Online Locality Meets Distributed Quantum Computing

Amirreza Akbari, Xavier Coiteux-Roy, Francesco d'Amore, François Le Gall, Henrik Lievonen, Darya Melnyk, Augusto Modanese, Shreyas Pai, Marc-Olivier Renou, Václav Rozhoň, et al.

Online Locality Meets Distributed Quantum Computing

AMIRREZA AKBARI, Aalto University, Finland
XAVIER COITEUX-ROY, Technical University of Munich, Germany and Munich Center for Quantum Science and Technology, Germany
FRANCESCO D'AMORE, Aalto University, Finland and Bocconi University, Italy
FRANÇOIS LE GALL, Nagoya University, Japan
HENRIK LIEVONEN, Aalto University, Finland
DARYA MELNYK, Technische Universität Berlin, Germany
AUGUSTO MODANESE, Aalto University, Finland
SHREYAS PAI, Aalto University, Finland
MARC-OLIVIER RENOU, Inria, France, Université Paris-Saclay, France, and École Polytechnique, Institut Polytechnique de Paris, France
VÁCLAV ROZHOŇ, ETH Zurich, Switzerland
JUKKA SUOMELA, Aalto University, Finland

We extend the theory of locally checkable labeling problems (LCLs) from the classical LOCAL model to a number of other models that have been studied recently, including the quantum-LOCAL model, finitely-dependent processes, non-signaling model, dynamic-LOCAL model, and online-LOCAL model [e.g. STOC 2024, ICALP 2023].

First, we demonstrate the advantage that finitely-dependent processes have over the classical LOCAL model. We show that all LCL problems solvable with locality $O(\log^* n)$ in the LOCAL model admit a finitely-dependent distribution (with constant locality). In particular, this gives a finitely-dependent coloring for regular trees, answering an open question by Holroyd [2023]. This also introduces a new formal barrier for understanding the distributed quantum advantage: it is not possible to exclude quantum advantage for any LCL in the $\Theta(\log^* n)$ complexity class by using non-signaling arguments.

Second, we put limits on the capabilities of all of these models. To this end, we introduce a model called randomized online-LOCAL, which is strong enough to simulate e.g. SLOCAL and dynamic-LOCAL, and we show that it is also strong enough to simulate any non-signaling distribution and hence any quantum-LOCAL algorithm. We prove the following result for trees: if we can solve an LCL problem with locality $o(\log^{(5)} n)$ in the randomized online-LOCAL model, we can solve it with locality $O(\log^* n)$ in the classical deterministic LOCAL model.

Put together, these results show that in trees the set of LCLs that can be solved with locality $O(\log^* n)$ is the same across all these models: locality $O(\log^* n)$ in quantum-LOCAL, non-signaling model, dynamic-LOCAL, or online-LOCAL is not stronger than locality $O(\log^* n)$ in the classical deterministic LOCAL model.

Authors' addresses: Amirreza Akbari, Aalto University, Espoo, Finland, amirreza.akbari@aalto.fi; Xavier Coiteux-Roy, School of Computation, Information and Technology, Technical University of Munich, Munich, Germany and Munich Center for Quantum Science and Technology, Munich, Germany, xavier.coiteux-roy@tum.de; Francesco d'Amore, Aalto University, Espoo, Finland and Bocconi University, Milan, Italy, francesco.damore2@unibocconi.it; François Le Gall, Nagoya University, Nagoya, Japan, legall@math.nagoya-u.ac.jp; Henrik Lievonen, Aalto University, Espoo, Finland, henrik.lievonen@aalto.fi; Darya Melnyk, Technische Universität Berlin, Berlin, Germany, melnyk@tu-berlin.de; Augusto Modanese, Aalto University, Espoo, Finland, augusto.modanese@aalto.fi; Shreyas Pai, Aalto University, Espoo, Finland, shreyas.pai@aalto.fi; Marc-Olivier Renou, Inria, Paris, France and Université Paris-Saclay, Paris, France, CPHT and École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France, marc-olivier.renou@inria.fr; Václav Rozhoň, ETH Zurich, Zurich, Switzerland, vaclavrozhon@gmail.com; Jukka Suomela, Aalto University, Espoo, Finland, jukka.suomela@aalto.fi.

1 INTRODUCTION

In this work, we connect three distinct lines of research that have recently explored extensions of the classical LOCAL model of distributed computing:

(1) Distributed quantum computing and non-signaling distributions [3, 24, 33].
(2) Finitely-dependent processes [40–42].
(3) Locality in online graph algorithms and dynamic graph algorithms [2, 20].

We prove new results on the capabilities and limitations of all of these models of computing, for *locally checkable labeling problems* (LCLs). Our work implies limitations on the *quantum advantage* in the distributed setting, and we also exhibit a new *barrier* for proving tighter bounds.

1.1 Classical models

Let us first recall the definitions of the classical models of distributed graph algorithms [50, 57] that form the foundation for our work; we will give brief and informal definitions here and postpone the formal definitions to Section 3:

- **Deterministic LOCAL:** Our input graph $G = (V, E)$ represents a computer network; each node $v \in V$ is a computer and each edge $\{u, v\} \in E$ is a communication link between two computers. Each node is labeled with a unique identifier from $\{1, 2, \ldots, \mathrm{poly}(|V|)\}$. All nodes follow the same distributed algorithm. Initially each node is only aware of its own identifier and its own degree. Computation proceeds in synchronous rounds, and in each round each node can send a message to each neighbor, receive a message from each neighbor, and update its own state. Eventually each node has to stop and announce its local output (its own part of the solution, e.g. in graph coloring its own color). The *running time*, *round complexity*, or *locality* of the algorithm is the (worst-case) number of rounds $T(n)$ until the algorithm stops in any n-node graph.
- **Randomized LOCAL:** As above, but each node has also access to its own private source of random bits.

It will be also useful to explicitly distinguish the following variants (see e.g. [46] for discussion on the knowledge of global information):

- **Deterministic LOCAL (shared):** Deterministic LOCAL with shared global information. The set of nodes and their unique identifiers is globally known. In particular, we know the value of $n = |V|$.
- **Randomized LOCAL (shared):** Randomized LOCAL with shared global information and shared randomness. The set of nodes and their unique identifiers is globally known, and in addition to the private sources of random bits, there is also a shared source of random bits that all nodes can access.

It may be helpful to interpret the shared versions of the models so that we get to *see* the set of nodes V and their unique identifiers in advance, and we can also *initialize* the nodes as we want based on this information (and hence in the randomized model, we can also initialize all nodes with the same shared random string), but the set of edges E is only revealed later. This interpretation will come useful when we switch to the quantum models.

1.2 Landscape of LCL problems

There has been more than three decades of work on understanding the capabilities and limitations of the classical deterministic and randomized LOCAL models, but for our purposes the most interesting is the recent line of work that has studied distributed algorithms for *locally checkable labeling problems*, or LCLs. This is a family of problems first introduced by Naor and Stockmeyer [53]. LCL

problems are graph problems that can be defined by specifying a *finite set of valid neighborhoods*—for example, the task of coloring graphs of maximum degree 5 with 7 colors is an example of an LCL problem.

Since 2016, we have seen a large body of work dedicated to understanding the computational complexity of LCL problems in the deterministic and randomized LOCAL models [4–8, 8–10, 15, 16, 18, 21, 32, 34, 35, 37, 58], and there are nowadays even algorithms and computer tools available for exploring such questions [5, 22, 55]. As a result of this large international research effort, a landscape of the distributed localities of LCL problems emerges [60]. We can now classify LCL problems in discrete classes based on their locality, and we also understand very well how much randomness can help in comparison with deterministic algorithms.

Our main goal in this work is to extend this understanding of LCL problems far beyond the classical models, and especially explore what can be computed very fast in models that are much stronger than deterministic or randomized LOCAL.

1.3 Quantum-LOCAL and finitely-dependent processes

Let us start our exploration of stronger models with distributed quantum computation. The key question is understanding the *distributed quantum advantage*: what can we solve faster if our nodes are quantum computers and our edges are quantum communication channels? There is a long line of prior work that has explored variants of this theme in different models of distributed computing [3, 17, 24, 31, 33, 44, 45, 47, 48, 51, 61–63], but for our purposes these are the models of interest:

- **Quantum-LOCAL:** Our model of computing is similar to the deterministic LOCAL model above, but now with quantum computers and quantum communication links. More precisely, the quantum computers manipulate local states consisting of an unbounded number of qubits with arbitrary unitary transformations, the communication links are quantum communication channels (adjacent nodes can exchange any number of qubits), and the local output can be the result of any quantum measurement.
- **Quantum-LOCAL (shared):** Quantum-LOCAL with shared global information and a shared quantum state. As above, but now the algorithm gets to inspect and manipulate the set of nodes (before seeing the set of edges). In particular, the algorithm can initialize the quantum computers with a globally shared entangled state.

Note that as quantum theory intrinsically involves randomness, quantum-LOCAL is at least as strong as randomized LOCAL.

It is known that there are some (artificial) problems that are known to be solvable much faster in quantum-LOCAL than deterministic or randomized LOCAL [48]; however, whether any LCL admits such a quantum advantage is a major open question in the field, and also one of the main motivations of our work.

1.3.1 Finitely-dependent processes and non-signaling model. Directly analyzing quantum-LOCAL is beyond the scope of current techniques. In essence, the only known technique for proving limitations of quantum-LOCAL is *sandwiching* it between the classical randomized-LOCAL model and more powerful models than quantum-LOCAL that do not explicitly refer to quantum information. These more powerful models are based on the *physical causality principle* (a.k.a. *non-signaling principle*). The idea is perhaps easiest to understand with the help of the following thought experiment:

Example 1.1. Fix a distributed algorithm $\mathcal{A}$ in the quantum-LOCAL model with shared global information and quantum state that runs in T rounds on graphs with n nodes. Let $G = (V, E)$ be some n-node input graph. Apply $\mathcal{A}$ repeatedly to G to obtain some probability distribution $Y(G)$ of outputs. Now fix some subset of nodes $U \subseteq V$, and consider the restriction of $Y(G)$ to U, in

notation $Y(G)\restriction_U$. Let $G[U,T]$ be the radius-T neighborhood of set U in G. Now modify G outside $G[U,T]$ to obtain a different n-node graph G' with $G[U,T] = G'[U,T]$. Apply $\mathcal{A}$ to G' repeatedly, and we obtain another probability distribution $Y(G')$ of outputs. If $Y(G)\restriction_U \neq Y(G')\restriction_U$, it would be possible to use $\mathcal{A}$ to transmit information in T time steps between two parties, Alice and Bob, that are within distance $T+1$ from each other: Bob's laboratory would hold all nodes of U, and he could, therefore, observe $Y(G)\restriction_U$, while Alice's laboratory would control the graph outside $G[U,T] = G'[U,T]$, and she could, therefore, instantiate either G or G'. This would enable Alice to send a signal to Bob even if no physical communication occurred from Alice's lab to Bob's lab (as they are at distance $T+1$ from each other and only T communication steps occurred), and thus violate the non-signaling principle.

This thought experiment suggests the following definition, also known as the ϕ-LOCAL model and the causal model [3, 33]:

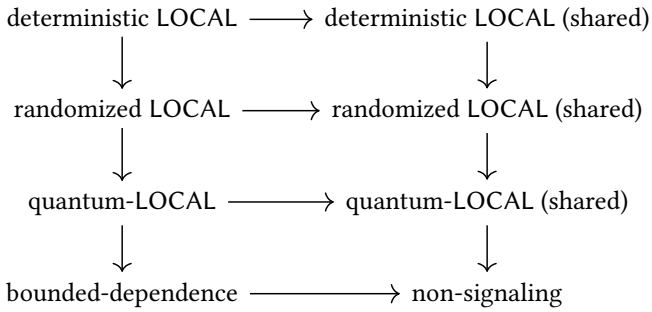
- **Non-signaling model:** We can produce an arbitrary output distribution as long as it does not violate the non-signaling principle: if we fix any set of nodes U, modifying the structure of the input graph at more than a distance $T(n)$ from U does not affect the output distribution of U.

We also need to introduce the following definition to better connect our work with the study of finitely-dependent processes and in particular finitely-dependent colorings [40–42]:

- **Bounded-dependence model:** We can produce an arbitrary output distribution as long as it does not violate the non-signaling principle, and, furthermore, distant parts are independent: if we fix any sets of nodes U_1 and U_2 such that their radius-$T(n)$ neighborhoods are disjoint, then the output labels of U_1 are independent of the output labels of U_2.

Now if we set $T(n) = O(1)$, algorithms in the bounded-dependence model are in essence what is usually called *finitely-dependent processes*.

1.3.2 Model hierarchy. Now we can connect all the above models with each others as follows, sandwiching the two versions of the quantum-LOCAL model between other models (see Appendix A for detailed arguments; the connection with the non-signaling model is known [3, 33] but the connection with the bounded-dependence model is to our knowledge new):



(1)

Here an arrow $M_1 \to M_2$ indicates that the existence of an algorithm with locality (or round complexity) $T(n)$ in model M_1 implies an algorithm with the same locality in M_2. While it is not surprising that we can indeed sandwich the quantum models between models that do not explicitly refer to quantum information, it is remarkable that at least for some problems we can prove near-tight lower bounds by using diagram (1). For example, a very recent work [24] used these connections to prove limits for the distributed quantum advantage in approximate graph

coloring: they proved an upper bound for the deterministic LOCAL model and a near-matching lower bound for the non-signaling model.

1.4 Contribution 1: symmetry breaking with finitely-dependent processes

Now we are ready to state our first contribution. Recall the following gap result by Chang et al. [19]: all LCL problems that can be solved with locality $o(\log n)$ in deterministic LOCAL or with locality $o(\log \log n)$ in randomized LOCAL can also be solved with locality $O(\log^* n)$ in deterministic LOCAL. The class of problems with locality $\Theta(\log^* n)$ contains in essence all *symmetry-breaking problems*: these are problems that could be solved with constant locality if only we had some means of breaking symmetry (e.g. distance-k coloring for some constant k would suffice).

In Section 5 we show the following result:

> Theorem 1.2. *Let Π be any LCL problem with locality $O(\log^* n)$ in the deterministic* LOCAL *model. Then Π can be also solved with locality $O(1)$ in the bounded-dependence model. Furthermore, the resulting finitely-dependent processes are invariant under subgraph isomorphism.*

Put otherwise, there is a finitely-dependent distribution over valid solutions of Π. Here the invariance under subgraph isomorphisms implies that, for any two graphs G, H that share some isomorphic subgraphs G' and H' such that their radius-$O(1)$ neighborhoods are still isomorphic, the finitely-dependent processes solving Π restricted to G' and H' are equal.

For any constant d, the task of coloring d-regular trees with $d + 1$ colors is an example of a problem with locality $O(\log^* n)$ in the deterministic LOCAL model. Hence, we can **answer the open question** by Holroyd [40]:

> Theorem 1.3. *For each $d \geq 2$, there is a finitely-dependent coloring with $d + 1$ colors in d-regular trees. Furthermore, the resulting process is invariant over subgraph isomorphisms.*

More specifically, there exists a 3-coloring finitely-dependent distribution of the infinite 3-regular tree that is invariant under automorphisms.

Theorem 1.2 also introduces a formal **barrier** for proving limitations on distributed quantum advantage. Recall that all current quantum-LOCAL lower bounds are, in essence, lower bounds in the non-signaling model. Before our work, there was a hope that we could discover a symmetry-breaking problem Π with the following properties: (1) its locality is $O(\log^* n)$ in deterministic LOCAL, and (2) we can show that its locality is $\Omega(\log^* n)$ in the non-signaling model, and therefore (3) Π cannot admit any distributed quantum advantage. However, our work shows that no such problem Π can exist. In particular, arguments related to non-signal distributions are not sufficient to exclude distributed quantum advantage in this region.

1.5 Locality in online and dynamic settings

Let us now switch gears and consider a very different line of work. Ghaffari et al. [36] introduced a *sequential* counterpart of the classical LOCAL model:

- **Deterministic SLOCAL model:** The nodes are processed in an adversarial order. When a node v is processed, the algorithm gets to see all information in its radius-$T(n)$ neighborhood. The algorithm has to label v with its local output, and the algorithm can also record other information in v, which it can exploit when other nodes near v are later processed.
- **Randomized SLOCAL model:** As above, but the algorithm has also access to a source of random bits.

It is easy to see that SLOCAL is stronger than LOCAL. One key feature is that the processing order of the nodes naturally breaks symmetry, and all symmetry-breaking LCLs can be solved with $O(1)$ locality in SLOCAL. One interpretation of our first contribution is that we also establish a new, unexpected similarity between SLOCAL and the bounded-dependence model: *both are strong enough to solve any symmetry-breaking LCL with constant locality.*

A recent work [2] introduced the following models that capture the notion of locality also in the context of centralized dynamic graph algorithms and centralized online graph algorithms:

- **Deterministic dynamic-LOCAL model:** The adversary constructs the graph one edge at a time. The algorithm has a global view of the graph, but it has to maintain a feasible solution after each update. The algorithm is restricted so that after a modification at node v, it can only update the solution within distance $T(n)$ from v.
- **Deterministic online-LOCAL model:** The adversary presents the input graph one node at a time. When a node v is presented, the adversary also reveals the radius-$T(n)$ neighborhood of v. The algorithm has to then choose the output label of v.

It turns out that both SLOCAL and dynamic-LOCAL can be sandwiched between LOCAL and online-LOCAL [2]:

$$\begin{array}{ccc} \text{deterministic LOCAL} & \longrightarrow & \text{deterministic SLOCAL} \\ \downarrow & & \downarrow \\ \text{deterministic dynamic-LOCAL} & \longrightarrow & \text{deterministic online-LOCAL} \end{array} \tag{2}$$

There are also some problems in which deterministic online-LOCAL is much stronger than deterministic LOCAL: for example, 3-coloring in bipartite graphs has locality $\tilde{\Theta}(\sqrt{n})$ in the deterministic LOCAL model [16, 24] but $O(\log n)$ in the online-LOCAL model [2]; very recently Chang et al. [20] also showed that this is tight in online-LOCAL.

1.6 Contribution 2: connecting all models for LCLs in trees

At first sight, the models discussed in Sections 1.3 and 1.5 seem to have very little in common; they seem to be orthogonal extensions of the classical deterministic LOCAL model. Furthermore, we have already seen evidence that online-LOCAL can be much stronger than deterministic LOCAL. Nevertheless, we can connect all these models in a unified manner, and prove strong limits on their expressive power. To this end, we introduce yet another model:

- **Randomized online-LOCAL model:** Like deterministic online-LOCAL, but the algorithm has also access to a source of random bits. Crucially, we are playing against an oblivious adversary (the adversary fixes the graph and the order in which the nodes are presented in the beginning, before the algorithm starts to flip coins).

