Distributed Computation with Local Advice

Alkida Balliu
Gran Sasso Science Institute, L'Aquila, Italy

Sebastian Brandt
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Fabian Kuhn
University of Freiburg, Germany

Krzysztof Nowicki
Unaffiliated

Dennis Olivetti
Gran Sasso Science Institute, L'Aquila, Italy

Eva Rotenberg
IT University of Copenhagen, Denmark

Jukka Suomela
Aalto University, Finland

Abstract

Algorithms with advice have received ample attention in the distributed and online settings, and they have recently proven useful also in dynamic settings. In this work we study *local computation with advice*: the goal is to solve a graph problem Π with a distributed algorithm in $T(\Delta)$ communication rounds, for some function T that only depends on the maximum degree Δ of the graph, and the key question is how many bits of advice per node are needed.

Some of our results regard *Locally Checkable Labeling problems* (LCLs), which is an important family of problems that includes various coloring and orientation problems on finite-degree graphs. These are constraint-satisfaction graph problems that can be defined with a finite set of valid input/output-labeled neighborhoods.

Our main results are:

1. Any *locally checkable labeling problem* can be solved with only 1 bit of advice per node in graphs with *sub-exponential growth* (the number of nodes within radius r is sub-exponential in r; for example, grids are such graphs). Moreover, we can make the set of nodes that carry advice bits arbitrarily sparse. As a corollary, any locally checkable labeling problem admits a *locally checkable proof* with 1 bit per node in graphs with sub-exponential growth.
2. The assumption of sub-exponential growth is complemented by a conditional lower bound: assuming the *Exponential-Time Hypothesis*, there are locally checkable labeling problems that cannot be solved in general with any constant number of bits per node.
3. In any graph we can find an *almost-balanced orientation* (indegrees and outdegrees differ by at most one) with 1 bit of advice per node, and again we can make the advice arbitrarily sparse. As a corollary, we can also *compress an arbitrary subset of edges* so that a node of degree d stores only $d/2 + 2$ bits, and we can *decompress* it locally, in $T(\Delta)$ rounds.
4. In any graph of maximum degree Δ, we can find a Δ-coloring (if it exists) with 1 bit of advice per node, and again, we can make the advice arbitrarily sparse.
5. In any 3-colorable graph, we can find a 3-coloring with 1 bit of advice per node. As a corollary, in bounded-degree graphs there is a locally checkable proof that certifies 3-colorability with 1 bit of advice per node, while prior work shows that this is not possible with a *proof labeling scheme* (PLS), which is a more restricted setting where the verifier can only see up to distance 1.

Our work shows that for many problems the key threshold is not whether we can achieve 1 bit of advice per node, but whether we can make the advice arbitrarily sparse. To formalize this idea, we develop a general framework of *composable* schemas that enables us to build algorithms for local computation with advice in a modular fashion: once we have (1) a schema for solving Π_1 and (2) a schema for solving Π_2 assuming an oracle for Π_1, we can also compose them and obtain (3) a schema

39th International Symposium on Distributed Computing (DISC 2025).
Editor: Dariusz R. Kowalski; Article No. 12; pp. 12:1–12:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

that solves Π_2 without the oracle. It turns out that many natural problems admit composable schemas, all of them can be solved with only 1 bit of advice, and we can make the advice arbitrarily sparse.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Distributed computing models; Mathematics of computing → Graph algorithms

Keywords and phrases Distributed graph algorithms, LOCAL model, computation with advice, locally checkable labeling problems, proof labeling schemes, locally checkable proofs, graph coloring, exponential-time hypothesis

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.12

Related Version *Full Version*: `https://arxiv.org/abs/2405.04519`

Funding This work was supported by the MUR (Italy) Department of Excellence 2023 - 2027 for GSSI, the PNRR MIUR research project GAMING "Graph Algorithms and MinINg for Green agents" (PE0000013, CUP D13C24000430001), the Independent Research Fund Denmark grant 2020-2023 (9131-00044B) "Dynamic Network Analysis" and the VILLUM Foundation grant (VIL37507) "Efficient Recomputations for Changeful Problems".

Acknowledgements This work was initiated at Dagstuhl Seminar 23071: "From Big Data Theory to Big Data Practice". We would like to thank Sameep Dahal, Laurent Feuilloley, and Augusto Modanese for insightful discussions, and the anonymous reviewers for their helpful comments on previous versions of this work.

1 Introduction

Our work explores *what can and cannot be computed locally with the help of advice*. Our main focus is understanding advice in the context of classic local graph problems, such as vertex coloring.

While computation with different forms of advice has been explored in a wide range of distributed settings [24, 23, 14, 31, 13, 35, 46, 48, 25, 36, 51, 22, 39, 47, 26], there is hardly any prior work on solving classic local graph problems. A rare example of prior work is [20] from 2007, which studied the question of how much advice is necessary to break Linial's [44] lower bound for coloring cycles.

We initiate a systematic study of exactly how much advice is needed in the context of a wide range of graph problems. As we will see in this work, the exploration of the advice complexity of graph problems opens up connections with many other topics – it is linked with *distributed proofs* [40, 41, 42, 43, 34, 17], *distributed decompression*, the notion of *order-invariant algorithms* [50], and also with the *exponential-time hypothesis* [37] in computational complexity theory.

1.1 Local computation with advice

Let us first formalize the setting we study:

> A graph problem Π can be *solved with β bits of advice* if there exists a $T(\Delta)$-round distributed algorithm $\mathcal{A}$, such that for any graph G that admits a solution to Π, there is an assignment of β-bit labels on vertices, such that the output of $\mathcal{A}$ on the labeled graph is a solution to Π.

We note that on graphs that do not admit a solution to Π, Algorithm $\mathcal{A}$ is allowed to behave arbitrarily. For instance, when we consider the 3-coloring problem, Algorithm $\mathcal{A}$ may produce an arbitrary output on graphs that are not 3-colorable.

We will work in the usual LOCAL model of distributed computing. In an n-node graph, the nodes are labeled with unique identifiers from $\{1, 2, \ldots, \mathrm{poly}(n)\}$. We emphasize that the advice may depend on the assignment of identifiers, and algorithm $\mathcal{A}$ can freely make use of both the advice and the identifiers.

Now we seek to understand this question:

> What is the smallest β such that Π can be solved with β bits of advice?

Note that if Π is, for example, the 3-coloring problem, it is trivial to solve with $\beta = 2$ bits of advice per node, as we can directly encode the solution. The key question is how much better we can do.

Our work shows that for many problems the key threshold is not whether we can achieve 1 bit of advice per node, but whether we can make the advice *arbitrarily sparse*, that is, make the ratio between 1s and 0s assigned to the nodes of the graph to be an arbitrarily small constant. This is particularly useful, as it enables us to *compose* multiple sparse advice schemes, so that it suffices to use just one bit per node in total (we will elaborate on this in Section 1.7). Hence, a large part of this paper addresses the following question:

> Which problems admit arbitrarily sparse advice?

The notion of the sparsity of the advice is discussed in more detail later in the paper. In this context, our paper shows that some problems can be solved with arbitrarily sparse advice. On the other hand, we also show that assuming the Exponential-Time Hypothesis, for any constant c, there exist problems that cannot be solved with c bits of advice. Finally, there are some problems like 3-coloring that can be solved with 1 bit of advice, but where it is not clear whether it can be solved with arbitrarily sparse advice. We discuss all of those points in more detail in the remaining part of this section; we refer to the full version of this work for the omitted proofs.

