A Breezing Proof of the KMW Bound

Corinna Coupette* Christoph Lenzen†

Abstract

In their seminal paper from 2004, Kuhn, Moscibroda, and Wattenhofer (KMW) proved a hardness result for several fundamental graph problems in the LOCAL model: For any (randomized) algorithm, there are graphs with n nodes and maximum degree Δ on which $\Omega(\min\{\sqrt{\log n/\log\log n}, \log\Delta/\log\log\Delta\})$ (expected) communication rounds are required to obtain polylogarithmic approximations to a minimum vertex cover, minimum dominating set, or maximum matching. Via reduction, this hardness extends to symmetry breaking tasks like finding maximal independent sets or maximal matchings.

Today, more than 15 years later, there is still no proof of this result that is easy on the reader. Setting out to change this, in this work, we provide a simplified proof of the main step in showing the KMW lower bound. Our key argument is algorithmic, and it relies on an invariant that can be readily verified from the generation rules of the lower bound graphs.

1 Introduction and Related Work

A key property governing the complexity of distributed graph problems is their *locality*: the distance up to which the nodes running a distributed algorithm need to explore the graph to determine their local output. Under the assumption that nodes have unique identifiers, the locality of any task is at most D, the diameter of the graph. However, many problems of interest have locality $o(D)$, and understanding the locality of such problems in the LOCAL model of computation has been a main objective of the distributed computing community since the inception of the field.

A milestone in these efforts is the 2004 article by Kuhn, Moscibroda, and Wattenhofer, proving a lower bound of $\Omega(\min\{\sqrt{\log n/\log\log n}, \log\Delta/\log\log\Delta\})$ on the locality of several fundamental graph problems [23], where n is the number of nodes and Δ is the maximum degree of the input graph. The bound holds under both randomization and approximation, and it is the first result of this generality beyond the classic $\Omega(\log^* n)$ bound on 3-coloring cycles [30].

1.1 A Brief Recap of the KMW Lower Bound

In a nutshell, in [26], the authors reason as follows.

1. **Define Cluster Tree (CT) graph family.** This graph family is designed such that in high-girth CT graphs, the k-hop neighborhoods of many nodes that are *not* part of a solution to, e.g., the minimum vertex cover problem, are isomorphic to the k-hop neighborhoods of nodes that *are* part of a solution.

2. **Prove that high-girth CT graphs have isomorphic node views.** If a CT graph G_k has girth at least $2k+1$, the isomorphisms mentioned in Step 1 exist. This implies that a distributed algorithm running for k rounds, which needs to determine the output at nodes based on their k-hop neighborhood, cannot distinguish between such nodes based on the graph topology.

3. **Show existence of high-girth CT graphs.** For each $k \in \mathbb{N}$, there exists a CT graph G_k with girth at least $2k+1$ that has sufficiently few nodes and low maximum degree.

4. **Infer lower bounds.** Under uniformly random node identifiers,[1] on a CT graph with girth at least $2k+1$, a k-round distributed algorithm cannot achieve a small expected approximation ratio for minimum vertex cover, maximum matching, or minimum dominating set, and it cannot find a maximal independent set or maximal matching with a small probability of failure.

The core of the technical argument lies in Step 2. A bird's-eye view of the reasoning for each of the steps is as follows.

1. **Define Cluster Tree (CT) graph family.** We want to have a large independent set of nodes—referred to as *cluster* C_0—which contains most of the nodes in the graph. The k-hop neighborhoods of these nodes should be isomorphic not only to each other but also to

*MPI for Informatics; Saarbrücken Graduate School of Computer Science; coupette@mpi-inf.mpg.de.

†MPI for Informatics; clenzen@mpi-inf.mpg.de.

[1] In the LOCAL model, nodes have unique identifiers. Without these, even basic tasks like computing the size of the graph are impossible.

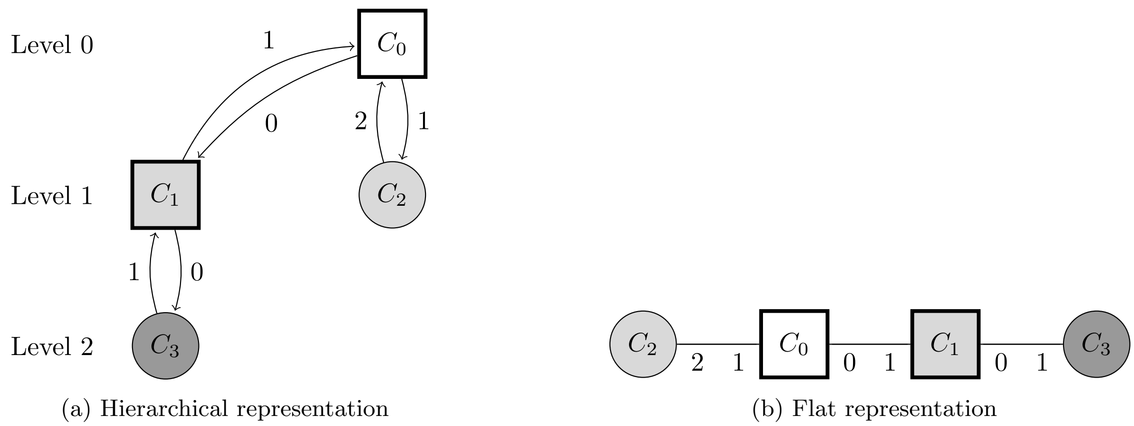


Figure 1: Representations of CT_1, which is parametrized by β, shaded by cluster size (darker means smaller). Cluster shapes indicate cluster position (*internal* or *leaf*). Edge label i is short for β^i, the number of neighbors that nodes in one cluster have in another. For example, nodes in cluster C_0 have β^0 neighbors in cluster C_1, and nodes in cluster C_1 have β^1 neighbors in cluster C_0.

the k-hop neighborhoods of nodes in a smaller cluster C_1. Each node in C_0 should have one neighbor in C_1, and the edges between the nodes from both clusters should form a biregular graph. In this situation, a k-round distributed algorithm computing, e.g., a vertex cover, cannot distinguish between the endpoints of edges connecting C_0 and C_1 based on the graph topology. This is all we need for Step 4 to succeed.

However, choosing the ratio $\beta := |C_0|/|C_1|$ larger than 1 entails that nodes in C_1 have more neighbors in C_0 than vice versa. To maintain the indistinguishability of nodes in C_0 and C_1 for a k-round distributed algorithm, we add clusters C_2 and C_3 providing the "right" number of additional neighbors to C_0 and C_1, respectively, which are by a factor of β smaller than their neighboring cluster to keep the overall number of non-C_0 nodes small. Now the nodes in C_0 and C_1 have the same number of neighbors, which implies that one round of communication is insufficient to distinguish between them.[2] See Figure 1 (p. 2) for an illustration of the resulting structure, CT_1.

