58th Annual IEEE Symposium on Foundations of Computer Science

A Time Hierarchy Theorem for the LOCAL Model

Yi-Jun Chang
University of Michigan
Ann Arbor, MI
Email: cyijun@umich.edu

Seth Pettie
University of Michigan
Ann Arbor, MI
Email: pettie@umich.edu

***Abstract*—The celebrated *Time Hierarchy Theorem* for Turing machines states, informally, that more problems can be solved given more time. The extent to which a time hierarchy-type theorem holds in the classic distributed LOCAL model has been open for many years. In particular, it is consistent with previous results that all natural problems in the LOCAL model can be classified according to a small *constant* number of complexities, such as $O(1), O(\log^* n), O(\log n), 2^{O(\sqrt{\log n})}$, etc.**

In this paper we establish the first time hierarchy theorem for the LOCAL model and prove that several *gaps* exist in the LOCAL time hierarchy. Our main results are as follows:

- **We define an infinite set of simple coloring problems called *Hierarchical* $2\frac{1}{2}$*-Coloring*. A correctly colored graph can be confirmed by simply checking the neighborhood of each vertex, so this problem fits into the class of *locally checkable labeling* (LCL) problems. However, the complexity of the k-level Hierarchical $2\frac{1}{2}$-Coloring problem is $\Theta(n^{1/k})$, for $k \in \mathbb{Z}^+$. The upper and lower bounds hold for both general graphs and trees, and for both randomized and deterministic algorithms.**
- **Consider any LCL problem on *bounded degree trees*. We prove an automatic-speedup theorem that states that any *randomized* $n^{o(1)}$-time algorithm solving the LCL can be transformed into a *deterministic* $O(\log n)$-time algorithm. Together with a previous result, this establishes that on trees, there are no natural deterministic complexities in the ranges $\omega(\log^* n)$—$o(\log n)$ or $\omega(\log n)$—$n^{o(1)}$.**
- **We expose a gap in the *randomized* time hierarchy on general graphs. Roughly speaking, any randomized algorithm that solves an LCL problem in sublogarithmic time can be sped up to run in $O(T_{LLL})$ time, which is the complexity of the distributed Lovász local lemma problem, currently known to be $\Omega(\log\log n)$ and $2^{O(\sqrt{\log\log n})}$ on bounded degree graphs.**

Finally, we revisit Naor and Stockmeyer's characterization of $O(1)$-time LOCAL algorithms for LCL problems (as *order-invariant w.r.t. vertex IDs*) and calculate the complexity gaps that are directly implied by their proof. For n-rings we see a $\omega(1)$—$o(\log^* n)$ complexity gap, for $(\sqrt{n} \times \sqrt{n})$-tori an $\omega(1)$—$o(\sqrt{\log^* n})$ gap, and for bounded degree trees and general graphs, an $\omega(1)$—$o(\log(\log^* n))$ complexity gap.

***Keywords*-distributed algorithm; locally checkable labeling; local model; time hierarchy theorem;**

I. Introduction

The goal of this paper is to understand the spectrum of natural problem complexities that can exist in the LOCAL model [1], [2] of distributed computation, and to quantify the value of randomness in this model. Whereas the time hierarchy of Turing machines is known[1] to be very "smooth", recent work [5], [6] has exhibited strange *gaps* in the LOCAL complexity hierarchy of LCL problems. Indeed, prior to this work it was not even known if the LOCAL model could support more than a small *constant* number of problem complexities (for LCL problems). Before surveying prior work in this area, let us formally define the deterministic and randomized variants of the LOCAL model, and the class of *locally checkable labeling* (LCL) problems, which are intuitively those graph problems that can be computed locally in *nondeterministic constant time*.

In both the DetLOCAL and RandLOCAL models the input graph $G = (V, E)$ and communications network are identical. Each vertex hosts a processor and all vertices run the same algorithm. Each edge supports communication in both directions. The computation proceeds in synchronized *rounds*. In a round, each processor performs some computation and sends a message along each incident edge, which is delivered before the beginning of the next round. Each vertex v is initially aware of its degree $\deg(v)$, a port numbering mapping its incident edges to $\{1, \dots, \deg(v)\}$, certain global parameters such as $n \overset{\text{def}}{=} |V|$, $\Delta \overset{\text{def}}{=} \max_{v \in V} \deg(v)$, and possibly other information. The assumption that global parameters are common knowledge can sometimes be removed; see Korman, Sereni, and Viennot [7]. The only measure of efficiency is the number of rounds. All local computation is free and the size of messages is unbounded. *Henceforth "time" refers to the number of rounds.* The differences between DetLOCAL and RandLOCAL are as follows.

- DetLOCAL: In order to avoid trivial impossibilities, all vertices are assumed to hold unique $\Theta(\log n)$-bit IDs. Except for the information about $\deg(v), \mathrm{ID}(v)$, and the port numbering, the initial state of v is identical to every other vertex. The algorithm executed at each vertex is deterministic.
- RandLOCAL: In this model each vertex may locally generate an unbounded number of independent truly

Supported by NSF Grants CCF-1514383 and CCF-1637546.

[1]For *any* time-constructible function $T(n)$, there is a problem solvable in $O(T(n))$ but not $o(T(n))$ time [3], [4].

random bits. There are no globally shared random bits. Except for the information about $\deg(v)$ and its port numbering, the initial state of v is identical to every other vertex. Algorithms in this model operate for a specified number of rounds and have some probability of *failure*, the definition of which is problem specific. We set the maximum tolerable global probability of failure to be $1/n$.

Clearly RandLOCAL algorithms can generate distinct IDs (w.h.p.) if desired. Observe that the role of "n" is different in the two LOCAL models: in DetLOCAL it affects the ID length whereas in RandLOCAL it affects the failure probability.

LCL Problems: Naor and Stockmeyer [8] introduced *locally checkable labelings* to formalize a large class of natural graph problems. Fix a class $\mathcal{G}$ of possible input graphs and let Δ be the maximum degree in any such graph. Formally, an LCL problem $\mathcal{P}$ for $\mathcal{G}$ has a radius r, constant size input and output alphabets $\Sigma_{\text{in}}, \Sigma_{\text{out}}$, and a set $\mathcal{C}$ of acceptable configurations. All of these parameters may depend on Δ. Each $C \in \mathcal{C}$ is a graph centered at a specific vertex, in which each vertex has a degree, a port numbering, and two labels from Σ_{in} and Σ_{out}. Given the input graph $G(V, E, \phi_{\text{in}})$ where $\phi_{\text{in}} : V(G) \to \Sigma_{\text{in}}$, an acceptable output is any function $\phi_{\text{out}} : V(G) \to \Sigma_{\text{out}}$ such that for each $v \in V(G)$, the subgraph induced by $N^r(v)$ (denoting the r-neighborhood of v together with information stored there: vertex degrees, port numberings, input labels, and output labels) is isomorphic to a member of $\mathcal{C}$. An LCL can be described explicitly by enumerating a finite number of acceptable configurations. LCLs can be generalized to graph classes with unbounded degrees.

Many natural symmetry breaking problems can be expressed as LCLs, such as MIS, maximal matching, (α, β)-ruling sets, $(\Delta+1)$-vertex coloring, and sinkless orientation.

A. The Complexity Landscape of LOCAL

The complexity landscape for LCL problems is defined by "natural" complexities (sharp lower and upper bounds for specific LCL problems) and provably empty *gaps* in the complexity spectrum. We now have an almost perfect understanding of the complexity landscape for two simple topologies: n-rings [9], [1], [10], [8], [5] and $(\sqrt{n} \times \sqrt{n})$-tori [8], [5], [6]. See Figure 1, Top and Middle. On the n-ring, the only possible problem complexities are $O(1)$, $\Theta(\log^* n)$ (e.g., 3-coloring), and $\Theta(n)$ (e.g., 2-coloring, if bipartite). The gaps between these three complexities are obtained by *automatic speedup theorems*. Naor and Stockmeyer's [8] characterization of $O(1)$-time LCL algorithms actually implies that any $o(\log^* n)$-time algorithm on the n-ring can be transformed to run in $O(1)$ time; see Appendix. Chang, Kopelowitz, and Pettie [5] showed that any $o(n)$-time RandLOCAL algorithm can be made to run in $O(\log^* n)$ time in DetLOCAL.

The situation with $(\sqrt{n} \times \sqrt{n})$-tori is almost identical [6]: every known LCL has complexity $O(1)$, $\Theta(\log^* n)$ (e.g., 4-coloring), or $\Theta(\sqrt{n})$ (e.g., 3-coloring). Whereas the gap implied by [8] is $\omega(1)$—$o(\log^* n)$ on the n-ring, it is $\omega(1)$—$o(\sqrt{\log^* n})$ on the $(\sqrt{n} \times \sqrt{n})$-torus; see Appendix.[2] Whereas randomness is known not to help in n-rings [8], [5], it is an open question on tori [6]. Whereas the classification question is decidable on n-rings (whether an LCL is $O(\log^* n)$ or $\Omega(n)$, for example) this question is *undecidable* on $(\sqrt{n} \times \sqrt{n})$-tori [8], [6].