1.2 Contribution 1: LCLs in bounded-growth graphs

Locally checkable labeling problems (LCL), first introduced by Naor and Stockmeyer [50] in the 1990s, are one of the most extensively studied families of problems in the theory of distributed graph algorithms. These are graph problems that can be specified by giving a *finite* set of valid local neighborhoods. Many key problems such as vertex coloring, edge coloring, maximal independent set, maximal matching, sinkless orientation, and many other splitting and orientation problems are examples of LCLs, at least when restricted to bounded-degree graphs. Thanks to the extensive research effort since 2016, we now understand very well the landscape of *all* LCL problems and their computational complexities across different models of distributed computing [5, 3, 11, 28, 2, 19, 54, 9, 10, 30, 4].

We design a schema that allows us to solve *any* LCL problem with just one bit of advice in graphs with a sub-exponential growth (the number of nodes in a radius-r neighborhood is sub-exponential in r):

> Any LCL problem can be solved with 1 bit of advice per node in sub-exponential growth graphs.

Furthermore, we show that the encoding can be made arbitrarily sparse. Note that e.g. grids have polynomial growth while e.g. regular trees have exponential growth, so the result is applicable in grids but not in trees.

1.2.1 Application: locally checkable proofs in bounded-growth graphs

One prominent application of this result is its connection with *distributed proofs* [40, 41, 42, 43, 17], and in particular with *locally checkable proofs* [34]. Consider any LCL Π. Assume that our task is to prepare a distributed proof that shows that in a graph G there exists a feasible solution of Π (for example, if Π is the task of 10-coloring, then the task is to certify that the chromatic number of G is at most 10). Now if G has sub-exponential growth, we can use our result to prepare a 1-bit advice that enables the algorithm to find a solution of Π. Our advice is the proof: to verify it, we simply try to recover a solution with the help of the advice, and then check that the output is feasible in all local neighborhoods (recall that Π is locally checkable). We obtain the following corollary:

> Any LCL problem admits a locally checkable proof with 1 bit per node in graphs with sub-exponential growth.

Note that this is not a proof labeling scheme as defined in [40, 41, 42, 43], as the verifier running at node u may need to see more than just the identifier and the proof label of u and the proof labels of u's immediate neighbors. However, in bounded-degree graphs it is a locally checkable proof (LCP) as defined in [34]; for a fixed Δ the verification radius is a constant $T(\Delta)$.

So to summarize, if we can solve some LCL problem Π with b bits of advice per node, then we also have an LCP for the graph property "G admits a feasible solution to Π" with b-bit proofs per node. The converse is not true: Consider the LCL problem Π that encodes the task "orient edges so that each node has indegree equal to outdegree". Now to prove "G admits a feasible solution to Π" one can use a 0-bit LCP, where even-degree nodes accept and odd-degree nodes reject. However, this LCP does not help us at all if we would like to solve Π with the help of advice. In this sense distributed computation with advice is a harder problem than local proofs.

It is also good to note that one can have LCPs for graph properties that are not of the form "G admits a feasible solution to some LCL Π." For example, planarity is such a property. Such LCPs are (to our knowledge) not directly connected with computation with local advice.

1.3 Contribution 2: LCLs in general graphs

At this point a natural question is whether the assumption about bounded growth is necessary. Could we solve all LCL problems in all graphs with 1 bit of advice? In Section 4 we show that the answer is likely to be no:

Fix any β. If all LCL problems can be solved locally with at most β bits of advice, then the Exponential-Time Hypothesis (ETH) is false.

The intuition here is that if some LCL problem Π can be solved with, say, 1 bit of advice per node with some local algorithm $\mathcal{A}$, then we could solve it with a centralized sequential algorithm as follows: check all 2^n possible assignments of advice, apply $\mathcal{A}$ to decode the advice, and see if the solution is feasible. The total running time (from the centralized sequential perspective) would be $2^n \cdot n \cdot s(n)$, where $s(n)$ is the time we need to simulate $\mathcal{A}$ at one node. Then we need to show that assuming the Exponential-Time Hypothesis, this is too fast for some LCL problem Π. However, the key obstacle is that it may be computationally expensive to simulate $\mathcal{A}$, as it might perform arbitrarily complicated calculations that depend on the numerical values of the unique identifiers, and we cannot directly bound $s(n)$.

Hence, we need to show that $\mathcal{A}$ can be made cheap to simulate. The key ingredient is the following technical result, which we prove using a Ramsey-type argument that is inspired by the proof of Naor and Stockmeyer [50]:

Assume that problem Π can be solved with β bits of advice per node, using some algorithm $\mathcal{A}$. Then the same problem can be also solved with β bits of advice using an *order-invariant* algorithm $\mathcal{A}'$, whose output does not depend on the numerical values of the identifiers but only on their relative order.

The key point here is that (for bounded-degree graphs) $\mathcal{A}'$ can be represented as a finite lookup table; hence the simulation of $\mathcal{A}'$ is cheap, and we can finally make a formal connection to the Exponential-Time Hypothesis. We refer to Section 4 for more details.

1.4 Contribution 3: balanced orientations

Now we move on to a specific graph problem: we study the task of finding balanced and almost-balanced orientations. The goal is to orient the edges so that for each node indegree and outdegree differ by at most 1. This is a hard problem to solve in a distributed setting, while slightly more relaxed versions of the problem admit efficient (but not constant-time) algorithms [29].

Here it is good to note that if we could place our advice on *edges*, then trivially one bit of advice per edge would suffice (simply use the single bit to encode whether the edge is oriented from lower to higher identifier). However, we are here placing advice on *nodes*, and encoding the orientation of each incident edge would require a number of bits proportional to the maximum degree. Surprisingly, we can do it, in any graph:

We can find almost-balanced orientations with 1 bit of advice per node.

Again, we can make the advice arbitrarily sparse.

1.4.1 Application: distributed decompression

Equipped with the advice schema for solving almost-balanced orientations, we can now make a formal connection to what we call *distributed decompression*. Here the task is to encode some graph labeling so that it can be decompressed locally (in $T(\Delta)$ rounds).

Local decompression is closely linked with local computation with advice. If we can compress some solution to Π with only β bits per node, and decompress it locally, then we can also solve Π with β bits of advice per node. Furthermore, if Π is a problem such that for any graph there is only one feasible solution, then the two notions coincide.

We will now show yet another connection between local decompression and local computation with advice. Consider the task of compressing an *arbitrary subset of edges* $X \subseteq E$. In a trivial encoding, we label each node v of degree d with a d-bit string that indicates which of the incident edges are present in X. On the other hand, we need a total of $|E|$ bits in order to distinguish all subsets of the edge-set E. In particular, for d-regular graphs, this means we need at least $d/2$ bits per node to recover an arbitrary subset of edges.

It turns out that once we can solve almost-balanced orientations, we can also compress a subset of edges efficiently. We simply use 1 bit of advice per node to encode an almost-balanced orientation. Now a node of degree d has outdegree $\delta \leq \lceil d/2 \rceil$, and it can simply store a δ-bit vector that indicates which of its outgoing edges are in X. Overall, we will need $\lceil d/2 \rceil + 1$, i.e. $\leq d/2 + 2$, bits per node:

> We can encode an arbitrary set of edges $X \subseteq E$ so that a node of degree d only needs to store $\lceil d/2 \rceil + 1$ bits, and we can decompress X locally, in $T(\Delta)$ rounds.