Unfortunately, looking up to distance two will now reveal the difference in degrees of neighbors: "Hiding" the asymmetry between C_0 and C_1 by adding C_2 and C_3 enforces a similar asymmetry between C_2 and C_3. This is overcome by inductively "growing" a *skeleton tree* structure on clusters, which encodes the topological requirements for moving the asymmetry in degrees further and further away from C_0 and C_1.

Because in a graph of girth at least $2k+1$, the k-hop neighborhoods of all nodes are trees, the symmetry in degrees thus established is sufficient to result in isomorphic k-hop neighborhoods between nodes in C_0 and C_1. The growth rules of the skeleton tree are chosen to meet the topological requirements, while increasing degrees and the total number of nodes as little as possible.

2. **Prove that high-girth CT graphs have isomorphic node views.** Using that k-hop neighborhoods of high-girth CT graphs are trees, the task of showing that $v \in C_0$ and $w \in C_1$ have isomorphic k-hop neighborhoods boils down to finding a *degree-preserving bijection* between these neighborhoods that maps v to w. At first glance, this seems straightforward: By construction, nodes in inner clusters of the skeleton tree have degrees of $\beta^0, \beta^1, \ldots, \beta^k$ towards their $k+1$ adjacent clusters, and for each leaf cluster that lies at distance $d \le k$ from $C(v)$ and has a degree of β^x towards its parent cluster, we can find a leaf cluster with the same degree towards its parent cluster at distance d from $C(w)$. Hence, mapping a node v'' with parent v' to a node w'' with parent w' if (1) the clusters $C(v')$ and $C(w')$ lie at the same distance $d' < k$ from $C(v)$ resp. $C(w)$ and (2) $C(v')$ and $C(w')$ have the same outdegree towards $C(v'')$ resp. $C(w'')$ seems to be a promising approach for finding the desired bijection.

However, when rooting the k-hop neighborhood of $v \in C_0$ ($w \in C_1$) at v (w) and constructing the isomorphism by recursing on subtrees, for each processed node, the image of its parent under the isomorphism has

[2] This only applies if nodes do *not* know the identities of their neighbors initially, known as KT0 (initial knowledge of topology up to distance 0). It is common to assume KT1, i.e., nodes *do* know the identifiers of their neighbors at the start of the algorithm. However, this weakens the lower bound by one round only, not affecting the asymptotics.

already been determined. The asymmetry discussed in Step 1 also shows up here: Some children of $v \in C_0$ and $w \in C_1$ that are mapped to each other will have different degrees towards their parents' clusters. This results in a mismatch for one pair of *their* neighbors when processing a node according to the proposed strategy.

Nonetheless, it turns out that mapping such "mismatched" nodes to each other results in the desired bijection. Proving this is, by a margin, the technically most challenging step in obtaining the KMW lower bound.

3. **Show existence of high-girth CT graphs.** In order to show that sufficiently small and low-degree CT graphs G_k of girth $2k+1$ exist, Kuhn et al. make use of *graph lifts.*[3] Graph H is a lift of graph G if there exists a *covering map* from H to G, i.e., a surjective graph homomorphism that is bijective when restricted to the neighborhood of each node of H. These requirements are stringent enough to ensure that a lift of a CT graph G_k (i) is again a CT graph, (ii) has at least the same girth, and (iii) has the same maximum degree. On the other hand, they are lax enough to allow for *increasing* the girth.[4] This is exploited by a combination of several known results as follows.

(a) Construct a low-girth CT graph G_k by connecting nodes in clusters that are adjacent in the skeleton tree using the edges of disjoint complete bipartite graphs whose dimensions are prescribed by the edge labels of the skeleton tree.[5] Choose the smallest such G_k.

(b) Embed G_k into a marginally larger regular graph, whose degree is the maximum degree of G_k (this is a folklore result).

(c) There exist Δ-regular graphs of girth g and fewer than Δ^g nodes [15].

(d) For any two Δ-regular graphs of n_1 and n_2 nodes, there is a common lift with $O(n_1 n_2)$ nodes [1]. Apply this to the above two graphs to obtain a high-girth lift of a supergraph of G_k.

(e) Restrict the covering map of this lift to the preimage of G_k to obtain a high-girth lift of G_k, which itself is a CT graph.

Doing the bookkeeping yields size and degree bounds for the obtained CT graph as a function of k.

4. **Infer lower bounds.** With the first three steps complete, the lower bound on the number of rounds for minimum vertex cover approximations follows by showing that the inability to distinguish nodes in C_0 and C_1 forces the algorithm to choose a large fraction of nodes from C_0, while a much smaller vertex cover exists. The former holds because under a uniformly random labeling, nodes in C_0 and C_1 are equally likely to be selected, while each edge needs to be covered with probability 1. Thus, at least $|C_0|/2$ nodes are selected in expectation. At the same time, the CT graph construction ensures that C_0 contributes the vast majority of the nodes. Hence, choosing all nodes *but* the independent set C_0 results in a vertex cover much smaller than $|C_0|/2$. The lower bounds for other tasks follow by similar arguments and reductions.[6]

1.2 Our Contribution

Despite its significance, apart from an early extension to maximum matching by the same authors [24], the KMW lower bound has not inspired follow-up results. We believe that one reason for this is that the result is not as well-understood as the construction by Linial [30], which inspired many extensions [2, 8, 9, 14, 20, 21, 29, 31] and alternative proofs [27, 36]. History itself appears to drive this point home: In a 2010 arXiv article [25], an improvement to $\Omega(\min\{\sqrt{\log n}, \log \Delta\})$ was claimed, which was refuted in 2016 by Bar-Yehuda et al. [7]. 2016 was also the year when finally a journal article covering the lower bound was published [26]—over a decade after the initial construction! In the journal article, the technical core of the proof spans six pages, involves convoluted notation, and its presentation suffers from a number of minor errors impeding the reader.[7]

[3] In the original paper [23], they instead use subgraphs of a high-girth family of graphs $D(r, q)$ given in [28]. Utilizing lifts as outlined here was proposed by Mika Göös and greatly simplifies a self-contained presentation.

[4] For instance, the cycle C_{3t} on $3t$ nodes is a lift of C_3, where the covering map sends the i^{th} node of C_{3t} to the $(i \bmod 3)^{\text{th}}$ node of C_3. Any graph G has an acyclic lift that is an infinite tree T, by adding a new "copy" of node $v \in V(G)$ to T for each walk leading to v (when starting from an arbitrary fixed node of G whose first copy is the root of T). The challenge lies in finding *small* lifts of high girth.

[5] E.g., the nodes in clusters C_0 and C_1, which themselves are connected by an edge with labels (β^0, β^1) (cf. Figure 1, p. 2), are connected using the edges of $|C_0|/\beta^1$ copies of K_{β^0,β^1}.