Trivially, this is at least as strong as all models in diagram (2). However, the big surprise is that it is also at least as strong as all models in diagram (1). In Section 4 we prove:

> Theorem 1.4. *Any LCL that can be solved in the non-signal model with locality $T(n)$ can also be solved in the randomized online-*LOCAL *model with the same locality.*

Then we zoom into the case of trees in Section 6 and prove:

> Theorem 1.5. *Any LCL on trees that can be solved in the randomized online-*LOCAL *model with locality $o(\log^{(5)} n)$ can also be solved in the deterministic* LOCAL *model with locality $O(\log^* n)$.*

Here $\log^{(1)} n = \log n$ and $\log^{(k+1)} n = \log\log^{(k)} n$. Now together with Theorem 1.2 and previously-known results, we also obtain the following corollaries:

Theorem 1.6. *In trees, the following families of LCLs are the same:*

- *Problems that can be solved with locality $O(\log^* n)$ in any of these models: deterministic and randomized* LOCAL*, and quantum-*LOCAL
- *Problems that can be solved with locality $O(1)$ in any of these models: bounded-dependence model, non-signaling model, deterministic and randomized* SLOCAL*, dynamic-*LOCAL*, and deterministic and randomized online-*LOCAL.

In trees, there is no LCL problem with locality between $\omega(\log^* n)$ and $o(\log^{(5)} n)$ in any of these models: deterministic and randomized LOCAL*, quantum-*LOCAL*, bounded-dependence model, non-signaling model, deterministic and randomized* SLOCAL*, dynamic-*LOCAL*, and deterministic and randomized online-*LOCAL.

In particular, when we look at LCLs in trees, $O(\log^* n)$-round quantum algorithms are not any stronger than $O(\log^* n)$-round classical algorithms. (However, it is still possible that there are some LCLs in trees that can be solved in $O(1)$ rounds in quantum-LOCAL and that require $\Theta(\log^* n)$ rounds in deterministic LOCAL; recall the discussion in Section 1.4.)

This also implies a new lower bound for the widely-studied *sinkless orientation* problem [12, 15] in a number of models—see Table 1 for more details on the complexity of sinkless orientation in different models:

Theorem 1.7. *Sinkless orientation has locality $\Omega(\log^{(5)} n)$ in all of these models: quantum-*LOCAL*, bounded-dependence model, non-signaling model, dynamic-*LOCAL*, and deterministic and randomized online-*LOCAL.

1.7 Contribution 3: a new lower bound for coloring

So far we have connected randomized online-LOCAL with other models through simulation arguments that only work in trees. Let us now put limitations on randomized online-LOCAL in a broader setting. Recall that in deterministic online-LOCAL we can 3-color bipartite graphs with locality $O(\log n)$ [2], and this is tight [20]. In Section 7 we show that randomness does not help:

Theorem 1.8. *3-coloring in bipartite graphs is not possible with locality $o(\log n)$ in the randomized online-*LOCAL *model.*

This demonstrates that even though randomized online-LOCAL is a very strong model, strong enough to simulate e.g. any non-signaling distribution, it is nevertheless possible to prove strong lower bounds in this model (which then imply lower bounds across the entire landscape of models).

1.8 Additional contributions

It is known that *deterministic* SLOCAL is strong enough to simulate *randomized* LOCAL. In Section 8 we show that the same holds also for dynamic-LOCAL:

Theorem 1.9. *Deterministic dynamic-*LOCAL *can simulate randomized* LOCAL.

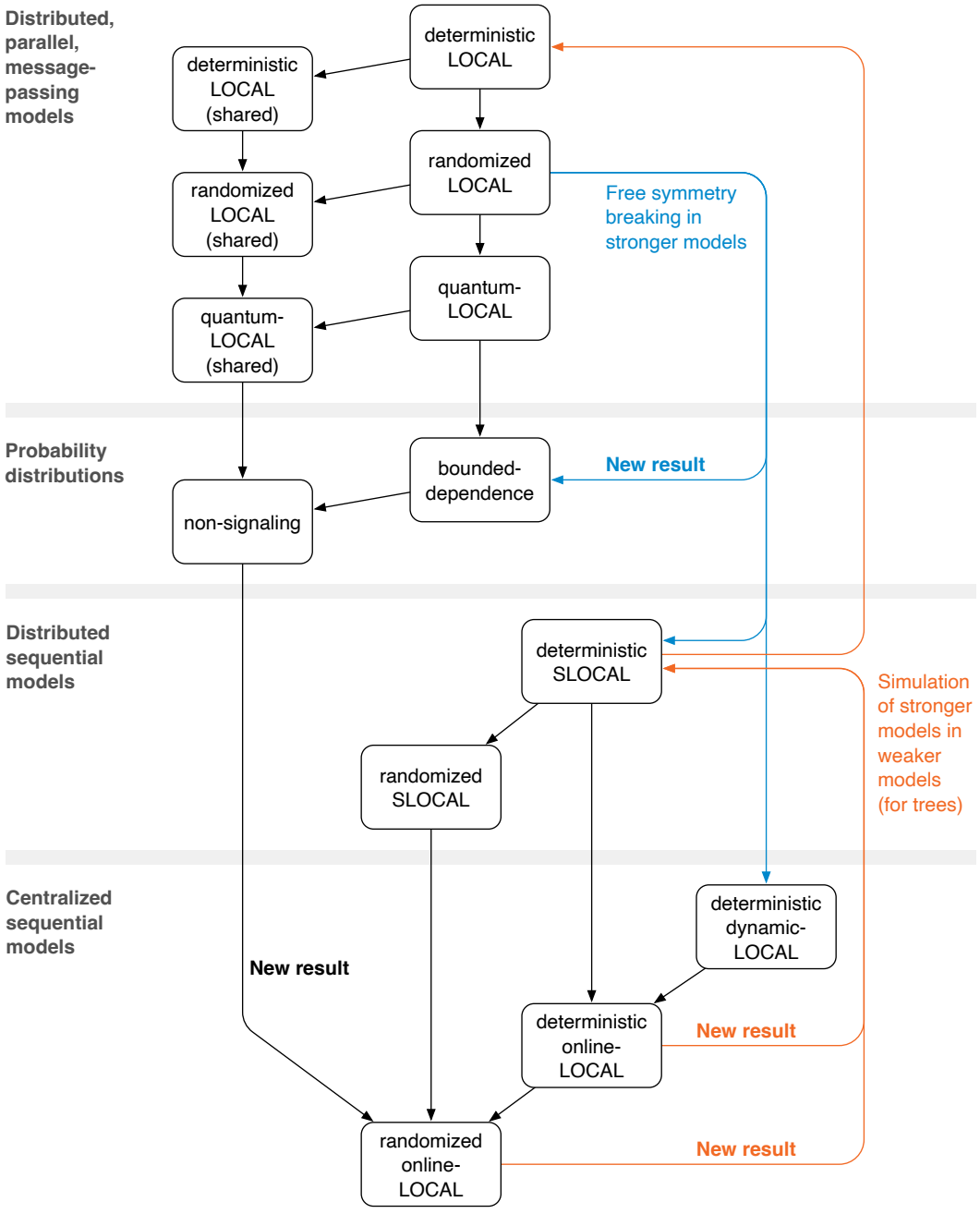


Fig. 1. Landscape of models and their relations for LCL problems. A black arrow $X \to Y$ indicates that model Y is at least as strong as model X and can simulate any algorithm designed there. A blue arrow $X \to Y$ indicates that we additionally get symmetry-breaking for free: locality $O(\log^* n)$ in model X implies locality $O(1)$ in model Y. An orange arrow $X \to Y$ indicates that *at least in trees* we can simulate highly-localized algorithms in model X using the weaker model Y so that e.g. locality $O(\log^* n)$ in model X implies locality $O(\log^* n)$ in model Y (see Theorems 6.1 and 6.2 and [36] for details).

The new model that we introduced, randomized online-LOCAL, is defined using an *oblivious* adversary. In Section 9 we show that this is also necessary: if we defined randomized online-LOCAL with an *adaptive* adversary, it would be as weak as the deterministic online-LOCAL model.

1.9 Discussion and open questions

By putting together all our contributions, the landscape shown in Fig. 1 emerges. We can sandwich all models between deterministic LOCAL and randomized online-LOCAL. Going downwards, we get symmetry breaking for free (as indicated by the blue arrows). And in the case of trees, for the low-locality region $o(\log^{(5)} n)$ we can also navigate upwards (as indicated by the orange arrows).

Table 1 gives an overview of the localities for three representative problems across the landscape of models: 3-coloring in cycles (a classic symmetry-breaking problem), sinkless orientation (an intermediate problem), and 3-coloring in bipartite graphs (a hard problem in which online-LOCAL helps). As shown in the table, our work implies new results also for sinkless orientations (indirectly, as implications of our general results) and for 3-coloring in bipartite graphs (directly).

Our work suggests a number of open questions; here are the most prominent ones:

Question 1.10. Is quantum-LOCAL stronger than randomized LOCAL for any LCL problem? In particular, is there any LCL problem with constant locality in quantum-LOCAL but super-constant locality in LOCAL? We conjecture that such a problem does not exist, but to show that, new proof techniques are needed.

Question 1.11. Can we *exclude* quantum advantage for any concrete LCL problem that has complexity $\Theta(\log^* n)$ in the classical LOCAL models? Again, we need new techniques to prove that, as non-signaling arguments cannot be used. Nevertheless, this could be easier to tackle than Question 1.10, as we can specifically engineer an LCL problem that would remain hard for quantum-LOCAL.

Question 1.12. Does shared global information, shared randomness, or shared quantum state ever help with any LCL problem? We conjecture that the answer is no.

Question 1.13. Is it possible to simulate deterministic or randomized online-LOCAL in SLOCAL and LOCAL also in a broader graph class than trees? Section 10 shows that we can also handle the case of a single cycle, but graphs with multiple cycles remain an open question. Here is a concrete question: if we have an LCL problem with locality $O(1)$ in *grids* in the online-LOCAL models, does that imply locality $O(1)$ in SLOCAL, or can we find a counterexample?

Table 1 also suggests a number of open questions, especially related to understanding the complexity of the sinkless orientation problem in the stronger models. The complexity of 3-coloring bipartite graphs in dynamic-LOCAL is yet another open question for future work.

2 OVERVIEW OF TECHNIQUES AND KEY IDEAS

In this section, we give an overview of the techniques and key ideas that we use to prove our main results, and we also provide a roadmap to the rest of this paper. We note that our first contribution is presented in Section 5, while the second contribution comes before it in Section 4—the proofs are ordered this way, since Section 4 also develops definitions that will be useful in Section 5.

2.1 Bounded-dependence model can break symmetry (Section 5)

Let us first present an overview of the proofs of Theorems 1.2 and 1.3 from Section 1.4. We show that the bounded-dependence model can break symmetry with constant locality; that is, there is a finitely-dependent process for any symmetry-breaking LCL.

Table 1. Localities of three representative problems.

Model	3-coloring in cycles		Sinkless orientation		3-coloring in bipartite graphs	
Deterministic LOCAL	$O(\log^* n)$	[28]	$O(\log n)$	[37]	$\tilde{O}(\sqrt{n})$	[24]
	$\Omega(\log^* n)$	[50]	$\Omega(\log n)$	[15, 19]	$\Omega(\sqrt{n})$	[24]
Randomized LOCAL	$O(\log^* n)$	[28]	$O(\log \log n)$	[37]	$\tilde{O}(\sqrt{n})$	[24]
	$\Omega(\log^* n)$	[52]	$\Omega(\log \log n)$	[15]	$\Omega(\sqrt{n})$	[24]
Quantum LOCAL	$O(\log^* n)$	[28]	$O(\log \log n)$	[37]	$\tilde{O}(\sqrt{n})$	[24]
			$\Omega(\log^{(5)} n)$	Theorem 6.2	$\Omega(\sqrt{n})$	[24]
Bounded-dependence	$O(1)$	[41, 42]	$O(\log \log n)$	[37]	$\tilde{O}(\sqrt{n})$	[24]
			$\Omega(\log^{(5)} n)$	Theorem 6.2	$\Omega(\sqrt{n})$	[24]
Deterministic SLOCAL	$O(1)$	trivial	$O(\log \log n)$	[34]	$\tilde{O}(\sqrt{n})$	[24]
			$\Omega(\log \log n)$	[34]	$n^{\Omega(1)}$	[2, 24]
Randomized SLOCAL	$O(1)$	trivial	$O(\log^{(3)} n)$	[34]	$\tilde{O}(\sqrt{n})$	[2, 24]
			$\Omega(\log^{(3)} n)$	[34]	$n^{\Omega(1)}$	[24]
Deterministic dynamic-LOCAL	$O(1)$	trivial	$O(\log \log n)$	Theorem 8.1	$\tilde{O}(\sqrt{n})$	[24]
			$\Omega(\log^{(3)} n)$	Theorem 6.1	$\Omega(\log n)$	[20]
Deterministic online-LOCAL	$O(1)$	trivial	$O(\log \log n)$	[34]	$O(\log n)$	[2]
			$\Omega(\log^{(3)} n)$	Theorem 6.1	$\Omega(\log n)$	[20]
Randomized online-LOCAL	$O(1)$	trivial	$O(\log^{(3)} n)$	[34]	$O(\log n)$	[2]
			$\Omega(\log^{(5)} n)$	Theorem 6.2	$\Omega(\log n)$	Theorem 7.1

It is well known that any LCL problem Π that has complexity $O(\log^* n)$ in the LOCAL model has the following property: there exists a constant $k \in \mathbb{N}_+$ (that depends only on the hidden constant in $O(\log^* n)$) such that, if the graph is given a distance-k coloring (with sufficiently small number of colors) as an input, then Π is solvable in time $O(1)$ in the LOCAL model (using the distance-k coloring as a local assignment of identifiers) [18]. Furthermore, no knowledge of the size of the input graph is required.

Finding a distance-k coloring in SLOCAL trivially requires locality exactly k, implying the well known fact that LCLs of complexity $O(\log^* n)$ in the LOCAL model have complexity $O(1)$ in SLOCAL.

In a similar way, we prove that for each bounded-degree graph, there is a finitely-dependent process providing a distance-k coloring for constant k. Then, we can just combine such process with the LOCAL algorithm that solves the problem (which has complexity $O(1)$ if given a distance-k coloring as an input) and prove that the resulting process is still a finitely-dependent distribution. Furthermore, we also prove that all these processes are invariant under subgraph isomorphisms (even those that do not preserve node identifiers), meaning that, for any two graphs G and H sharing two isomorphic subgraphs with isomorphic radius-$O(1)$ neighborhoods, the restrictions of the finitely-dependent processes solving Π over G and H restricted to G' and H' are equal in law.

One of the key observations that we use is that LOCAL algorithms that do not exploit the specific assignment of node identifiers and do not depend on the size of the graph provide finitely-dependent

distributions that are invariant under subgraph isomorphisms whenever the input labeling for the graphs is invariant under subgraph isomorphisms.

Overview. The cornerstone of our proof is a surprising result by Holroyd and Liggett [42] and its follow-up in [41], that state that there exist k-dependent distributions giving a q-coloring of the infinite path and of cycles for $(k, q) \in \{(1, 4), (2, 3)\}$ that are invariant under subgraph isomorphisms.

Recently, Holroyd has combined the finitely-dependent distributions of infinite paths to provide a finitely-dependent 4-coloring of the d-dimensional lattice [40]. Getting a translation invariant distribution is quite easy: First, use the distributions for the paths on each horizontal and vertical path obtaining a distance-k coloring (with k being a large enough constant) of the lattice with constantly many colors as shown in [42, Corollary 20]. Second, apply some LOCAL algorithm that starts from a distance-k coloring and reduces the number of colors to 4 while keeping the resulting distribution symmetric (e.g., the algorithms from [11, 16]). The major contribution of [40] is transforming such a distribution into a process that is invariant under subgraph isomorphisms. However, this *symmetrization* phase is quite specific to the considered topology.

We come up with a new approach that obtains similar results in all bounded-degree graphs through the following steps:

(1) We show that the finitely-dependent coloring of paths and cycles can be combined to obtain finitely-dependent 3-coloring distributions of rooted pseudoforests of bounded-degree that are invariant under subgraph isomorphisms.
(2) We observe that all graphs of bounded-degree admit a random decomposition in rooted pseudoforests that satisfies some required symmetry properties.
(3) We prove that such random decomposition can be combined with the finitely-dependent 3-coloring of rooted pseudoforests to obtain finitely dependent distributions that give a $(\Delta + 1)$-coloring of graphs of maximum degree Δ that are invariant under subgraph isomorphism.
(4) We show that we can use this finitely dependent $(\Delta + 1)$-coloring distributions to provide a distance-k coloring for graphs of maximum degree Δ which serves as an input for LOCAL algorithms solving any LCL Π of complexity $O(\log^* n)$ in LOCAL. Such combination results in finitely-dependent processes that are invariant under subgraph-isomorphisms and solve Π.

Notice that, in spirit, steps 1 to 3 are similar to the steps needed to produce a $(\Delta + 1)$-coloring in time $O(\log^* n)$ in the LOCAL model [39, 56]: however, the detailed way these steps are obtained in the bounded-dependence model is quite different and requires a careful ad-hoc analysis.

1. Finitely-dependent 3-coloring distributions of rooted pseudoforests. A pseudotree is a tree that contains at most one cycle. A rooted pseudotree is an oriented pseudotree in which each node as outdegree at most 1. A rooted pseudoforest is just a collection of disjoint rooted pseudotrees. Let us now fix any rooted pseudoforest of maximum degree Δ. Consider the following process: each node v colors its indegree neighbors with a uniformly sampled permutation of the elements of $\{1, \ldots, \mathrm{indeg}(v)\}$. The graph G_i induced by nodes colored with color i is just a disjoint union of directed paths and cycles (see also Fig. 2 for an example) and, hence, admits a finitely-dependent 4-coloring given by [42] that is invariant under subgraph isomorphisms: if a node is isolated, it can deterministically join any of the G_is, say G_1. The sequence of graphs $(G_1, \ldots, G_{\Delta_{\mathrm{in}}})$ is said to be a random Δ_{in}-decomposition of the rooted pseudoforest. Furthermore, if two graphs G, H have isomorphic subgraphs G', H' (together with some constant-radius neighborhoods), the decompositions in directed paths and cycles induced in G' and H' have the same distribution (because node

colors are locally chosen uniformly). We prove that the combination of the random decomposition and the finitely-dependent coloring yields a finitely-dependent 4Δ-coloring which is invariant under subgraph isomorphisms: by further combining such distribution with the Cole–Vishkin color reduction LOCAL algorithm [28, 39] (that has complexity $O(\log^* k)$ with k being the size of the input coloring), we can obtain a finitely-dependent 3-coloring distribution for rooted pseudoforests of maximum degree Δ that is invariant under subgraph isomorphisms.