1.5 Contribution 4: vertex Δ-coloring

Next we study the problem of Δ-coloring graphs of maximum degree Δ:

> In any graph of maximum degree Δ, we can find a Δ-coloring (if it exists) with 1 bit of advice per node.

Again, we can make the advice arbitrarily sparse.

Our schema for encoding Δ-colorings consists of three steps. First, we compute a vertex coloring with $O(\Delta^2)$ colors, with the help of advice. Then we reduce the number of colors down to $\Delta + 1$, using the algorithm by [21, 6, 45]. Finally, we follow the key idea of the algorithm by Panconesi and Srinivasan [52] to turn $(\Delta + 1)$-coloring into a Δ-coloring, and again we will need some advice to make this part efficient.

1.6 Contribution 5: vertex 3-coloring

So far we have seen primarily results of two flavors: many problems can be solved with 1 bit of advice so that we can make the advice arbitrarily sparse, while there are also some problems that require arbitrarily many bits of advice.

We now turn our attention to a problem that seems to lie right at the boundary of what can be done with only 1 bit per node: vertex 3-coloring in any 3-colorable graph. Note that this is a problem that is hard to solve without advice not only in the distributed setting (it is a global problem) but also in the centralized setting (it is an NP-hard problem).

In the centralized setting, 1 bit of advice per node makes the problem easy. To see this, we can simply use the bit to indicate which nodes are of color 3. Then the rest of the graph has to be bipartite, and we can simply find a proper 2-coloring in polynomial time.

In the distributed setting, the trivial solution does not work: 2-coloring in bipartite graphs is still a global problem. Nevertheless, we show that 3-coloring is still doable with 1 bit of advice:

In any 3-colorable graph, we can find a 3-coloring with 1 bit of advice per node.

Our encoding essentially uses one bit to encode one of the color classes, but we adjust the encoding slightly so that throughout the graph there are *local hints* that help us to also choose the right parity for the region that we need to 2-color.

Here, our encoding genuinely needs one bit per node (it just barely suffices); we cannot make our advice arbitrarily sparse.

1.6.1 Application: locally checkable proofs for 3-coloring

Following the same idea as in Section 1.2.1, we can now also certify that a graph is 3-colorable with a proof that only takes 1 bit per node. Furthermore, the verifier only needs to see up to distance $T(\Delta)$, and hence if $\Delta = O(1)$, this is a locally checkable proof in the sense of [34]:

There is a locally checkable proof that certifies 3-colorability with 1 bit per node in graphs of maximum degree $\Delta = O(1)$.

Now it is interesting to compare this with *proof labeling schemes* (PLS). In essence, a PLS is an LCP in which the verifier only sees the identifier of the present node, and the proof labels within radius 1. The above result provides a 1-bit LCP but not a 1-bit PLS.

This connects directly with the work done by Ardévol Martínez et al. [1] and Bousquet et al. [7], which study the same question: how many bits per node are needed to certify k-colorability. Here Bousquet et al. [7] shows that $\Theta(\log k)$ bits per node are necessary for a PLS that certifies k-colorability, and [1] shows that in particular 3-colorability cannot be certified with 1 bit per node with any PLS (while 2 bits per node is trivial). Our work complements the latter result, and provides a separation between PLSs and LCPs in this setting: now we know that while 1 bit per node does not suffice to certify 3-colorability with any PLS, it is sufficient for an LCP (with the caveat that we need to be in a bounded-degree graph).

1.7 Key technique: composability framework

We already discussed some of the proof ingredients above. However, there is one additional technique that we use in many of our algorithms: the framework of *composable schemas*.

It turns out that for many problems, it is easier to work with advice schemas in which only a few nodes carry advice bits, but they may carry many bits of advice. In Definition 2 we give the formal definition of such a schema, and in Definition 4 we give the formal definition of *composable* schemas, which satisfy the additional property that the ratio between the total number of bits held by the nodes, and the total number of nodes, can be made arbitrarily small in every large-enough neighborhood.

While the definition is a bit technical, it has two key properties, which we discuss in more detail in the full version of this work:

1. As the name suggests, composable schemas can be easily composed, in the following sense: once we have (1) a composable schema for solving Π_1 and (2) a composable schema for solving Π_2 assuming an oracle for Π_1, we can also compose them and obtain (3) a composable schema that solves Π_2 without the oracle. This way we can solve problems in a modular fashion, in essence using schemas as "subroutines."

2. A composable schema can be then encoded with only 1 bit of advice per node, and we can make the advice arbitrarily sparse.

For example, our algorithms for finding almost-balanced orientations and Δ-coloring with advice are based on the framework of composable schemas.

1.8 Open questions

Our work suggests a number of open questions:

1. Our negative result in Section 4 assumes the Exponential-Time Hypothesis. Is it possible to prove an unconditional lower bound without such assumptions? Can we exhibit a concrete LCL problem Π that is unconditionally hard?
2. We conjecture that the advice for vertex 3-coloring in 3-colorable graphs cannot be made arbitrarily sparse. Is this true?
3. We also conjecture that for a sufficiently large d, vertex d-coloring in d-colorable graphs cannot be encoded with one bit per node. Is this true? This is closely linked with the discussion on the trade-off conjecture in e.g. [1, 7].
4. Our schema for distributed decompression is asymptotically optimal, but there is room for improving additive constants. Here is a concrete open question: Let $G = (V, E)$ be a 3-regular graph, and let $X \subseteq E$ be an arbitrary set of edges. Is it possible to encode X using only 2 bits per node so that it can be decompressed locally? (Note that 1 bit per node is trivially impossible, while 3 bits per node is trivial. If we delete one edge from each connected component, an encoding with 2 bits per node follows from 2-degeneracy.)

2 Additional related work

Computing with advice is not really a well-defined area of research, as one can consider many kinds of advice and various models of computations. In this paper, we focus on existential advice and the distributed LOCAL model: the advice is provided by an all-knowing oracle, and it is designed to enable algorithms with low locality (i.e., low time complexity). More broadly, one can consider a variant in which the restriction on the size of advice is more relaxed, but it needs to be computable by an oracle with some limitations (realizable in some model of computation), which then can be used as a building block of algorithms.

Advice and locality in distributed computing. Feuilloley et al. [18] consider the framework of Proof Labeling Schemes (PLS). In this framework, there is a *prover* and a *verifier*: the prover is a centralized entity that assigns labels to the nodes, while the verifier is a distributed algorithm that in T rounds is able to verify the validity of the collection of labels. If the predicate we want to investigate holds true, then there must exist an assignment of labels (or certificates) such that each node, in T rounds, "accepts" the given labeling; while if the predicate we want to investigate does not hold, then for any given labeling assignment there must exist a node that rejects it. The authors study the tradeoff between the size of the labels given at each node and the round-complexity of the verifier. In their work, they, too, discover how this tradeoff is remarkably different depending on the growth rate of the graph.