The gap theorems of Chang et al. [5] show that no LCL problem on general graphs has DetLOCAL complexity in the range $\omega(\log^* n)$—$o(\log_\Delta n)$, nor RandLOCAL complexity in the range $\omega(\log^* n)$—$o(\log_\Delta \log n)$. Some problems exhibit an exponential separation ($O(\log_\Delta \log n)$ vs. $\Omega(\log_\Delta n)$) between their RandLOCAL and DetLOCAL complexities, such as Δ-coloring degree-Δ trees [11], [5] and sinkless orientation [11], [12]. More generally, Chang et al. [5] proved that the RandLOCAL complexity of *any* LCL problem on graphs of size n is, holding Δ fixed, at least its deterministic complexity on instances of size $\sqrt{\log n}$. Thus, on the class of degree $\Delta = O(1)$ graphs there were only five known natural complexities: $O(1)$, $\Theta(\log^* n)$, randomized $\Theta(\log\log n)$, $\Theta(\log n)$, and $\Theta(n)$. For non-constant Δ, the RandLOCAL lower bounds of Kuhn, Moscibroda, and Wattenhofer [13] imply $\Omega(\min\{\frac{\log \Delta}{\log\log \Delta}, \sqrt{\frac{\log n}{\log\log n}}\})$ lower bounds on $O(1)$-approximate vertex cover, MIS, and maximal matching. This $\Omega(\log \Delta / \log\log \Delta)$ lower bound is only known to be tight for $O(1)$-approximate vertex cover [14]; the best maximal matching [15] and MIS [16] algorithms' dependence on Δ is $\Omega(\log \Delta)$. The $\Omega(\sqrt{\frac{\log n}{\log\log n}})$ lower bound is not known to be tight for any problem, but is almost tight for maximal matching on bounded arboricity graphs [15], e.g., trees or planar graphs.

New Results: In this paper we study the LOCAL complexity landscape on more general topologies: bounded degree trees and general graphs; see Figure 1, Bottom. We establish a new complexity gap for trees, a complexity gap for general graphs based on the distributed complexity of the constructive Lovász local lemma, and a new infinite hierarchy of coloring problems with polynomial time complexities. In more detail,

- We prove that on the class of degree bounded trees, no LCL has complexity in the range $\omega(\log n)$—$n^{o(1)}$. Specifically, any $n^{o(1)}$-time RandLOCAL algorithm can be converted to an $O(\log n)$-time DetLOCAL algorithm. Moreover, given a description of an LCL problem $\mathcal{P}$, it is *decidable* whether the RandLOCAL complexity of $\mathcal{P}$ is $n^{\Omega(1)}$ or the DetLOCAL complexity

[2]J. Suomela (personal communication, 2017) has a proof that there is an $\omega(1)$—$o(\log^* n)$ complexity gap for tori, at least for LCLs that do not use port numberings or input labels. The issues that arise with port numbering and input labels can be very subtle.

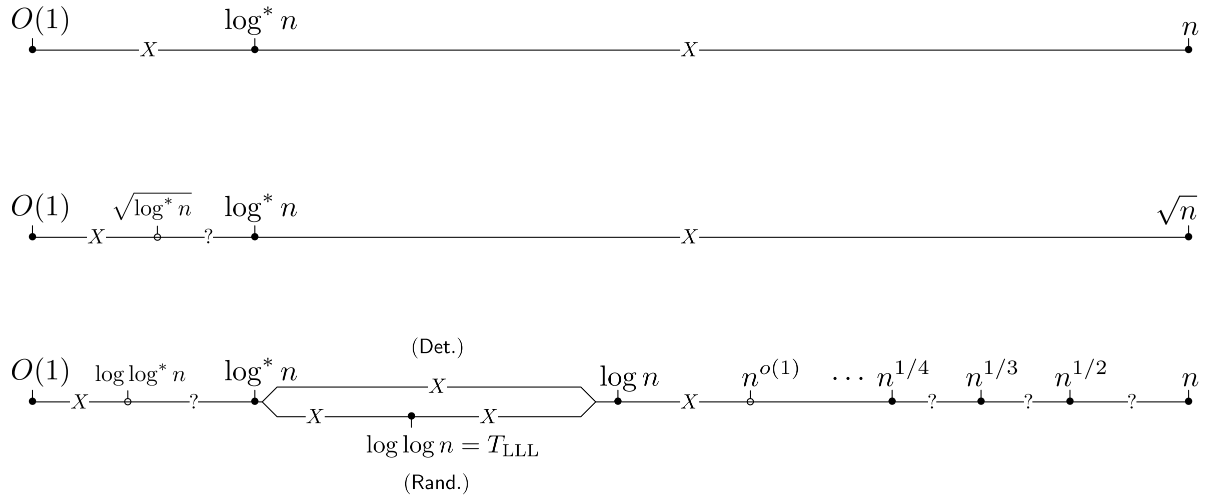


Figure 1. **Top:** the complexity landscape for LCL problems on the n-ring. **Middle:** the complexity landscape for LCL problems on the $(\sqrt{n} \times \sqrt{n})$-torus. Refer to [8], [5], [6] and Appendix for proofs of the complexity gaps ('X') on rings and tori. **Bottom:** the complexity landscape for LCL problems on bounded degree trees. The $\omega(\log^* n)$—$o(\log n)$ DetLOCAL gap and $\omega(\log^* n)$—$o(\log\log n)$ RandLOCAL gap are due to [5]. The $\omega(T_{LLL})$—$o(\log n)$ and $\omega(\log n)$—$n^{o(1)}$ gaps are new. Refer to Appendix for the $\omega(1)$—$o(\log(\log^* n))$ gap. It is unknown whether there are $\omega(n^{1/(k+1)})$—$o(n^{1/k})$ gaps. With the exception of the $\omega(\log n)$—$n^{o(1)}$ gap and the complexity of T_{LLL}, this is exactly the known complexity landscape for general bounded degree graphs as well.

of $\mathcal{P}$ is $O(\log n)$. It turns out that this gap is maximal: we cannot extend it lower than $\omega(\log n)$ [1], [5], nor higher than $n^{o(1)}$, as we show below.

- We define an infinite class of LCL problems called *Hierarchical* $2\frac{1}{2}$*-Coloring*. We prove that k-level Hierarchical $2\frac{1}{2}$-Coloring has complexity $\Theta(n^{1/k})$. The upper bound holds in DetLOCAL on general graphs, and the lower bound holds even on degree-3 trees in RandLOCAL. Thus, in contrast to rings and tori, trees and general graphs support an *infinite* number of natural problem complexities.
- Suppose we have a RandLOCAL algorithm for general graphs running in $C(\Delta) + o(\log_\Delta n)$ time. We can transform this algorithm to run in $O(C(\Delta) \cdot T_{LLL})$ time, where T_{LLL} is the complexity of a weak (i.e., "easy") version of the constructive Lovász local lemma. The complexity T_{LLL} seems to be sensitive to the precise LLL criterion, whether randomness is allowed, the dependency graph topology and its maximum degree; refer to [17] for a survey of distributed LLL algorithms [18], [19], [11], [5], [20]. At present, T_{LLL} is known to be $\Omega(\log\log n)$ [11] (even on trees [17]), $2^{O(\sqrt{\log\log n})}$ on *bounded degree graphs* [20], and $\Theta(\log\log n)$ on trees [17]. Therefore, our result implies new RandLOCAL complexity gaps $2^{\omega(\sqrt{\log\log n})}$—$o(\log n)$ for bounded degree graphs and $\omega(\log\log n)$—$o(\log n)$ for bounded degree trees.

Finally, it seems to be folklore that Naor and Stockmeyer's work [8] implies *some kind* of complexity gap, which has been cited as $\omega(1)$—$o(\log^* n)$ [6, p. 2]. However, to our knowledge, no proof of this complexity gap has been published. We show how Naor and Stockmeyer's approach implies complexity gaps that depend on the graph topology:

— $\omega(1)$—$o(\log^* n)$ on rings.
— $\omega(1)$—$o(\sqrt{\log^* n})$ on tori.
— $\omega(1)$—$o(\log(\log^* n))$ on bounded degree trees and general graphs.

These gaps apply to the general class of LCL problems defined in this paper, in which vertices initially hold an input label and possible port numbering. Port numberings are needed to represent "edge labeling" problems (like maximal matching, edge coloring, and sinkless orientation) unambiguously as vertex labelings. They are not needed for native "vertex labeling" problems like $(\Delta + 1)$-coloring or MIS. J. Suomela (personal communication) gave a proof that the $\omega(1)$—$o(\log^* n)$ gap exists in tori as well, for the class of LCL problems without input labels or port numbering; see Appendix.

Commentary: Our $\omega(\log n)$—$n^{o(1)}$ complexity gap for trees is interesting from both a technical and greater philosophical perspective, due to the fact that many natural problems have been "stuck" at $n^{o(1)}$ complexities for decades. Any DetLOCAL algorithm that relies on network decompositions [21] currently takes $2^{O(\sqrt{\log n})}$ time. If our automatic speedup theorem could be extended to the class of all graphs, this would immediately yield $O(\log n)$-time algorithms for MIS, $(\Delta+1)$-coloring, and many other LCLs.

All the existing automatic speedup theorems are quite different in terms of proof techniques. Naor and Stockmeyer's approach is based on Ramsey theory. The speedup theorems of [5], [6] use the fact that $o(\log_\Delta n)$ algorithms on general graphs (and $o(n)$ algorithms on n-rings and $o(\sqrt{n})$ algorithms on $(\sqrt{n} \times \sqrt{n})$-tori) cannot "see" the whole graph, and can therefore be efficiently tricked into thinking the graph has constant size. Our $n^{o(1)} \to O(\log n)$ speedup theorem introduces an entirely new set of techniques based on classic automata theory. We show that any LCL problem gives rise to a regular language that represents partial labelings of the tree that can be consistently extended to total lablelings. By applying the pumping lemma for regular languages, we can "pump" the input tree into a much larger tree that behaves similar to the original tree. The advantage of creating a larger imaginary tree is that each vertex can (mentally) simulate the behavior of an $n^{o(1)}$-time algorithm on the *imaginary* tree, merely by inspecting its $O(\log n)$-neighborhood in the *actual* tree. Moreover, because the pumping operation preserves properties of the original tree, a labeling of the imaginary tree can be efficiently converted to a labeling of the original tree.

