v, G) \leq r$ *for every* $w \in W$.

See Fig. 1.1 for an illustration. For example, an MIS is a $(2, 1)$ ruling-set for the set of nodes V. Note that in the sequential setting, a simple greedy algorithm[3] gives an $(\alpha, \alpha - 1)$ ruling set for every $\alpha \geq 1$. In the deterministic and distributed setting, we can have the following:

[1]Here each node has an access to an unbounded pool of (private) random coins and can base its decisions using those coins.
[2]Throughout the course, *high probability* refers to probability of at least $1 - 1/n^c$ for some constant $c > 1$.
[3]E.g., traverse the nodes of the graph one by one, and add it to the current ruling set only if their distance to this set is at least α.

Lemma 1.3 (Distributed Computation of Ruling Sets) *Given a graph $G = (V, E)$, and subset of nodes $W \subseteq V$, for every α one can compute an $(\alpha, \alpha \cdot \log n)$ ruling set within $O(\alpha \log n)$ rounds.*

Proof: The constriction is recursive. We partition the set W into subsets W_0 and W_1, where W_0 contains all nodes in W whose last bit in their ID is 0, and W_1 contains all nodes in W whose last bit in their ID is 1. Now, we compute in parallel an $(\alpha, \alpha(\log n - 1))$ ruling-set R_0 for W_0, and an $(\alpha, \alpha(\log n - 1))$ ruling set R_1 for W_1. The stopping condition is when the set W consists of a single vertex and in this case, we simply take this vertex into the ruling set. The final ruling set for W is given by:

$$R = R_0 \cup \{u \in R_1 \mid \texttt{dist}(u, R_0) \geq \alpha\} .$$

Correctness: The correctness is shown by induction on the number of bits in the *current* ID of the vertex. We imagine that after dividing the nodes into W_0 and W_1, we omit the last bit in the ID of those nodes (as the last bit of all nodes in W_0 (resp., W_1) is the same, it is no longer informative), and continue working on nodes with $\log n - 1$ bits. In each level j of the recursion, we consider sets W' with $\log n - j$ many bits, thus the recursion has depth $O(\log n)$. We will prove by induction on the number of bits i, that the algorithm computes an $(\alpha, \alpha \cdot i)$ ruling sets for the sets W' in level $\log n - i$ (i.e., sets in which nodes have ID with i bits). Base (1 bit): a single vertex is simply taken into the ruling set (the leaf sets of the recursion tree) and thus the claim holds immediately. Step ($i + 1$ bits): Consider a set W' in level $\log n - (i + 1)$ with nodes of $i + 1$ bits. This set is partitioned into W'_0 and W'_1, each such set contains nodes with i many bits. By induction assumption for i bits, the algorithm computes a set R'_0 which is an $(\alpha, \alpha \cdot i)$ ruling set of W'_0, and set R'_0 which is an $(\alpha, \alpha \cdot i)$ ruling set of W'_1. Recall that the combined set R' takes all nodes in R_0 and omits nodes in R'_1 that are at distance at most α from R'_0. The separation property follows by definition, i.e., $\texttt{dist}(u, v, G) \geq \alpha$ for every $u, v \in R'$. Also the ruling property for the nodes in W'_0 is immediate as all the nodes in R'_0 are taken. It remains to show that $\texttt{dist}(u, R', G) \leq \alpha \cdot (i + 1)$ for every $u \in W'_1$. Let v be the closest vertex to u in R'_1. By induction assumption, $\texttt{dist}(v, u, G) \leq \alpha \cdot i$. If $v \in R'$, we are done. Otherwise, if $v \notin R'$, there some must be some vertex v' in R'_0 such that $\texttt{dist}(v, v', G) \leq \alpha$. We get that:

$$\texttt{dist}(u, R', G) \leq \texttt{dist}(u, v', G) \leq \texttt{dist}(u, v, G) + \texttt{dist}(v, v', G) \leq \alpha + \alpha \cdot i = \alpha \cdot (i + 1) .$$

See Fig. 1.1 for an illustration. Intuitively, in every step of the recursion, in the merging part, we get an additive of $+\alpha$ in the distance between the vertex to the current ruling set. Since there are $O(\log n)$ recursion levels, the overall distance to the closest vertex in the ruling set becomes $O(\alpha \log n)$.