[6] For example, as any maximal matching yields a 2-approximation to a minimum vertex cover, the minimum vertex cover lower bound extends to maximal matching.

[7] The refutation of the improved lower bound in [7] came to the attention of the authors of [26] *after* the article had been accepted by *J. ACM* with the incorrect result; the authors were forced to revise the article on short notice before publication, leading to the corrected material receiving no review [22]. Taking into account the complexity of the proof in [26], despite minor flaws, we feel that the authors did a commendable job.

Symbol	Definition	Meaning
$[k] :=$	$\{i \in \mathbb{N} \mid i \leq k\}$	Set of positive integers not larger than k
$[k]_0 :=$	$\{i \in \mathbb{N}_0 \mid i \leq k\}$	Set of nonnegative integers not larger than k
$G :=$	$(V(G), E(G))$	Graph with node set $V(G)$ and edge set $E(G)$
$G[S] :=$	$(S, \{\{v, w\} \in E(G) \mid v, w \in S\}$	Subgraph of G induced by $S \subseteq V(G)$
$\Gamma_G(v) :=$	$\{w \in V(G) \mid \{v, w\} \in E(G)\}$	Neighborhood of v in G (non-inclusive)
$\Gamma_G^k(v) :=$	$\{w \in V(G) \mid d_G(v, w) \leq k\}$	k-hop neighborhood of v in G (inclusive)
$G^k(v) :=$	$G[\Gamma^k(v)] \setminus \{\{w, u\} \in E(G) \mid d_G(v, w) = d_G(v, u) = k\}$	k-hop subgraph of a node v in G
$\exists p_G(u, w, k) :=$	$\exists(\{u, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, w\}) \in E(G)^k$	Existence of k-hop path from u to w in G
$d_G(u, w) :=$	$\min\{k \mid \exists p_G(u, w, k)\}$	Distance between node u and node w in G
$g_G :=$	$\inf\{k > 0 \mid \exists v \in V(G), p_G(v, v, k)\}$	Girth of G (length of its shortest cycle)

Table 1: General notation used in this work (subscript or parenthesized G may be omitted when clear from context).

A constructive proof of the key graph isomorphism. In this work, we present a novel proof for Step 2 of the KMW bound. That is, we revise the heart of the argument, which shows that nodes in C_0 and C_1 have indistinguishable k-hop neighborhoods. The proof in [26] uses an inductive argument that is based on a number of notation-heavy derivation rules to describe the k-hop neighborhoods of nodes in C_0 and C_1 and map subtrees of these neighborhoods onto each other. The proofs of the derivation rules, which together enable the inductive argument, rely crucially on notation and verbal description.

In contrast, our proof is based on a simple algorithmic invariant. We give an algorithm that constructs the graph isomorphism between the nodes' neighborhoods in the natural way suggested by the CT graph construction. The key observation is that one succinct invariant is sufficient to overcome the main obstacle, namely the "mismatched" nodes that are mapped to each other by the constructed isomorphism. This not only substantially simplifies the core of the proof, it also has explanatory power: In the proof from [26], the underlying intuition is buried under heavy notation and numerous indices.

Simplified notation and improved visualization. Capitalizing on the new proof of the key graph isomorphism, as a secondary contribution, we clean up and simplify notation also outside of the indistinguishability argument. We complement this effort with improved visualizations of the utilized graph structures. Overall, we expect these modifications to make the lower bound proof much more accessible, and we hope to provide a solid foundation for work extending the KMW result.

1.3 Further Related Work

The KMW bound applies to fundamental graph problems that are locally checkable in the sense of Naor and Stockmeyer [31]. Balliu et al. give an overview of the known time complexity classes for such problems [3–5], extending a number of prior works [10–12, 16–19, 34], and Suomela surveys the state of the art attainable via constant-time algorithms [35]. Bar-Yehuda et al. provide algorithms that compute $(2+\varepsilon)$-approximations to minimum (weighted) vertex cover and maximum (weighted) matching in $\mathcal{O}(\log \Delta / \varepsilon \log\log \Delta)$ and $\mathcal{O}(\log \Delta / \log\log \Delta)$ deterministic rounds, respectively [6, 7], demonstrating that the KMW bound is tight when parametrized by Δ even for constant approximation ratios. For symmetry breaking tasks, the classic algorithm by Panconesi and Rizzi [32] to compute maximal matchings and maximal independent sets in $\mathcal{O}(\log^* n + \Delta)$ deterministic rounds has recently been shown to be optimal for a wide range of parameters [2].

1.4 Organization of this Article

Since our main contribution is a novel proof establishing the indistinguishability of k-hop neighborhoods of nodes in C_0 and C_1, we confine the remainder of the exposition to this topic; readers interested in a complete and self-contained presentation are invited to an extended version of this article on arXiv [13].

After introducing basic graph theoretical concepts and notation in Section 2, we define the lower bound graphs in Section 3.1. This sets the stage for our main contribution: In Section 3.2, we prove the indistinguishability of the k-hop neighborhoods of nodes in the clusters C_0 and C_1 under the assumption of high girth.

2 Preliminaries

The basic graph theoretic notation used in this work is summarized in Table 1 (p. 4); all our graphs are finite and simple.

We operate in the LOCAL model of computation, our presentation of which follows Peleg [33]. The LOCAL model is a stylized model of network communication designed to capture the locality of distributed computing. In this model, a communication network is abstracted as a simple graph $G = (V, E)$, with nodes representing network devices and edges representing bidirectional communication links. To eliminate all computability restrictions that are not related to locality, the model makes the following assumptions:

- Network devices have unique identifiers and unlimited computation power.
- Communication links have infinite capacity.
- Computation and communication takes place in synchronous rounds.
- All network devices start computing and communicating at the same time.
- There are no faults.

In each round, a node can

1. perform an internal computation based on its currently available information,
2. send messages to its neighbors,
3. receive all messages sent by its neighbors, and
4. potentially terminate with some local output.

A k-round distributed algorithm in the LOCAL model can be interpreted as a function from k-hop subgraphs to local outputs:

Definition 2.1. (k-round distributed algorithm) *A* k-round distributed algorithm $\mathcal{A}$ *is a function mapping k-hop subgraphs $G^k(v)$, labeled by unique node identifiers (and potentially some local input), to local outputs. For a randomized algorithm, nodes are also labeled by (sufficiently long) strings of independent, unbiased random bits.*

We assume that at the start of the algorithm, nodes do *not* know their incident edges. Assuming that nodes *do* know these edges in the beginning weakens the lower bound by one round only, not affecting the asymptotics.