2–3. Finitely-dependent $(\Delta+1)$-coloring distribution of bounded-degree graphs. First, if the input graph is undirected, make it a directed graph by duplicating all edges and assigning both orientations to duplicated edges. Since a coloring of the nodes can be given in both cases equivalently, we focus on the directed case for simplicity. Second, consider the following process: each node v labels its outdegree edges with a uniformly sampled permutation of the elements of $\{1, \dots, \text{outdeg}(v)\}$. For each i, observe that edges labelled with i form a rooted pseudoforest (see also Fig. 3 for an example) that admits a finitely-dependent 3-coloring distribution (according to step 1) that is invariant under subgraph isomorphisms, inducing a random decomposition of the graph. Furthermore, if two graphs G, H have isomorphic subgraphs G', H' (together with some constant-radius neighborhoods), the decompositions induced in G' and H' have the same distribution (because edge labelings are locally chosen uniformly). We prove that the combination of the random decomposition and the finitely-dependent coloring yields a finitely-dependent 3^Δ-coloring of the bounded-degree graph which is invariant under subgraph isomorphisms: by further combining such distribution with a variant of the Cole–Vishkin color reduction LOCAL algorithm [56] (that has complexity $O(\log^* k)$ with k being the size of the input coloring), we obtain a finitely-dependent $(\Delta+1)$-coloring distribution for bounded-degree graphs of maximum degree Δ that is invariant under subgraph isomorphisms.

4. Finitely-dependent distribution solving Π. Consider any graph G of maximum degree Δ, and its k-th power graph defined as follows: simply add edges to G between each pair of nodes at distance at most k, where k is some large enough constant. Observe that G^k is a graph of maximum degree Δ^k. Now, step 3 implies that there is a finitely-dependent (Δ^k+1)-coloring of G^k that is invariant under subgraph isomorphisms: such distribution yields a distance-k coloring of G. For any LCL Π that has complexity $O(1)$ in SLOCAL, we simply define a LOCAL algorithm $\mathcal{A}'$ solving Π that does not need node identifiers, does not depend on n, and has complexity $O(1)$: the combination of the input distance-k coloring of G with such an algorithm yields a finitely-dependent distribution solving Π that is invariant under subgraph isomorphisms. Basically, we take $k = T$ where T is the locality of some constant-time SLOCAL algorithm $\mathcal{A}$ solving Π, and we define a LOCAL algorithm $\mathcal{A}'$ as follows: nodes colored with color i perform T rounds of communication between round $(i-1)T$ and round $iT-1$ simulating the behavior of $\mathcal{A}$, while all other rounds of communications are useless. We prove that $\mathcal{A}'$ is correct and provides the desired properties.

Random decomposition of a graph. In steps 1 and 3 we proceed in an analogous way: First, we construct a process that induces a random decomposition of a graph. Second, we consider finitely-dependent distributions of output labelings over the outputs of the random decomposition. The combination of the random decomposition and the finitely-dependent distributions gives rise to a process over the whole graph. In Section 5, we derive a general result (Lemma 5.8) which gives sufficient conditions that the random decomposition and the finitely-dependent distributions we use must satisfy to ensure that the final process is still finitely-dependent (possibly with symmetry properties). Lemma 5.8 is then the tool used in practice in steps 1 and 3.

2.2 The non-signaling model can be simulated in randomized online-LOCAL (Section 4)

Let us now give the intuition behind the proof of Theorem 1.4 from Section 1.6: we show that the non-signaling model model can be simulated in randomized online-LOCAL without any loss in the locality.

A randomized online-LOCAL algorithm is given in input the size of the input graph, and a distribution that is non-signaling beyond distance T and that solves some problem Π over some graph family $\mathcal{F}$. When the adversary picks any node v_1 and shows to the randomized online-LOCAL algorithm its radius-T neighborhood, the randomized online-LOCAL algorithm simply goes over all graphs of n nodes in $\mathcal{F}$ until it finds one, say H_1, that includes the radius-T neighborhood of v_1: then, it samples an output according to the restriction of the non-signaling distribution in H_1 to v_1. Notice that such distribution does not change if the topology of the graph changes outside the radius-T neighborhood of v_1. Recursively, when the adversary picks the i-th node v_i, the randomized online-LOCAL algorithm goes over all graphs of n nodes in $\mathcal{F}$ until it finds one, say H_i, that includes the union of radius-T neighborhoods of $v_1, \dots, v_i$ (it must necessarily exist as the graph chosen by the adversary is a valid input): hence, it samples an output according to the restriction of the non-signaling distributions in H_i to v_i conditional on the outputs of $v_1, \dots, v_{i-1}$. We prove that the non-signaling property ensures that the algorithm described above fails with at most the same probability of failure of the non-signaling distribution.

2.3 Online-LOCAL can be simulated in SLOCAL for trees (Section 6)

Next, we give an overview of the proof of Theorem 1.5 from Section 1.6: we show that a randomized online-LOCAL algorithm that solves an LCL problem in trees with locality $o(\log^{(5)} n)$ can be simulated in the deterministic SLOCAL model with locality $O(1)$, and therefore also in the deterministic LOCAL model with locality $O(\log^* n)$.

Online-LOCAL algorithms can be seen as SLOCAL algorithms with a global memory. The key idea in the proof of Theorem 1.5 is to make this global memory useless by showing the algorithm so many neighborhoods that it can no longer distinguish between them. In a sense, we make the online-LOCAL algorithm *amnesiac*, i.e. to lose its memory and process each neighborhood as if it was the first one it ever processed. We form a spectrum between $(0, C)$-amnesiac algorithms that are just regular online-LOCAL algorithms, and (n, C)-amnesiac algorithms, where n is the size of the input instance, that are very close to being SLOCAL.

Constructing amnesiac algorithms. To get some intuition on how we are going to do, start by considering an LCL problem Π on forests and a *deterministic* online-LOCAL algorithm $\mathcal{A}$ solving Π with locality $T(n)$ in n-node graphs. Note that the output label $\mathcal{A}$ chooses for a node may depend arbitrarily on the information the algorithm has seen so far. This is in contrast with SLOCAL algorithms in which the output may depend only on the local information around the node.

We now show how to turn algorithm $\mathcal{A}$ into an algorithm whose output for *isolated* nodes depends only on the local topology and inputs, not on any previously-processed nodes; we call such algorithms $(1, C)$-amnesiac. A node v is isolated if all nodes withing ball $B(v, T)$ are new for algorithm $\mathcal{A}$, that is it doesn't know how v connects to other nodes it has seen before. We defer the general construction of (a, C)-amnesiac algorithms to Lemma 6.5 in Section 6.

Let $\mathcal{G}$ be the set of all possible n-node forests with inputs and ordering of nodes. We denote $g = |\mathcal{G}|$ and remark that $g \leq 2^{n^2} |\Sigma_{\text{in}}|^n$. Now consider the following experiment: Pick a natural number C and construct $f(C) = |\Sigma_{\text{out}}| \cdot C$ copies of $\mathcal{G}$. Then reveal the first node on each of those graphs to $\mathcal{A}$ in an arbitrary order with locality T that is to be determined later. Now for each *type of neighborhood* $\mathcal{T} = B(v, T)$, there exists some output label $\sigma_{\mathcal{T}}$ that occurs at least C times; we call such neighborhoods *good*.

We can now imagine a new online-LOCAL algorithm $\mathcal{B}$ that starts by running the above experiment with algorithm $\mathcal{A}$. Afterwards, $\mathcal{B}$ proceeds exactly like $\mathcal{A}$, showing nodes to $\mathcal{A}$ as they arrive and labeling them accordingly, except for nodes that are isolated.

Let node v be such isolated node. Algorithm $\mathcal{B}$ starts by finding a good neighborhood matching $B(v, T)$ in the experiment. It then takes that neighborhood, removes all edges just outside the neighborhood, identifies all nodes with the revealed neighborhood, and then labels v accordingly. Given that the constant C was selected to be sufficiently large, such neighborhood exists. In effect, algorithm $\mathcal{B}$ cuts and pastes the neighborhood from the experiment to the actual graph without algorithm $\mathcal{A}$ noticing.

The algorithm $\mathcal{B}$ we just described labels the graph correctly by the virtue of $\mathcal{A}$ working correctly. Moreover, when labeling an isolated node, it always labels it in a consistent way that depends only on the local structure and inputs of the graph, as long as it has not seen more than C isolated neighborhoods that look exactly the same.

Randomized online-LOCAL. For deterministic online-LOCAL algorithms, we can adaptively pick the good neighborhoods before processing further. As the randomized online-LOCAL model calls for an oblivious adversary, we cannot do this. Instead, we repeat the experiment triply exponentially many times to force the algorithm to produce good labelings with probability high enough. We then extract a good set of labelings from this experiment and use those to produce a deterministic SLOCAL algorithm.

2.4 Lower bound on 3-coloring in randomized online-LOCAL (Section 7)

Let us now look at the proof of Theorem 1.8 from Section 1.7. We present a lower bound for 3-coloring in $\sqrt{n} \times \sqrt{n}$ grids, showing that it takes $\Omega(\log n)$ locality in the randomized online-LOCAL model.

Here it is important to assume that the adversary is oblivious, i.e., it cannot see the random decisions of the randomized algorithm. This lower bound complements a recent result of Chang et al. [20] showing the same bound for the deterministic online-LOCAL model.

In this proof, we use the notion of a b-value defined in [20] as a measure of the number of incompatible boundaries present in a region of a grid. We start with the assumption that a grid can be 3-colored with locality $o(\log n)$ and derive a contradiction. The high level idea is to construct two path segments below each other, where one path segment has a large count of incompatible boundaries (a high b-value) and the other segment has a low boundary count (incompatible b-value to the upper path). This forces an algorithm to make boundaries escape on the side between the two segments. We show that the boundary count is, however, too large compared to the distance between the two segments and thus the boundaries "cannot escape".

Two difficulties arise in the randomized case compared to the deterministic lower bound: (i) In order to create a path with a large b-value we have to use a probabilistic construction that produces a segment with a large b-value with high probability; (ii) Since the first construction is probabilistic and the adversary oblivious, we cannot "see" the large b-value segment constructed in (i), that is, we can neither predict its position nor its size. We therefore need to use another probabilistic construction that positions the segment in a position that forces a contradiction (and which succeeds with constant probability).

2.5 Deterministic dynamic-LOCAL can derandomize randomized LOCAL (Section 8)

Let us now say a few words about the proof of Theorem 1.9 from Section 1.8: we show that deterministic dynamic-LOCAL can simulate randomized LOCAL.

The idea behind this result is the same as that for SLOCAL explained in Section 2.1. Any LCL problem Π that has complexity $O(\log^* n)$ in the LOCAL model has the following property: there exists a constant $k \in \mathbb{N}_+$ (that depends only on the hidden constant in $O(\log^* n)$) such that, if the input graph is given a distance-k coloring in the input, then Π is solvable in time $O(1)$ in the LOCAL model (using the distance-k coloring as a local assignment of identifiers) [18]. The proof consists in showing that dynamic-LOCAL can indeed provide a distance-k coloring in constant time.

2.6 Randomized online-LOCAL with adaptive adversary equals deterministic online-LOCAL (Section 9)

As mentioned in Section 1.8, it is necessary to work with an oblivious adversary in randomized online-LOCAL: we show that an algorithm that works against an adaptive adversary can be turned into a deterministic algorithm.

We show that an adaptive adversary in randomized online-LOCAL is so strong that a succeeding randomized randomized online-LOCAL algorithm of locality T would admit a single random-bit string that outputs a good solution for all possible graphs of a given size: hence, a correct deterministic randomized online-LOCAL algorithm exists. Since deterministic online-LOCAL algorithms of locality T for graphs of n nodes are only finitely many, the deterministic online-LOCAL algorithm, in the initialization phase, can go over all of them until it finds the one working for all graphs of n nodes: it then uses that one.

2.7 Randomized online-LOCAL in paths and cycles (Section 10)

Our final technical part shows that LCL problems in paths and cycles have complexity either $O(1)$ or $\Theta(n)$ in the randomized online-LOCAL model; moreover the locality is $O(1)$ in randomized online-LOCAL if and only if it is $O(\log^* n)$ in the deterministic LOCAL model. Together with prior work, this also shows that locality of an LCL problem in paths and cycles is *decidable* across all models [4] (with the caveat that we cannot distinguish between $O(1)$ and $\Theta(\log^* n)$ for quantum-LOCAL).

The proof is a reworking of its deterministic variant from [2]. The main take-home message of this result is the following: cycles are not a fundamental obstacle for simulating randomized online-LOCAL in weaker models. Hence, there is hope for generalizing the simulation result of Section 6 from trees to a broader class of graphs.

2.8 Quantum models vs. non-signaling and bounded-dependence models (Appendix A)

Appendix A aims at serving a dual purpose. First, it aims at formally introducing the non-signaling model based on the non-signaling principle, and at explaining *why* it is more powerful than the quantum-LOCAL model. This is not a new result, but included for completeness and to clarify the early works of Gavoille et al. [33] and Arfaoui and Fraigniaud [3].

Second, it formally introduces the bounded-dependence model based on finitely-dependent processes and argues why the relations in diagram (1) hold, and in particular why quantum-LOCAL without shared quantum state is contained not only in the non-signaling model but also in the bounded-dependence model. While all the ingredients are well-known, to our knowledge this relation between the quantum-LOCAL model and the bounded-dependence model is not made explicit in the literature before.

3 PRELIMINARIES

In this section we give some preliminaries.

We consider the set $\mathbb{N}$ of natural numbers to start with 0. We also define $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. For any positive integer $n \in \mathbb{N}_+$, we denote the set $\{1, \ldots, n\}$ by $[n]$.

Graphs. In this work, a graph $G = (V, E)$ can be either directed ($E \subseteq V^2$) or undirected ($E \subseteq \binom{V}{2}$). If the set of nodes and the set of edges are not specified, we refer to them by $V(G)$ and $E(G)$, respectively. For any edge $e = (u, v) \in E(G)$ we say that e is directed from u to v and is incident to u and to v (the latter holds also for undirected edges). All graphs in this paper are simple graphs without self-loops unless differently specified. The degree of a node v is the number of edges that are incident to v and is denoted by $\deg_G(v)$, or simply by $\deg(v)$ when G is clear from the context. The indegree of a node v is the number of directed edges that are directed towards v and is denoted by $\text{indeg}_G(v)$, while the outdegree of v is the number of directed edges that are incident to v but directed to some other vertex and is denoted by $\text{outdeg}_G(v)$. Again, we omit the suffix G if the graph is clear from the context.

If G is a subgraph of H, we write $G \subseteq H$. For any subset of nodes $A \subseteq V$, we denote by $G[A]$ the subgraph induced by the nodes in A. For any nodes $u, v \in V$, $\text{dist}_G(u, v)$ denotes the distance between u and v in G (i.e., the number of edges of any shortest path between u and v in G—it doesn't need to be a directed path); if u and v are disconnected, then $\text{dist}_G(u, v) = +\infty$. If G is clear from the context, we may also simply write $\text{dist}(u, v) = \text{dist}_G(u, v)$. Also, for subset of nodes $A, B \subseteq V$ and any node $v \in V$, we can define $\text{dist}_G(v, A) = \min_{u \in A} \text{dist}_G(u, v)$ and, by extension, $\text{dist}_G(A, B) = \min_{u \in A} \text{dist}_G(u, B)$. We assume that $\text{dist}_G(A, \emptyset) = +\infty$. Similarly, for subgraphs $G_1, G_2 \subseteq G$, we define $\text{dist}_G(G_1, G_2) = \text{dist}_G(V(G_1), V(G_2))$: here, we also assume $\text{dist}_G(G_1, \emptyset) = +\infty$ where $\emptyset$ is now the empty graph. For $T \in [n]$, the T-neighborhood of a node $u \in V$ of a graph G is the set $\mathcal{N}_T(u, G) = \{v \in V \mid \text{dist}_G(u, v) \leq T\}$. The T-neighborhood of a subset $A \subseteq V$ is the set $\mathcal{N}_T(A, G) = \{v \in V \mid \exists u \in A : \text{dist}_G(u, v) \leq T\}$. Similarly, the T-neighborhood of a subgraph $H \subseteq G$ is the set $\mathcal{N}_T(H, G) = \{v \in V \mid \exists u \in V(H) : \text{dist}_G(u, v) \leq T\}$. If G is clear from the context, we just write $\mathcal{N}_T(u)$, $\mathcal{N}_T(A)$, and $\mathcal{N}_T(H)$. If $u \notin V$, $A \cap V = \emptyset$, or $V(H) \cap V(G) = \emptyset$, then the neighborhood $\mathcal{N}_T(u) = \emptyset$, $\mathcal{N}_T(A) = \emptyset$, or $\mathcal{N}_T(H) = \emptyset$, respectively. We make use of some graph operations: For any two graphs G, H, we denote by $G \cap H$ the intersection graph defined by $G \cap H = (V(G) \cap V(H), E(G) \cap E(H))$. The graph union is defined by $G \cup H = (V(G) \cup V(H), E(G) \cup E(H))$. Moreover, the graph difference is the graph $G \setminus H = (V(G) \setminus V(H), E(G) \setminus E(H))$.

Finally, for any two graphs G and H, we write $G \sim_f H$ to denote that G and H are isomorphic and $f : V(G) \to V(H)$ is an isomorphism.

Labeling problems. We start with the notion of labeling problem.

Definition 3.1 (Labeling problem). Let Σ_{in} and Σ_{out} two sets of input and output labels, respectively. A *labeling problem* Π is a mapping $(G, \lambda_{\text{in}}) \mapsto \{\lambda_{(\text{out},i)}\}_{i \in I}$, with I being a discrete set of indexes, that assigns to every graph G with any input labeling $\lambda_{\text{in}} : V(G) \to \Sigma_{\text{in}}$ a set of permissible output vectors $\lambda_{(\text{out},i)} : V(G) \to \Sigma_{\text{out}}$ that might depend on (G, λ_{in}). The mapping must be closed under graph isomorphism, i.e., if $\varphi : V(G) \to V(G')$ is an isomorphism between G and G', and $\lambda_{(\text{out},i)} \in \Pi((G', \lambda_{\text{in}}))$, then $\lambda_{(\text{out},i)} \circ \varphi \in \Pi((G, \lambda_{\text{in}} \circ \varphi))$.

A labeling problem can be thought as defined for *any* input graph of *any* number of nodes. If the set of permissible output vectors is empty for some input (G, λ_{in}), we say that the problem is not solvable on the input (G, λ_{in}): accordingly, the problem is solvable on the input (G, λ_{in}) if $\Pi(G, \lambda_{\text{in}}) \neq \emptyset$.

One observation on the generality of definition of labeling problem follows: one can actually consider problems that require to output labels on edges.