Fraigniaud et al. [20] investigate the problem of distributedly 3-coloring a cycle. Without advice, it is known that this problem requires $\Omega(\log^* n)$ rounds [44], and the paper seeks to understand whether advice can help to break this barrier. In the context studied in [20], each node receives some advice as input, and the total advice is measured as the sum of the lengths of the bit-strings given to all nodes. The authors show that, for any constant k, $O(n/\log^{(k)} n)$ bits of total advice are not sufficient to beat the $\Omega(\log^* n)$ lower bound, where $\log^{(k)} n$ denotes k recursive iterations of $\log n$.

Algorithm design. Parnas and Ron [53] show that one can use distributed algorithms to derive (by simulation) Local Computation Algorithms. Quite naturally, one can use the same approach to derive parallel graph algorithms or graph algorithms for dynamic data sets (or graph algorithms for any model of computation that can leverage the fact that the output is defined by some small neighborhood of a vertex or edge). However, as is, the reduction from [53] did not immediately imply many new state-of-the-art algorithms, as the size of the neighborhood grows exponentially with locality, and, for many problems, the locality of the fastest-known algorithms is fairly large. One of the ways to address this issue is to rely on analysis of algorithms with advice.

Christiansen et al. [12] use distributed algorithms with advice to design faster dynamic algorithms for vertex coloring, by combining dynamic graph orientations with ideas for simulating distributed algorithms on the resulting directed graph.

While this approach in principle can be used in the design of parallel algorithms, so far it is used implicitly or in a somewhat trivial way (e.g. Ghaffari et al. [27] mention that 4-coloring of a graph can be trivially used to compute a maximal independent set).

Advice in other models of computation with partial knowledge. More generally, one can consider the impact of advice that captures some global knowledge about the whole input (or distribution of inputs) on the performance of algorithms. Similarly to the case of distributed computing, one can consider existential advice and advice that can be realized in a somewhat practical model of computation.

Emek et al. [15] introduced the notion of advice in the context of online algorithms. The authors studied the impact of advice in the competitive ratio of some classical online problems, namely *metrical task systems* and *k-server*, and they show tradeoffs between the number of bits of advice and the achievable competitive ratio. We refer to the survey by Boyar et al. [8] for more work on online algorithms with advice.

Mitzenmacher and Vassilvitskii [49] show that algorithms using advice realized by machine learning models can be used to provide algorithms that on average perform better than their traditional counterparts while keeping the same worst-case bounds. However, this particular advice model allows querying an oracle at each step. As such, in the context of online algorithms, it is in some sense both weaker than the existential advice for online algorithms (as it is produced by some machine learning model) and stronger as it gives a lot of bits of advice, and has access to the prefix of the input.

3 Preliminaries

Let $G = (V, E)$ be a graph, where V is the set of nodes and E is the set of edges. We denote with n the number of nodes of the graph, and with Δ its maximum degree. We may use the notation $V(G)$ to denote the set of nodes of G, and $E(G)$ to denote the set of edges of G. With $\mathrm{dist}_G(u, v)$ we denote the distance between $u \in V$ and $v \in V$ in G, that is, the length of the shortest path between u and v in G, where the length of a path is the number of edges of the path. For an integer k, the power graph G^k of a graph G is the graph that contains the same nodes as G, and there is an edge $\{u, v\}$ in G^k if and only if $1 \leq \mathrm{dist}_G(u, v) \leq k$. Two nodes u and v are neighbors in G if there is an edge $\{u, v\} \in E(G)$.

A maximal independent set (MIS) of G is a subset S of nodes of G satisfying that no nodes of S are neighbors in G, and that all nodes in $V \setminus S$ have at least one neighbor that is in S. An (α, β)-ruling set is a subset S of nodes of G satisfying that each pair of nodes

from S have distance at least α, and that all nodes in $V \setminus S$ have at least one node in S at distance at most β. Observe that Maximal Independent Set and $(2,1)$-ruling set is the same problem. By β-ruling set we denote the $(2,\beta)$-ruling set problem.

One important tool that we will use is the Lovász Local Lemma, which states the following.

▶ **Lemma 1** (Lovász Local Lemma [55, 56, 16]). *Let $\{E_1, \ldots, E_k\}$ be a set of events satisfying the following conditions: each event E_i depends on at most d other events; each event E_i happens with probability at most p. Then, if $epd \leq 1$, there is non-zero probability that none of the events occur.*

3.1 LOCAL model

In the LOCAL model of distributed computation, each node of an n-node graph is equipped with a unique ID, typically in $\{1, \ldots, n^c\}$, for some constant c. Initially, each node in the graph knows its own ID, its degree (i.e., the number of neighboring nodes), the maximum degree Δ in the graph, and the total number of nodes. Then, the computation proceeds in synchronous rounds where at each round each node exchanges messages with its neighbors and performs some local computation. The size of the messages can be arbitrarily large and the local computation can be arbitrarily heavy. The runtime of an algorithm is defined as the number of rounds it requires such that all nodes terminate and produce an output.

3.2 Locally checkable labelings

We formally define an LCL problem Π as a tuple $(\Sigma_{\text{in}}, \Sigma_{\text{out}}, C, r)$ where each element of the tuple is defined as follows. The parameters Σ_{in} and Σ_{out} are finite sets of input and output labels, respectively. The parameter r is a positive integer called *checkability radius* of Π, i.e., it determines how far in the graph each node needs to check in order to verify the validity of a given solution. The parameter C determines the constraints of Π, that is, C is a finite set of labeled graphs H containing a vertex of eccentricity at most r, where each edge-endpoint pair $(uv, v) \in E_H \times V_H$ has a label $\ell_{\text{in}} \in \Sigma_{\text{in}}$ and a label $\ell_{\text{out}} \in \Sigma_{\text{out}}$.

Let $\Pi = (\Sigma_{\text{in}}, \Sigma_{\text{out}}, C, r)$ be an LCL problem and let $G = (V, E)$ be a graph where each edge-endpoint pair (uv, v) is labeled with a label from Σ_{in}. The task of solving Π on G requires labeling each edge-endpoint pair $(uv, v) \in E \times V$ with a label in Σ_{out} such that, for each node $v \in V$ it holds that the graph induced by the nodes at distance at most r from v and edges that have at least one endpoint at distance at most $r-1$ from v is isomorphic to some (labeled) graph in C.

3.3 Advice schema

We now formally define the notion of advice schema.

▶ **Definition 2** (Advice Schema). *A $(\mathcal{G}, \Pi, \beta, T)$-advice schema is a function f that receives as input a (possibly input-labeled) graph $G = (V, E, I) \in \mathcal{G}$ (where I is an input for the nodes and/or the edges of the graph), and outputs a function $\ell_G := f(G)$ that satisfies the following.*

- *The function ℓ_G maps each node $v \in V$ into a bit-string of length at most β, where β is a function of Δ.*
- *There exists a LOCAL algorithm $\mathcal{A}$ that, for each $G \in \mathcal{G}$, if we label each node $v \in V$ with the bit-string $\ell_G(v)$, then $\mathcal{A}$ runs in at most T rounds and outputs a valid solution for Π, where T is a function of Δ.*

Moreover, we distinguish three possible types of advice schemas:

1. If all nodes of the graph receive bit-strings of the same length, then the schema is called *uniform fixed-length*.
2. If a subset of nodes receives bit-strings of the same length, and all the others receive bit-strings of length 0, then the schema is called *subset fixed-length*.
3. If a subset of nodes receives a bit-string of possibly different positive lengths, and all the others receive bit-strings of length 0, then the schema is called *variable-length*. Observe that an advice schema, as defined in Definition 2 and without further assumptions, is variable-length.