The key concept used to show that a graph problem is difficult to solve (exactly or approximately) for a k-round distributed algorithm in the LOCAL model is the *k-hop indistinguishability* of nodes' neighborhoods.[8]

[8]LOCAL algorithms may also make use of nodes' local inputs and identifiers. However, so far, the KMW construction has been applied to tasks without additional inputs only. For such tasks, assigning node identifiers uniformly at random translates the stated purely topological notion of indistinguishability to identical *distributions* of k-hop subgraphs labeled by identifiers.

Definition 2.2. (k-hop indistinguishability in G) *Two nodes v and w in G are indistinguishable to a k-round distributed algorithm (*k-hop indistinguishable*) if and only if there exists an isomorphism $\phi : V(G^k(v)) \to V(G^k(w))$ with $\phi(v) = w$.*

Accordingly, our goal in Section 3.2 will be to establish that the nodes in C_0 and C_1 are k-hop indistinguishable.

3 Cluster Trees

Cluster Trees (CTs) are the main concept in the derivation of the KMW bound. For $k \in \mathbb{N}$, the *Cluster Tree skeleton* CT_k describes sufficient constraints on the topology of graphs G_k (beyond high girth, which ensures that the k-hop neighborhoods of all nodes are trees) to enable the indistinguishability proof in Section 3.2.

Definition 3.1. (Cluster Trees) *For $k \in \mathbb{N}$, a cluster tree skeleton (*CT skeleton*) is a tree $CT_k = (\mathcal{C}_k, \mathcal{A}_k)$, rooted at $C_0 \in \mathcal{C}_k$, which describes constraints imposed on a corresponding* CT graph *G_k.*

- *For each* cluster[9] *$C \in \mathcal{C}_k$, there is a corresponding independent set in G_k.*
- *An edge connecting clusters C and C' in CT_k is labeled with $\{(C, x), (C', y)\}$ for $x, y \in \mathbb{N}$. This expresses the constraint that in G_k, C and C' must be connected as a biregular bipartite graph, where each node in C has x neighbors in C' and each node in C' has y neighbors in C. We say that C (C') is connected to C' (C) via* outgoing label *x (y).*
- *G_k contains no further nodes or edges.*

Note that CT_k imposes many constraints on G_k. Choosing the size of C_0 determines the number of nodes and edges in G_k, and node degrees are fully determined by CT_k as well. However, there is substantial freedom regarding how to realize the connections between adjacent clusters. As mentioned earlier, this permits leveraging graph lifts to obtain cluster tree graphs G_k of high girth in Step 3 of the KMW construction.

3.1 Construction of Cluster Tree Skeletons

Definition 3.1 (p. 5) does not detail the structure of CT_k. To specify this structure, we use the following terminology.

Definition 3.2. (Cluster position, level, and parent) *A leaf cluster C in CT_k has position* leaf, *while internal clusters have position* internal. *The* level *of C*

[9]"Cluster" here is used in the sense of "associated set of nodes," referring to its role in G_k. We use the term to refer to both nodes in CT_k and the corresponding independent sets in G_k.

is its distance to C_0. The parent cluster *of $C \neq C_0$ is its parent in CT_k.*

Given $\beta \geq 2(k+1)$, the structure of CT_k is now defined inductively. The base case of the construction is CT_1.

DEFINITION 3.3. (BASE CASE CT_1) *$CT_1 = (\mathcal{C}_1, \mathcal{A}_1)$, where $\mathcal{C}_1 := \{C_0, C_1, C_2, C_3\}$ and*

$$\mathcal{A}_1 := \{\{(C_0, \beta^0), (C_1, \beta^1)\}, \{(C_0, \beta^1), (C_2, \beta^2)\}, \{(C_1, \beta^0), (C_3, \beta^1)\}\}.$$

Based on CT_{k-1}, for $k \geq 2$, CT_k is grown as follows.

DEFINITION 3.4. (GROWTH RULES FOR CT_k GIVEN CT_{k-1})

1. *To each internal cluster C in CT_{k-1}, attach a new neighboring cluster C' via an edge $\{(C, \beta^k), (C', \beta^{k+1})\}$.*
2. *To each leaf cluster C in CT_{k-1} that is connected to its parent cluster via outgoing label β^q, add a total of k neighboring clusters: one cluster C' with an edge $\{(C, \beta^p), (C', \beta^{p+1})\}$ for each $p \in [k]_0 \setminus \{q\}$.*

Note that with this definition, CT_k is a regular tree but a CT graph G_k is not regular. Figure 1 (p. 2) shows CT_1 in its hierarchical and flat representations, and flat representations of CT_2 and CT_3 are given in Figure 2 (p. 7) to illustrate the growth process.[10] In all figures, we write i for outgoing label β^i to reduce visual clutter, and in the flat representations, outgoing labels are depicted like port numbers, i.e., the edge label corresponding to C is depicted next to C.

3.2 Indistinguishability given High Girth As observed by Kuhn et al. [23, 26], showing k-hop indistinguishability becomes easier when the nodes' k-hop subgraphs are trees, i.e., the girth is at least $2k+1$. Notably, in a CT graph G_k with $g \geq 2k+1$, the topology of a node's k-hop subgraph is determined entirely by the structure of the skeleton CT_k. Hence, we will be able to establish the following theorem without knowing the details of G_k.

THEOREM 3.1. (k-HOP INDISTINGUISHABILITY OF NODES IN C_0 AND C_1) *Let G_k be a CT graph of girth $g \geq 2k+1$. Then any $v_0 \in C_0$ and $v_1 \in C_1$ are k-hop indistinguishable.*

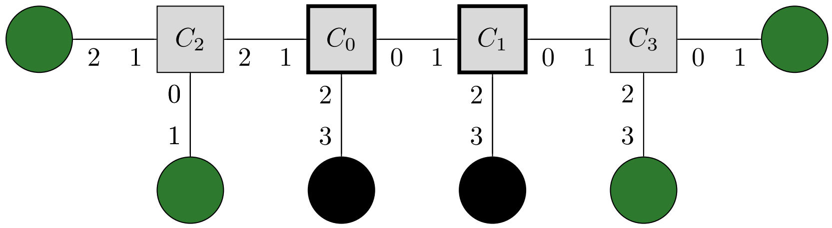
[10] The labels of the edges connecting leaf clusters in CT_3 to the rest of CT_3 are omitted in the drawing. They are such that every internal cluster has outgoing labels $\{\beta^i \mid i \in [3]_0\}$, and if a leaf cluster C is connected to an internal cluster C' with label β^i outgoing from C', then C has outgoing label β^{i+1}.

By Definition 2.2 (p. 5), $v_0 \in C_0$ and $v_1 \in C_1$ are k-hop indistinguishable if and only if there exists an isomorphism $\phi : V(G_k^k(v_0)) \rightarrow V(G_k^k(v_1))$ with $\phi(v_0) = v_1$. We prove the theorem constructively by showing the correctness of Algorithm 1 (p. 8), which purports to find such an isomorphism.