B. Related Results

There are several LOCAL lower bounds for natural problems that do not quite fit in the LCL framework. Göös, Hirvonen, and Suomela [22] proved a sharp $\Omega(\Delta)$ lower bound for fractional maximal matching and Göös and Suomela proved $\Omega(\log n)$ lower bounds on $(1+\delta)$-approximating the minimum vertex cover, $\delta > 0$, even on degree-3 graphs. See [23], [24] for lower bounds on coloring problems that apply to constrained algorithms or a constrained version of the LOCAL model.

In recent years there have been efforts to develop a complexity theory of locality. The gap theorems of [8], [5], [6] have already been discussed. Suomela surveys [25] the class of problems that can be computed with $O(1)$ time. Fraigniaud et al. [26] defined a distributed model for locally deciding graph properties; see [27] for a survey of variants of the local distributed decision model. Göös and Suomela [28] considered the *proof* complexity (measured in terms of bits-per-vertex label) of locally verifying graph properties. Very recently, Ghaffari, Kuhn, and Maus [29] defined the SLOCAL model (sequential LOCAL) and exhibited several *complete* problems for this model, inasmuch as a $\mathrm{polylog}(n)$-time DetLOCAL algorithm for any complete problem implies a $\mathrm{polylog}(n)$ DetLOCAL algorithm for every $\mathrm{polylog}(n)$-time problem in SLOCAL.[3]

C. Organization

In Section II we introduce Hierarchical $2\frac{1}{2}$-Coloring and prove that the k-level variant of this problem has complexity $\Theta(n^{1/k})$. In Section III we prove the $n^{o(1)} \to O(\log n)$ speedup theorem for bounded degree trees. In Section IV we discuss the constructive Lovász local lemma and prove the $o(\log_\Delta n) \to T_{LLL}$ randomized speedup theorem. In Section V we discuss open problems and outstanding conjectures. Appendix reviews Naor and Stockmeyer's characterization of $O(1)$-time LCL algorithms, using Ramsey theory, and explains how it implies gaps in the complexity hierarchy that depend on graph topology.

II. An Infinitude of LOCAL Complexities: Hierarchical $2\frac{1}{2}$-Coloring

In this section we give an infinite sequence $(\mathcal{P}_k)_{k\in\mathbb{Z}^+}$ of LCL problems, where the complexity of $\mathcal{P}_k$ is precisely $\Theta(n^{1/k})$.[4] The upper bound holds on general graphs in DetLOCAL and the lower bound holds in RandLOCAL, even on degree-3 trees. Informally, the task of $\mathcal{P}_k$ is to 2-color (with $\{♀, ♂\}$) certain specific subgraphs of the input graph. Some vertices are *exempt* from being colored (in which case they are labeled ♄), and in addition, it is possible to decline to 2-color certain subgraphs, by labeling them ☿.

There are no input labels. The output label set is $\Sigma_{\text{out}} = \{♀, ♂, ☿, ♄\}$.[5] The problem $\mathcal{P}_k$ is an LCL defined by the following rules.

Levels. Subsequent rules depend on the *levels* of vertices. Let V_i, $i \in \{1, \ldots, k+1\}$, be the set of vertices on level i, defined as follows.

$$\begin{aligned} G_1 &= G \\ G_i &= G_{i-1} - V_{i-1}, && \text{for } i \in [2, k+1] \\ V_i &= \{v \in V(G_i) \mid \deg_{G_i}(v) \leqslant 2\}, && \text{for } i \in [1, k] \\ V_{k+1} &= V(G_{k+1}) \end{aligned}$$

Remember that vertices know their degrees, so a vertex in V_1 deduces this with 0 rounds of communication. In general the level of v can be calculated from information in $N^k(v)$.

Exemption. A vertex labeled ♄ is called *exempt*. No V_1 vertex is labeled ♄; all V_{k+1} vertices are labeled ♄. Any V_i vertex is labeled ♄ iff it is adjacent to a lower level vertex labeled ♀, ♂, or ♄. Define $D_i \subseteq V_i$ to be the set of level i exempt vertices.

Two-Coloring. Vertices not covered by the exemption rule are labeled one of ♀, ♂, ☿.

— Any vertex in V_i, $i \in [1, k]$, labeled ♀ has no neighbor in V_i labeled ♀ or ☿.
— Any vertex in V_i, $i \in [1, k]$, labeled ♂ has no neighbor in V_i labeled ♂ or ☿.
— Any vertex in $V_k - D_k$ with exactly 0 or 1 neighbors in $V_k - D_k$ must be labeled ♀ or ♂.

[3]The class of $O(1)$-time SLOCAL algorithms is, roughly speaking, those graph labelings that can be computed sequentially, by a truly local algorithm. This class is a *strict* subset of LCLs.

[4]Brandt et al. [6, Appendix A.3] described an LCL that has complexity $\Theta(\sqrt{n})$ on general graphs, but not trees. It may be possible to generalize their LCL to any complexity of the form $\Theta(n^{1/k})$.

[5]Venus, Mars, Mercury, Saturn.

Commentary: The Level rule states that the graph induced by V_i consists of paths and cycles. The Two-Coloring rule implies that each component of non-exempt vertices in the graph induced by $V_i - D_i$ must either (a) be labeled uniformly by ⚥ or (b) be properly 2-colored by $\{♀, ♂\}$. Every *path* in $V_k - D_k$ must be properly 2-colored, but *cycles* in $V_k - D_k$ are allowed to be labeled uniformly by ⚥. This last provision is necessary to ensure that *every* graph can be labeled according to $\mathcal{P}_k$ since there is no guarantee that cycles in $V_k - D_k$ are bipartite.

Remark 1. *As stated $\mathcal{P}_k$ is an LCL with an alphabet size of 4 and a radius k, since the coloring rules refer to levels, which can be deduced by looking up to radius k. On the other hand, we can also represent $\mathcal{P}_k$ as an LCL with radius 1 and alphabet size $4k$ by including a vertex's level in its output label. A correct level assignment can be verified within radius 1. For example, level 1 vertices are those with degree at most 2, and a vertex is labeled $i \in [2, k]$ iff all but at most 2 neighbors have levels less than i.*

Theorem 1. *The* DetLOCAL *complexity of $\mathcal{P}_k$ on general graphs is $O(n^{1/k})$.*

Proof: The algorithm fixes the labeling of $V_1, \ldots, V_k, V_{k+1}$ in order, according to the following steps. Assume that all vertices in $V_1, \ldots, V_{i-1}$ have already been labeled.

- Compute D_i according to the Exemption rule. (E.g., $D_1 = \emptyset$, $D_{k+1} = V_{k+1}$.)
- Each path in the subgraph induced by $V_i - D_i$ calculates its length. If it contains at most $\lceil 2n^{1/k} \rceil$ vertices, it properly 2-colors itself with $\{♀, ♂\}$; longer paths and cycles in $V_i - D_i$ label themselves uniformly by ⚥.

This algorithm correctly solves $\mathcal{P}_k$ provided that it never labels a path in $V_k - D_k$ with ⚥. Let U_i be the subgraph induced by those vertices in $V_1 \cup \cdots \cup V_i$ labeled ⚥. Consider a connected component C in U_i whose V_i-vertices are arranged in a path (not a cycle). We argue by induction that C has at least $2n^{i/k}$ vertices. This is clearly true in the base case $i = 1$: if a path component of U_1 were colored ⚥, it must have more than $\lceil 2n^{1/k} \rceil$ vertices. Now assume the claim is true for $i - 1$ and consider a component C of U_i. If the V_i-vertices in C form a path, it must have length greater than $2n^{1/k}$. Each vertex in that path must be adjacent to an endpoint of a V_{i-1} path. Since V_{i-1} paths have two endpoints, the V_i path is adjacent to at least $\lceil 2n^{1/k} \rceil / 2 \geqslant n^{1/k}$ components in U_{i-1}, each of which has size at least $2n^{(i-1)/k}$, by the inductive hypothesis. Thus, the size of C is at least $n^{1/k} \cdot 2n^{(i-1)/k} + 2n^{1/k} > 2n^{i/k}$. Because there are at most n vertices in the graph, it is impossible for V_k vertices arranged in a path to be colored ⚥. ■

Theorem 2. *The* RandLOCAL *complexity of $\mathcal{P}_k$ on trees with maximum degree $\Delta = 3$ is $\Omega(n^{1/k})$.*

Proof: Fix an integer parameter x and define a sequence of graphs $(H_i)_{1 \leqslant i \leqslant k}$ as follows. Each H_i has a *head* and a *tail*.

- H_1 is a path (or *backbone*) of length x. One end of the path is the head and the other end the tail.
- To construct H_i, $i \in [2, k-1]$, begin with a *backbone* path $(v_1, v_2, \ldots, v_x)$, with head v_1 and tail v_x. Form $x + 1$ copies $(H_{i-1}^{(j)})_{1 \leqslant j \leqslant x+1}$ of H_{i-1}, where $v^{(j)}$ is the head of $H_{i-1}^{(j)}$. Connect $v^{(j)}$ to v_j by an edge, for $j \in [1, x]$, and also connect $v^{(x+1)}$ to v_x by an edge.
- H_k is constructed exactly as above, except that we generate $x + 2$ copies of H_{k-1} and connect the heads of two copies of H_{k-1} to both v_1 and v_x. See Figure 2 for an example with $k = 3$.