Running time: The algorithm has $O(\log n)$ recursion levels. In each level, it works on all level-i subgraphs in parallel. The time consuming part in each level is to compute the merged ruling set R by excluding the nodes in R_1 that are at distance at most α from R_0. This step takes $O(\alpha)$ rounds, thus overall the running time is $O(\alpha \cdot \log n)$. Formally, the time complexity is $T(n) = T(n/2) + \alpha$, thus $T(n) = O(\alpha \log n)$. Note that in the sequential setting, when breaking up a problem into two problems of size $n/2$, we need to pay $2T(n/2)$, however, in the distributed setting, we are working on the two sub-problems in parallel, thus the total round complexity is $O(\alpha \log n)$. ■

Ruling sets allow one to compute vertex-disjoint clusters of small depths that are centered at the ruling-set nodes.

Definition 1.4 ((s, r) ruling forests) *Given a (s, r) ruling set R for W, an (s, r) ruling forest is a collection of trees $T_1, \ldots, T_{|R|}$, where each T_i is rooted at $u_i \in R$ such that: (i) $W \subseteq \bigcup_i V(T_i)$, (ii) the diameter of each tree is $O(r)$ and (iii) all trees are vertex disjoint.*

Lemma 1.5 (From Ruling Set to Ruling Forest) *Given a (s, r) ruling set for W, one can compute the (s, r) ruling forest in $O(r)$ rounds.*

Proof: All nodes in the ruling set R compute a BFS tree up to depth r simultaneously. Every vertex u joins the tree of the root of its closest vertex in R where ties are broken based on IDs. The parent of u in the chosen tree of root r is its neighbor u' that lies on the r-u shortest path (i.e., the neighbor from which it got a BFS message from the tree of u, if there are several such neighbors, break ties based on ID). It is easy to verify that the resulting trees are vertex-disjoint, cover all the nodes in W and each tree has depth at most r. ■

See Fig. 1.1 for illustration of ruling sets and their corresponding ruling forests.

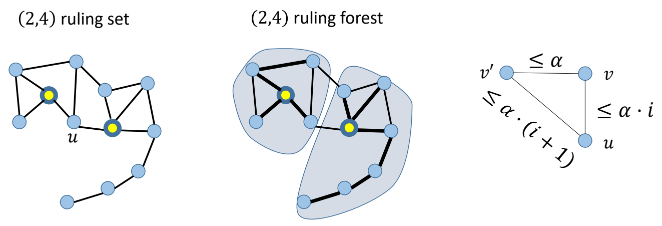

Figure 1.1: Left: Marked nodes are $(2,4)$ ruling set. Middle: bold edges are in the $(2,4)$ forest. Each vertex in V joins the tree of its closest representative in the ruling set, breaking ties based on ID. Right: Illustration for the correctness proof of Lemma 1.3.

Working on cluster graphs. In what follows, we will be working on graphs obtained by contracting vertex-disjoint clusters of small depth into a node. Recall that for a collection of vertex-disjoint clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ of (weak) diameter d, the cluster graph is given by $\widetilde{G} = (\mathcal{C}, \mathcal{E})$ where $\mathcal{E} = \{(C_i, C_j) \in \mathcal{E} \mid \exists u \in C_i, v \in C_j \text{ such that } (u,v) \in G\}$. We will repeatedly use the following observation:

Observation 1.6 *Every communication round in $\widetilde{G}$ can be simulated in the base graph G using $O(d)$ communication rounds. Thus any r-round algorithm on the graph $\widetilde{G}$ can be simulated in G using $O(d \cdot r)$ many rounds.*

Proof: A single round of communication in $\widetilde{G}$ is specified by a list of messages exchanged by neighbors in $\widetilde{G}$. To simulate this in G, we have a cluster leader in each cluster that plays the rule of the corresponding cluster node in $\widetilde{G}$. To send a message M from cluster C_i to its neighboring cluster C_j, we do as follows. The cluster of C_i sends the message M to all its cluster members. Let $u \in C_i$ and $v \in C_j$ be such that $(u,v) \in E(G)$. Since C_i and C_j are neighbors in the cluster graph $\widetilde{G}$ such u and v exist. The vertex $u \in C_i$ sends the message M to v, and then v sends the message to the leader of its cluster C_j. Since we have no bandwidth limitation, this can be done in parallel for all messages M' exchanged between neighboring clusters. Overall, exchanging messages between neighboring clusters takes $O(d)$ rounds. See Fig. 1.2. ■

High level description of the (d,c) network decomposition by [ALGP89]. Set a parameter λ, to be formally given later. The final number of colors in the decomposition will be bounded by λ. Partition the nodes into *light* nodes of degree at most $\lambda - 1$ and *heavy* nodes of degree at least λ:

$$L = \{v \in V \mid \deg(v,G) \leq \lambda - 1\} \text{ and } H = \{v \in V \mid \deg(v,G) \geq \lambda\} .$$

The algorithm handles the heavy and light nodes as follows.