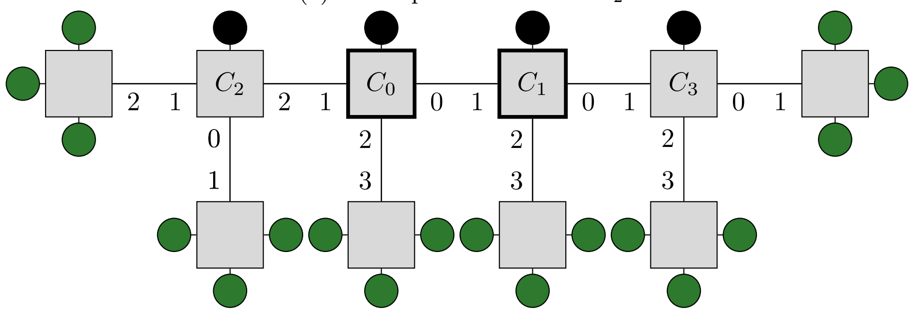
Algorithm 1 (p. 8) implements a *coupled depth-first search* on the k-hop subgraphs of $v_0 \in C_0$ and $v_1 \in C_1$. Its main function, FINDISOMORPHISM(G_k, k, v_0, v_1), receives a CT graph G_k with high girth, along with the parameter k, and one node from each of C_0 and C_1 as input, and it outputs the ϕ we are looking for. To obtain ϕ, FINDISOMORPHISM maps v_0 to v_1 and then calls the function WALK(v_0, v_1, $\perp$, k) before it returns ϕ. The WALK function extends ϕ by mapping the *newly discovered* nodes in the neighborhoods of its first two input parameters (v and $w := \phi(v)$, initially: v_0 and v_1) to each other with the help of the function MAP. The third parameter of WALK (*prev*, initially: $\perp$) ensures that we only define ϕ for newly discovered nodes, while the fourth parameter (*depth*, initially: k) controls termination when WALK calls itself recursively on the newly discovered neighbors (and the newly discovered neighbors of these neighbors, and so on) until the entire k-hop subgraph of v_0 has been visited.

The tricky part now is to ascertain that the interplay between the functions WALK and MAP makes ϕ a bijection from $V(G_k^k(v_0))$ to $V(G_k^k(v_1))$, i.e., nodes that are paired up always have the same degree. Here, the representation of node neighborhoods used by the WALK function is key, which is based on the insight that the set of nodes neighboring v (resp. w) can be partitioned by the outgoing labels in CT_k through which neighboring nodes are discovered from v (w). Since these labels lie in $\{\beta^i \mid i \in [k+1]_0\}$, WALK represents the neighborhood of v (w) as a list N_v (N_w) of $k+2$ (possibly empty) lists (Algorithm 1, l. 9–13, p. 8). The list at index i holds all *previously undiscovered* nodes (we require $v' \neq prev$ and $w' \neq \phi(prev)$) connected to v (w) via v's (w's) outgoing label β^i, in arbitrary order.

The WALK function passes N_v and N_w to the function MAP (Algorithm 1, l. 14, p. 8), which sets $\phi(N_v[i][j]) := N_w[i][j]$ where possible (Algorithm 1, l. 19–21, p. 8). It then treats the special case that some nodes in N_v and N_w remain unmatched (Algorithm 1, l. 22–25, p. 8). By construction, without this special case, the ϕ returned by FINDISOMORPHISM is already an isomorphism between the subgraphs of $G_k^k(v_0)$ and $G_k^k(v_1)$ induced by the nodes of the domain for which ϕ is defined (and their images under ϕ). However, we still need to show that our special case suffices to extend this restricted isomorphism to a full isomorphism between $G_k^k(v_0)$ and $G_k^k(v_1)$. To facilitate our reasoning, we



(a) Flat representation of CT_2



(b) Flat representation of CT_3

Figure 2: Representations of CT_2 and CT_3, colored by cluster types; grey: internal, black: first growth rule, green: second growth rule.

introduce *cluster identities*:

Definition 3.5. (Cluster identity $C(v)$) *Given a node v in a CT graph G_k, we refer to its cluster in CT_k as its* cluster identity, *denoted as $C(v)$. For example, for $v_0 \in C_0$ and $v_1 \in C_1$, we have $C(v_0) = C_0$, $C(v_1) = C_1$, and $C(v_0) \neq C(v_1)$.*

Our argument will crucially rely on the concepts of *node position* and *node history*:

Definition 3.6. (Node position) *For $i \in \{0, 1\}$, the* position *of a node v in $G_k^k(v_i)$ is the position of its cluster $C(v)$ in the CT skeleton (*internal *or* leaf*).*

Definition 3.7. (Node history) *For $i \in \{0, 1\}$, the* history *of a node $v \neq v_i$ in $G_k^k(v_i)$ is the outgoing label of the edge connecting $C(v)$ to $C(prev)$, i.e., β^x if the corresponding edge is $\{(C(v), \beta^x), (C(prev), \beta^{x'})\}$.*

We begin with a simple observation:

Lemma 3.1. (Variables determining node neighborhoods) *For v in $G_k^k(v_0) \setminus \{v_0\}$, let $w := \phi(v)$. When* Map *is called with parameters N_v and N_w (Algorithm 1 l. 14, p. 8), the numbers of nodes in $N_v[i]$ and $N_w[i]$ for $i \in [k+1]_0$ are uniquely determined the* position *and the* history *of v and w. If v and w agree on* position *and* history, *$len(N_v[i]) = len(N_w[i])$ for all $i \in [k+1]_0$. If v and w agree on position* internal *but disagree on* history, *where v has history β^x and w has history β^y, we have $len(N_v[i]) = len(N_w[i])$ for all $i \in [k+1]_0 \setminus \{x, y\}$, $len(N_v[x]) = len(N_w[x]) - 1$, and $len(N_v[y]) - 1 = len(N_w[y])$.*

Proof. If $u \in \{v, w\}$ has position *internal*, we know that $C(u)$ has outgoing labels $\{\beta^i \mid i \in [k]_0\}$ by the construction of the CT skeleton. Denoting by $z \in \{x, y\}$ the exponent of u's *history*, we have that there are β^i nodes in $N_u[i]$ for $i \in [k]_0 \setminus \{z\}$, $\beta^z - 1$ nodes in $N_u[z]$ (as *prev* or $\phi(prev)$ are removed, respectively), and zero nodes in $N_u[k+1]$.

If $u \in \{v, w\}$ has position *leaf*, all nodes in N_u belong to the same cluster C', u has β^z neighbors in this cluster, and *prev* (resp. $\phi(prev)$) lies in this cluster as well. Hence, $len(N_u[z]) = \beta^z - 1$ and $len(N_u[i]) = 0$ for all $i \in [k+1]_0 \setminus \{z\}$.