Let us make several observations about the construction of H_k. First, it is a tree with maximum degree 3. Second, when decomposing $V(H_k)$ into levels $(V_1, \ldots, V_k, V_{k+1})$, V_i is precisely the union of the backbones in all copies of H_i, and $V_{k+1} = \emptyset$. Third, the number of vertices in H_k is $\Theta(x^k)$, so a $o(n^{1/k})$ algorithm for $\mathcal{P}_k$ must run in $o(x)$ time on H_k.

Consider a RandLOCAL algorithm $\mathcal{A}$ solving $\mathcal{P}_k$ on H_k within $t < x/5 - O(1)$ time, that fails with probability p_{fail}. If $\mathcal{A}$ is a good algorithm then $p_{\text{fail}} \leqslant 1/|V(H_k)|$. However, we will now show that p_{fail} is constant, independent of $|V(H_k)|$.

Define $\mathcal{E}_i$ to be the event that $D_i \neq \emptyset$ and $p_i = \Pr(\mathcal{E}_i)$. By an induction from $i = 2$ to k, we prove that $p_i \leqslant 2(i - 1) \cdot p_{\text{fail}}$.

Base case. We first prove that $\Pr(H_k$ is not correctly colored according to $\mathcal{P}_k \mid \mathcal{E}_2) \geqslant 1/2$.

Conditioning on $\mathcal{E}_2$ means that $D_2 \neq \emptyset$. Fix any $v \in D_2$ and let P be a copy of H_1 (a path) adjacent to v. In order for $v \in D_2$, it must be that P is properly 2-colored with $\{♀, ♂\}$. Since $t < x/5 - O(1)$, there exist two vertices u and u' in P such that

1) $N^t(u)$, $N^t(u')$, and $N^t(v)$ are disjoint sets,
2) the subgraphs induced by $N^t(u)$ and $N^t(u')$ are isomorphic, and
3) the distance between u and u' is odd.

Let $p_♀$ and $p_♂$ be the probabilities that u/u' is labeled ♀ and ♂, respectively. A proper 2-coloring of P assigns u and u' different colors, and that occurs with probability $2p_♀ p_♂ \leqslant 2p_♀(1 - p_♀) \leqslant 1/2$. Moreover, this holds independent of the random bits generated by vertices in $N^t(v)$. The algorithm fails unless u, u' have different colors, thus $p_{\text{fail}} \geqslant p_2/2$, and hence $p_2 \leqslant 2 \cdot p_{\text{fail}}$.

Inductive Step. Let $3 \leqslant i \leqslant k$. The inductive hypothesis states that $p_{i-1} \leqslant 2(i-2) \cdot p_{\text{fail}}$. By a proof similar to the base case, we have: $\Pr(H_k$ is not correctly colored according to $\mathcal{P}_k \mid \mathcal{E}_i \backslash \mathcal{E}_{i-1}) \geqslant 1/2$.

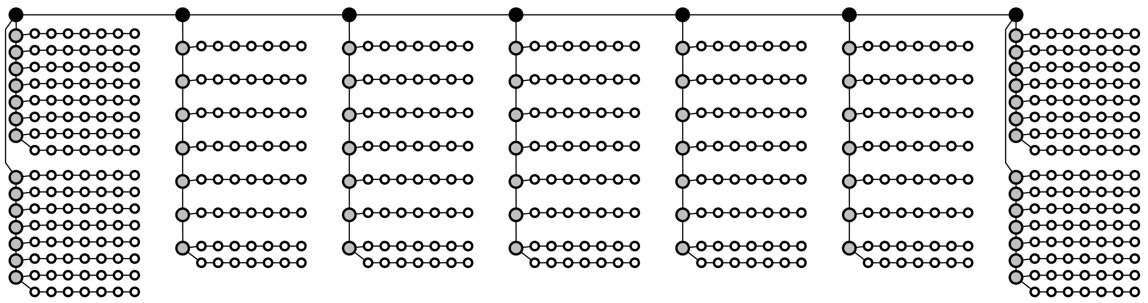


Figure 2. The graph H_k with parameters $k = 3, x = 7$. White vertices are in V_1, gray in V_2, and black in V_3. $V_4 = V_{k+1}$ is empty.

We are conditioning on $\mathcal{E}_i \backslash \mathcal{E}_{i-1}$. If this event is empty, then $p_i \leqslant p_{i-1} \leqslant 2(i-2) \cdot p_{\text{fail}}$ and the induction is complete. On the other hand, if $\mathcal{E}_i \backslash \mathcal{E}_{i-1}$ holds then there is some $v \in D_i$ adjacent to a copy of H_{i-1} with backbone path P, where $P \cap D_{i-1} = \varnothing$. In other words, if H_k is colored according to $\mathcal{P}_k$ then P must be properly 2-colored with $\{♀, ♂\}$. The argument above shows this occurs with probability at least 1/2. Thus,

$$\begin{aligned} p_{\text{fail}} &= \Pr(H_k \text{ is incorrectly colored}) \\ &\geqslant \Pr(\mathcal{E}_i \backslash \mathcal{E}_{i-1})/2 \\ &\geqslant (p_i - p_{i-1})/2, \end{aligned}$$

or $p_i \leqslant 2p_{\text{fail}} + p_{i-1} \leqslant 2(i-1)p_{\text{fail}}$, completing the induction.

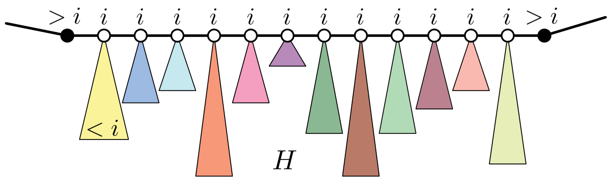
Finally, let P be the path induced by vertices in V_k. The probability that $\mathcal{E}_k$ holds ($P \cap D_k \neq \varnothing$) is $p_k \leqslant 2(k-1) \cdot p_{\text{fail}}$. On the other hand, $\Pr(H_k \text{ not colored correctly} \mid \overline{\mathcal{E}_k}) \geqslant 1/2$ by the argument above, hence $p_{\text{fail}} \geqslant (1-p_k)/2$, or $p_k \geqslant 1 - 2p_{\text{fail}}$. Combining the upper and lower bounds on p_k we conclude that $p_{\text{fail}} \geqslant (2k)^{-1}$ is constant, independent of $|V(H_k)|$. Thus, algorithm $\mathcal{A}$ cannot succeed with high probability. ■

III. A Complexity Gap on Bounded Degree Trees

In this section we prove an $n^{o(1)} \to O(\log n)$ speedup theorem for LCL problems on bounded degree trees. Due the page limit, we only present the high level structure of the proof and the key ideas. See [30] for the full proof. Throughout, $\mathcal{P}$ is a radius-r LCL and $\mathcal{A}$ is an $n^{o(1)}$-time algorithm for $\mathcal{P}$ on bounded degree trees.

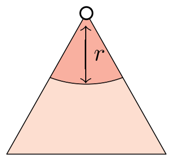
Consider this simple way to decompose a tree in $O(\log n)$ time, inspired by Miller and Reif [31]. Iteratively remove paths of degree-2 vertices (*compress*) and vertices with degree 0 or 1 (*rake*). Vertices removed in iteration i are at *level* i. If $O(\log n)$ *rakes* alone suffice to decompose a tree then it has $O(\log n)$ diameter and any LCL can be solved in $O(\log n)$ time on such a graph. Thus, we mainly have to worry about the situation where *compress* removes very long ($\omega(1)$-length) paths.

The first observation is that it is easy to split up long degree-2 paths of level-i vertices into constant length paths, by artificially promoting a well-spaced subset of level-i vertices to level $i+1$. Thus, we have a situation that looks like this: level-i vertices are arranged in an $O(1)$-length path, each the root of a (colored) subtree of level-($< i$) vertices that were removed in previous rake/compress steps, and bookended by level-($> i$) (black) vertices. Call the subgraph between the bookends H.

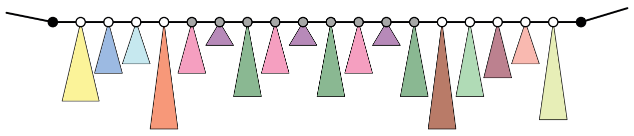


In our approach it is the level-($> i$) vertices that are in charge of coordinating the labeling of level-($\leqslant i$) vertices in their purview. In this diagram, H is in the purview of both black bookends. We only have one tool available for computing a labeling of this subgraph: an $n^{o(1)}$-time RandLOCAL algorithm $\mathcal{A}$ that works w.h.p. What would happen if we *simulated* $\mathcal{A}$ on H? The simulation would fail catastrophically of course, since it needs to look up to an $n^{o(1)}$ radius, to parts of the graph far outside of H.

Note that the colored subtrees are unbounded in terms of size and depth. Nonetheless, they fall into a *constant* number of equivalence classes in the following sense. The *class* of a rooted tree is the set of all labelings of the r-neighborhood of its root that can be extended to total labelings of the tree that are consistent with $\mathcal{P}$.