In the second and third cases, the nodes having non-zero bit-strings assigned are called *bit-holding* nodes. Moreover, observe that Type 1 is a special case of Type 2, which is a special case of Type 3.

For uniform fixed-length advice schemas where all nodes receive one bit, we define the notion of sparsity, which captures the fact that the ratio between nodes receiving a 1 and nodes receiving a 0 can be made arbitrarily small.

▶ **Definition 3** (Sparse schema). *A uniform fixed-length $(\mathcal{G}, \Pi, \beta, T)$-advice schema is called* ε-sparse *if the following holds:*

- *$\beta = 1$, that is, each node receives exactly one bit.*
- *For any graph $G \in \mathcal{G}$, let n_0 be the number of nodes to which the schema (i.e., the function f of Definition 2) assigns a 0, and let n_1 be the number of nodes to which f assigns a 1. Then, $\frac{n_1}{n_0+n_1} \leq \varepsilon$.*

With abuse of notation, we call a uniform fixed-length $(\mathcal{G}, \Pi, \beta, T)$-advice schema (where T is a function that receives as input ε and returns a function of Δ) sparse *if, for any constant $\varepsilon > 0$, there exists an ε-sparse $(\mathcal{G}, \Pi, \beta, T(\varepsilon))$-advice schema.*

3.4 Composability

The main idea that we will use to devise our advice schemas is the following. Given some problem Π, it may be cumbersome to directly define an advice schema for it. Instead, it may be easier to do this operation gradually. In more detail, many problems can be solved as follows: first, find a solution for some subproblem; then, use the solution for the subproblem in order to solve the problem more easily. As a running example, consider the following problem Π.

- The input is a bipartite graph where all nodes have even degree.
- It is required to output a coloring of the edges, say red and blue, such that each node has the same number of red and blue incident edges.

Consider the following three problems.

- Let Π_{v} be the problem of computing a 2-coloring of the nodes, say, black and white.
- Let Π_{e} be the problem of outputting a 2-coloring of the edges, such that each node has the same number of red and blue incident edges, assuming that we are given as input a 2-coloring of the nodes and a balanced orientation of the edges, and assuming that all nodes have even degree.
- Let Π_{o} be the problem of orienting all edges such that each node has the same number of incoming and outgoing edges, assuming that all nodes have even degree.

Consider the following algorithm. First, solve Π_{v} and Π_{o}. Then, solve Π_{e}, by coloring *red* the edges oriented from black to white, and by coloring *blue* the edges oriented from white to black. Observe that this algorithm solves Π.

In other words, we decomposed Π into three different subproblems: two of them are "hard" problems (Π_{v} and Π_{o}), for which some advice is needed if we want to solve them fast, and one (Π_{e}) is trivial once we are given as input the solution for the other two. The idea now is to devise two different advice schemas for Π_{v} and Π_{o} separately.

- For Π_{v} there is a trivial advice schema: use 1 bit to encode the 2-coloring of the nodes.
- For Π_{o} the advice schema is non-trivial, and we will show how it can be done by using 1 bit per node.

While both schemas use only 1 bit per node, we cannot directly combine these two schemas into a single schema that uses 1 bit per node. For this reason, we will devise our schemas in a more high-level way. In particular, for most of the results of this paper, we will not directly provide a uniform fixed-length advice schema that uses one bit per node, but we will start by devising variable-length schemas. Variable-length schemas provide a simpler way to encode information: we can choose a subset of nodes and assign bit-strings to them, without worrying about how these strings will be later encoded by giving one bit to each node. Moreover, the variable-length schemas that we provide satisfy a property called *composability*, which we will formally define later. Intuitively, this property requires the ability to make the set of bit-holding nodes of a variable-length schema to be arbitrarily sparse. For example, for Π_{v}, we do not really need to provide the color of all nodes: we assign 1 bit to a sparse set of nodes (encoding their color), and to all other nodes we do not assign any bit. The nodes that have no bit assigned can still recover a 2-coloring by simple propagation. For Π_{o}, things are more complicated, but a composable variable-length schema can be obtained as well.

Once we have a composable schema for all subproblems that are used to solve our problem of interest, we apply, as a black box, a lemma to obtain a single variable-length schema for our problem. Then, again as a black box, we convert such a schema into a uniform fixed-length schema that uses a single bit per node.

Summarizing, many of the results that we provide about schemas that use a single bit per node are based on the following idea:

- We first decompose a problem of interest into many subproblems;
- We show that, for each of the subproblems, it is possible to devise a variable-length schema that satisfies some desirable properties;
- We prove that such properties imply that we can compose many variable-length schemas into a single one that satisfies the same properties;
- We prove that the resulting variable-length schema implies a uniform fixed-length schema that uses a single bit per node.

Note that the last two steps are done in a problem-independent way. The conditions that allow to combine multiple variable-length schemas are expressed in Definition 4, that, on a high level, states the following. We want our variable-length schema to be tunable as a function of three parameters α, γ, and c. The schema needs to satisfy that in each α-radius neighborhood there are at most γ bit-holding nodes, and that the number of bits held by these nodes is upper bounded by $c\alpha/\gamma^3$. Hence, a composable schema is a collection of schemas, one for each choice of parameters α, γ, and c.

▶ **Definition 4** (Composability). *A $(\mathcal{G}, \Pi, \gamma_0, A, T)$-composable advice schema is a collection $\mathcal{S}$ of advice schemas satisfying the following. For any constant $c > 0$, any $\gamma \geq \gamma_0$, and any $\alpha \geq A(c, \gamma)$, there exists $\beta \leq c\alpha/\gamma^3$ such that:*

- *The collection $\mathcal{S}$ contains a variable-length $(\mathcal{G}, \Pi, \beta, T(\alpha, \Delta))$-advice schema S.*
- *For each $G \in \mathcal{G}$, the assignment given by S to the nodes of G satisfies that, in each α-radius neighborhood of G, there are at most γ_0 bit-holding nodes.*

4 Structural results and lower bounds

Our main goal here is to show that if we were able to solve all LCLs with some constant number of bits of advice per node, it would violate the *Exponential-Time Hypothesis* (ETH) [37]. However, to do that we need to first show a structural result, which is also of independent interest: algorithms that make use of advice can be made *order-invariant*, i.e., they do not need the numerical values of the identifiers.

4.1 Order-invariance

In general, in an advice schema our advice bits may depend on the numerical values of the identifiers, and the algorithm $\mathcal{A}$ that solves the problem may make use of the advice bits and the numerical values of the identifiers. However, here we will show that we can essentially for free eliminate the dependency on the numerical values of the identifiers.

A distributed algorithm is called *order-invariant* [50] if the output remains the same if we change the numerical values of the identifiers but preserve their relative order. Put otherwise, an order-invariant algorithm may make use of the graph structure, local inputs (which in our case includes the advice), and the relative order of the identifiers, but not the numerical values of the identifiers.

Naor and Stockmeyer [50] showed that constant-time distributed algorithms that solve LCL problems can be made order-invariant, and since then order-invariant algorithms have played a key role in many lower-bound results related to constant-time distributed algorithms, see e.g. [32, 33].