Taking care of the heavy nodes. Clearly, we cannot expect coloring the heavy subgraph $G[H]$ with λ colors. The strategy is to cluster the heavy nodes into small depth clusters, such that the number of clusters is λ factor smaller than the number of the heavy nodes. We will then continue computing the decomposition recursively on the cluster graph obtained by contracting each cluster into a vertex. Towards that goal we compute a $(3, 3\log n)$ ruling set $R \subset H$ for the heavy nodes in G. The reason for setting the separation parameter to 3 is to guarantee that the number of nodes in R, and thus also the number of clusters, is small. Specifically, since the degree of each vertex in R is at least λ, and since every two nodes in the ruling-set do not have a common neighbor, we get that $|R| \leq n/\lambda$. The clusters containing all heavy nodes correspond to the trees of the $(3, 3\log n)$ ruling forest. Note that those trees might contain light vertices as well. The algorithm contract each such tree into a node and essentially continues working on a graph with n/λ many vertices. Note that while reducing the number of "nodes" by a factor of λ, when working on the cluster graph, we now pay a logarithmic factor for the communication in the cluster graph. As we will see, each recursive call reduces the number of "heavy" clusters by a factor of λ, but increases the diameter of the clusters in the cluster graph by a factor of $O(\log n)$.

Taking care of the remaining light nodes. It remains to take care of all unclustered nodes, i.e., nodes that do not belong to any of the clusters (trees of the ruling-set) in the previous step. By definition, all these nodes are *light*. Let us denote these nodes by $L' \subset L$. Since we have a budget of λ colors, we can color the graph $G[L']$ with λ colors using $O(\lambda \cdot \log n)$ rounds[4]. The clusters are simply the singleton clusters $\mathcal{C}(L') = \{\{v\} \mid v \in L'\}$.

Combining the Two Decompositions. So far, we have computed recursively a network decomposition for all nodes in $V \setminus L'$. We have also computed a network decomposition for $G[L']$. To combine these decompositions, we take the union of clusters obtained by both decompositions. All these clusters are indeed vertex disjoint (why?) and of small diameter. It remains to define the coloring of the clusters. The coloring of the heavy clusters (obtained by the decomposition of the heavy nodes) remains as is, and we will adapt the colors of the lights clusters. To do that, we will iterate over the λ color classes of L'. In each color class i, we consider all light clusters (singleton clusters in the first recursion level) of color i, and let them simultaneously pick a free color in $[1, \lambda]$. By the definition, each light cluster has at most $\lambda - 1$ neighbors, and clearly their degree cannot increase. After λ iterations, all colors are fixed.

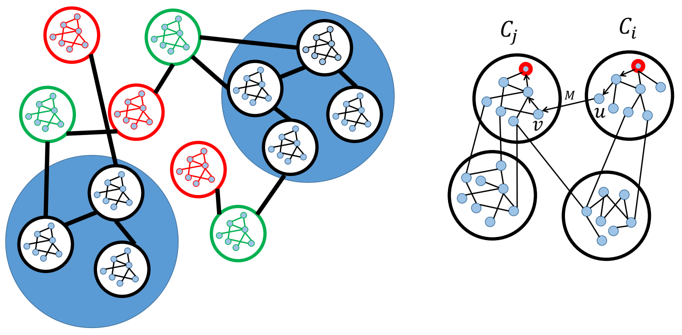

Figure 1.2: Left: Illustration for the second recursion level. Each vertex in the graph corresponds to a tree of depth $O(\log n)$ in the base graph G. Every communication step on the cluster graph takes $O(\log n)$ communication rounds in the base graph. The cluster graph is again partitioned into light and heavy nodes. The heavy "nodes" are clustered and the light nodes are colored with at most λ colors. Right: Communication in the cluster graph. Illustration for Obs. 1.6. The marked nodes are the cluster centers.

Analysis. We begin with several quick observations that can be shown by induction.

Observation 1.7 *(1) Every i-level cluster has at least λ^i many nodes; (2) The diameter of each i-level cluster is $O(\log n)^i$.*

Proof: Both claims follow by induction. Level-1 clusters contain at least λ nodes, since those clusters are formed by computing $(3, 3\log n)$ ruling set on the heavy nodes. Since the degree of each node in the ruling set is at least λ, and the distance between two nodes in the ruling set is at least 3, each cluster contains at least λ nodes (e.g., the unique neighbors of the root). In addition, the depth of each cluster is $O(\log n)$.