We actually focus on labeling problems where, for any input graph, an output vector λ_{out} is permissible if and only if the restrictions of the problem on any local neighborhoods can be solved and there exist compatible local permissible output vectors whose combination provides λ_{out}. This concept is grasped by the notion of locally checkable labeling (LCL) problems, first introduced by

Naor and Stockmeyer [53]. For any function $f : A \to B$ and any subset $A' \subseteq A$, let us denote the restriction of f to A' by $f \restriction_{A'}$. Furthermore, we define a centered graph to be a pair (H, v_H) where H is a graph and $v_H \in V(H)$ is a vertex of H that we name the *center* of H. The *radius* of a centered graph is the maximum distance from v_H to any other node in H.

Definition 3.2 (Locally checkable labeling problem). Let $r, \Delta \in \mathbb{N}$. Let Σ_{in} and Σ_{out} two finite sets of input and output labels, respectively, and Π a labeling problem. Π is *locally checkable* with checking radius r if there exists a family $\mathcal{S} = \{((H, v_H), \bar{\lambda}_{\text{in}}, \bar{\lambda}_{\text{out}})_i\}_{i \in I}$ of tuples, where (H, v_H) is a centered graph of radius at most r and maximum degree at most Δ, $\bar{\lambda}_{\text{in}} : V(H) \to \Sigma_{\text{in}}$ is an input labeling for H, $\bar{\lambda}_{\text{out}} : V(H) \to \Sigma_{\text{out}}$ is an output labeling for H (which can depend on $\bar{\lambda}_{\text{in}}$) with the following property

- for any input (G, λ_{in}) to Π with $\deg(G) \leq \Delta$, an output vector $\lambda_{\text{out}} : V(G) \to \Sigma_{\text{out}}$ is permissible (i.e., $\lambda_{\text{out}} \in \Pi((G, \lambda_{\text{in}}))$) if and only if, for each node $v \in V(G)$, the tuple $((G[\mathcal{N}_r(v)]), \lambda_{\text{in}} \restriction_{\mathcal{N}_r(v)}, \lambda_{\text{out}} \restriction_{\mathcal{N}_r(v)})$ belongs to $\mathcal{S}$ (up to graph isomorphisms).

Notice that the family $\mathcal{S}$ can be always thought to be finite up to graph isomorphisms, as r and Δ are fixed and the set of input/output labels are finite. We now define the computational models we work in.

The port-numbering model. A port-numbered network is a triple $N = (V, P, p)$ where V is the set of nodes, P is the set of *ports*, and $p : P \to P$ is a function specifying connections between ports. Each element $x \in P$ is a pair (v, i) where $v \in V$, $i \in \mathbb{N}_+$. The connection function p between ports is an involution, that is, $p(p(x)) = x$ for all $x \in P$. If $(v, i) \in P$, we say that (v, i) is port number i in node v. The degree of a node v in the network N is $\deg_N(v)$ is the number of ports in v, that is, $\deg_N(v) = |\{i \in \mathbb{N} : (v, i) \in P\}|$. Unless otherwise mentioned, we assume that port numbers are consecutive, i.e., the ports of any node $v \in V$ are $(v, 1), \ldots, (v, \deg_N(v))$. Clearly, a port-numbered network identifies an *underlying graph* $G = (V, E)$ where, for any two nodes $u, v \in V$, $\{u, v\} \in E$ if and only if there exists ports $x_u, x_v \in P$ such that $p(x_u) = x_v$. Clearly, the degree of a node $\deg_N(v)$ corresponds to $\deg_G(v)$.

In the port-numbering model we are given distributed system consisting of a port-numbered network of $|V| = n$ *processors* (or *nodes*) that operates in a sequence of synchronous rounds. In each round the processors may perform unbounded computations on their respective local state variables and subsequently exchange of messages of arbitrary size along the links given by the underlying input graph. Nodes identify their neighbors by using ports as defined before, where port assignment may be done adversarially. Barring their degree, all nodes are identical and operate according to the same local computation procedures. Initially all local state variables have the same value for all processors; the sole exception is a distinguished local variable $x(v)$ of each processor v that encodes input data.

Let Σ_{in} be a set of input labels. The input of a problem is defined in the form of a labeled graph (G, x) where $G = (V, E)$ is the system graph, V is the set of processors (hence it is specified as part of the input), and $x \colon V \to \Sigma_{\text{in}}$ is an assignment of an input label $\lambda_{\text{in}}(v) \in \Sigma_{\text{in}}$ to each processor v. The output of the algorithm is given in the form of a vector of local output labels $\lambda_{\text{out}} \colon V \to \Sigma_{\text{out}}$, and the algorithm is assumed to terminate once all labels $\lambda_{\text{out}}(v)$ are definitely fixed. We assume that nodes and their links are fault-free. The local computation procedures may be randomized by giving each processor access to its own set of random variables; in this case, we are in the *randomized* port-numbering model as opposed to the *deterministic* port-numbering model.

The running time of an algorithm is the number of synchronous rounds required by all nodes to produce output labels. If an algorithm running time is T, we also say that the algorithm has locality T. Notice that T can be a function of the size of the input graph. We say that a problem Π over

some graph family $\mathcal{F}$ has complexity T in the port-numbering model if there is a port-numbering algorithm running in time T that solves Π over $\mathcal{F}$. If the algorithm is randomized, we also require that the failure probability is at most $1/n$, where n is the size of the input graph.

We remark that the notion of an (LCL) problem is a graph problem, and does not depend on the specific model of computation we consider (hence, the problem cannot depend on, e.g., port numbers).

The LOCAL *model.* The LOCAL model was first introduced by Linial [49]: it is just the port-numbering model augmented with an assignment of unique identifiers to nodes. Let $c \geq 1$ be a constant, and let Σ_{in} be a set of input labels. The input of a problem is defined in the form of a labeled graph (G, x) where $G = (V, E)$ is the system graph, V is the set of processors (hence it is specified as part of the input), and $x \colon V \to [n^c] \times \Sigma_{\text{in}}$ is an assignment of a *unique* identifier $\text{id}(v) \in [n^c]$ and of an input label $\lambda_{\text{in}}(v) \in \Sigma_{\text{in}}$ to each processor v. The output of the algorithm is given in the form of a vector of local output labels $\lambda_{\text{out}} \colon V \to \Sigma_{\text{out}}$, and the algorithm is assumed to terminate once all labels $\lambda_{\text{out}}(v)$ are definitely fixed. We assume that nodes and their links are fault-free. The local computation procedures may be randomized by giving each processor access to its own set of random variables; in this case, we are in the *randomized* LOCAL (randomized LOCAL) model as opposed to *deterministic* LOCAL (deterministic LOCAL). Notice that the knowledge of n makes the randomized port-numbering model roughly equivalent to the randomized LOCAL model, as unique identifiers can be produced with high probability. We say that a problem Π over some graph family $\mathcal{F}$ has complexity T in the LOCAL model if there is a LOCAL algorithm running in time T that solves Π over $\mathcal{F}$. If the algorithm is randomized, we also require that the failure probability is at most $1/n$, where n is the size of the input graph.

The sequential LOCAL *model.* The sequential LOCAL model was first introduced by [36]: it is a sequential version of the LOCAL model. Nodes are processed according to an adversarial order $\sigma = v_1, \ldots, v_n$. When processing a node v_i, a T-round algorithm collects all inputs in the radius-T neighborhood of v_i (including outputs of those $v_j \in \mathcal{N}_T(V_i)$ for $j < i$): we say that such an algorithm has complexity T. Note that the algorithm might store all inputs in $\mathcal{N}_T(v_i)$ in the output of v_i: hence, when processing v_i, it can see the input of v_j, $j < i$, if and only if there is a subsequence of nodes $\{v_{h_k}\}_{k \in [m]}$ with $j = h_k < h_{k+1} < \cdots < h_{k_m} = i$ such that $v_{h_k} \in \mathcal{N}_T(v_{h_{k+1}})$ for all $k \in [m]$.

If the algorithm is given an infinite random bit string, we talk about the randomized SLOCAL model, as opposed to the deterministic SLOCAL model. We assume that the adversarial order according to which nodes are processed is oblivious to the random bit string, as in the original definition of the model. We say that a problem Π over some graph family $\mathcal{F}$ has complexity T in the SLOCAL model if there is an SLOCAL algorithm running in time T that solves Π over $\mathcal{F}$. If the algorithm is randomized, we also require that the failure probability is at most $1/n$, where n is the size of the input graph.

*The online-*LOCAL *model.* The (deterministic) online-LOCAL model was introduced in [2]. This is a centralized model where the algorithm initially knows only the set of nodes of the input graph G. The nodes are processed with respect to an adversarial input sequence $\sigma = v_1, v_2, \ldots, v_n$, such that the label of the v_i depends on $G_i = G[\bigcup_{j=1}^{i} \mathcal{N}_T(v_j)]$, i.e., the subgraph induced by the radius-T neighborhoods of $v_1, v_2, \ldots, v_i$ (including all input data).

We define the randomized online-LOCAL model as a randomized variant of the online-LOCAL model where the label assigned by the algorithm to v_i is a random outcome. Note that this model is oblivious to the randomness used by the algorithm. In particular this means that the graph $G \setminus G_i$ cannot be changed depending on the label assigned to v_i. One could also define the randomized

online-LOCAL model in an adaptive manner, but it turns out that this is equivalent to the deterministic online-LOCAL model as we show in Section 9. We say that a problem Π over some graph family $\mathcal{F}$ has complexity T in the online-LOCAL (randomized online-LOCAL) model if there is an online-LOCAL (randomized online-LOCAL) algorithm running in time T that solves Π over $\mathcal{F}$. If the algorithm is randomized, we also require that the failure probability is at most $1/n$, where n is the size of the input graph.

4 SIMULATION OF THE NON-SIGNALING MODEL IN RANDOMIZED ONLINE-LOCAL

4.1 Framework

In this section we give the necessary framework to define the non-signaling model model and is largely inspired by the definitions given in [24, 33]. Next definition introduces the concept of outcome.

Definition 4.1 (Outcome). Let Σ_{in} and Σ_{out} be two sets of input and output labels, respectively, and let $\mathcal{F}$ be a family of graphs. An *outcome* O over $\mathcal{F}$ is a mapping $(G, x) \mapsto \{(\lambda_{(\text{out},i)}, p_i)\}_{i\in I}$, with I being a discrete set of indexes, assigning to every input graph $G \in \mathcal{F}$ with any input data $x = (\text{id} : V(G) \to [|V(G)|^c], \lambda_{\text{in}} : V(G) \to \Sigma_{\text{in}})$, a discrete probability distribution $\{p_i\}_{i\in I}$ over output vectors $\lambda_{(\text{out},i)} : V(G) \to \Sigma_{\text{out}}$ such that:

(1) for all $i \in I$, $p_i > 0$;
(2) $\sum_{i\in I} p_i = 1$;
(3) p_i represents the probability of obtaining $\lambda_{(\text{out},i)}$ as the output vector of the distributed system.

We say that an outcome O over some graph family $\mathcal{F}$ *solves* problem Π over $\mathcal{F}$ *with probability* p if, for every $G \in \mathcal{F}$ and any input data $x = (\text{id}, \lambda_{\text{in}})$, it holds that

$$\sum_{\substack{(\lambda_{(\text{out},i)}, p_i)\in \mathrm{O}((G,x)) : \\ \lambda_{(\text{out},i)} \in \Pi((G,\lambda_{\text{in}}))}} p_i \geq p.$$

When $p = 1$, we will just say that O *solves* problem Π over the graph family $\mathcal{F}$.

The next computational model tries to capture the fundamental properties of any *physical* computational model (in which one can run either deterministic, random, or quantum algorithms) that respects causality. The defining property of such a model is that, for any two (labeled) graphs (G_1, x_1) and (G_2, x_2) that share some identical subgraph (H, y), every node u in H must exhibit identical behavior in G_1 and G_2 as long as its *local view*, that is, the set of nodes up to distance T away from u together with input data and port numbering, is fully contained in H. As the port numbering can be computed with one round of communication through a fixed procedure (e.g., assigning port numbers $1, 2, \ldots, \deg(v)$ based on neighbor identifiers in ascending order) and we care about asymptotic bounds, we will omit port numbering from the definition of local view.

The model we consider has been introduced by [33]. In order to proceed, we first define the *non-signaling* property of an outcome. Let $T \geq 0$ be an integer, and I a set of indices. For any set of nodes V, subset $S \subseteq V$, and for any input $(G = (V, E), x)$, we define its *T-local view* as the set

$$\mathbf{v}_T(G, x, S) = \{(u, x(u)) \mid \exists\, u \in V,\ v \in S \text{ such that } \operatorname{dist}_G(u, v) \leq T\},$$

where $\operatorname{dist}_G(u, v)$ is the distance in G. Furthermore, for any subset of nodes $S \subseteq V$ and any output distribution $\{\lambda_{(\text{out},i)}, p_i\}_{i\in I}$, we define the *marginal distribution* of $\{\lambda_{(\text{out},i)}, p_i\}_{i\in I}$ on set S as the unique output distribution defined as follows: for any $\lambda_{\text{out}} : V(G) \to \Sigma_{\text{out}}$, the probability of $\lambda_{\text{out}} \restriction_S$

on S is given by

$$p(\lambda_{\text{out}}, S) = \sum_{i \in I \,:\, \lambda_{\text{out}} \restriction_S = \lambda_{(\text{out},i)} \restriction_S} p_i,$$

where $\lambda_{\text{out}} \restriction_S$ and $\lambda_{(\text{out},i)} \restriction_S$ are the restrictions of λ_{out} and $\lambda_{(\text{out},i)}$ on S, respectively.

Definition 4.2 (Non-signaling outcome). Let $\mathcal{F}$ be a family of graphs. An outcome $\mathsf{O} : (G, x) \mapsto \{(\lambda_{(\text{out},i)}, p_i)\}_{i \in I}$ over $\mathcal{F}$ is *non-signaling* beyond distance $T = T(G, x)$ if for any pair of inputs $(G_1 = (V_1, E_1), x_1)$, $(G_2 = (V_2, E_2), x_2)$, with the same number of nodes, such that $\mathbf{v}_{T(G_1,x_1)}(G_1, x_1, S)$ is isomorphic to $\mathbf{v}_{T(G_1,x_1)}(G_2, x_2, S)$ and $G_1, G_2 \in \mathcal{F}$, the output distributions corresponding to these inputs have identical marginal distributions on the set S.

Definition 4.2 is also the more general definition for the locality of an outcome: an outcome O has locality T if it is non-signaling beyond distance T.

*The φ-*LOCAL *model.* The φ-LOCAL model is a computational model that produces non-signaling outcomes over some family of graphs $\mathcal{F}$. Let $p \in [0, 1]$. A problem Π over some graph family $\mathcal{F}$ has complexity T (and success probability p) if there exists an outcome O that is non-signaling beyond distance T which solves Π over $\mathcal{F}$ (with probability at least p).

As every (deterministic or randomized) algorithm running in time at most T in the LOCAL model produces an outcome which has locality T, we can provide lower bounds for the LOCAL model by proving them in the φ-LOCAL model.

Notice that algorithms in the LOCAL model can be always thought as producing outputs for *any* input graph: when the computation at any round is not defined for some node, we can make the node output some garbage label, say $\perp$. If we require that also outcomes are defined for every possible graph, then we are restricting the power of φ-LOCAL because outcomes must be defined accordingly. This gives rise to a slightly weaker model, the non-signaling model, which was considered in other works such as [24] and is still stronger than any classical or quantum variation of the LOCAL model.

Theorem 4.3. *Let Π be any LCL problem over any family of graphs $\mathcal{F}$. Let $\mathsf{O} : (G, \lambda_{in}) \mapsto \{(\lambda_{(out,h)}, p_i)\}_{h \in H}$ be an outcome over $\mathcal{F}$ which solves Π with failure probability at most ε and is non-signaling at distance greater than T. There exists a randomized online-*LOCAL *algorithm $\mathcal{A}$ with complexity T that solves Π over $\mathcal{F}$ and has failure probability at most ε.*

We want to design a randomized online-LOCAL algorithm $\mathcal{A}$ with complexity T that solves Π over $\mathcal{F}$ and has failure probability at most ε that somehow simulates O. We need to assume that the number of nodes of the input graph is known by $\mathcal{A}$.

Fix any graph $G \in \mathcal{F}$ of n nodes and any input labeling λ_{in}, and fix any sequence of nodes $v_1, \ldots, v_n$ the adversary might choose to reveal to the algorithm.

The adversary initially shows to the algorithm $G[\mathcal{N}_T(v_1)]$ (including input labels and identifiers) and asks to label v_1. For the algorithm, it is sufficient to find any graph $H_1 \in \mathcal{F}$ of n nodes that contains a subgraph isomorphic to $G[\mathcal{N}_T(v_1)]$ (including input labels and identifiers): the non-signaling property ensures that the restriction of the output distribution $\mathsf{O}(H_1, \lambda_{\text{in}})$ over $\mathcal{N}_T(v_1)$ does not change if the topology of the graph outside $G[\mathcal{N}_T(v_1)]$ differs. Notice that H_1 exists necessarily as G itself is such a graph.

For any $\lambda_{\text{out}} : V(G) \to \Sigma_{\text{out}}$, the probability that $\mathcal{A}$ labels v_1 with $\lambda_{\text{out}}(v_1)$ is

$$p(\lambda_{\text{out}}, v_1) = \sum_{k \,:\, \lambda_{\text{out}} \restriction_{\{v_1\}} = \lambda_{(\text{out},k)} \restriction_{\{v_1\}}} p_k,$$

where $\{(\lambda_{(\text{out},k)}, p_k)\}_{k \in K_1} = \mathsf{O}(H_1, \lambda_{\text{in}})$.

Let Λ_i be the random variable yielding the label assigned to v_i by $\mathcal{A}$. In general, for any $\lambda_{\text{out}} : V(G) \to \Sigma_{\text{out}}$ let $p(\lambda_{\text{out}}, v_i)$ denote the probability that $\Lambda_i = \lambda_{\text{out}}(v_i)$. Assume now that $G[\mathcal{N}_T(v_{i+1})]$ is shown to the algorithm. Conditional on $\Lambda_1, \ldots, \Lambda_i$, for any $\lambda_{\text{out}} : V(G) \to \Sigma_{\text{out}}$ such that $\lambda_{\text{out}}(v_j) = \Lambda_j$ for all $j = 1, \ldots, i$, the probability that $\Lambda_{i+1} = \lambda_{\text{out}}(v_{i+1})$ is

$$p(\lambda_{\text{out}}, v_{i+1}) = \sum_{\substack{k\colon \lambda_{\text{out}} \restriction_{\{v_j\}} = \lambda_{(\text{out},k)} \restriction_{\{v_j\}} \\ \forall\ 1 \le j \le i+1}} \frac{p_k}{\prod_{1 \le l \le i} p(\lambda_{\text{out}}, v_l)},$$

where $\{(\lambda_{(\text{out},k)}, p_k)\}_{k \in K_{i+1}} = \mathrm{O}(H_{i+1}, \lambda_{\text{in}})$ for any graph $H_{i+1} \in \mathcal{F}$ of n nodes that contains a subgraph isomorphic to $G[\cup_{i=1}^{i+1} \mathcal{N}_T(v_j)]$ (including input labels and identifiers). The specific choice of H_{i+1} does not matter since the property of a non-signaling outcome is exactly that the probability distribution of any output over any subset S of nodes does not change if the topology of the graph is modified "far enough" from S: here, $S = \{v_1, \ldots, v_{i+1}\}$.