We say that graph problem Π is *component-wise-defined* if the following holds: a valid solution for graph G is also a valid solution for a connected component that is isomorphic to G. Put otherwise, we can always solve Π by splitting the graph into connected components and solving Π separately in each component. In essence all graph problems of interest (especially in the distributed setting) are component-wise defined, but one can come up with problems that do not satisfy this property (a trivial example being the task of outputting the number of nodes in the graph).

We say that graph problem Π is a *finite-input* problem if the nodes and edges are either unlabeled, or they are labeled with labels from some finite set Σ_{in}.

Note that Theorem 5 below is applicable to LCL problems (they are component-wise defined, and in any LCL we have a bounded Δ and bounded alphabets $\Sigma_{\text{in}}, \Sigma_{\text{out}}$), but also to many other problems that are not locally checkable. We emphasize that we will assume some bound on the maximum degree Δ, but if we have an encoding schema that works for any Δ, we can apply the following result separately for each Δ.

▶ **Theorem 5.** *Fix a maximum degree Δ and the number of advice bits β. Assume that Π is a finite-input component-wise defined graph problem, in which the task is to label nodes with labels from some finite set Σ_{out}. Assume that we can solve some graph problem Π with some distributed algorithm $\mathcal{A}$ in T rounds using β bits of advice. Then we can also solve Π with an order-invariant algorithm $\mathcal{A}'$ in T rounds using β bits of advice.*

Proof. Let E be the *encoder* that, given a graph G (with some unique identifiers), produces β-bit advice strings that $\mathcal{A}$ can then use to solve Π.

In the first steps, we follow the basic idea of Naor and Stockmeyer [50] to manipulate $\mathcal{A}$. Algorithm $\mathcal{A}$ is a mapping from labeled radius-T neighborhoods to local outputs; we write here $N^T(v)$ for the radius-T neighborhood of node v, and we let $s = \Delta^{T+1}$ be an upper bound on the number of nodes in $N^T(v)$.

In general, we can encode all information in $N^T(v)$ as follows:

1. The *set* $X \subseteq \{1, 2, \dots\}$ *of unique identifiers* present in the neighborhood, with $|X| \leq s$.
2. The *structure* S of the neighborhood, which includes the graph topology, the relative order of the identifiers, local inputs, and the advice bits.

So we can reinterpret $\mathcal{A}$ as a function that maps a pair (X, S) to the local output $\mathcal{A}(X, S) \in \Sigma_{\text{out}}$. But we can equally well interpret $\mathcal{A}$ as a function A that maps X to a *type* $A(X) = F$, where F is a function that maps S to the local output $F(S) \in \Sigma_{\text{out}}$; we simply let $A(X)(S) = \mathcal{A}(X, S)$. The second interpretation turns out to be convenient.

Notice that we can always pad X with additional identifiers (that use values larger than any value in X), so that we will have $|X| = s$.

Then we make the key observation: *there are only finitely many different structures* S. This follows from the assumptions that Δ is a constant, β is a constant, Π is a finite-input problem, and T is a constant. Furthermore, Σ_{out} is finite, so F is a mapping from a finite set to a finite set, and follows that *there are only finitely many different types* F. Let m be the number of possible types, and identify the types with $1, 2, \dots, m$.

With this interpretation, A defines a *coloring* (in the Ramsey-theoretic sense) of all s-subsets of natural numbers with m colors, that is, it assigns to each subset $X \subseteq \{1, 2, \dots\}$ with $|X| = s$ some *color* $A(X) \in \{1, 2, \dots, m\}$. By applying the infinite multigraph version of Ramsey's theorem, it follows that there exists an infinite set of natural numbers I and a single *canonical type* F^* such that for any $X \subseteq I$ we have $A(X) = F^*$. Set I is called a *monochromatic set*.

Now let us stop for a moment and digest what we have learned so far: if our unique identifiers came from the set I, then we will have $\mathcal{A}(X, S) = A(X)(S) = F^*(S)$. That is, $\mathcal{A}$ will then ignore the numerical values of the identifiers and only pay attention to the structure S. However, this is a big if; in general our identifiers can be arbitrary, and even adversarial.

Let us now continue; we will now modify the encoder E. We construct a new encoder E' that works as follows. Assume we are given a graph G, together with some assignment of unique identifiers. The encoder E' first constructs graph G_1 by renumbering the identifiers (but preserving their order) so that all identifiers in G_1 come from the monochromatic set I.

Now we would like to apply encoder E to G_1, but we cannot. Our encoder may assume that the identifiers in an n-node graph come from the set $\{1, 2, \dots, \text{poly}(n)\}$, while now we have made our identifiers astronomically large. But to fix this we exploit the fact that Π is component-wise solvable. We simply construct a new graph H that consists of sufficiently many copies of G, such that the first copy is G_1, with unique identifiers coming from I, while in the other copies we assign identifiers from $\{1, 2, \dots\} \setminus I$. This way we can arrange things so that H has N nodes, for some (very large) number N, and the identifiers are assigned from $\{1, 2, \dots, N\}$.

Now H is a valid instance, and we can feed it to the encoder E, which will label it with β-bit labels so that if we apply $\mathcal{A}$ to H and these labels, we will correctly solve Π in each component of H. In particular, we will correctly solve Π in G_1. Moreover, in component G_1, algorithm $\mathcal{A}$ will apply order-invariant algorithm F^* in all neighborhoods, ignoring the numerical values of the identifiers.

However, we needed an encoding for our *original* graph G, with the *original* set of identifiers. To do that, we proceed as follows. We make the above thought experiment, to construct the advice for H. Then we simply copy the advice bits from component G_1 to the original graph G.

If we now applied $\mathcal{A}$ to solve Π in G, it does not necessarily work. However, if we apply $\mathcal{A}'(X, S) = F^*(S)$ to solve Π in G, it will behave in exactly the same way as applying $\mathcal{A}$ in G_1. Hence, $\mathcal{A}'$ will also solve Π correctly in G. Furthermore, $\mathcal{A}'$ is by construction order-invariant. ◀

We note that the encoder E' constructed above is not practical or efficient (even if the original encoder E is). As we will see next, merely knowing that $\mathcal{A}'$ exists can be sufficient.

4.2 Hardness assuming the Exponential-Time Hypothesis

In the full version of this work, we show that in graphs with sub-exponential growth, any LCL can be solved with one bit of advice per node. We now show that one bit does not suffice in general, assuming the Exponential-Time Hypothesis [37].

Recall that the Exponential-Time Hypothesis states that there is a positive constant $\delta > 0$ such that 3-SAT cannot be solved in time $O(2^{\delta n})$, where n is the number of variables in the 3-SAT instance.

▶ **Theorem 6.** *Fix any β. Assume that for every LCL problem Π there is some T such that, on any input, Π can be solved in T rounds with β bits of advice. Then the Exponential-Time Hypothesis is false.*

Proof. Suppose there is some β such that all LCL problems can be solved with β bits of advice. We show how to then solve 3-SAT in time $O(2^{\varepsilon n} \cdot n \cdot f(\varepsilon, \beta))$ for an arbitrarily small $\varepsilon > 0$, violating the Exponential-Time Hypothesis.

Consider some 3-SAT instance ϕ, with n variables and m clauses. By applying the sparsification lemma by [38], we can assume w.l.o.g. that $m = O(n)$.