In other words, the large and complex graph H can be succinctly encoded as a simple class vector $(c_1, c_2, \ldots, c_\ell)$, where c_j is the class of the jth colored tree. Consider the set of all labelings of H that are consistent with $\mathcal{P}$. This set can also be succinctly represented by listing the labelings of the r-neighborhoods of the bookends that can be extended to all of H, while respecting $\mathcal{P}$. The set of these partial labelings defines the *type* of H. We show that the type of H can be computed by a finite automaton that reads the class vector $(c_1, \ldots, c_\ell)$ one character at a time. By the pigeonhole principle, if ℓ is sufficiently large then the automaton loops, meaning that $(c_1, \ldots, c_\ell)$ can be written as $x \circ y \circ z$, which has the same type as every $x \circ y^j \circ z$, for all $j \geqslant 1$. This *pumping lemma for trees* lets us dramatically expand the size of H without affecting its type, i.e., how it interacts with the outside world beyond the bookends.



This diagram illustrates the pumping lemma with a substring of $|y| = 3$ trees (rooted at gray vertices) repeated $j = 3$ times. Now let us reconsider the simulation of $\mathcal{A}$. If we first pump H to be long enough, and then simulate $\mathcal{A}$ on the middle section of pumped-H, $\mathcal{A}$ must, according to its $n^{o(1)}$ time bound, compute a labeling *without needing any information outside of pumped-H*, i.e., beyond the bookends. Thus, we can use $\mathcal{A}$ to *pre-commit* to a labeling of a small (radius-r) subgraph of pumped-H. Given this pre-commitment, the left and right bookends no longer need to coordinate their activities: everything left (right) of the pre-committed zone is now in the purview of the left (right) bookend. Interestingly, these manipulations (tree surgery and pre-commitments) can be repeated for each i, yielding a hierarchy of *imaginary* trees such that a proper labeling at one level of the hierarchy implies a proper labeling at the previous level.

Theorem 3. *Let $\mathcal{P}$ be any LCL problem on trees with $\Delta = O(1)$. If there exists a* RandLOCAL *algorithm $\mathcal{A}$ that solves $\mathcal{P}$ in $n^{o(1)}$ rounds, then there exists a* DetLOCAL *algorithm $\mathcal{A}'$ that solves $\mathcal{P}$ in $O(\log n)$ rounds. Moreover, given a description of $\mathcal{P}$, it is decidable whether the* RandLOCAL *complexity of $\mathcal{P}$ is $n^{\Omega(1)}$ or the* DetLOCAL *complexity of $\mathcal{P}$ is $O(\log n)$.*

Our $O(\log n)$-time decomposition algorithm also works on graphs of girth at least $c \log n$, where c is a sufficiently large constant depending on $\mathcal{P}$. This implies that Theorem 3 also applies to the class of n-vertex graphs with girth $\omega(\log n)$.

IV. A Gap in the RandLOCAL Complexity Hierarchy

Consider a set $\mathcal{V}$ of independent random variables, and a set $\mathcal{X}$ of *bad events*, where $A \in \mathcal{X}$ depends only on some subset $\text{vbl}(A) \subset \mathcal{V}$ of variables.[6] The *dependency graph* $G_{\mathcal{X}} = (\mathcal{X}, \{(A, B) \mid \text{vbl}(A) \cap \text{vbl}(B) \neq \varnothing\})$ joins events by an edge if they depend on at least one common variable. The *Lovász local lemma* (LLL) and its variants give criteria under which $\Pr(\bigcap_{A \in \mathcal{X}} \overline{A}) > 0$, i.e., it is possible that all bad events do not occur. We will narrow our discussion to *symmetric* criteria, expressed in terms of p and d, where $p = \max_{A \in \mathcal{X}} \Pr(A)$ and d is the maximum degree in $G_{\mathcal{X}}$. A standard version of the LLL states that if $ep(d+1) < 1$, then $\Pr(\bigcap \overline{A}) > 0$. Given that all bad events *can* be avoided, it is often desirable to constructively find a point in the probability space (i.e., an assignment to variables in $\mathcal{V}$) that avoids them. This problem has been thoroughly investigated in the sequential context [18], [32], [33], [34], [35], [36], [37], but somewhat less so from the point of view of parallel and distributed computation [19], [16], [11], [5], [38], [20], [17].

The *distributed* constructive LLL problem is the following. The communications network is precisely $G_{\mathcal{X}}$. Each vertex (event) A knows the number of bad events in $G_{\mathcal{X}}$ and the distribution of those variables appearing in $\text{vbl}(A) \subset \mathcal{V}$. Vertices communicate for some number of rounds, and collectively reach a consensus on an assignment to $\mathcal{V}$ in which no bad event occurs. Moser and Tardos's [18] parallel resampling algorithm implies an $O(\log^2 n)$ time RandLOCAL algorithm under the LLL criterion $ep(d+1) < 1$. Chung, Pettie, and Su [19] gave an $O(\log_{1/epd^2} n)$ time algorithm under the LLL criterion $epd^2 < 1$ and an $O(\log n / \log\log n)$ time algorithm under criterion $p \cdot \text{poly}(d) 2^d = O(1)$. They observed that under *any* criterion of the form $p \cdot f(d) < 1$, $\Omega(\log^* n)$ time is necessary. Ghaffari's [16] weak MIS algorithm, together with [19], implies an $O(\log d \cdot \log_{1/ep(d+1)} n)$ algorithm under LLL criterion $ep(d+1) < 1$. Brandt et al. [11] proved that $\Omega(\log_d \log n)$ time in RandLOCAL is necessary, even under the permissive LLL criterion $p2^d \leqslant 1$. Chang et al. [5]'s results imply that $\Omega(\log_d n)$ time is necessary in DetLOCAL, again, under the LLL criterion $p2^d \leqslant 1$. We *define* $T_{LLL}(n, d)$ to be the RandLOCAL time to compute a point in the probability space avoiding all bad events (w.h.p.), under any "polynomial" LLL criterion of the form

$$pd^c = O(1), \tag{1}$$

where c can be an arbitrarily large constant. Earlier prior results [19], [11] imply that $T_{LLL}(n, d)$ is $\Omega(\log_{\log d} \log n)$, $\Omega(\log^* n)$, and $O(\log_{1/epd^2} n)$. Very recently, it has been shown that LLL can be solved in sublogarithmic time. In particular, $T_{LLL}(n, d) = 2^{O(\sqrt{\log\log n})}$ for $d < (\log\log n)^{1/5}$ on general graphs [20], and $T_{LLL}(n, d) = O(\log\log n)$ for tree-structured instances [17].

In this section we prove an automatic speedup theorem for sublogarithmic RandLOCAL algorithms. We do *not* assume

[6]Each variable $V \in \mathcal{V}$ may have a different distribution and range, so long as the range is some finite set.

that $\Delta = O(1)$ in this section.

Theorem 4. *Suppose that $\mathcal{A}$ is a* RandLOCAL *algorithm that solves some LCL problem $\mathcal{P}$ (w.h.p.), in $T_\Delta(n)$ time. For a sufficiently small constant $\epsilon > 0$, suppose $T_\Delta(n)$ is upper bounded by $C(\Delta)+\epsilon \log_\Delta n$, for some function C. It is possible to transform $\mathcal{A}$ into a new* RandLOCAL *algorithm $\mathcal{A}'$ that solves $\mathcal{P}$ (w.h.p.) in $O(C(\Delta) \cdot T_{LLL}(n, \Delta^{O(C(\Delta))}))$ time.*

Proof: Suppose that $\mathcal{A}$ has a local probability of failure $1/n$, that is, for any $v \in V(G)$, the probability that $N^r(v)$ is inconsistent with $\mathcal{P}$ is $1/n$, where r is the radius of $\mathcal{P}$. Once we settle on the LLL criterion exponent c in (1), we fix $\epsilon = O((2c)^{-1})$. Define $n^\star$ as the minimum value for which

$$t^\star = T_\Delta(n^\star) < (1/2c) \cdot \log_\Delta n^\star - r.$$

It follows that $t^\star = O(C(\Delta))$ and $n^\star = \Delta^{O(C(\Delta))}$.

The algorithm $\mathcal{A}'$ applied to an n-vertex graph G works as follows. Imagine an experiment where we run $\mathcal{A}$, but lie to the vertices, telling them that "n" $= n^\star$. Any $v \in V(G)$ will see a $t^\star$-neighborhood $N^{t^\star}(v)$ that is consistent with some $n^\star$-vertex graph. However, the *bad event* that $N^r(v)$ is incorrectly labeled is $1/n^\star$, not $1/\text{poly}(n)$, as desired. We now show that this system of bad events satisfies the LLL criterion (1). Define the following events, graph, and quantities:

$$\begin{aligned}
&\mathcal{E}_v \ : \text{ the event that } N^r(v) \text{ is incorrectly labeled according to } \mathcal{P}\\
&\mathcal{X} = \{\mathcal{E}_v \mid v \in V(G)\} \quad \text{(the set of bad events)}\\
&G_\mathcal{X} = (\mathcal{X}, \{(\mathcal{E}_u, \mathcal{E}_v) \mid N^{r+t^\star}(u) \cap N^{r+t^\star}(v) \neq \varnothing\})\\
&\quad \text{(the dependency graph)}\\
&d \leqslant \Delta^{2(r+t^\star)}\\
&p = 1/n^\star
\end{aligned}$$