From these observations, the claims of the lemma follow immediately. ∎

Using Lemma 3.1 (p. 7), we can rephrase the task of proving Theorem 3.1 (p. 6) as a simple condition on the pairs of nodes on which Walk is called recursively.

Corollary 3.1. (Sufficient condition for correctness of Algorithm 1) *Given a CT graph G_k with girth at least $2k+1$, if all pairs of nodes created by* Map *on which* Walk *is called recursively (i) agree on* position *and* history *or (ii) agree on position* internal, *Algorithm 1 (p. 8) produces an isomorphism between $G_k^k(v_0)$ and $G_k^k(v_1)$.*

Algorithm 1: Find an isomorphism $\phi : V(G_k^k(v_0)) \to V(G_k^k(v_1))$

1 **Function** FindIsomorphism(G_k, k, v_0, v_1):
  Input: A CT graph G_k with $g \geq 2k+1$, $k \in \mathbb{N}$, $v_0 \in C_0$, $v_1 \in C_1$
  Output: Isomorphism $\phi : V(G_k^k(v_0)) \to V(G_k^k(v_1))$
2   $\phi \leftarrow$ empty map
3   $\phi(v_0) \leftarrow v_1$
4   Walk(v_0, v_1, $\perp$, k)
5   **return** ϕ

6 **Function** Walk(v, w, $prev$, $depth$):
7   **if** $depth = 0$ **then**
8   **return**
9   $N_v \leftarrow$ empty list of length $k+2$
10   $N_w \leftarrow$ empty list of length $k+2$
11   **for** $i \leftarrow 0$ **to** $k+1$ **do**
    // if edge β^i does not exist, $N_v[i]$ (resp. $N_w[i]$) is empty
12     $N_v[i] \leftarrow$ list of new nodes $v' \neq prev$ found using edge β^i from v
13     $N_w[i] \leftarrow$ list of new nodes $w' \neq \phi(prev)$ found using edge β^i from w
14   Map(N_v, N_w)
15   **for** $i \leftarrow 0$ **to** $k+1$ **do**
16   **for** v' **in** $N_v[i]$ **do**
17   Walk(v', $\phi(v')$, v, $depth - 1$)

18 **Function** Map(N_v, N_w):
19   **for** $i \leftarrow 0$ **to** $k+1$ **do**
    // $zip(\cdot,\cdot)$ yields element tuples until the shorter list ends
20   **for** v', w' **in** $zip(N_v[i], N_w[i])$ **do**
21       $\phi(v') \leftarrow w'$
  // $len(\cdot)$ returns the length of a list
22   **if** $\exists\, i \in [k+1]_0 : len(N_v[i]) \neq len(N_w[i])$ **then**
    // we will prove that $len(L_v[i]) = len(L_w[i])$ for $i \in [k+1]_0 \setminus \{i_v, i_w\}$
23     $i_v \leftarrow i \in [k+1]_0 : len(N_v[i]) = len(N_w[i]) + 1$
24     $i_w \leftarrow i \in [k+1]_0 : len(N_v[i]) + 1 = len(N_w[i])$
    // $L[i][-1]$ retrieves the last element from list i in L
25     $\phi(N_v[i_v][-1]) \leftarrow N_w[i_w][-1]$

Proof. Due to the assumed high girth, Algorithm 1 (p. 8) produces an isomorphism between $G_k^k(v_0)$ and $G_k^k(v_1)$ if $\phi\big|_{N_v}$ (i.e., ϕ with its domain restricted to the neighborhood of v) is a bijection from N_v to $N_{\phi(v)}$ for all v in $G_k^k(v_0)$ with $d(v, v_0) < k$. For v_0 and $\phi(v_0) = v_1$, this holds because they both have β^i neighbors in the clusters connected to them via outgoing edge label β^i for $i \in [k]_0$, i.e., $len(N_v[i]) = len(N_w[i])$ for $i \in [k]_0$ (and $len(N_v[k+1]) = len(N_w[k+1]) = 0$). Hence, Map ensures that $\phi(N_v) = N_w$. For nodes $v \neq v_0$ and $w := \phi(v)$ paired by Map that agree on *position* and *history*, Lemma 3.1 (p. 7) shows that $len(N_v[i]) = len(N_w[i])$ for all $i \in [k+1]_0$, so again Map succeeds. The last case is that v and w agree on position *internal*. In this case, applying Lemma 3.1 (p. 7) and noting that Map takes care of the resulting mismatch in list lengths in Lines 22–25 proves that Map succeeds here, too. ◻

The next step in our reasoning is to craft an algorithmic invariant establishing the preconditions of Corollary 3.1 (p. 7). Reflecting the inductive construction of cluster trees, we will prove it inductively. To this end, for $i \in [k]$, we interpret CT_i as a subgraph of CT_k by simply stripping away all clusters that were added after constructing CT_i.

Recall that $G_k^k(v_0)$ and $G_k^k(v_1)$ are trees, because the girth of G_k is at least $2k+1$. Treating these trees as rooted at v_0 and v_1, respectively, Algorithm 1 (p. 8) maps nodes at depth d in $G_k^k(v_0)$ to nodes at depth d in $G_k^k(v_1)$. Accordingly, the following notion will be useful.

Definition 3.8. (Node parent) *For $v \in G_k^k(v_i)$, $i \in \{0, 1\}$, with $d(v_i, v) > 0$, the* parent *of v in $G_k^k(v_i)$, denoted $p_i(v)$, is the node through which v is discovered from v_i in Algorithm 1 (p. 8).*

We are now ready to state the main invariant of Algorithm 1 (p. 8).

Definition 3.9. (Main Invariant of Algorithm 1) *For $0 < d < k$, suppose that v and $w := \phi(v)$ lie at distance d from v_0 and v_1, respectively. Then exactly one of the following holds:*

1. *$C(v), C(w) \in CT_d$, and v and w agree on* history *or both have* history *$\leq \beta^{d+1}$.*
2. *There is some i with $d < i \leq k$ such that $C(v), C(w) \in CT_i \setminus CT_{i-1}$, v and w agree on* history*, and $C(v)$ and $C(w)$ are connected from CT_{i-1} with outgoing labels $(\beta^{j'}, \beta^{j'+1})$ for the same $j' \in [i]_0$.*

Intuitively, the first case tracks how asymmetry propagates through the construction, and counts down how many levels of the iterative construction remain that hide it: If there is a mismatch in history, it is confined to CT_d, i.e., not only $C(v), C(w) \in CT_d$, but also the history of v and w has labels corresponding to CT_d.