Now, consider $\mathrm{O}(G, \lambda_{\text{in}}) = \{(\lambda_{(\text{out},h)}, p_h)\}_{h \in H}$ for the right input graph G, and take any

$$(\lambda_{(\text{out},h^\star)}, p_{h^\star}) \in \mathrm{O}(G, \lambda_{\text{in}}).$$

The probability that the outcomes of $\Lambda_1, \ldots, \Lambda_n$ give exactly $\lambda_{(\text{out},h^\star)}$ is

$$\begin{aligned}
&\Pr\left[\Lambda_1 = \lambda_{(\text{out},h^\star)}(v_1), \ldots, \Lambda_n = \lambda_{(\text{out},h^\star)}(v_n)\right] \\
&= \Pr\left[\Lambda_n = \lambda_{(\text{out},h^\star)}(v_n) \middle| \Lambda_1 = \lambda_{(\text{out},h^\star)}(v_1), \ldots, \Lambda_{n-1} = \lambda_{(\text{out},h^\star)}(v_{n-1})\right] \\
&\cdot \Pr\left[\Lambda_{n-1} = \lambda_{(\text{out},h^\star)}(v_{n-1}) \middle| \Lambda_1 = \lambda_{(\text{out},h^\star)}(v_1), \ldots, \Lambda_{n-2} = \lambda_{(\text{out},h^\star)}(v_{n-2})\right] \\
&\quad \ldots \\
&\cdot \Pr\left[\Lambda_1 = \lambda_{(\text{out},h^\star)}(v_1)\right] \\
&= p(\lambda_{(\text{out},h^\star)}, v_1) \cdot \ldots \cdot p(\lambda_{(\text{out},h^\star)}, v_n) \\
&= \prod_{1 \le l \le n-1} p(\lambda_{\text{out}}, v_l) \cdot \sum_{\substack{k_n\colon \lambda_{(\text{out},h^\star)} \restriction_{\{v_j\}} = \lambda_{(\text{out},k_j)} \restriction_{\{v_j\}} \\ \forall\ 1 \le j \le n}} \frac{p_{k_n}}{\prod_{1 \le l \le n-1} p(\lambda_{\text{out}}, v_l)} \\
&= \sum_{\substack{k_n\colon \lambda_{(\text{out},h^\star)} \restriction_{\{v_j\}} = \lambda_{(\text{out},k_j)} \restriction_{\{v_j\}} \\ \forall\ 1 \le j \le n}} p_{k_n}.
\end{aligned}$$

Notice that

$$\sum_{\substack{k_n\colon \lambda_{(\text{out},h^\star)} \restriction_{\{v_j\}} = \lambda_{(\text{out},k_j)} \restriction_{\{v_j\}} \\ \forall\ 1 \le j \le n}} p_{k_n} = p_{h^\star}$$

as H_n is necessarily isomorphic to G. By the hypothesis,

$$\sum_{h\colon \lambda_{(\text{out},h)} \text{ is valid for } (G,\lambda_{\text{in}})} p_h \ge 1 - \varepsilon,$$

implying that $\mathcal{A}$ succeeds with probability at least $1 - \varepsilon$.

5 BOUNDED-DEPENDENCE MODEL CAN BREAK SYMMETRY

For any graph $G = (V, E)$, a *random process* (or *distribution*) *on the vertices* of G is a family of random variables $\{X_v\}_{v \in V}$, indexed by V, while a *random process on the edges* of G is a family of random variables $\{X_e\}_{e \in E}$ indexed by E. More generally, a random process on G is a family of random variables $\{X_y\}_{y \in V \cup E}$ indexed by $V \cup E$. The variables of a random process live in the same

probability space and take values in some label set Σ. In general, we will consider random processes over vertices of graphs unless otherwise specified.

We now introduce the notion of T-dependent distribution. To do so, we extend the definition of distance to edges as follows: For any two edges $e = \{v_1, v_2\}, e' = \{u_1, u_2\} \in E$, $\text{dist}_G(e, e') = \min_{i,j\in[2]} \text{dist}_G(v_i, u_j)$. Similarly, the distance between any edge $e = (v_1, v_2)$ and a vertex v is $\text{dist}_G(e, v) = \min_{i\in[2]} \text{dist}_G(v_i, v)$. The definition extends easily to subsets containing vertices and edges.

Definition 5.1 (T-dependent distribution). Let $T \in \mathbb{N}$ be a natural number and $G = (V, E)$ be any graph. A random process $\{X_v\}_{v\in V}$ on the vertices G is said to be a T-dependent distribution if, for all subsets $S, S' \subseteq V$ such that $\text{dist}_G(S, S') > T$, the two processes $\{X_v\}_{v\in S}$ and $\{X_v\}_{v\in S'}$ are independent. Analogous definitions hold for random processes on the edges of a graph and for random processes on the whole graph.

A way to define T-dependent distributions that uses the same notation of Section 4.1 is by describing the output probability of global labelings: Let I be a discrete set of indices, and $G = (V, E)$ some graph. A distribution $\{(\lambda_i, p_i)\}_{i\in I}$ over output labelings, where $\lambda_i : V \to \Sigma$ is an output labeling, is *T-dependent* if the following holds: for all output labelings λ in $\{(\lambda_i, p_i)\}$, for every two subsets of nodes $S_1, S_2 \subseteq V(G)$ such that $\text{dist}_G(S_1, S_2) > T$, we have that

$$p(\lambda, S_1 \cup S_2) = p(\lambda, S_1) \cdot p(\lambda, S_2).$$

Notice that outcomes, as defined in Definition 4.1, output a random process for every input. Often, we have a family of graphs of arbitrarily large size n on which a $T(n)$-dependent distribution is defined.[1]

Now we define the hypothetical computational model that outputs T-dependent distributions on some input graph.

The bounded-dependence model. The bounded-dependence model is a computational model that, for a given family of graphs $\mathcal{F}$, produces an outcome (as defined in Definition 4.1) over $\mathcal{F}$ that is non-signaling beyond distance $T = T(G, x)$ (as defined in Definition 4.2), where $G \in \mathcal{F}$ and x represents the input, which, in turn, produces $T(G, x)$-dependent distributions. The random processes produced by an outcome are said to be *finitely-dependent* if $T = O(1)$ for all graphs in $\mathcal{F}$ and all input data. If an outcome with the aforementioned properties solves a problem Π over a graph family $\mathcal{F}$ with probability at least p, we say that the pair $(\Pi, \mathcal{F})$ has complexity T (with success probability p), and T is also called the *locality* or the *complexity* of the corresponding output distributions.

We are particularly interested in T-dependent distributions that satisfy invariance properties: We say that a random process $\{X_v\}_{v\in V}$ over vertices of a graph $G = (V, E)$ is *invariant under automorphisms* if, for all automorphisms $f : V \to V$ of $G = (V, E)$, the two processes $\{X_v\}_{v\in V}$ and $\{Y_v = X_{f(v)}\}_{v\in V}$ are equal in law. The definition is easily extendable to random processes on edges and random processes on the whole graph.

A stronger requirement is the *invariance under subgraph isomorphism*: Suppose we have an outcome $\mathsf{O} : (G, x) \mapsto \{X_v\}_{v\in V(G)}$ that maps each input graph G from family of graphs $\mathcal{F}$ and any input data x to a $T = T(G, x)$-dependent distribution over G from a family of random process $\mathcal{R}$. We say that the random process over vertices in $\mathcal{R}$ are invariant under subgraph isomorphisms if, given any two graphs $G_1, G_2 \in \mathcal{F}$ of size n_1, n_2 with associated process $\{X_v^{(1)}\}_{v\in V(G_1)}$ and $\{X_v^{(2)}\}_{v\in V(G_2)}$, and any two subgraphs $H_1 \subseteq G_1, H_2 \subseteq G_2$ such that $G_1[\mathcal{N}_{T(n_1)}(H_1)]$ and $G_2[\mathcal{N}_{T(n_2)}(H_2)]$ are

[1]Note that the dependency T might depend on other graph parameters such as the maximum degree Δ, the chromatic number χ, etc.

isomorphic (with the isomorphism that brings H_1 into H_2), then $\{X_v^{(1)}\}_{v\in V(H_1)}$ and $\{X_v^{(2)}\}_{v\in V(H_2)}$ are equal in law.[2] Trivially, invariance under subgraph isomorphisms implies invariance under automorphisms and the non-signaling property. This definition is again easily extendable to the case of families of random processes over edges or over graphs in general.

The baseline of our result is a finitely-dependent distribution provided by [41, 42] on paths and cycles.

Theorem 5.2 (Finitely-dependent coloring of the integers and of cycles [41, 42]). *Let $G = (V, E)$ be a graph that is either a cycle with at least 3 nodes or has $V = \mathbb{Z}$ and $E = \{\{i, i+1\} : i \in \mathbb{Z}\}$. For $(k, q) \in \{(1, 4), (2, 3)\}$, there exists a k-dependent distribution $\{X_v^{(G)}\}_{v\in V(G)}$ that gives a q-coloring of G. Furthermore, such distributions can be chosen to meet the following property: if $\{X_i\}_{i\in[n]}$ is the k-dependent q-coloring of the n-cycle and $\{Y_i\}_{i\in\mathbb{Z}}$ is the k-dependent q-coloring of the integers, then $\{X_i\}_{i\in[n-k]}$ is equal in law to $\{Y_i\}_{i\in[n-k]}$.*

It is immediate that the disjoint union of any number of paths and cycles (even countably many) admits such distributions as well.

Corollary 5.3. *Let $\mathcal{F}$ be the family of graphs formed by the disjoint union (possibly uncountably many) paths with countably many nodes and cycles of any finite length. For all $G \in \mathcal{F}$ and for $(k, q) \in \{(1, 4), (2, 3)\}$, there exists a k-dependent distribution $\{X_v^{(G)}\}_{v\in V(G)}$ that gives a q-coloring of G. Furthermore, such distributions can be chosen to meet all the following properties:*

1. *On each connected component $H \subseteq G$, $\{X_v^{(G)}\}_{v\in V(H)}$ is given by Theorem 5.2.*
2. *$\{\{X_v^{(G)}\}_{v\in V(G)}\}_{G\in\mathcal{F}}$ is invariant under subgraph isomorphisms.*

We will use this result to provide "fake local identifiers" to the nodes of the graph to simulate SLOCAL through a random process with constant dependency. In order to do that, we introduce some results on the composition of T-dependent distributions.

Trivially, every T-round (deterministic and randomized) port-numbering algorithm defines a $2T$-dependent distribution over the input graph. Furthermore, if the underlying port-numbering model has access to random bits, the distribution can be made invariant under subgraph isomorphisms (provided that, whenever a distribution over input labelings is given together with the input graph, such distribution is also invariant under subgraph isomorphisms). The composition of the T_2-dependent distribution obtained by a T_2-round port-numbering algorithm and any T_1-dependent distribution yields a $(2T_2 + T_1)$-dependent distribution.

Lemma 5.4. *Let $\Sigma^{(1)}$ and $\Sigma^{(2)}$ be two label sets with countably many labels. Let $\mathcal{F}$ be any family of graphs, and let $\mathcal{R} = \{\{X_v\}_{v\in V(G)} : G \in \mathcal{F}\}$ be a family of T_1-dependent distributions taking values in $\Sigma^{(1)}$, where T_1 might depend on parameters of G. Consider any T_2-round port-numbering algorithm $\mathcal{A}$ that takes as an input $G \in \mathcal{F}$ labelled by $\{X_v\}_{v\in V(G)}$: it defines another distribution $\{Y_v\}_{v\in V(G)}$ taking values in some label set $\Sigma^{(2)}$. Then, $\{Y_v\}_{v\in V(G)}$ is a $(2T_2 + T_1)$-dependent distribution on G. Furthermore, the following property holds:*

1. *If the processes in $\mathcal{R}$ are invariant under subgraph isomorphisms, T_2 is constant, $\mathcal{A}$ does not depend on the size of the input graph and permutes port numbers locally u.a.r. at round 0, then the processes in $\{\{Y_v\}_{v\in V(G)} : G \in \mathcal{F}\}$ are invariant under subgraph isomorphisms.*

Proof. Fix any $G = (V, E) \in \mathcal{F}$, and the corresponding T_2-dependent distribution $\{X_v\}_{v\in V} \in \mathcal{R}$. Fix any two subsets $S, S' \subseteq V$ such that $\mathrm{dist}_G(S, S') > 2T_2 + T_1$. Consider any output labeling $\lambda^{(2)} : V \to \Sigma^{(2)}$. Then,

$$\Pr\left[\cap_{v\in S\cup S'}\{Y_v = \lambda^{(2)}(v)\}\right]$$

[2] We remark that, as opposed to Definition 4.2, the isomorphism must preserve the input but *not* the node identifiers.

$$
\begin{aligned}
&= \sum_{\lambda^{(1)}\,:\,V\to\Sigma^{(1)}} \Pr\left[\cap_{v\in S\cup S'}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in V}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in V}\{X_v=\lambda^{(1)}(v)\}\right] \\
&= \sum_{\lambda^{(1)}\,:\,V\to\Sigma^{(1)}} \Pr\left[\cap_{v\in S\cup S'}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S\cup S')}\{X_v=\lambda^{(1)}(v)\}\right] \qquad (3)\\
&\quad\cdot \Pr\left[\cap_{v\in V}\{X_v=\lambda^{(1)}(v)\}\right] \\
&= \sum_{\lambda^{(1)}\,:\,\mathcal{N}_{T_2}(S\cup S')\to\Sigma^{(1)}} \Pr\left[\cap_{v\in S\cup S'}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S\cup S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in \mathcal{N}_{T_2}(S\cup S')}\{X_v=\lambda^{(1)}(v)\}\right],
\end{aligned}
$$

where Eq. (3) holds because the output of $\{Y_v\}_{v\in S\cup S'}$ is independent of $\{X_v\}_{v\notin\mathcal{N}_{T_2}(S\cup S')}$. Since $\mathrm{dist}_G(S,S')>2T_2$,

$$
\begin{aligned}
&\sum_{\lambda^{(1)}\,:\,\mathcal{N}_{T_2}(S\cup S')\to\Sigma^{(1)}} \Pr\left[\cap_{v\in S\cup S'}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S\cup S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in \mathcal{N}_{T_2}(S\cup S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&= \sum_{\lambda^{(1)}\,:\,\mathcal{N}_{T_2}(S\cup S')\to\Sigma^{(1)}} \Pr\left[\cap_{v\in S}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S\cup S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in S'}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S\cup S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in \mathcal{N}_{T_2}(S\cup S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&= \sum_{\lambda^{(1)}\,:\,\mathcal{N}_{T_2}(S\cup S')\to\Sigma^{(1)}} \Pr\left[\cap_{v\in S}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S)}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in S'}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in \mathcal{N}_{T_2}(S\cup S')}\{X_v=\lambda^{(1)}(v)\}\right].
\end{aligned}
$$

Now observe that $\mathrm{dist}_G(\mathcal{N}_{T_2}(S),\mathcal{N}_{T_2}(S'))>T_1$. Since $\{X_v\}_{v\in V}$ is a T_1-dependent distribution, it holds that

$$
\begin{aligned}
&\sum_{\lambda^{(1)}\,:\,\mathcal{N}_{T_2}(S\cup S')\to\Sigma^{(1)}} \Pr\left[\cap_{v\in S}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S)}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in S'}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in \mathcal{N}_{T_2}(S\cup S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&= \sum_{\lambda^{(1)}\,:\,\mathcal{N}_{T_2}(S\cup S')\to\Sigma^{(1)}} \Pr\left[\cap_{v\in S}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S)}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in S'}\{Y_v=\lambda^{(2)}(v)\}\middle|\cap_{v\in \mathcal{N}_{T_2}(S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&\quad\cdot \Pr\left[\cap_{v\in \mathcal{N}_{T_2}(S)}\{X_v=\lambda^{(1)}(v)\}\right]\Pr\left[\cap_{v\in \mathcal{N}_{T_2}(S')}\{X_v=\lambda^{(1)}(v)\}\right] \\
&= \Pr\left[\cap_{v\in S}\{Y_v=\lambda^{(2)}(v)\}\right]\cdot\Pr\left[\cap_{v\in S'}\{Y_v=\lambda^{(2)}(v)\}\right].
\end{aligned}
$$

Now, given any $G,H\in\mathcal{F}$ of size n_G and n_H, respectively, consider any subgraph isomorphism $f:\mathcal{N}_{2T_2+T_1(n_G)}(G',G)\to\mathcal{N}_{2T_2+T_1(n_H)}(H',H)$ for any two $G'\subseteq G, H'\subseteq H$ such that f restricted to G' is

an isomorphism to H'. Let $\{X_v^{(G)}\}_{v\in V(G)}$ and $\{X_v^{(H)}\}_{v\in V(H)}$, and $\{Y_v^{(G)}\}_{v\in V(G)}$ and $\{Y_v^{(H)}\}_{v\in V(H)}$ be the corresponding distributions of interest before and after the combination with the port-numbering algorithm, respectively. Fix any subset of nodes $U \subseteq V(G')$ and consider any family of labels $\{\lambda_u\}_{u\in U}$ indexed by U. In order to prove property 1, it is sufficient to show that

$$\Pr\left[\cap_{u\in U}\{Y_u^{(G)} = \lambda_u\}\right] = \Pr\left[\cap_{u\in U}\{Y_{f(u)}^{(H)} = \lambda_u\}\right].$$

Notice that $\Pr[\cap_{u\in U}\{Y_u = \lambda_u\}]$ depends solely on the graph $G[\mathcal{N}_{T_2}(U,G)] = \cup_{u\in U} G[\mathcal{N}_{T_2}(u,G)]$, the distribution of the port numbers in $G[\mathcal{N}_{T_2}(U,G)]$, and the random process $\{X_u\}_{u\in\mathcal{N}_{T_2}(U)}$. By hypothesis, $\{X_u^{(G)}\}_{u\in\mathcal{N}_{T_2}(U,G)}$ is equal in law to $\{X_{f(u)}^{(H)}\}_{u\in\mathcal{N}_{T_2}(U,G)}$. Furthermore, the restriction of f to $G[\mathcal{N}_{T_2}(U,G)]$ defines an isomorphism from $G[\mathcal{N}_{T_2}(U,G)]$ to $H[\mathcal{N}_{T_2}(f(U),H)]$, hence the distribution of the port numbers in $G[\mathcal{N}_{T_2}(U,G)]$ is the same as that in $H[\mathcal{N}_{T_2}(f(U),H)]$ because each node permutes the port numbers locally u.a.r. Thus, $\Pr\left[\cap_{u\in U}\{Y_u^{(G)} = \lambda_u\}\right]$ must be the same as $\Pr\left[\cap_{u\in U}\{Y_{f(u)}^{(H)} = \lambda_u\}\right]$. □

T-dependent distributions over different graphs can be combined to obtain a T-dependent distribution over the graph union.