The event $\mathcal{E}_v$ is determined by the labeling of $N^r(v)$ and the label of each $v' \in N^r(v)$ is determined by $N^{t^\star}(v')$, hence $\mathcal{E}_v$ is determined by (the data stored in, and random bits generated by) vertices in $N^{r+t^\star}(v)$. Clearly $\mathcal{E}_v$ is independent of any $\mathcal{E}_u$ for which $N^{r+t^\star}(u) \cap N^{r+t^\star}(v) = \varnothing$, which justifies the definition of the edge set of $G_\mathcal{X}$. Since the maximum degree in G is Δ, the maximum degree d in $G_\mathcal{X}$ is less than $\Delta^{2(r+t^\star)}$. By definition of $\mathcal{A}$, $\Pr(\mathcal{E}_v) \leqslant 1/n^\star = p$. This system satisfies LLL criterion (1) since, by definition of $t^\star$,

$$pd^c = p\Delta^{2c(r+t^\star)} < (1/n^\star) \cdot n^\star = 1.$$

The algorithm $\mathcal{A}'$ now simulates a constructive LLL algorithm on $G_\mathcal{X}$ in order to find a labeling such that no bad event occurs. Since a virtual edge $(\mathcal{E}_u, \mathcal{E}_v)$ exists iff u and v are at distance at most $2(r + t^\star) = O(C(\Delta))$, any RandLOCAL algorithm in $G_\mathcal{X}$ can be simulated in G with $O(C(\Delta))$ slowdown. Thus, $\mathcal{A}'$ runs in $O(C(\Delta) \cdot T_{LLL}(n, \Delta^{O(C(\Delta))}))$ time. ∎

Theorem 4 shows that when $\Delta = O(1)$, $o(\log n)$-time RandLOCAL algorithms can be *sped up* to run in $O(T_{LLL}(n, O(1)))$ time. Another consequence of this same technique is that sublogarithmic RandLOCAL algorithms with *large messages* can be converted to (possibly slightly slower) algorithms with small messages. The statement of Theorem 5 reflects the use of a particular distributed LLL algorithm, namely [19, Corollary 1 and Algorithm 2]. It may be improvable using future distributed LLL technology.

The LLL algorithm of [19] works under the assumption that $epd^2 < 1$, and that each bad event $A \in \mathcal{X}$ is associated with a unique ID. The algorithm starts with a random assignment to the variables $\mathcal{V}$. In each iteration, let $\mathcal{F}$ be the set of bad events that occur under the current variable assignment; let $\mathcal{I}$ be the subset of $\mathcal{F}$ such that $A \in \mathcal{I}$ if and only if $\text{ID}(A) < \text{ID}(B)$ for each $B \in \mathcal{F}$ such that $\text{vbl}(A) \cap \text{vbl}(B) \neq \varnothing$. The next variable assignment is obtained by *resampling* all variables in $\bigcup_{A \in \mathcal{I}} \text{vbl}(A)$. After $O(\log_{1/epd^2} n)$ iterations, no bad event occurs with probability $1 - 1/\text{poly}(n)$.

Theorem 5. *Let $\mathcal{A}$ be a $(C(\Delta) + \epsilon \log_\Delta n)$-time* RandLOCAL *algorithm that solves some LCL problem $\mathcal{P}$ with high probability, where $\epsilon > 0$ is a sufficiently small constant. Each vertex locally generates $r_\Delta(n)$ random bits and sends $m_\Delta(n)$-bit messages. It is possible to transform $\mathcal{A}$ into a new* RandLOCAL *algorithm $\mathcal{A}'$ that solves $\mathcal{P}$ (w.h.p.) in $O(\log_\Delta n)$ time, where each vertex generates $O(\log n + r_\Delta(\zeta) \cdot \log_\zeta n)$ random bits, and sends $O(\min\{\log(|\Sigma_{\text{out}}|) \cdot \Delta^{O(1)} + m_\Delta(\zeta) + \zeta,\ r_\Delta(\zeta) \cdot \zeta\})$-bit messages, where $\zeta = \Delta^{O(C(\Delta))}$ depends on Δ.*

Proof: We continue to use the notation and definitions from Theorem 4, and fix $c = 3$ in the LLL criterion (1). Since $d = \Omega(\Delta^{O(C(\Delta))}) = \Omega(\zeta)$ and we selected $t^\star$ w.r.t. $c = 3$ (i.e., LLL criterion $pd^3 < 1$), we have $1/epd^2 = \Omega(\zeta)$. If $\mathcal{A}'$ uses the LLL algorithm of [19], each vertex $v \in V(G)$ will first generate an $O(\log n)$-bit unique identifier $\text{ID}(\mathcal{E}_v)$ (which costs $O(\log n)$ random bits) and generate $r_\Delta(n^\star) \cdot O(\log_{1/epd^2} n) = O(r_\Delta(\zeta) \cdot \log_\zeta n)$ random bits throughout the computation. Thus, the total number of random bits per vertex is $O(\log n + r_\Delta(\zeta) \cdot \log_\zeta n)$.

In each resampling step of $\mathcal{A}'$, in order for v to tell whether $\mathcal{E}_v \in \mathcal{I}$, it needs the following information: (i) $\text{ID}(\mathcal{E}_u)$ for all $u \in N^{2(r+t^\star)}(v)$, and (ii) whether $\mathcal{E}_u$ occurs under the current variable assignment, for all $u \in N^{2(r+t^\star)}(v)$. We now present two methods to execute one resampling step of $\mathcal{A}'$; they both take $O(C(\Delta))$ time using a message size that depends on Δ but is independent of n. There are $O(\log_{1/epd^2} n) = O(\log_\zeta n) = O(\frac{\log_\Delta n}{C(\Delta)})$ resampling steps, so the total time is $O(\log_\Delta n)$, independent of the function C.

Method 1. Before the LLL algorithm proper begins, we do the following preprocessing step. Each vertex v gathers up all IDs and random bits in its $3(t^\star + r)$-neighborhood. This takes $O((\log n + r_\Delta(\zeta) \cdot \log_\zeta n) \cdot \zeta/b)$ time with b-bit messages (recall that $\Delta^{O(t^\star+r)} = \Delta^{O(C(\Delta))} = \zeta$). In particular, the runtime can be made $O(\log_\Delta n)$ if we set $b = O(r_\Delta(\zeta) \cdot \zeta)$.

During the LLL algorithm, each vertex u owns one random variable: an $r_\Delta(n^\star)$-bit string V_u. In order for v to tell whether $\mathcal{E}_u$ occurs for each $u \in N^{2(r+t^\star)}(v)$ under the current variable assignment, it only needs to know how many times each V_u, $u \in N^{3(r+t^\star)}(v)$, has been resampled. Whether the output labeling of $u \in N^{2(r+t^\star)}(v)$ is locally consistent depends on the output labeling of vertices in $N^r(u)$, which depends on the random bits and the graph topology within $N^{r+t^\star}(u) \subseteq N^{3(r+t^\star)}(v)$. Given the graph topology, IDs, and the random bits within $N^{3(r+t^\star)}(v)$, the vertex v can locally simulate $\mathcal{A}$ and decides whether $\mathcal{E}_v \in \mathcal{I}$.

Thus, in each iteration of the LLL algorithm, each vertex v simply needs to alert its $3(r+t^\star)$-neighborhood whether V_v is resampled or not. This can be accomplished in $O(r+t^\star) = O(C(\Delta))$ time with ζ-bit messages.

Method 2. In the second method, vertices keep their random bits private. Similar to the first method, we do a preprocessing step to let each vertex gathers up all IDs in its $2(t^\star+r)$-neighborhood. This can be done in $O(\log_\Delta n)$ time using ζ-bit messages.

During the LLL algorithm, in order to tell which subset of bad events $\{\mathcal{E}_v\}_{v \in V(G)}$ occur under the current variable assignment, all vertices simulate $\mathcal{A}$ for $t^\star$ rounds, sending $m_\Delta(n^\star)$-bit messages. After the simulation, for a vertex v to tell whether $\mathcal{E}_v$ occurs, it needs to gather the output labeling of the vertices in $N^r(v)$. This can be done in $r = O(1)$ rounds, sending $\log(|\Sigma_{\text{out}}|) \cdot \Delta^{O(1)}$-bit messages.[7] Next, for a vertex v to tell whether $\mathcal{E}_v \in \mathcal{I}$, it needs to know whether $\mathcal{E}_u$ occurs for all $u \in N^{2(r+t^\star)}(v)$. This information can be gathered in $O(C(\Delta))$ time using messages of size $O(\zeta)$. To summarize, the required message size is $O(\log(|\Sigma_{\text{out}}|) \cdot \Delta^{O(1)} + m_\Delta(\zeta) + \zeta)$. ∎

An interesting corollary of Theorem 5 is that when $\Delta = O(1)$, randomized algorithms with unbounded length messages can be simulated with 1-bit messages.

Corollary 1. *Let $\mathcal{P}$ be any LCL problem. When $\Delta = O(1)$, any $o(\log n)$ algorithm solving $\mathcal{P}$ w.h.p. using* unbounded length messages *can be made to run in $O(\log n)$ time with 1-bit messages.*

V. Conclusion

We now have a very good understanding of the LOCAL complexity landscape for cycles, tori, bounded degree trees, and to a lesser extent, general bounded degree graphs. See Figure 1. However, there are some very critical gaps in our understanding.

Our randomized speedup theorem of Section IV depends on the complexity of a relatively weak version of the Lovász local lemma. Since the LLL is essentially a "complete" problem for sublogarithmic RandLOCAL algorithms, understanding the distributed complexity of the LLL is a significant open problem.

Conjecture 1. *There exists a sufficiently large constant c such that the distributed LLL problem can be solved in $O(\log\log n)$ time on bounded degree graphs, under the symmetric LLL criterion $pd^c < 1$.*

After the initial publication of this work [30], Fischer and Ghaffari [20] gave an LLL algorithm for bounded degree graphs running in $2^{O(\sqrt{\log\log n})}$ time.[8] Building on [20], Chang, He, Li, Pettie, and Uitto [17] proved the $O(\log\log n)$ bound of Conjecture 1 for the special case of *tree-structured* dependency graphs. The results of [20], [17] make us more optimistic that Conjecture 1 is true.