[4]Using a standard deterministic algorithm of $(\Delta + 1)$ coloring.

```
Algorithm NetworkDecomposition(G)
 1. Let
      $L = \{v \in V \mid \deg(v,G) \le \lambda - 1\}$ and $H = \{v \in V \mid \deg(v,G) \ge \lambda\}$ .
 2. Compute an $(3, 3\log n)$ ruling-set $R$ for $H$ in $G$.
 3. Compute an $(3, 3\log n)$ ruling-forest $\mathcal{T} = \{T_1, \ldots, T_\ell\}$ (where the trees might contain light nodes).
 4. $(\mathcal{C}(H), \Phi(H))$ = NetworkDecomposition($\widehat{G}$) where $\widehat{G} = (\mathcal{T}, \mathcal{E})$ is the cluster graph of the forest $\mathcal{T}$
    obtained by contracting each tree into a forest. Formally,
      $\mathcal{E} = \{(T_i, T_j) \mid \exists u \in T_i, v \in T_j \mid (u,v) \in G\}$.
 5. Let $L' = V \setminus \bigcup_i T_i$ be the light nodes not in the ruling forest.
 6. Set $\mathcal{C}(L') = \{\{v\} \mid v \in L'\}$.
 7. Compute a $\lambda$-coloring in $G[L']$.
 8. Let $\mathcal{C}(V) = \mathcal{C}(L') \cup \mathcal{C}(H)$.
 9. For every $i \in [1, \lambda]$:
      • Each color-$i$ vertex in $L'$ pick a free color in the currently colored cluster neighbors in $\mathcal{C}(V)$.
10. Denoting the new coloring by $\Phi^*(L')$, the final coloring $\Phi(V)$ is obtained by the coloring of $\Phi(H)$
    and the modified coloring $\Phi^*(L')$.
```

Figure 1.3: Construction of (d, c) Network Decomposition

Assume that those properties hold for level-$(i-1)$ clusters and consider level i. Each cluster of level-i contains at least λ level-$(i-1)$ clusters, (1) follows. Each cluster of level-i has depth of $O(\log n)$ in the cluster graph induced on the level-$(i-1)$ clusters, thus the final diameter of a level-i cluster is $O(\log n)^i$, (2) follows. ■

By Observation 1.7(i), the depth of the recursion is $O(\log_\lambda n)$. Note that in this last level of the recursion, every single communication round in the cluster graph is simulated in the base graph G within $(\log n)^{O(\log_\lambda n)}$ rounds (e.g., due to internal communication inside clusters). Specifically, computing a λ-coloring on the light "nodes" of this cluster graph now takes $\lambda \cdot \log n \cdot (\log n)^{O(\log_\lambda n)}$ rounds.

To minimize the running time we pick $\lambda = 2^{O(\sqrt{\log n \cdot \log\log n})}$. By plugging it in the bound of diameter, we get that the diameter of the decomposition is bounded by $2^{\sqrt{\log n \cdot \log\log n}}$ as well. Formally, letting $T(n)$ be the number of rounds for computing the decomposition on n-vertex graph, it holds that:

$$T(n) = O(\lambda \cdot \log n) + O(\log n) \cdot T(n/\lambda) ,$$

solving this recursive formula yields the same bounds of $2^{O(\sqrt{\log n \cdot \log\log n})}$ for the diameter, the number of colors and the number of rounds.

References

[ALGP89] Baruch Awerbuch, Michael Luby, Andrew V Goldberg, and Serge A Plotkin. Network decomposition and locality in distributed computation. In *30th Annual Symposium on Foundations of*

Computer Science, pages 364–369. IEEE, 1989.

[EN16] Michael Elkin and Ofer Neiman. Distributed strong diameter network decomposition. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 211–216. ACM, 2016.

[Lin87] Nathan Linial. Distributive graph algorithms global solutions from local data. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 331–335. IEEE, 1987.

[LS93] Nathan Linial and Michael Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

[Pel00] David Peleg. *Distributed Computing: A Locality-sensitive Approach.* SIAM, 2000.

[PS96] Alessandro Panconesi and Aravind Srinivasan. On the complexity of distributed network decomposition. *Journal of Algorithms*, 20(2):356–374, 1996.