Lemma 5.5. *Let $\{X_v\}_{v\in V_1}$ and $\{Y_v\}_{v\in V_2}$ be a T_1-dependent distribution over a graph $G_1 = (V_1, E_1)$ taking values in $\Sigma^{(1)}$ and a T_2-dependent distribution over a graph $G_2 = (V_2, E_2)$ taking values in $\Sigma^{(2)}$, respectively. Assume $\{X_v\}_{v\in V_1}$ and $\{Y_v\}_{v\in V_2}$ to be independent processes. Consider the graph $H = (V = V_1 \cup V_2, E = E_1 \cup E_2)$ and a distribution $\{Z_v\}_{v\in V}$ over H taking values in $\Sigma = (\Sigma^{(1)} \cup \{0\}) \times (\Sigma^{(2)} \cup \{0\})$ defined by*

$$Z_v = \begin{cases} (X_v, 0) & \text{if } v \in V_1 \setminus V_2, \\ (X_v, Y_v) & \text{if } v \in V_1 \cap V_2, \\ (0, Y_v) & \text{if } v \in V_2 \setminus V_1. \end{cases}$$

Then, $\{Z_v\}_{v\in V}$ is $\max(T_1, T_2)$-dependent.

Proof. For any vector $\mathbf{v} \in \Sigma_1 \times \cdots \times \Sigma_n$, we write $\mathbf{v}[i]$ to denote its i-th entry. Fix any output labeling $\lambda : V \to \Sigma$ such that $\lambda(v)[2] = 0$ for all $v \in V_1 \setminus V_2$ and $\lambda(v)[1] = 0$ for all $v \in V_2 \setminus V_1$. Observe that for any subset $S \subseteq V$ it holds that

$$\begin{aligned} &\Pr\left[\cap_{v\in S}\{Z_v = \lambda(v)\}\right] \\ &= \Pr\left[\left(\cap_{v\in S\cap V_1}\{X_v = \lambda(v)[1]\}\right) \cap \left(\cap_{v\in S\cap V_2}\{Y_v = \lambda(v)[2]\}\right)\right] \\ &= \Pr\left[\cap_{v\in S\cap V_1}\{X_v = \lambda(v)[1]\}\right] \cdot \Pr\left[\cap_{v\in S\cap V_2}\{Y_v = \lambda(v)[2]\}\right], \end{aligned} \tag{4}$$

where the latter equality follows by independence between $\{X_v\}_{v\ in V_1}$ and $\{Y_v\}_{v\ in V_2}$. W.l.o.g., suppose $T_1 \geq T_2$. Consider two subsets $S, S' \subseteq V$ such that $\mathrm{dist}_G(S, S') > T_1$. Fix any output labeling $\lambda : V \to \Sigma$ such that $\lambda(v)[2] = 0$ for all $v \in V_1 \setminus V_2$ and $\lambda(v)[1] = 0$ for all $v \in V_2 \setminus V_1$. Using Eq. (4), we have that

$$\begin{aligned} &\Pr\left[\cap_{v\in S\cup S'}\{Z_v = \lambda(v)\}\right] \\ &= \Pr\left[\cap_{v\in (S\cup S')\cap V_1}\{X_v = \lambda(v)[1]\}\right] \cdot \Pr\left[\cap_{v\in (S\cup S')\cap V_2}\{Y_v = \lambda(v)[2]\}\right] \\ &= \Pr\left[\cap_{v\in (S\cap V_1)\cup (S'\cap V_1)}\{X_v = \lambda(v)[1]\}\right] \cdot \Pr\left[\cap_{v\in (S\cap V_2)\cup (S'\cap V_2)}\{Y_v = \lambda(v)[2]\}\right] \\ &= \Pr\left[\cap_{S\cap V_1}\{X_v = \lambda(v)[1]\}\right] \cdot \Pr\left[\cap_{S'\cap V_1}\{X_v = \lambda(v)[1]\}\right] \\ &\quad \cdot \Pr\left[\cap_{S\cap V_2}\{Y_v = \lambda(v)[2]\}\right] \cdot \Pr\left[\cap_{S'\cap V_2}\{Y_v = \lambda(v)[2]\}\right] \end{aligned} \tag{5}$$

$$
\begin{aligned}
&= \Pr\left[\left(\cap_{v\in S\cap V_1}\{X_v = \lambda(v)[1]\}\right) \cap \left(\cap_{v\in S\cap V_2}\{Y_v = \lambda(v)[2]\}\right)\right] \qquad (6)\\
&\quad \cdot \Pr\left[\left(\cap_{v\in S'\cap V_1}\{X_v = \lambda(v)[1]\}\right) \cap \left(\cap_{v\in S'\cap V_2}\{Y_v = \lambda(v)[2]\}\right)\right]\\
&= \Pr\left[\cap_{v\in S}\{Z_v = \lambda(v)\}\right] \cdot \Pr\left[\cap_{v\in S'}\{Z_v = \lambda(v)\}\right],
\end{aligned}
$$

where Eq. (5) holds because $\{X_v\}_{v\in V_1}$ is T_1-dependent and $\{Y_v\}_{v\in V_2}$ is T_2-dependent, while Eq. (6) holds because $\{X_v\}_{v\in V_1}$ and $\{Y_v\}_{v\in V_2}$ are independent. □

Now we present a final lemma on the composition of random processes. To do so, we first introduce the notation of *random decomposition* of a graph.

Definition 5.6 (Random decomposition). Let $G = (V, E)$ be any graph and $\mathcal{P}$ a family of subgraphs of G. For any $k \in \mathbb{N}$, let $\Gamma(G)$ be a random variable taking values in $\mathcal{P}^k$ that is sampled according to any probability distribution. We say that $\Gamma(G)$ is a *random k-decomposition* of G in $\mathcal{P}$.

Given a random k-decomposition $\Gamma(G)$ of G in $\mathcal{P}$, for any $y \in V \cup E$, we define the random variable $\Gamma(G)_y \in \{0,1\}^k$ as follows: $\Gamma(G)_y[i] = 1$ if y belongs to $\Gamma(G)[i]$ and 0 otherwise. Notice that $\{\Gamma(G)_y\}_{y\in V\cup E}$ is a random process on G. If $\{\Gamma(G)_y\}_{y\in V\cup E}$ is invariant under automorphisms, then we say that the random k-decomposition $\Gamma(G)$ is invariant under automorphisms. If, for a family of graphs $\mathcal{F}$ and any graph $G \in \mathcal{F}$, $\{\Gamma(G)_y\}_{y\in V(G)\cup E(G)}$ is T-dependent (with T being a function of the size of G) and the processes in $\{\{\Gamma(G)_y\}_{y\in V(G)\cup E(G)} : G \in \mathcal{F}\}$ are invariant under subgraph isomorphisms, then we say that the random k-decompositions in $\{\Gamma(G) : G \in \mathcal{F}\}$ are T-dependent and invariant under subgraph isomorphisms.

For a random decomposition, we define the notion of induced random process.

Definition 5.7 (Induced process). Let $G = (V, E)$ be a graph that admits a family of subgraphs $\mathcal{P}$. Suppose there exists a random process $\{X_v\}_{v\in V(H(\mathcal{P}))}$ with $H(\mathcal{P})$ being the graph obtained by the disjoint union of all elements of $\mathcal{P}$. Let $\Gamma(G)$ be a random k-decomposition of G in $\mathcal{P}$. For all $v \in V$ and $G' \in \mathcal{P}$, define the random process $\{X_v^{(G')}\}_{v\in V}$ by setting $X_v^{(G')} = X_{f_{G'}(v)}$ for all $v \in V(G')$, where $f_{G'} : V(G') \to V(H(\mathcal{P}))$ is the natural immersion of G' into $H(\mathcal{P})$ otherwise set $X_v^{(G')} = 0$. Let $\{Y_v^{(\Gamma(G))}\}_{v\in V}$ be a random process that we define conditional on the output of $\Gamma(G)$: for all $\mathbf{G} \in \mathcal{P}^k$, conditional on $\Gamma(G) = \mathbf{G}$,

$$Y_v^{(\Gamma(G))} = Y_v^{(\mathbf{G})} = (X_v^{(\mathbf{G}[1])}, \ldots, X_v^{(\mathbf{G}[k])}).$$

The random process $\{Y_v^{(\Gamma(G))}\}_{v\in V}$ on G is said to be induced by the action of $\{X_v\}_{v\in V(H(\mathcal{P}))}$ over the random k-decomposition $\Gamma(G)$.

Now we present a result on the induced random process whenever the random decomposition and the family of random processes that acts on the random decomposition meet some invariance properties.

Lemma 5.8. *Let $\mathcal{F}$ be a family of graphs. For any $G = (V, E) \in \mathcal{F}$, let $\mathcal{P}_G$ be any family of subgraphs of G that is closed under node removal and disjoint graph union, that is, if $G_1, G_2 \in \mathcal{P}_G$ and $V(G_1) \cap V(G_2) = \emptyset$, then $G_1 \cup G_2 \in \mathcal{P}_G$. Furthermore, suppose that, for each pair of isomorphic subgraphs $G_1, G_2 \subseteq, G_1 \in \mathcal{P}_G \implies G_2 \in \mathcal{P}_G$. Moreover, let $\Gamma(G)$ be a random k-decomposition of G in $\mathcal{P}_G$ that is T_1-dependent, and suppose that the random decompositions in $\{\Gamma(G) : G \in \mathcal{F}\}$ are invariant under subgraph isomorphism. Suppose there is T_2-dependent distribution $\{X_v\}_{v\in V(H(\mathcal{P}_G))}$, taking values in a finite set Σ, such that $\{\{X_v\}_{v\in V(H(\mathcal{P}_G))} : G \in \mathcal{F}\}$ is invariant under subgraph isomorphism, where $H(\mathcal{P}_G)$ is obtained by the disjoint union of a copy of each element of $\mathcal{P}_G$. Let $\{Y_v^{(\Gamma(G))}\}_{v\in V(G)}$ be the random process induced by the action of $\{X_v\}_{v\in V(H(\mathcal{P}_G))}$ over $\Gamma(G)$. Then, $\{Y_v^{(\Gamma(G))}\}_{v\in V(G)}$*

is a $(T_1 + 2T_2)$-dependent distribution. Furthermore, the processes in $\{\{X_v\}_{v\in V(H(\mathcal{P}_G))} : G \in \mathcal{F}\}$ are invariant under subgraph isomorphism.

Proof. Fix $G \in \mathcal{F}$ and let $\Gamma = \Gamma(G)$, $\mathcal{P} = \mathcal{P}_G$. Since $\mathcal{P}^k$ might be uncountable, we consider the density function f_Γ and the probability measure $\mathbb{P}$ associated to Γ. Consider two subsets of nodes $S, S' \subseteq V$ at distance at least $\max(T_1, T_2) + 1$. Fix any labeling $\lambda : V \to \Sigma^k$. It holds that

$$\begin{aligned}
&\Pr\left[\cap_{v\in S\cup S'}\{Y_v^{(\Gamma)} = \lambda(v)\}\right] \\
&= \int_{\mathcal{P}^k} \Pr\left[\cap_{v\in S\cup S'}\{Y_v^{(\Gamma)} = \lambda(v)\} \,\middle|\, \Gamma = \mathbf{G}\right] f_\Gamma(\mathbf{G})\,\mathrm{d}\mathbb{P} && (7)\\
&= \int_{\mathcal{P}^k} \Pr\left[\cap_{v\in S\cup S'}\{Y_v^{(\mathbf{G})} = \lambda(v)\}\right] f_\Gamma(\mathbf{G})\,\mathrm{d}\mathbb{P} \\
&= \int_{\mathcal{P}^k} \Pr\left[\cap_{v\in S}\{Y_v^{\mathbf{G}} = \lambda(v)\}\right] \Pr\left[\cap_{v\in S'}\{Y^{\mathbf{G}} = \lambda(v)\}\right] f_\Gamma(\mathbf{G})\,\mathrm{d}\mathbb{P} && (8)\\
&= \int_{\mathcal{P}^k} \prod_{i\in[k]} \Pr\left[\cap_{v\in S}\{X_v^{\mathbf{G}[i]} = \lambda(v)[i]\}\right] \Pr\left[\cap_{v\in S'}\{X_v^{\mathbf{G}[i]} = \lambda(v)[i]\}\right] f_\Gamma(\mathbf{G})\,\mathrm{d}\mathbb{P}, && (9)
\end{aligned}$$

where Eq. (7) holds by the law of total probability, Eq. (8) holds since $\{Y_v^{(\mathbf{G})}\}_{v\in V}$ is T_2-dependent for all $\mathbf{G} \in \mathcal{P}^k$ by Lemma 5.5, and Eq. (9) holds since $\{X_v^{\mathbf{G}[i]}\}_{v\in V}$ and $\{X_v^{\mathbf{G}[j]}\}_{v\in V}$ are independent if $i \neq j$. Notice that, for any set $S \subseteq V$, $G' = \mathbf{G}[i][\mathcal{N}_{T_2}(S, \mathbf{G}[i])] \in \mathcal{P}$ and the event $\cap_{v\in S}\{X_v^{\mathbf{G}[i]} = \lambda(v)[i]\}$ has the same probability as $\cap_{v\in S}\{X_v^{G'} = \lambda(v)[i]\}$ because $\{X_v\}_{v\in V(H(\mathcal{P}))}$ is invariant under subgraph isomorphisms.

For any subset $U \subseteq V$ and any integer $T \geq 0$, let

$$\mathcal{H}(T, U) = \left\{(\mathbf{G}[1][\mathcal{N}_T(U, \mathbf{G}[1])], \ldots, \mathbf{G}[k][\mathcal{N}_T(U, \mathbf{G}[k])]) \,\middle|\, \mathbf{G} \in \mathcal{P}^k\right\} \subseteq \mathcal{P}^k,$$

and let $\mathbb{P}[\mathcal{H}(T, U)]$ be the restriction of $\mathbb{P}$ to $\mathcal{H}(T, U)$ defined as follows: for any event E,

$$\mathbb{P}[\mathcal{H}(T, U)](E) = \mathbb{P}(E \cap \mathcal{H}(T, U))/\mathbb{P}(\mathcal{H}(T, U)).$$

Finally, define the random variable $\Gamma^{(T,U)}$ to be a k-dimensional vector, taking values in $\mathcal{H}(T, U)$, whose i-th entry is $\Gamma[i][\mathcal{N}_T(U, \Gamma[i])]$, and let $f_{\Gamma^{(T,U)}}$ be its density function. Define $U = S \cup S'$: Eq. (9) becomes

$$\begin{aligned}
&\int_{\mathcal{P}^k} \prod_{i\in[k]} \Pr\left[\cap_{v\in S}\{X_v^{\mathbf{G}[i]} = \lambda(v)[i]\}\right] \Pr\left[\cap_{v\in S'}\{X_v^{\mathbf{G}[i]} = \lambda(v)[i]\}\right] f_\Gamma(\mathbf{G})\,\mathrm{d}\mathbb{P} && (10)\\
&= \int_{\mathcal{H}(T_2,U)} \prod_{i\in[k]} \Pr\left[\cap_{v\in S}\{X_v^{\mathbf{H}[i]} = \lambda(v)[i]\}\right] \Pr\left[\cap_{v\in S'}\{X_v^{\mathbf{H}[i]} = \lambda(v)[i]\}\right] f_{\Gamma^{(T_2,U)}}(\mathbf{H})\,\mathrm{d}\mathbb{P}[\mathcal{H}(T_2, U)]\,.
\end{aligned}$$

Now notice that, since the random decomposition Γ is T_1-dependent and $\mathrm{dist}_G(S, S') > T_1 + 2T_2$, the probability space $\mathcal{H}(T_2, U)$ (with measure $\mathbb{P}[\mathcal{H}(T_2, U)]$) is isomorphic to the product space $\mathcal{H}(T_2, S) \times \mathcal{H}(T_2, S')$ (with product measure $\mathbb{P}[\mathcal{H}(T_2, S)] \times \mathbb{P}[\mathcal{H}(T_2, S')]$). The isomorphism brings any element

$$(\mathbf{G}[1][\mathcal{N}_{T_2}(U, \mathbf{G}[1])], \ldots, \mathbf{G}[k][\mathcal{N}_{T_2}(U, \mathbf{G}[k])])$$

of $\mathcal{H}(T_2, U)$ into

$$\left((\mathbf{G}[1][\mathcal{N}_{T_2}(S, \mathbf{G}[1])], \ldots, \mathbf{G}[k][\mathcal{N}_{T_2}(S, \mathbf{G}[k])]), (\mathbf{G}[1][\mathcal{N}_{T_2}(S', \mathbf{G}[1])], \ldots, \mathbf{G}[k][\mathcal{N}_{T_2}(S', \mathbf{G}[k])])\right)$$

which belongs to $\mathcal{H}(T_2, S) \times \mathcal{H}(T_2, S')$. Hence, we get

$$\begin{aligned}
&\int_{\mathcal{H}(T_2,U)} \prod_{i\in[k]} \Pr\left[\cap_{v\in S}\{X_v^{\mathbf{H}[i]} = \lambda(v)[i]\}\right] \Pr\left[\cap_{v\in S'}\{X_v^{\mathbf{H}[i]} = \lambda(v)[i]\}\right] f_{\Gamma(T_2,U)}(\mathbf{H})\, \mathrm{d}\mathbb{P}[\mathcal{H}(T_2,S)] \\
&= \int_{\mathcal{H}(T_2,S)} \int_{\mathcal{H}(T_2,S')} \prod_{i\in[k]} \Pr\left[\cap_{v\in S}\{X_v^{\mathbf{H}[i]} = \lambda(v)[i]\}\right] \Pr\left[\cap_{v\in S'}\{X_v^{\mathbf{H}[i]} = \lambda(v)[i]\}\right] \\
&\quad \cdot f_{\Gamma(T_2,S)}(\mathbf{H}_1) f_{\Gamma(T_2,S')}(\mathbf{H}_2)\, \mathrm{d}\mathbb{P}[\mathcal{H}(T_2,S)]\, \mathrm{d}\mathbb{P}[\mathcal{H}(T_2,S')]\,.
\end{aligned}$$

The latter becomes

$$\begin{aligned}
&\int_{\mathcal{H}(T_2,S)} \int_{\mathcal{H}(T_2,S')} \prod_{i\in[k]} \Pr\left[\cap_{v\in S}\{X_v^{\mathbf{H}[i]} = \lambda(v)[i]\}\right] \Pr\left[\cap_{v\in S'}\{X_v^{\mathbf{H}[i]} = \lambda(v)[i]\}\right] \\
&\quad \cdot f_{\Gamma(T_2,S)}(\mathbf{H}_1) f_{\Gamma(T_2,S')}(\mathbf{H}_2)\, \mathrm{d}\mathbb{P}[\mathcal{H}(T_2,S)]\, \mathrm{d}\mathbb{P}[\mathcal{H}(T_2,S')] \\
&= \int_{\mathcal{H}(T_2,S)} \prod_{i\in[k]} \Pr\left[\cap_{v\in S}\{X_v^{\mathbf{H}[i]} = \lambda(v)[i]\}\right] f_{\Gamma(T_2,S)}(\mathbf{H}_1)\, \mathrm{d}\mathbb{P}[\mathcal{H}(T_2,S)] \\
&\quad \cdot \int_{\mathcal{H}(T_2,S')} \prod_{i\in[k]} \Pr\left[\cap_{v\in S'}\{X_v^{\mathbf{H}[i]} = \lambda(v)[i]\}\right] f_{\Gamma(T_2,S')}(\mathbf{H}_2)\, \mathrm{d}\mathbb{P}[\mathcal{H}(T_2,S')] \\
&= \int_{\mathcal{P}^k} \prod_{i\in[k]} \Pr\left[\cap_{v\in S}\{X_v^{\mathbf{G}[i]} = \lambda(v)[i]\}\right] f_\Gamma(\mathbf{G})\, \mathrm{d}\mathbb{P} \qquad (11) \\
&\quad \cdot \int_{\mathcal{P}^k} \prod_{i\in[k]} \Pr\left[\cap_{v\in S'}\{X_v^{\mathbf{G}[i]} = \lambda(v)[i]\}\right] f_\Gamma(\mathbf{G})\, \mathrm{d}\mathbb{P} \\
&= \Pr\left[\cap_{v\in S}\{Y_v^{(\Gamma(G))} = \lambda(v)\}\right] \Pr\left[\cap_{v\in S'}\{Y_v^{(\Gamma(G))} = \lambda(v)\}\right],
\end{aligned}$$

where Eq. (11) follows by the same reasoning as for Eq. (10) but in reverse.