The new polynomial complexities introduced in Section II are of the form $\Theta(n^{1/k})$, $k \in \mathbb{Z}^+$. Is this set of polynomial complexities exhaustive? Is it possible to engineer problems with complexity $\Theta(n^q)$ for any given rational q? We think the answer is no, and resolving Conjecture 2 would be the first step.

Conjecture 2. *Any $o(n)$-time* DetLOCAL *algorithm for an LCL problem can be automatically sped up to run in $O(\sqrt{n})$ time. In general, there is an $\omega(n^{1/(k+1)})$—$o(n^{1/k})$ gap in the* DetLOCAL *complexity hierarchy.*

References

[1] N. Linial, "Locality in distributed graph algorithms," *SIAM J. Comput.*, vol. 21, no. 1, pp. 193–201, 1992.

[2] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.

[3] J. Hartmanis and R. E. Stearns, "On the computational complexity of algorithms," *Trans. Amer. Math. Soc.*, vol. 117, pp. 285–306, 1965.

[4] M. Fürer, "Data structures for distributed counting," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 231–243, 1984.

[5] Y.-J. Chang, T. Kopelowitz, and S. Pettie, "An exponential separation between randomized and deterministic complexity in the LOCAL model," in *Proceedings 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2016, pp. 615–624.

[6] S. Brandt, J. Hirvonen, J. H. Korhonen, T. Lempiäinen, P. R. J. Östergård, C. Purcell, J. Rybicki, J. Suomela, and P. Uznanski, "LCL problems on grids," in *Proceedings 36th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2017.

[7] An output label can be encoded as a $\log(|\Sigma_{\text{out}}|)$-bit string. We do not assume that Δ is constant so $|\Sigma_{\text{out}}|$, which may depend on Δ but not directly on n, is also not constant. E.g., consider the $O(\Delta)$ vertex coloring problem.

[8] as well as somewhat slower algorithms for arbitrary graphs.

[7] A. Korman, J.-S. Sereni, and L. Viennot, "Toward more localized local algorithms: removing assumptions concerning global knowledge." *Distributed Computing*, vol. 26, no. 5–6, pp. 289–308, 2013.

[8] M. Naor and L. J. Stockmeyer, "What can be computed locally?" *SIAM J. Comput.*, vol. 24, no. 6, pp. 1259–1277, 1995.

[9] R. Cole and U. Vishkin, "Deterministic coin tossing with applications to optimal parallel list ranking," *Information and Control*, vol. 70, no. 1, pp. 32–53, 1986.

[10] M. Naor, "A lower bound on probabilistic algorithms for distributive ring coloring," *SIAM J. Discrete Mathematics*, vol. 4, no. 3, pp. 409–412, 1991.

[11] S. Brandt, O. Fischer, J. Hirvonen, B. Keller, T. Lempiäinen, J. Rybicki, J. Suomela, and J. Uitto, "A lower bound for the distributed Lovász local lemma," in *Proceedings 48th ACM Symposium on the Theory of Computing (STOC)*, 2016, pp. 479–488.

[12] M. Ghaffari and H.-H. Su, "Distributed degree splitting, edge coloring, and orientations," in *Proceedings 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017, pp. 2505–2523.

[13] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Local computation: Lower and upper bounds," *J. ACM*, vol. 63, no. 2, pp. 17:1–17:44, 2016.

[14] R. Bar-Yehuda, K. Censor-Hillel, and G. Schwartzman, "A distributed $(2+\epsilon)$-approximation for vertex cover in $O(\log\Delta/\epsilon\log\log\Delta)$ rounds," *to appear in J. ACM*, 2017.

[15] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider, "The locality of distributed symmetry breaking," *J. ACM*, vol. 63, 2016, article 20.

[16] M. Ghaffari, "An improved distributed algorithm for maximal independent set," in *Proceedings 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2016, pp. 270–277.

[17] Y.-J. Chang, Q. He, W. Li, S. Pettie, and J. Uitto, "The complexity of distributed edge coloring with small palettes," *CoRR*, vol. abs/1708.04290, 2017. [Online]. Available: http://arxiv.org/abs/1708.04290

[18] R. A. Moser and G. Tardos, "A constructive proof of the general Lovász local lemma," *J. ACM*, vol. 57, no. 2, 2010.

[19] K.-M. Chung, S. Pettie, and H.-H. Su, "Distributed algorithms for the Lovász local lemma and graph coloring," *Distributed Computing*, vol. 30, pp. 261–280, 2017.

[20] M. Fischer and M. Ghaffari, "Sublogarithmic distributed algorithms for Lovász local lemma with implications on complexity hierarchies," in *Proceedings 31st International Symposium on Distributed Computing (DISC)*, 2017.

[21] A. Panconesi and A. Srinivasan, "On the complexity of distributed network decomposition," *J. Algor.*, vol. 20, no. 2, pp. 356–374, 1996.

[22] M. Göös, J. Hirvonen, and J. Suomela, "Linear-in-Δ lower bounds in the LOCAL model," *Distributed Computing*, pp. 1–14, 2015.

[23] F. Kuhn and R. Wattenhofer, "On the complexity of distributed graph coloring," in *Proceedings 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2006, pp. 7–15.

[24] D. Hefetz, F. Kuhn, Y. Maus, and A. Steger, "Polynomial lower bound for distributed graph coloring in a weak LOCAL model," in *Proceedings 30th International Symposium on Distributed Computing (DISC)*, 2016, pp. 99–113.

[25] J. Suomela, "Survey of local algorithms," *ACM Comput. Surv.*, vol. 45, no. 2, pp. 24:1–24:40, 2013.

[26] P. Fraigniaud, A. Korman, and D. Peleg, "Towards a complexity theory for local distributed computing," *J. ACM*, vol. 60, no. 5, pp. 35:1–35:26, 2013.

[27] L. Feuilloley and P. Fraigniaud, "Survey of distributed decision," *CoRR*, vol. abs/1606.04434, 2016.

[28] M. Göös and J. Suomela, "Locally checkable proofs in distributed computing," *Theory of Computing*, vol. 12, no. 1, pp. 1–33, 2016.

[29] M. Ghaffari, F. Kuhn, and Y. Maus, "On the complexity of local distributed graph problems," in *Proceedings 49th ACM Symposium on Theory of Computing (STOC)*, 2017.

[30] Y.-J. Chang and S. Pettie, "A time hierarchy theorem for the LOCAL model," *CoRR*, vol. abs/1704.06297, 2017. [Online]. Available: http://arxiv.org/abs/1704.06297

[31] G. L. Miller and J. H. Reif, "Parallel tree contraction–Part I: Fundamentals," *Advances in Computing Research*, vol. 5, pp. 47–72, 1989.

[32] D. G. Harris and A. Srinivasan, "A constructive algorithm for the Lovász local lemma on permutations," in *Proceedings 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014, pp. 907–925.

[33] D. G. Harris, "Lopsidependency in the Moser-Tardos framework: Beyond the lopsided Lovász local lemma," *ACM Trans. Algorithms*, vol. 13, no. 1, pp. 17:1–17:26, 2016.

[34] K. B. R. Kolipaka and M. Szegedy, "Moser and Tardos meet Lovász," in *Proceedings 43rd ACM Symposium on Theory of Computing (STOC)*, 2011, pp. 235–244.

[35] V. Kolmogorov, "Commutativity in the algorithmic Lovász local lemma," in *Proceedings 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2016, pp. 780–787.

[36] N. J. A. Harvey and J. Vondrák, "An algorithmic proof of the Lovász local lemma via resampling oracles," in *Proceedings 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015, pp. 1327–1346.

[37] D. Achlioptas and F. Iliopoulos, "Random walks that find perfect objects and the Lovász local lemma," in *Proceedings 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2014, pp. 494–503.

[38] B. Haeupler and D. G. Harris, “Parallel algorithms and concentration bounds for the Lovász local lemma via witness-DAGs,” in *Proceedings 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017, pp. 1170–1187.

[39] R. L. Graham, B. L. Rothschild, and J. H. Spencer, *Ramsey Theory*, 2nd ed. New York: John Wiley and Sons, 1990.

Appendix

Let $\mathcal{A}$ be any $T(n)$-round DetLOCAL algorithm. Let η and η' be any two *order-indistinguishable* assignments of distinct IDs to $N^{T(n)}(v)$, i.e., for $u, w \in N^{T(n)}(v)$, $\eta(u) > \eta(w)$ if and only if $\eta'(u) > \eta'(w)$. If, for every possible input graph fragment induced by $N^{T(n)}(v)$, the output label of v is identical under every pair of order-indistinguishable η, η', then $\mathcal{A}$ is *order-invariant*.

Suppose that there exists a number $n' = O(1)$ such that $\Delta^{T(n')+r} \leqslant n'$. If $\mathcal{A}$ is order-invariant then it can be turned into an $O(1)$-round DetLOCAL algorithm $\mathcal{A}'$, since we can pretend that the total number of vertices is n' instead of n.

Naor and Stockmeyer [8] proved that any DetLOCAL algorithm that takes $\tau = O(1)$ rounds on a bounded degree graph can be turned into an order-invariant τ-round DetLOCAL algorithm. A more careful analysis shows that the proof still works when τ is a slowly growing function of n.

Requirement for Automatic Speedup: The muticolor hypergraph Ramsey number $R(p, m, c)$ is the minimum number such that the following holds. Let H be a complete p-uniform hypergraph of at least $R(p, m, c)$ vertices. Then any c-edge-coloring of H contains a monochromatic clique of size m.