However, we need to take into account that attaching leaf clusters to internal clusters by the first growth rule is needed, as otherwise "older" clusters could be easily recognized without having to inspect the far-away clusters added by the second growth rule. These leaves then recursively sprout their own subtrees, which again sprout their own subtrees, and so on. If the recursive construction of the isomorphism visits such clusters before reaching distance k from root C_0 (resp. C_1), by design, it enters subtrees of CT_k that started to grow in the same iteration. Thus, these subtrees are completely symmetric, and they are entered with matching history. This is captured by the second case of the invariant. The intuition of the invariant and its interplay with Corollary 3.1 (p. 7) are illustrated in Figure 3 (p. 10).

Recall that the growth rules only attach leaves, and do so for each cluster. Hence, the clusters in CT_{k-1} are exactly the internal clusters in CT_k, while $CT_k \setminus CT_{k-1}$ contains all leaves. Thus, in the first case of the invariant, v and w agree on position *internal*, and in the second case, they agree on *position* and *history*. Therefore, Theorem 3.1 (p. 6) follows from Corollary 3.1 (p. 7) once the invariant is established.

Having carved out the crucial properties of the construction in this invariant, we can now complete the proof of the theorem with much less effort than in [26].

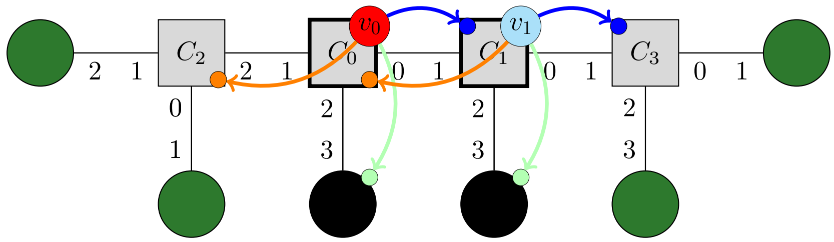
Lemma 3.2. (Main invariant holds) *Algorithm 1 (p. 8) satisfies the invariant stated in Definition 3.9 (p. 8).*

Proof. We prove the claim for fixed k by induction on d. For v and $w := \phi(v)$ at distance $d = 1$ from $v_0 = p_0(v)$ and $v_1 = p_1(w)$, respectively, v and w are matched in the initial call to Walk with v_0 and v_1 as arguments. In this call, $len(N_{v_0}[i]) = len(N_{v_1}[i])$ for all $i \in [k+1]_0$, i.e., only nodes corresponding to the same outgoing labels get matched. Inspecting CT_1 and taking into account the CT growth rules, we see that for $i \in \{0, 1\}$, the matched nodes lie in clusters that are present already in CT_1 and have outgoing labels of at most β^2 (i.e., the first case of the invariant holds), while for $i > 1 = d$, both nodes lie in clusters from $CT_i \setminus CT_{i-1}$ with outgoing labels of β^{i+1} and their clusters are connected from CT_{i-1} with outgoing labels (β^i, β^{i+1}) (i.e., the second case of the invariant holds).

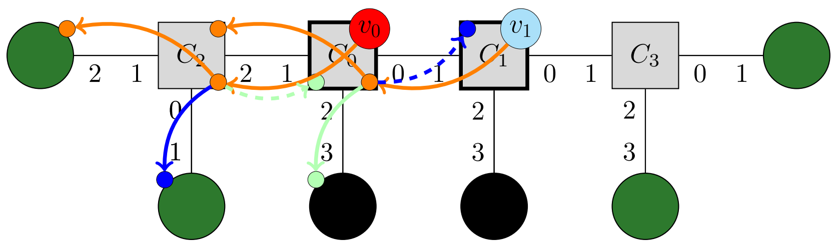
For the inductive step, assume that the invariant is established up to distance d for $1 \le d < k-1$, and consider v, $w := \phi(v)$ at distance $d+1$ from v_0 and v_1, respectively. We apply the invariant to $v' := p_0(v)$ and $w' := p_1(w)$ and distinguish between its two cases.

(1) Suppose that $C(v'), C(w') \in CT_d$, and v' and w' agree on *history* or both have *history* $\le \beta^{d+1}$. As $d < k$, we know that v' and w' agree on position *internal*. By Lemma 3.1 (p. 7), the call to Walk on v' and w' thus satisfies that $len(N_{v'}[i]) = len(N_{w'}[i])$ for all $i \in [k+1]_0 \setminus \{j, j'\}$, where $\beta^j, \beta^{j'}$ for $j, j' \le d+1$ are the histories of v' and w', respectively. If $C(v) \in CT_{d+1}$, Lemma 3.1 (p. 7) entails that $v \in N_{v'}[i]$ for some $i \le d+1$, and Walk chooses $w = \phi(v)$ from $N_{w'}[i']$ for some $i' \le d+1$. Due to the CT growth rules, since $C(v'), C(w') \in CT_d$, the incident edges of $C(v')$ and $C(w')$ with outgoing labels of at most β^{d+1} lead to clusters in CT_{d+1}, and the history of nodes discovered by traversing these edges is at most β^{d+2}. Hence, if $C(v) \in CT_{d+1}$, it follows that the first case of the invariant holds for v and w. If $C(v) \notin CT_{d+1}$, we have that $C(v) \in CT_i \setminus CT_{i-1}$ for some $i > d+1$, yielding $len(N_{v'}[i]) = len(N_{w'}[i])$, and thus, $w \in N_{w'}[i]$. As $C(v')$ and $C(w')$ are internal clusters in CT_{d+1}, we can conclude that both $C(v)$ and $C(w)$ have been added to the cluster tree in the i^{th} construction step using growth rule 1. Hence, we get that $C(v), C(w) \in CT_i \setminus CT_{i-1}$ with v and w agreeing on *history* β^{i+1}, and since $C(v'), C(w') \in CT_d \subseteq CT_{i-1}$, $C(v')$ and $C(w')$ are connected from CT_{i-1} with outgoing labels (β^i, β^{i+1}), and the second case of the invariant holds for v and w.