Now we want to prove that $\{\{Y_v^{(\Gamma(G))}\}_{v\in V} : G \in \mathcal{F}\}$ is invariant under subgraph isomorphism. Fix any two graphs $G, H \in \mathcal{F}$ of sizes n_G and n_H, respectively. Consider any isomorphism α between the radius-$(T_1(n_G) + 2T_2(n_G))$ neighborhoods of any subgraph $G' \subseteq G$ and the radius-$(T_1(n_H)+2T_2(n_H))$ neighborhoods of any subgraph $H' \subseteq H$, such that the restriction of α to G' is an isomorphism to H'. With an abuse of notation, for any subgraph $K \subseteq G'$, let us denote $\alpha(K) \subseteq H'$ its isomorphic image in H' through α. Let $T_G = T_1(n_G) + 2T_2(n_G)$ and $T_H = T_1(n_H) + 2T_2(n_H)$. Fix any labeling $\lambda : V \to \Sigma^k$. We have that

$$\begin{aligned}
&\Pr\left[\cap_{v\in V(G')}\{Y_v^{(\Gamma(G))} = \lambda(v)\}\right] \\
&= \int_{\mathcal{P}_G^k} \Pr\left[\cap_{v\in V(G')}\{Y_v^{(\Gamma(G))} = \lambda(v)\} \,\middle|\, \Gamma(G) = \mathbf{G}\right] f_{\Gamma(G)}(\mathbf{G})\, \mathrm{d}\mathbb{P}_G\,. \\
&= \int_{\mathcal{P}_G^k} \Pr\left[\cap_{v\in V(G')}\{Y_v^{(\mathbf{G})} = \lambda(v)\}\right] f_{\Gamma(G)}(\mathbf{G})\, \mathrm{d}\mathbb{P}_G \\
&= \int_{\mathcal{P}_G^k} \prod_{i\in[k]} \Pr\left[\cap_{v\in V(G')}\{X_v^{(\mathbf{G}[i])} = \lambda(v)[i]\}\right] f_{\Gamma(G)}(\mathbf{G})\, \mathrm{d}\mathbb{P}_G \\
&= \int_{\mathcal{H}_G(T_G,V(G'))} \prod_{i\in[k]} \Pr\left[\cap_{v\in V(G')}\{X_v^{(\mathbf{H}[i])} = \lambda(v)[i]\}\right] f_{\Gamma(G)^{(T_G,V(G'))}}(\mathbf{H})\, \mathrm{d}\mathbb{P}_G(\mathcal{H}_G(T_G, V(G')))
\end{aligned}$$

$$= \int_{\mathcal{H}_G(T_G,V(G'))} \prod_{i\in[k]} \Pr\left[\cap_{v\in V(G')}\{X_{\alpha(v)}^{(\alpha(\mathbf{H}[i]))} = \lambda(v)[i]\}\right] f_{\Gamma(G)^{(T_G,V(G'))}}(\mathbf{H})\, \mathrm{d}\mathbb{P}_G(\mathcal{H}_G(T_G,V(G'))),$$

where the latter holds because

$$\Pr\left[\cap_{v\in V(G')}\{X_v^{(\mathbf{H}[i])} = \lambda(v)[i]\}\right] = \Pr\left[\cap_{v\in V(G')}\{X_{\alpha(v)}^{(\alpha(\mathbf{H}[i]))} = \lambda(v)[i]\}\right]$$

as $\{\{X_v\}_{v\in V(H(\mathcal{P}_G))} : G \in \mathcal{F}\}$ is T_2-dependent and invariant under subgraph isomorphism. Furthermore, since the processes in $\{\Gamma(G) : G \in \mathcal{F}\}$ are T_1-dependent and invariant under subgraph isomorphism, we have that $f_{\Gamma(G)^{(T_G,V(G'))}}(\mathbf{H}) = f_{\Gamma(H)^{(T_H,V(H'))}}((\alpha(\mathbf{H}[1]),\dots,\alpha(\mathbf{H}[k])))$ almost everywhere in $\mathcal{H}_G(T_G,V(G'))$, and that the probability space $\mathcal{H}_G(T_G,V(G'))$ with measure $\mathrm{d}\mathbb{P}_G(\mathcal{H}_G(T_G,V(G')))$ is isomorphic to $\mathcal{H}_H(T_H,V(H'))$ with measure $\mathrm{d}\mathbb{P}_H(\mathcal{H}_H(T_H,V(H')))$, where the isomorphism brings $\mathbf{H}$ into $(\alpha(\mathbf{H}[1]),\dots,\alpha(\mathbf{H}[k]))$. Hence,

$$\begin{aligned}
&\int_{\mathcal{H}_G(T_G,V(G'))} \prod_{i\in[k]} \Pr\left[\cap_{v\in V(G')}\{X_{\alpha(v)}^{(\alpha(\mathbf{H}[i]))} = \lambda(v)[i]\}\right] f_{\Gamma(G)^{(T_G,V(G'))}}(\mathbf{H})\, \mathrm{d}\mathbb{P}_G(\mathcal{H}_G(T_G,V(G'))) \\
&= \int_{\mathcal{H}_H(T_H,V(H'))} \prod_{i\in[k]} \Pr\left[\cap_{v\in V(H')}\{X_v^{(\mathbf{H}[i])} = \lambda(v)[i]\}\right] f_{\Gamma(H)^{(T_H,V(H'))}}(\mathbf{H})\, \mathrm{d}\mathbb{P}_H(\mathcal{H}_H(T_H,V(H'))) \\
&= \Pr\left[\cap_{v\in V(H')}\{Y_v^{(\Gamma(H))} = \lambda(v)\}\right],
\end{aligned}$$

concluding the proof. □

Remark 5.9. When the underlying graph is a directed graph, Lemma 5.8 guarantees invariance under subgraph isomorphisms that keep edge orientation.

We are going to work on rooted pseudotrees and pseudoforests. A *pseudotree* is a graph that is connected and contains at most one cycle. A *pseudoforest* is a graph obtained by the disjoint union of pseudotrees; an equivalent definition of pseudoforest is a graph in which each connected component has no more edges than vertices. Note that a pseudotree might contain multiple edges: however, we assume it does not contain self-loops as self-loops are useless communication links in the LOCAL model. A *rooted tree* is a tree where each edge is oriented and all nodes have outdegree at most 1: it follows that all but one node have outdegree exactly 1 and one node (the *root*) has outdegree 0. Trivially, a tree can be rooted by selecting one node and orienting all edges towards it. A *rooted pseudotree* is a pseudotree where each edge is oriented and each node has outdegree at most 1: if the pseudotree contains a cycle, then all nodes necessarily have outdegree exactly 1. Any pseudotree can be oriented so that it becomes rooted: just orient the cycle first (if it exists) in a consistent way, then remove it, and make the remaining trees rooted at nodes that belonged to the cycle. A *rooted pseudoforest* is the union of rooted pseudotrees.

We will show that pseudoforests of maximum degree Δ admit a $O(\log^* \Delta)$-dependent 3-coloring distributions. In order to do so, we use a color reduction technique that follows by known revisions of the Cole–Vishkin technique [28, 39].

Lemma 5.10 (port-numbering algorithm for color reduction in pseudoforests [28, 39]). *Let G be a pseudoforest with countably many nodes. Assume G is given as an input a k-coloring for some $k \geq 3$. There exists a deterministic port-numbering algorithm that does not depend on the size of G and outputs a 3-coloring of G in time $O(\log^* k)$.*

Now we are ready to prove our result on pseudoforests.

Lemma 5.11 (Finitely-dependent coloring of rooted pseudoforests). *Let $\mathcal{F}$ be a family of rooted pseudoforests of countably many nodes of maximum degree Δ. Then, there exists an outcome*

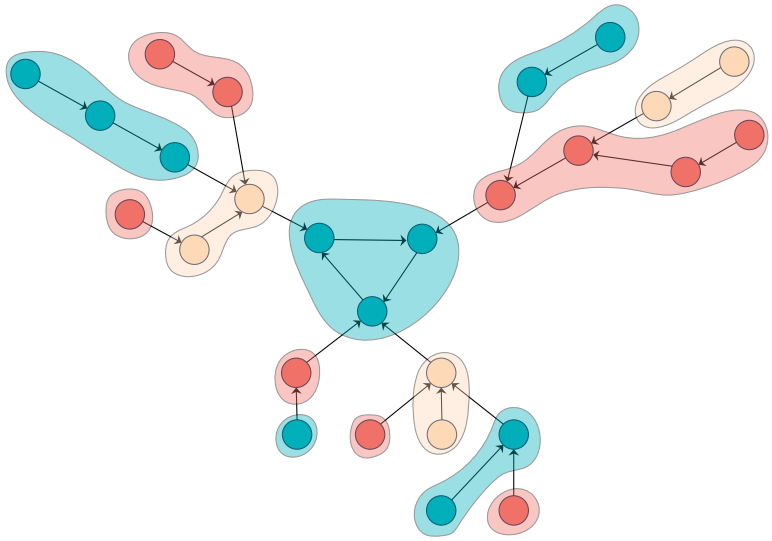
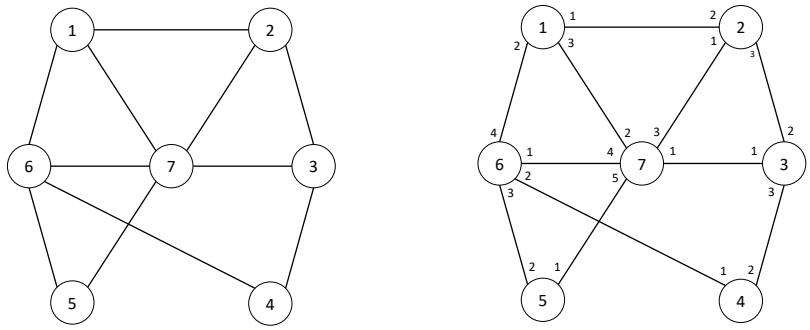


Fig. 2. A decomposition of rooted pseudoforests in directed paths and cycles: each node v colors its indegree neighbors with a uniformly sampled permutation of the elements in $[\text{indeg}(v)]$. The graph induced by nodes colored with color i is a disjoint union of directed paths and cycles.

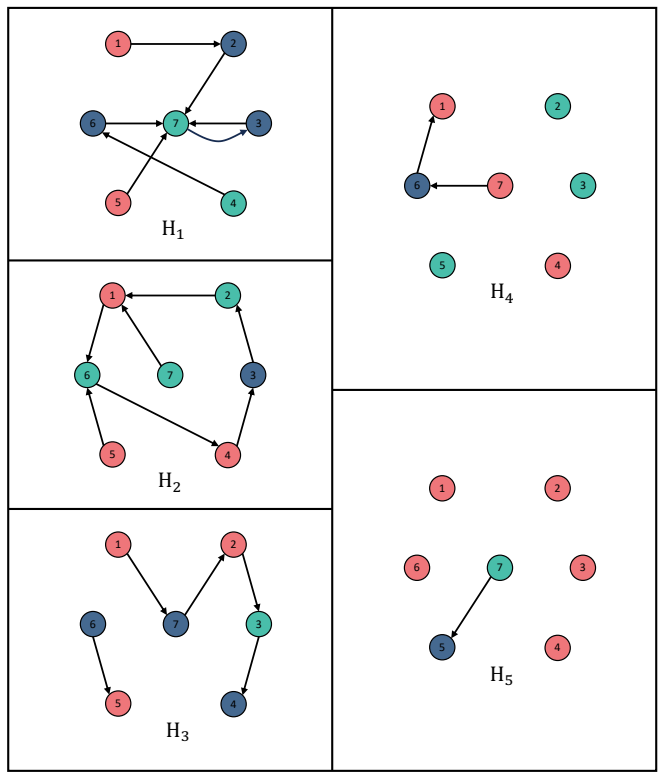
that associates to each graph $G \in \mathcal{F}$ a $O(\log^* \Delta)$-dependent distribution on the vertices of G that gives a 3-coloring of G. Furthermore, the family of distributions outputted by the outcome are invariant under subgraph isomorphisms.

Proof. Let us fix the rooted pseudoforest $G \in \mathcal{F}$. Let $\mathcal{P}_G$ be the family of all subgraphs of G formed by the disjoint union of directed paths and cycles. Notice that $\mathcal{P}_G$ is closed under node removal and disjoint graph union. Furthermore, for any two pair of isomorphic subgraphs $G_1, G_2 \subseteq G$, $G_1 \in \mathcal{P}_G \implies G_2 \in \mathcal{P}_G$. Let $H(\mathcal{P}_G)$ be the graph formed by the disjoint union of a copy of each element of $\mathcal{P}_G$. By Corollary 5.3, $H(\mathcal{P}_G)$ admits a 1-dependent 4-coloring distribution $\{X_v\}_{v \in H(\mathcal{P}_G)}$ (with colors in $[4]$), such that $\{\{X_v\}_{v \in H(\mathcal{P}_G)} : G \in \mathcal{F}\}$ is invariant under subgraph isomorphism.

Consider now a (non-proper) coloring of the pseudoforest in which each node u colors its indegree neighbors with a permutation of $\{1, \ldots, \text{indeg}(u)\}$ sampled uniformly at random: if a node has outdegree zero, then it is deterministically colored with color 1. Such a coloring is described by a 2-dependent distribution $\{Z_v\}_{v \in V}$ that is (trivially) invariant under subgraph isomorphisms that keep edge orientation. Also, $\{Z_v\}_{v \in V}$ identifies Δ_{in} disjoint random subset of nodes $\mathsf{V}_1, \ldots, \mathsf{V}_{\Delta_{\text{in}}}$, where nodes in V_i are colored with the color i, and Δ_{in} is the maximum indegree of the graph. Let $\Gamma(G) = (G[\mathsf{V}_1], \ldots, G[\mathsf{V}_{\Delta_{\text{in}}}])$, where $\Gamma(G)[i]$ is the random graph induced by V_i. Furthermore, observe that, the output of $\Gamma(G)[i]$ is the disjoint union of oriented paths and/or oriented cycles, with $\Gamma(G)[i]$ being the i-th entry of the Δ_{in}-tuple $\Gamma(G)$ (see Fig. 2). Notice that process $\Gamma(G)$ is 2-dependent and is a random Δ_{in}-decomposition of G in $\mathcal{P}_G$ (according to Definition 5.6), such that the random decompositions in $\{\Gamma(G) : G \in \mathcal{F}\}$ are invariant under subgraph isomorphism. By Lemma 5.8, the random process $\{Y_v^{(\Gamma(G))}\}_{v \in V}$, that is induced by the action of $\{X_v\}_{v \in H(\mathcal{P}_G)}$ over the random Δ_{in}-decomposition $\Gamma(G)$ (according to Definition 5.7), is a 4-dependent distribution



(a) Each node v rearranges its port numbers uniformly at random.



(b) The $\Delta = 5$ rooted pseudoforests that we take by considering port numbers $i \in [\Delta]$.

Fig. 3. A decomposition of a graph of maximum degree $\Delta = 5$ in rooted pseudoforests: for the sake of image clarity, we focus on the undirected case. In Fig. 3a, each node v rearranges its port-numbers with a uniformly sampled permutation of the elements in $[\deg(v)]$. As shown in Fig. 3b, edges hosting port number i at some endpoint are oriented away from that port (in case both endpoints host port number i, the edge is duplicated) and form a rooted pseudoforest.

that gives a $4\Delta_{\text{in}}$-coloring of G: in fact, $\Gamma(G)[i]$ and $\Gamma(G)[j]$ are disjoint if $i \neq j$, hence only one entry of $Y_v^{(\Gamma(G))}$ is non-zero, for all $v \in V$.

We then combine $\{Y_v^{(\Gamma(G))}\}_{v\in V}$ and a modified version of the port-numbering algorithm from Lemma 5.10 where at round 0 each node permutes port numbers locally u.a.r.: by Lemma 5.4, we obtain an $O(\log^* \Delta)$-dependent 3-coloring distribution $\{Q_v^{(G)}\}_{v\in V(G)}$ of G such that processes in $\{\{Q_v^{(G)}\}_{v\in V(G)} : G \in \mathcal{F}\}$ are invariant under subgraph isomorphisms. □

Our finitely-dependent coloring of pseudoforests can be used as a baseline to provide a $(\Delta+1)$-coloring of graphs with maximum degree Δ. The tool we use is again an application of the Cole–Vishkin color reduction technique [56].