Given the number $\tau \geqslant 2$, the three parameters p, m, and c are selected as follows. (See the proof of [8, Lemma 3.2] for more details.)

- The number p is the maximum number of vertices in $N^{\tau}(v)$, over all vertices $v \in V(G)$ and all graphs G under consideration. For rings, $p = 2\tau + 1$. For tori, $p \leqslant 2(\tau + 1)^2$. For trees or general graphs, $p \leqslant \Delta^{\tau}$.
- The number m is the maximum number of vertices in $N^{\tau+r}(v)$, over all vertices $v \in V(G)$ and all graphs G under consideration. E.g., for rings, $p = 2\tau + 2r + 1$ and for general graphs, $p \leqslant \Delta^{\tau+r}$.
- The number z counts the distinguishable radius-τ centered subgraphs, disregarding IDs. For example, for LCLs on the ring without input labels or port numbering, $z = 1$, whereas with input labels and port numbering it is $(2|\Sigma_{\text{in}}|)^{2\tau+1}$ since each vertex has one of $|\Sigma_{\text{in}}|$ input labels and 2 port numberings. In general z is less than $2^{\binom{\Delta^{\tau}}{2}} \cdot (\Delta! |\Sigma_{\text{in}}|)^p$.
- The number c is defined as $|\Sigma_{\text{out}}|^{p! z}$. Intuitively, we can use a number in $[c]$ to encode a function that maps a radius-τ centered subgraph (that is equipped with unique vertex IDs from a set S with cardinality p) to an output label in Σ_{out}.

Recall that vertices in DetLOCAL have $O(\log n)$-bit IDs, i.e., they can be viewed as elements of $[n^k]$ for some $k = O(1)$. Naor and Stockmeyer's proof implies that, as long as $n^k \geqslant R(p, m, c)$, any DetLOCAL τ-round algorithm on a bounded degree graph can be turned into an order-invariant τ-round DetLOCAL algorithm, which then implies an $O(1)$-round DetLOCAL algorithm.

The Ramsey number $R(p, m, c)$: According to the proof of [39, §1, Theorem 2], we have:

$$\text{For } p = 1, R(p, m, c) = c(m - 1) + 1$$

$$\text{For } p > 1, R(p, m, c) \leqslant 2c^x$$

$$\text{where } x = \sum_{i=p-1}^{R(p-1,m,c)-1} \binom{i+1}{p-1} < R(p-1, m, c)^p$$

Therefore, $\log^*(R(p, m, c)) \leqslant p + \log^* m + \log^* c + O(1)$.

Automatic Speedup Theorems: Observe that in all scenarios, if the running time $\tau = \tau(n) = \omega(1)$, we have $\log^* m + \log^* c = o(p)$. Therefore, having $p \leqslant \epsilon \log^* n$ for some small enough constant ϵ suffices to meet the condition $n^k \geqslant R(p, m, c)$. We conclude that the complexity of any LCL problem (with or without input labels and port numbering) in the LOCAL model never falls in the following gaps:

$\omega(1)$—$o(\log^* n)$	for n-rings.
$\omega(1)$—$o(\sqrt{\log^* n})$	for $(\sqrt{n} \times \sqrt{n})$-tori.
$\omega(1)$—$o(\log(\log^* n))$	for bounded degree trees or general graphs.

Due to the “Stepping-Up Lemma” (see [39, §4, Lemma 17]), we have $\log^*(R(p, m, 2)) = \Omega(p)$. Thus, Naor and Stockmeyer's approach *alone* cannot give an $\omega(1)$—$o(\log^* n)$ gap for general graphs.

However, for a certain class of LCL problems on $(\sqrt{n} \times \sqrt{n})$-tori, the gap can be widened to $\omega(1)$—$o(\log^* n)$ [6, p. 2]. The following proof is due to Jukka Suomela (personal communication).

Theorem 6 (J. Suomela)**.** *Let $\mathcal{P}$ be any LCL problem on $(\sqrt{n} \times \sqrt{n})$-tori that does not refer to input labels or port-numbering. The* DetLOCAL *and* RandLOCAL *complexity of $\mathcal{P}$ is either $O(1)$ or $\Omega(\log^* n)$.*

Proof: Given a $(\sqrt{n} \times \sqrt{n})$-torus G, we associate each vertex $v \in V(G)$ with a coordinate (α, β), where $\alpha, \beta \in \{0, \ldots, \sqrt{n} - 1\}$. We consider the following special way to generate unique $k \log n$-bit IDs. Let ϕ_x and ϕ_y be two functions mapping integers in $\{0, \ldots, \sqrt{n} - 1\}$ to integers in $\{0, \ldots, n^{k/2} - 1\}$. We additionally require that $\phi_x(0) < \ldots < \phi_x(\sqrt{n} - 1) < \phi_y(0) < \ldots < \phi_y(\sqrt{n} - 1)$. If v is at position (α, β), it has ID $\phi_x(\alpha) \cdot n^{k/2} + \phi_y(\beta)$. Notice that the IDs of all vertices in $N^{\tau}(v)$ can be deduced from just $4\tau + 2$ numbers: $\phi_x(i)$, $i \in [\alpha - \tau, \alpha + \tau]$ and $\phi_y(j)$, $j \in [\beta - \tau, \beta + \tau]$.

Suppose that the complexity of $\mathcal{P}$ is $o(\log^* n)$. Let $\mathcal{A}$ be any τ-round DetLOCAL algorithm for solving $\mathcal{P}$, where $\tau = o(\log^* n)$. Notice that the algorithm $\mathcal{A}$ works correctly even when we restrict ourselves to the above special ID assignment. Our goal is to show that $\mathcal{P}$ is actually *trivial* in the sense that there exists an element $\sigma \in \Sigma_{\text{out}}$ such that labeling all vertices by σ gives a legal labeling, assuming w.l.o.g. that $\sqrt{n} > 2r + 1$. Thus, $\mathcal{P}$ can be solved in $O(1)$ rounds.

In subsequent discussion, we let v be any vertex whose position is (α, β), where $\tau + r \leqslant \alpha \leqslant (\sqrt{n} - 1) - (\tau + r)$ and $\tau + r \leqslant \beta \leqslant (\sqrt{n} - 1) - (\tau + r)$. That is, v is sufficiently far way from the places where the coordinates wrap around.

Given $\mathcal{A}$, we construct a function f as follows. Let $S = (s_1, \ldots, s_{4\tau+2})$ be a vector of $4\tau + 2$ numbers in $\{0, \ldots, n^{k/2} - 1\}$ such that $s_k < s_{k+1}$ for each $k \in [4\tau + 2]$. Then $f(S) \in \Sigma_{\text{out}}$ is defined as the output labeling of v resulting from executing $\mathcal{A}$ with the following ID assignment of vertices in $N^\tau(v)$. We set $\phi_x(\alpha - \tau - 1 + i) = s_i$ for each $i \in [2\tau + 1]$ and set $\phi_y(\beta - \tau - 1 + j) = s_{j+2\tau+1}$ for each $j \in [2\tau + 1]$ Recall that $\mathcal{P}$ does not use port-numbering and input labeling, so the output labeling of v depends only on IDs of vertices in $N^\tau(v)$.

We set $p = 4\tau + 2$, $m = 4\tau + 4r + 2$, and $c = |\Sigma_{\text{out}}|$. Notice that the calculation of the parameter c here is different from the original proof of Naor and Stockmeyer. Since we already force that $\phi_x(0) < \ldots < \phi_x(\sqrt{n} - 1) < \phi_y(0) < \ldots < \phi_y(\sqrt{n} - 1)$, we do not need to consider all $p!$ permutations of the set S.

We have $R(p, m, c) \ll n^{k/2}$ (since $p = o(\log^* n)$). Thus, there exists a set S' of m distinct numbers in $\{0, \ldots, n^{k/2}\}$ such that the following is true. We label these m numbers $\phi_x(i)$, $i \in [\alpha - \tau - r, \alpha + \tau + r]$, and $\phi_y(j)$, $j \in [\beta - \tau - r, \beta + \tau + r]$ by the set S' such that $\phi_x(\alpha - \tau - r) < \ldots < \phi_x(\alpha + \tau + r) < \phi_y(\beta - \tau - r) < \ldots < \phi_y(\beta + \tau + r)$. Then the output labels of all vertices in $N^r(v)$ assigned by $\mathcal{A}$ are identical.

Therefore, there exists an element $\sigma \in \Sigma_{\text{out}}$ such that labeling all vertices by σ yields a legal labeling of G. Thus, $\mathcal{P}$ can be solved in $O(1)$ rounds. ■

Discussion: It still remains an outstanding open problem whether the gap for other cases can also be widened to $\omega(1)$—$o(\log^* n)$.

The proof of Theorem 6 extends easily to d-dimensional tori, but does not extend to bounded degree trees, since there is a non-trivial problem that can be solved in $O(1)$ rounds on a subset of bounded degree trees. Naor and Stockmeyer [8] showed that on any graph class in which all vertex degrees are odd, *weak* $2^{O(\Delta \log \Delta)}$*-coloring* can be solved in 2 rounds and *weak* 2*-coloring* can be solved in $O(\log^* \Delta)$ rounds in DetLOCAL.[9] This problem is *non-trivial* in the sense that coloring all vertices by the same color is not a legal solution. Since the d-dimensional torus is Δ-regular, $\Delta = 2d$, we conclude that the complexity of weak $O(1)$-coloring on Δ-regular graphs is $\Theta(\log^* n)$ for every fixed even number $\Delta \geqslant 2$.

[9]A weak coloring is one in which every vertex is colored differently than at least one neighbor.