Now turn ϕ into a bipartite graph, with nodes representing variables on one side and nodes representing clauses on the other side. Each clause is connected by edges with the three variables it contains. Each node is labeled by its type (variable or clause) and each edge vc is labeled to indicate whether the variable v is negated in the clause c. This results in a bipartite graph with $n + m = O(n)$ nodes, and each clause-node has degree at most 3. Hence, the total number of edges is at most $3m$.

Variable-nodes may have arbitrarily high degrees, but the sum of their degrees is bounded by $3m$. We replace each variable-node that has degree $d > 3$ by a cycle of d variable-nodes, with the new edges labeled by equality constraints. This results in a graph in which all nodes have degree at most 3, and the total number of nodes is bounded by $3m + m = O(n)$. Let G_0 be the resulting graph, let $n_0 = O(n)$ be the number of nodes in G_0, let $\Delta_0 = 3$ be the maximum degree, and let $\ell_0 = O(1)$ be the number of node and edge labels that we used to encode the instance.

Now it is easy to define an LCL problem Π_0 such that a solution of Π_0 in graph G_0 can be interpreted as a satisfying assignment of the variables in formula ϕ, and vice versa.

Now assume that we have defined an LCL problem Π_i and a graph G_i with n_i nodes, maximum degree Δ_i, and ℓ_i labels; the base case $i = 0$ was presented above. Define a new LCL problem Π_{i+1} and a new graph G_{i+1} as follows. We contract edges in G_i to construct G_{i+1} so that we satisfy two properties: each node in G_{i+1} represents $O(1)$ nodes of G_i, but the number of nodes is $n_{i+1} \leq n_i/2$. This can be achieved by e.g. greedily contracting edges, favoring the edges whose endpoints currently represent the smallest number of nodes of G_i.

We label the nodes of G_{i+1} so that given the input labels of the nodes, we can also recover the original graph G_i. As each node represents a bounded number of original nodes, and the original graph had maximum degree Δ_i, the new graph will have maximum degree Δ_{i+1} that only depends on i. Also, the number of labels ℓ_{i+1} will be bounded by a constant that only depends on i. To see that everything can be indeed encoded with constant-size labels, note that each new node v in G_{i+1} can have its own *local numbering* of the original nodes v' that were contracted to v, that is, each node v' can be represented as a pair (v, a), where v is a new node and a is a sequence number. This way, the new label of node v can use

constant-size triples of the form (a, b, x) to encode that graph G_i had an edge from (v, a) to (v, b) with label x, and the new edge label of edge (u, v) can also use constant-size triples of the form (a, b, x) to encode that graph G_i had an edge from (u, a) to (v, b) with label x. Finally, the new node label of v will use constant-size pairs of the form (a, x) to encode that the node label of (v, a) in G_i was x. This way we can use constant-size node and edge labels in G_{i+1} to encode the structure and node and edge labels of G_i.

Now we can define an LCL problem Π_{i+1} such that a valid solution of Π_{i+1} in G_{i+1} can be mapped to a valid solution of Π_i in G_i, and hence eventually to a satisfying assignment of ϕ, and conversely a satisfying assignment of ϕ can be turned into a valid solution of Π_{i+1}. In essence, the output label of a node in G_{i+1} captures the output labels of all $O(1)$ nodes of G_i that it represents.

Continuing this way for $\Theta(\log(\beta/\varepsilon))$ steps, we can construct a graph G_i with fewer than $\varepsilon n/\beta$ nodes. Furthermore, there is some LCL problem Π_i such that ϕ is a yes-instance if and only if there is a valid solution of Π_i in G_i.

By assumption, there exists a distributed algorithm $\mathcal{A}$ that solves Π_i with β bits of advice per node. Using Theorem 5, we can also assume that $\mathcal{A}$ is order-invariant. Hence, $\mathcal{A}$ is a finite function, and we can compute $\mathcal{A}$ in constant time (where the constant depends on i, which depends on β and ε, but is independent of n).

Now, we simply try out all possible strings of advice; there are at most $2^{\varepsilon n}$ such strings. For each advice combination, we try to apply $\mathcal{A}$ to solve Π_i in G_i; we simulate $\mathcal{A}$ at each of the $n_i = O(n)$ nodes. If and only if ϕ is satisfiable, we will find an advice string such that $\mathcal{A}$ succeeds in solving Π_i. The overall running time is $O(2^{\varepsilon n} \cdot n \cdot f(\varepsilon, \beta))$. ◀

References

1. Virginia Ardévol Martínez, Marco Caoduro, Laurent Feuilloley, Jonathan Narboni, Pegah Pournajafi, and Jean-Florent Raymond. A lower bound for constant-size local certification. *Theoretical Computer Science*, 971:114068, 2023. doi:10.1016/J.TCS.2023.114068.
2. Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. How much does randomness help with locally checkable problems? In Yuval Emek and Christian Cachin, editors, *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, pages 299–308. ACM, 2020. doi:10.1145/3382734.3405715.
3. Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. Almost global problems in the LOCAL model. *Distributed Computing*, 34(4):259–281, 2021. doi:10.1007/S00446-020-00375-2.
4. Alkida Balliu, Keren Censor-Hillel, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Locally checkable labelings with small messages. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, volume 209 of *LIPIcs*, pages 8:1–8:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.DISC.2021.8.
5. Alkida Balliu, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. New classes of distributed time complexity. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1307–1318. ACM, 2018. doi:10.1145/3188745.3188860.
6. Leonid Barenboim, Michael Elkin, and Uri Goldenberg. Locally-iterative distributed $(\Delta + 1)$-coloring and applications. *Journal of the ACM*, 69(1):5:1–5:26, 2022. doi:10.1145/3486625.
7. Nicolas Bousquet, Laurent Feuilloley, and Sébastien Zeitoun. Local certification of local properties: Tight bounds, trade-offs and new parameters. In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024, March 12-14, 2024, Clermont-Ferrand,*

France, volume 289 of *LIPIcs*, pages 21:1–21:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.STACS.2024.21.

8 Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online algorithms with advice: A survey. *ACM Computing Surveys*, 50(2):19:1–19:34, 2017. doi:10.1145/3056461.

9 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 Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 479–488. ACM, 2016. doi:10.1145/2897518.2897570.

10 Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the LOCAL model. *SIAM Journal on Computing*, 48(1):122–143, 2019. doi:10.1137/17M1117537.

11 Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the LOCAL model. *SIAM Journal on Computing*, 48(1):33–69, 2019. doi:10.1137/17M1157957.

12 Aleksander Bjørn Grodt Christiansen, Krzysztof Nowicki, and Eva Rotenberg. Improved dynamic colouring of sparse graphs. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1201–1214. ACM, 2023. doi:10.1145/3564246.3585111.

13 Dariusz Dereniowski and Andrzej Pelc. Drawing maps with advice. *Journal of Parallel and Distributed Computing*, 72(2):132–143, 2012. doi:10.1016/J.JPDC.2011.10.004.

14 Stefan Dobrev, Rastislav Královic, and Euripides Markou. Online graph exploration with advice. In Guy Even and Magnús M. Halldórsson, editors, *Structural Information and Communication Complexity - 19th International Colloquium, SIROCCO 2012, Reykjavik, Iceland, June 30-July 2, 2012, Revised Selected Papers*, volume 7355 of *Lecture Notes in Computer Science*, pages 267–278. Springer, 2012. doi:10.1007/978-3-642-31104-8_23.

15 Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 427–438. Springer, 2009. doi:10.1007/978-3-642-02927-1_36.