(2) Assume that there is some i with $d < i \le k$ such that $C(v'), C(w') \in CT_i \setminus CT_{i-1}$, v' and w' agree on *history*, and $C(v')$ and $C(w')$ are connected from CT_{i-1} with outgoing labels $(\beta^{j'}, \beta^{j'+1})$ for the same $j' \in [i]_0$. Since $C(v')$ and $C(w')$ were added in the same growth round, v' and w' also agree on *position*, so $v \in N_{v'}[j]$ and $w \in N_{w'}[j]$ for the same $j \in [k+1]_0$ by Lemma 3.1 (p. 7), and similarly, as $C(v')$ and $C(w')$ are both added when forming CT_i from CT_{i-1} and connected from CT_{i-1} with the same labels, v and w agree on *history*. Hence, if $j \neq j'+1$, then $C(v), C(w) \in CT_{i'+1} \setminus CT_{i'}$ for some $i' \ge i+1$, $C(v)$ and $C(w)$ are connected from CT_i with the same labels, and since $i+1 > d+1$, the second case of the invariant holds for v and w. If $j = j'+1$, $C(v) = C(p_0(v'))$ and $C(w) = C(p_1(w'))$. As Walk mapped v' to w', we have that $p_0(v')$ was mapped to $\phi(p_0(v')) = p_1(w')$, where $p_0(v')$ and $p_1(w')$ lie at distance $d-1$ from v_0 and v_1, respectively. Applying the invariant to these nodes, the first case and the second case with $i' \le d+1$ both imply that $C(v), C(w) \in CT_{d+1}$, establishing the first case of the invariant for v and w. And if the second case applies with $i' > d+1$, then the second case of the invariant holds for v and w. ∎



(a) $d = 1$ from v_0 and v_1: for the blue nodes, the first case of the invariant holds *with* agreement on *history*; for the orange nodes, the first case of the invariant holds *without* agreement on *history*; and for the green nodes, the second case of the invariant holds.



(b) $d = 2$ from orange nodes at distance $d = 1$: because the invariant holds for $d = 1$, Corollary 3.1 (p. 7) ensures that Algorithm 1 (p. 8) produces an isomorphism between $G_2^2(v_0)$ and $G_2^2(v_1)$ by mapping exactly one node in $G_2^2(v_0)$ discovered via the solid blue arrow to one node in $G_2^2(v_1)$ discovered via the solid green arrow (Algorithm 1, p. 8, l. 22–25).

Figure 3: Illustration of Definition 3.9 (p. 8) for CT_2. Cluster colors, shapes, and borders drawn as in Figure 2 (p. 7). Nodes $v_0 \in C_0$ and $v_1 \in C_1$ are depicted as medium-size circles; representatives of nodes seen via a certain outgoing edge are depicted as small circles and connected to their parents by arrows. Node and arrow colors show outgoing edge labels (e.g., blue nodes are seen via the outgoing edge β^0); dashed arrows indicate that $\beta^i - 1$, rather than β^i, nodes are discovered via the outgoing label indicated by the arrow color.

With this result, we can summarize:

Proof of Theorem 3.1 (p. 6). Follows from the correctness of Algorithm 1 (p. 8) for CT graphs G_k with girth $\geq 2k + 1$, established via Lemma 3.2 (p. 9) and Corollary 3.1 (p. 7). ∎

References

[1] Dana Angluin and A. Gardiner. Finite common coverings of pairs of regular graphs. *Journal of Combinatorial Theory, Series B*, 30(2):184–187, 1981.

[2] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. In *60th IEEE Annual Symposium on Foundations of Computer Science (FOCS '19)*, pages 481–497, 2019.

[3] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. Almost Global Problems in the LOCAL Model. In *32nd International Symposium on Distributed Computing (DISC '18)*, pages 9:1–9:16, 2018.

[4] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. How much does randomness help with locally checkable problems? In *Proceedings of the 39th Symposium on Principles of Distributed Computing (PODC '20)*, pages 299–308, 2020.

[5] Alkida Balliu, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. New classes of distributed time complexity. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC '18)*, pages 1307–1318, 2018.

[6] Reuven Bar-Yehuda, Keren Censor-Hillel, Mohsen Ghaffari, and Gregory Schwartzman. Distributed approximation of maximum independent set and maximum matching. In *Proceedings of the 36th ACM Symposium on Principles of Distributed Computing (PODC '17)*, pages 165–174, 2017.

[7] Reuven Bar-Yehuda, Keren Censor-Hillel, and Gregory Schwartzman. A distributed $(2+\epsilon)$-approximation for vertex cover in

$O(\log \Delta/\epsilon \log\log \Delta)$ rounds. In *Proceedings of the 35th ACM Symposium on Principles of Distributed Computing (PODC '16)*, pages 3–8, 2016.

[8] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed lovász local lemma. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC '16)*, page 479–488, 2016.

[9] Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. The complexity of distributed edge coloring with small palettes. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '18)*, pages 2633–2652, 2018.

[10] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the local model. In *57th IEEE Annual Symposium on Foundations of Computer Science (FOCS '16)*, pages 615–624, 2016.

[11] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the local model. *SIAM J. Comput.*, 48(1):122–143, 2019.

[12] Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the local model. *SIAM J. Comput.*, 48(1):33–69, 2019.

[13] Corinna Coupette and Christoph Lenzen. A Breezing Proof of the KMW Bound. *CoRR*, abs/2002.06005, 2020.

[14] Andrzej Czygrinow, Michal Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC '08)*, page 78–92, 2008.

[15] Paul Erdős and Horst Sachs. Reguläre graphen gegebener taillenweite mit minimaler knotenzahl. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 12(3):251–257, 1963.

[16] Laurent Feuilloley, Pierre Fraigniaud, and Juho Hirvonen. A Hierarchy of Local Decision. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP '16)*, pages 118:1–118:15, 2016.

[17] Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *J. ACM*, 60(5):35:1–35:26, October 2013.

[18] Mohsen Ghaffari, David G Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS '18)*, pages 662–673, 2018.

[19] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC '17)*, pages 784–797, 2017.

[20] Mika Göös, Juho Hirvonen, and Jukka Suomela. Lower bounds for local approximation. *J. ACM*, 60(5):39:1–39:23, October 2013.

[21] Mika Göös, Juho Hirvonen, and Jukka Suomela. Linear-in-Δ lower bounds in the local model. *Distributed Computing*, 30(5):325–338, 2017.

[22] Fabian Kuhn. Personal communication, 2020.

[23] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC '04)*, pages 300–309, 2004.

[24] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA '06)*, pages 980–989, 2006.

[25] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *CoRR*, abs/1011.5470, 2010.

[26] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *J. ACM*, 63(2):17:1–17:44, March 2016.

[27] Juhana Laurinharju and Jukka Suomela. Brief announcement: Linial's lower bound made easy. In *Proceedings of the 33rd ACM Symposium on Principles of Distributed Computing (PODC '14)*, page 377–378, 2014.

[28] Felix Lazebnik and Vasiliy A. Ustimenko. Explicit construction of graphs with an arbitrary large girth and of large size. *Discrete Applied Mathematics*, 60(1):275–284, 1995.

[29] Christoph Lenzen and Roger Wattenhofer. Leveraging linial's locality limit. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC '08)*, page 394–407, 2008.

[30] Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

[31] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM J. Comput.*, 24(6):1259–1277, December 1995.

[32] Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distributed Computing*, 14(2):97–100, 2001.

[33] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

[34] Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20)*, pages 350–363, 2020.

[35] Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, March 2013.

[36] Jukka Suomela. Distributed algorithms (online textbook), 2019.