16 P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vols. I, II, III*, volume Vol. 10 of *Colloq. Math. Soc. János Bolyai*, pages 609–627. North-Holland, Amsterdam-London, 1975.

17 Laurent Feuilloley and Pierre Fraigniaud. Survey of distributed decision, 2017. doi:10.48550/arXiv.1606.04434.

18 Laurent Feuilloley, Pierre Fraigniaud, Juho Hirvonen, Ami Paz, and Mor Perry. Redundancy in distributed proofs. *Distributed Computing*, 34(2):113–132, 2021. doi:10.1007/S00446-020-00386-Z.

19 Manuela Fischer and Mohsen Ghaffari. Sublogarithmic distributed algorithms for Lovász local lemma, and the complexity hierarchy. In Andréa W. Richa, editor, *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.DISC.2017.18.

20 Pierre Fraigniaud, Cyril Gavoille, David Ilcinkas, and Andrzej Pelc. Distributed computing with advice: information sensitivity of graph coloring. *Distributed Computing*, 21(6):395–403, 2009. doi:10.1007/S00446-008-0076-Y.

21 Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local conflict coloring. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016,*

9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA, pages 625–634. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.73.

22 Pierre Fraigniaud, David Ilcinkas, and Andrzej Pelc. Tree exploration with advice. *Information and Computation*, 206(11):1276–1287, 2008. doi:10.1016/J.IC.2008.07.005.

23 Pierre Fraigniaud, David Ilcinkas, and Andrzej Pelc. Communication algorithms with advice. *Journal of Computer and System Sciences*, 76(3-4):222–232, 2010. doi:10.1016/J.JCSS.2009.07.002.

24 Pierre Fraigniaud, Amos Korman, and Emmanuelle Lebhar. Local MST computation with short advice. In Phillip B. Gibbons and Christian Scheideler, editors, *SPAA 2007: Proceedings of the 19th Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Diego, California, USA, June 9-11, 2007*, pages 154–160. ACM, 2007. doi:10.1145/1248377.1248402.

25 Emanuele G. Fusco and Andrzej Pelc. Trade-offs between the size of advice and broadcasting time in trees. *Algorithmica*, 60(4):719–734, 2011. doi:10.1007/S00453-009-9361-9.

26 Emanuele G. Fusco, Andrzej Pelc, and Rossella Petreschi. Topology recognition with advice. *Information and Computation*, 247:254–265, 2016. doi:10.1016/J.IC.2016.01.005.

27 Mohsen Ghaffari, Christoph Grunau, and Ce Jin. Improved MPC algorithms for mis, matching, and coloring on trees and beyond. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 34:1–34:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.DISC.2020.34.

28 Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 662–673. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00069.

29 Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, Yannic Maus, Jukka Suomela, and Jara Uitto. Improved distributed degree splitting and edge coloring. *Distributed Computing*, 33(3-4):293–310, 2020. doi:10.1007/S00446-018-00346-8.

30 Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2505–2523. SIAM, 2017. doi:10.1137/1.9781611974782.166.

31 Christian Glacet, Avery Miller, and Andrzej Pelc. Time vs. information tradeoffs for leader election in anonymous trees. *ACM Transactions on Algorithms*, 13(3):31:1–31:41, 2017. doi:10.1145/3039870.

32 Mika Göös, Juho Hirvonen, and Jukka Suomela. Lower bounds for local approximation. *Journal of the ACM*, 60(5):39:1–39:23, 2013. doi:10.1145/2528405.

33 Mika Göös, Juho Hirvonen, and Jukka Suomela. Linear-in-Δ lower bounds in the LOCAL model. *Distributed Computing*, 30(5):325–338, 2017. doi:10.1007/S00446-015-0245-8.

34 Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory of Computing*, 12(1):1–33, 2016. doi:10.4086/TOC.2016.V012A019.

35 Barun Gorain and Andrzej Pelc. Deterministic graph exploration with advice. *ACM Transactions on Algorithms*, 15(1):8:1–8:17, 2019. doi:10.1145/3280823.

36 David Ilcinkas, Dariusz R. Kowalski, and Andrzej Pelc. Fast radio broadcasting with advice. *Theoretical Computer Science*, 411(14-15):1544–1557, 2010. doi:10.1016/J.TCS.2010.01.004.

37 Russell Impagliazzo and Ramamohan Paturi. Complexity of k-sat. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999*, pages 237–240. IEEE Computer Society, 1999. doi:10.1109/CCC.1999.766282.

38 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/JCSS.2001.1774.

39 Dennis Komm, Rastislav Královic, Richard Královic, and Jasmin Smula. Treasure hunt with advice. In Christian Scheideler, editor, *Structural Information and Communication*

Complexity - 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015, Post-Proceedings, volume 9439 of *Lecture Notes in Computer Science*, pages 328–341. Springer, 2015. doi:10.1007/978-3-319-25258-2_23.

40 Amos Korman and Shay Kutten. On distributed verification. In Soma Chaudhuri, Samir R. Das, Himadri S. Paul, and Srikanta Tirthapura, editors, *Distributed Computing and Networking, 8th International Conference, ICDCN 2006, Guwahati, India, December 27-30, 2006*, volume 4308 of *Lecture Notes in Computer Science*, pages 100–114. Springer, 2006. doi:10.1007/11947950_12.

41 Amos Korman and Shay Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20(4):253–266, 2007. doi:10.1007/S00446-007-0025-1.

42 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010. doi:10.1007/S00446-010-0095-3.

43 Amos Korman, David Peleg, and Yoav Rodeh. Constructing labeling schemes through universal matrices. *Algorithmica*, 57(4):641–652, 2010. doi:10.1007/S00453-008-9226-7.

44 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.

45 Yannic Maus and Tigran Tonoyan. Linial for lists. *Distributed Computing*, 35(6):533–546, 2022. doi:10.1007/S00446-022-00424-Y.

46 Avery Miller and Andrzej Pelc. Fast rendezvous with advice. *Theoretical Computer Science*, 608:190–198, 2015. doi:10.1016/J.TCS.2015.09.025.

47 Avery Miller and Andrzej Pelc. Tradeoffs between cost and information for rendezvous and treasure hunt. *Journal of Parallel and Distributed Computing*, 83:159–167, 2015. doi:10.1016/J.JPDC.2015.06.004.

48 Avery Miller and Andrzej Pelc. Election vs. selection: How much advice is needed to find the largest node in a graph? In Christian Scheideler and Seth Gilbert, editors, *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pages 377–386. ACM, 2016. doi:10.1145/2935764.2935772.

49 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, pages 646–662. Cambridge University Press, 2020. doi:10.1017/9781108637435.037.

50 Moni Naor and Larry J. Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

51 Nicolas Nisse and David Soguet. Graph searching with advice. *Theoretical Computer Science*, 410(14):1307–1318, 2009. doi:10.1016/J.TCS.2008.08.020.

52 Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 581–592. ACM, 1992. doi:10.1145/129712.129769.

53 Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1-3):183–196, 2007. doi:10.1016/J.TCS.2007.04.040.

54 Václav Rozhon and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 350–363. ACM, 2020. doi:10.1145/3357713.3384298.

55 James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985. doi:10.1007/BF02579368.

56 Joel Spencer. Asymptotic lower bounds for ramsey functions. *Discrete Mathematics*, 20:69–76, 1977. doi:10.1016/0012-365X(77)90044-9.