Lemma 5.12 (port-numbering algorithm for color reduction of general graphs [56]). *Let $G = (V, E)$ be a graph with maximum degree Δ and countably many nodes. Suppose G is given in input a k-coloring for some $k \geq \Delta + 1$. There exists a deterministic port-numbering algorithm that does not depend on the size of the input graph and outputs a $(\Delta+1)$-coloring of G in time $O(\log^* k + \Delta^2)$.*

Lemma 5.13 (Finitely-dependent coloring of bounded-degree graphs). *Let $\mathcal{F}$ be a family of graphs of countably many nodes and maximum degree Δ. Then, there exists an outcome that associates to each graph $G \in \mathcal{F}$ a $O(\Delta^2)$-dependent distribution on the vertices of G that gives a $(\Delta+1)$-coloring of G. Furthermore, the family of distributions outputted by the outcome are invariant under subgraph isomorphisms.*

Proof. Let us fix $G = (V, E) \in \mathcal{F}$ First, if G is not directed, then duplicate each edge and give to each pair of duplicates different orientations. Since a coloring of the original graph is a proper coloring if and only if the same coloring is proper in the directed version, w.l.o.g., we can assume G to be directed.

Let $\mathcal{P}_G$ be a family of all subgraphs of G formed by rooted pseudotrees and disjoint union of rooted pseudotrees. Notice that $\mathcal{P}_G$ is closed under node removal and disjoint graph union. Furthermore, for any two pair of isomorphic subgraphs $G_1, G_2 \subseteq G$, $G_1 \in \mathcal{P}_G \implies G_2 \in \mathcal{P}_G$. The graph $H(\mathcal{P}_G)$ that is the disjoint union of all elements of copies of each $\mathcal{P}_G$ is a rooted pseudoforest of maximum degree Δ and, by Lemma 5.11, admits a 3-coloring $O(\log^* \Delta)$-dependent distribution $\{X_v\}_{v\in H(\mathcal{P}_G)}$ such that processes in $\{\{X_v\}_{v\in H(\mathcal{P}_G)} : G \in \mathcal{F}\}$ are invariant under subgraph isomorphism.

Now, consider a process in which each node v samples uniformly at random a permutation of a port numbering from $\{1, \ldots, \text{outdeg}(v)\}$ for its outdegree edges. For each $i \in [\Delta_{\text{out}}]$, consider the graph G_i induced by edges that host port i: notice that G_i is a rooted pseudoforest as each node has at most one outdegree edge with port i. If a node has degree 0, it deterministically joins G_1, which remains a pseudoforest. The random choice of port numbering defines a random variable $\Gamma \in \mathcal{P}_G^k$, where $\Gamma[i]$ is the graph induced by port number i: according to Definition 5.6, we obtain a random Δ_{out} -decomposition $\Gamma(G)$ of G in $\mathcal{P}_G$ which is 2-dependent (by construction): also, the random decompositions in $\{\Gamma(G) : G \in \mathcal{F}\}$ are invariant under subgraph isomorphisms. For an example of a possible output of the random decomposition, see Fig. 3.

Hence, the random process $\{Y_v^{(\Gamma(G))}\}_{v\in V}$ from Definition 5.7 is well defined, and provides a proper 3^Δ-coloring of G. By Lemma 5.8, the random process $\{Y_v^{(\Gamma(G))}\}_{v\in V}$ is $O(\log^*)$-dependent and, when $G \in \mathcal{F}$ varies, the processes $\{Y_v^{(\Gamma(G))}\}_{v\in V}$ are invariant under subgraph isomorphisms.

We then combine $\{Y_v^{(\Gamma(G))}\}_{v\in V}$ and a modified version of the port-numbering algorithm from Lemma 5.12 where at round 0 each node permutes port numbers locally u.a.r.: by Lemma 5.4, we obtain an $O(\Delta^2)$-dependent $(\Delta+1)$-coloring distribution of G that is invariant under subgraph isomorphisms (as $G \in \mathcal{F}$ varies). □

Consider any graph $G = (V, E)$. For any $k \in \mathbb{N}_+$, a distance-k coloring of G is an assignment of colors $c : V \to \Sigma$ such that, for each node $v \in V$, all nodes in $\mathcal{N}_k(v) \setminus \{v\} = \{u \in V : \text{dist}_G(u, v) \leq k\} \setminus \{v\}$ have colors that are different from $c(v)$.

It is well known that any LCL problem Π that has complexity $O(\log^* n)$ in the LOCAL model has the following property: there exists a constant $k \in \mathbb{N}_+$ (that depends only on the hidden constant in $O(\log^* n)$) such that, if the input graph is given a distance-k coloring, then Π is solvable in time $O(1)$ in the port-numbering model [18]. Furthermore, no knowledge of the size of the input graph is required.

Finding a distance-k coloring in SLOCAL trivially requires locality exactly k, implying the well known fact that LCLs of complexity $O(\log^* n)$ in the LOCAL model have complexity $O(1)$ in SLOCAL. In SLOCAL is indeed easier to address the case of infinite graphs for such problems.

Theorem 5.14. *Consider any LCL problem Π with checking radius r that has complexity $T \geq r$ in the* SLOCAL *model over a family $\mathcal{F}$ of graphs with maximum degree Δ, where $T = O(1)$ is a constant. For each $G \in \mathcal{F}$, there exists an $O(T^2\Delta^{2T})$-dependent distribution $\{Y_v^{(G)}\}_{v \in V(G)}$ that solves Π over G. Furthermore, the processes in $\{\{Y_v^{(G)}\}_{v \in V(G)} : G \in \mathcal{F}\}$ are invariant under subgraph isomorphisms.*

Proof. Fix $G \in \mathcal{F}$ of size n. Consider the power graph G^k, where $k \geq T$, defined by $G^k = \{V, E^k\}$ with $E^k = \{\{u, v\} : u, v \in V, \text{dist}_G(u, v) \leq k\}$. The maximum degree of G^k is Δ^k. By Lemma 5.13, G^k admits an $O(\Delta^{2k})$-dependent $(\Delta^k + 1)$-coloring distribution $\{X_v\}_{v \in V}$ that is invariant under subgraph isomorphisms (as $G \in \mathcal{F}$ varies). Notice that such a coloring provides a distance-k coloring for G.

Let $\mathcal{A}$ be the algorithm in SLOCAL that solves Π in time T. Consider a port-numbering algorithm $\mathcal{A}'$ that simulates $\mathcal{A}$ and is defined as follows: At round 0, $\mathcal{A}'$ permutes ports locally u.a.r. Then, nodes colored with color i perform T rounds of communication from round $(i-1)T+1$ to round iT and output a label as if they were the i-th node shown by the adversary to the SLOCAL algorithm and had identifier i: all other communications are useless for such nodes. Notice that, for each $v \in V(G)$, identifiers in $\mathcal{N}_T(v)$ are unique and belong to $\{1, \ldots, \Delta^k\}$. The running time of $\mathcal{A}'$ is kT. Now we prove that the $\mathcal{A}'$ outputs a correct labeling by induction on the colors of the nodes.

Let us focus on nodes colored with color 1 and, by contradiction, suppose one of such nodes (node u) fails: a failure means that $G[\mathcal{N}_r(u)]$ contains a non-valid labeling. Notice that no node other than u outputs anything in $G[\mathcal{N}_r(u)]$, as nodes in $\mathcal{N}_r(u) \setminus \{u\}$ have color greater than 1. Then $\mathcal{A}$ fails in G where the adversary picks u as the first node in G, reaching a contradiction.

Suppose $c > 1$ and that nodes with color $i < c$ output a correct label. Let us focus on nodes with color c and assume, by contradiction, that one of such nodes (node u) fails: a failure means that $G[\mathcal{N}_r(u)]$ contains a non-valid labeling. Notice that in $G[\mathcal{N}_r(u)]$ only nodes with colors strictly smaller than c have already outputted a label, and such a label is correct by the inductive hypothesis. Furthermore, colors in $\mathcal{N}_r(u)$ are all different as $r \leq k$. Then $\mathcal{A}$ fails in G where the adversary picks nodes in $\mathcal{N}_k(u)$ according to the order given by the coloring, and then continues outside $\mathcal{N}_k(u)$ following any arbitrary order, reaching a contradiction.

Notice that $\{Y_v\}_{v \in V}$, the random process induced by combining $\{X_v\}_{v \in V}$ and $\mathcal{A}'$ is a $O(kT\Delta^{2k})$-dependent distribution on G by Lemma 5.4 with the required invariance properties, and we get the thesis by choosing $k = T$. □

Theorem 5.14 answers an open question formulated by Holroyd [40].

Corollary 5.15. *Let $G = (V, E)$ be the infinite d-regular trees. There exists a finitely-dependent distribution giving a $(d+1)$-coloring of G that is invariant under automorphisms.*

Proof. Finding a $(d+1)$-coloring of any d-regular trees has complexity $O(1)$ in SLOCAL: the coloring can just be performed greedily. Theorem 5.14 yields the desired result. □

6 SIMULATION OF ONLINE-LOCAL IN SLOCAL FOR TREES

In this section, we show how to turn an online-LOCAL algorithm solving an LCL problem in forests into a deterministic SLOCAL algorithm solving the same problem. More concretely, we prove the following two theorems:

Theorem 6.1. *Let Π be an LCL problem with degree constraint Δ, input label set Σ_{in} and output label set Σ_{out}. Let $\mathcal{A}$ be a deterministic online-*LOCAL *algorithm solving Π with locality $T(n)$ on forests. Then there exists an* SLOCAL *algorithm solving Π with locality $T(O(2^{n^4}))$.*

Theorem 6.2. *Let Π be an LCL problem with degree constraint Δ, input label set Σ_{in} and output label set Σ_{out}. Let $\mathcal{A}$ be a randomized online-*LOCAL *algorithm solving Π with locality $T(n)$ on forests. Then there exists an* SLOCAL *algorithm solving Π with locality $T(O((2|\Sigma_{out}|)^{2^{2^n}}))$.*

Theorem 1.5 follows as a simple corollary of Theorem 6.2, along with the fact that an algorithm with locality $o(\log\log n)$ in the SLOCAL model implies the existence of a locality $o(\log n)$ algorithm in the LOCAL model [34].

6.1 Amnesiac algorithms

We start by formalizing what an online-LOCAL algorithm sees when run for a fixed number of steps on a graph. We then define formally what we mean by (a, C)-amnesiac algorithms:

*Definition 6.3 (Partial online-*LOCAL *run of length ℓ).* Let G be a graph with an ordering of nodes $v_1, v_2, \ldots, v_n$. Consider the subgraph $G_\ell \subseteq G$ induced by the radius-T neighborhoods of the first ℓ nodes $v_1, \ldots, v_\ell$. We call $(G_\ell, (v_1, \ldots, v_\ell))$ the *partial online-*LOCAL *run of length ℓ of G* as this is exactly the information that an online-LOCAL algorithm would know about G when deciding the output for node v_ℓ. We denote by G_ℓ^- the partial online-LOCAL run formed by $v_1, \ldots, v_{\ell-1}$, that is, one step shorter than G_ℓ.

If G_ℓ contains only one connected component, we call $(G_\ell, (v_1, \ldots, v_\ell))$ a *partial one-component online-*LOCAL *run of length ℓ of G*.

We denote by $\mathcal{G}_\ell$ the set of all partial one-component online-LOCAL runs of length ℓ.

Definition 6.4 (Amnesiac algorithm). Let $\mathcal{A}$ be an online-LOCAL algorithm, and let $a, C \in \mathbb{N}$ be constants.

Let v be a node such that the connected component of nodes that $\mathcal{A}$ has seen around v corresponds to some partial online-LOCAL run $(G_a, (v_1, \ldots, v_a))$. We say that algorithm $\mathcal{A}$ is (a, C)*-amnesiac* if the output of $\mathcal{A}$ for node v depends only on this local connected component and not anything else the algorithm has seen. For technical reasons, we relax this a little by allowing the output of algorithm $\mathcal{A}$ depend on anything after it has seen C identical neighborhoods.

Intuitively, algorithm $\mathcal{A}$ is (a, C)-amnesiac if its output for the components in which it has labeled at most $a-1$ nodes depends only on the component and nothing else. In a sense, the algorithm *forgets* about all other components it has seen. After C isomorphic components, the output can have arbitrary dependencies on the input.

We say that every online-LOCAL algorithm is $(0, C)$-amnesiac for every $C \in \mathbb{N}$.

6.2 Online to amnesiac

We are now ready to generalize the intuition we provided in Section 2.3 for turning online-LOCAL algorithm to $(1, C)$-amnesiac algorithms into a way to turn any online-LOCAL algorithm into an

(a, C)-amnesiac algorithm. We later show how this can be used to construct an SLOCAL algorithm in the proof of Theorem 6.1:

Lemma 6.5. *Let Π be an LCL problem with degree constraint Δ, input label set Σ_{in} and output label set Σ_{out}. Let $\mathcal{A}$ be an online-*LOCAL *algorithm solving Π with locality $T(n)$ on n-node forests. Then for every $a, C \in \mathbb{N}$ there exists an (a, C)-amnesiac online-*LOCAL *algorithm $\mathcal{A}'$ that Π on n-node forests with locality $T(C2^{O(n^2 a)})$.*

Proof. Let $T = T(N)$ be the locality of the following experiment; we determine the size N of the experiment graph later. Let $\mathcal{G}_\ell$ be the set of all partial one-component online-LOCAL runs of length ℓ for n-node forests. We denote $g_\ell = |\mathcal{G}_\ell|$ and remark that $g_\ell \leq g = 2^{n^2}|\Sigma_{\text{in}}|^n$. We set $N_\ell = (3|\Sigma_{\text{out}}|ng)^{a-\ell+1}C$.

We now adaptively construct an experiment graph H in a phases. In the first phase, we construct N_1 copies of $\mathcal{G}_1$, that is we construct $N_1 g_1$ disjoint components describing all possible isolated neighborhoods an online-LOCAL algorithm might see. We then show the centers of these regions to $\mathcal{A}$ and let it label them. As there are N_ℓ copies of each neighborhood, for each neighborhood $\mathcal{T}$ there exists a label $\sigma_{\mathcal{T}}$ that appears at least $N_1/|\Sigma_{\text{out}}|$ times, by the pigeonhole principle. We say that such neighborhoods are *good*, and ignore the rest.

In each subsequent phase ℓ, we again construct N_ℓ copies of $\mathcal{G}_\ell$, with the catch that for each graph $(G, (v_1, \ldots, v_\ell)) \in \mathcal{G}_\ell$, we take the components of G^- from the good graphs of previous phases. We defer arguing why such good neighborhoods exist for later. Then we show node v_ℓ to algorithm $\mathcal{A}$ and let it label the node. Finally, again by the pigeonhole principle, for each neighborhood $\mathcal{T}$ we find a canonical label $\sigma_{\mathcal{T}}$ that appears at least $N_\ell/|\Sigma_{\text{out}}|$ times, and mark those neighborhoods as good.

Now our new online-LOCAL algorithm $\mathcal{B}$ works as follows: Before even processing the first node of the input, algorithm $\mathcal{B}$ runs the above-described experiment with graph H on algorithm $\mathcal{A}$ with locality T.

When processing node v, algorithm $\mathcal{B}$ first checks whether the connected component formed by nodes seen by $\mathcal{B}$ correspond to a good, unused, neighborhood in the experiment graph H. This is the case especially when the neighborhood contains at most $a - 1$ previously-processed nodes. In this case, algorithm $\mathcal{B}$ identifies this good neighborhood with the input nodes and marks it used. Note that if the neighborhood contains previously-processed nodes, those must have corresponded to a good neighborhood, too, and hence they also must use the good, canonical label, and the identification must succeed. Otherwise, algorithm $\mathcal{B}$ shows the new node and the neighborhood around it to algorithm $\mathcal{A}$ and copies the output from $\mathcal{A}$. This construction ensures that $\mathcal{B}$ is (a, C)-amnesiac.

The only things left to do is to argue that we can always find a good neighborhood in the construction and to compute the final size of the experiment graph, and hence the locality of $\mathcal{B}$. We start with the former: Consider phase ℓ. All subsequent phases contain $\sum_{k=\ell+1}^{a} N_k g_k$ graphs, the simulation uses at most C graphs, and each of those may use at most n previously-used components. Hence, nodes from phase ℓ can be used at most

$$C + n \sum_{k=\ell+1}^{a} N_k g_k \leq C + ng \sum_{k=\ell+1}^{a} N_k \leq C + \frac{N_\ell}{2|\Sigma_{\text{out}}|}$$

Assuming that $C \ll N_\ell$, we get that

$$\frac{N_\ell}{|\Sigma_{\text{out}}|} \geq C + n \sum_{k=\ell+1}^{a} N_k g_k,$$

that is the future phases won't run out of components.

Finally, the size of the experiment graph is

$$N = n \sum_{\ell=1}^{a} N_\ell g_\ell \leq N_0 = C(3|\Sigma_{\text{out}}|gn)^{a+1} = C2^{O(n^2 a)}. \qquad \square$$

6.3 Deterministic online-LOCAL

We can now prove Theorem 6.1:

Proof. Let Π be an LCL problem with degree constraint Δ, input label set Σ_{in} and output label set Σ_{out}. Let $\mathcal{A}$ be a deterministic online-LOCAL algorithm solving Π with locality $T(n)$ on forests.

We can apply Lemma 6.5 to get an (n, n)-amnesiac algorithm $\mathcal{A}'$ that solves Π with locality $T = T(n2^{O(n^3)}) = T(O(2^{n^4}))$.

We can now turn $\mathcal{A}'$ into an SLOCAL algorithm $\mathcal{B}$ with locality $2T$ as follows: Algorithm $\mathcal{B}$ stores in each node everything it knows. When processing node v, algorithm $\mathcal{B}$ gets to see a connected component of the graph, possibly with some nodes that have been previously processed. Now $\mathcal{B}$ simulates algorithm $\mathcal{A}'$ locally on this component in the same order. As $\mathcal{A}'$ is (n, n)-amnesiac, its output depends only on the structure and processing order of nodes in this component. Moreover, as $\mathcal{A}'$ solves problem Π correctly, the produced output must be locally correct. As Π is an LCL problem, this also ensures that the solution is globally correct.

The factor 2 in the locality comes from the fact that an online-LOCAL algorithm with locality T that sees two nodes within distance $2T$ sees how they connect while an SLOCAL algorithm requires locality $2T$. $\square$

6.4 Randomized online-LOCAL

We now modify the construction from the proof of Lemma 6.5 to prove Theorem 6.2:

Proof. We start by considering what would happen if we used the construction from the proof of Lemma 6.5 with a randomized online-LOCAL algorithm? The obstacle turns out to be the fact that the adversary needs to be oblivious while the previous construction was highly adaptive: only $1/|\Sigma_{\text{out}}| = q'$ fraction of components in each phase are good, and we cannot adaptively choose to use only those.

To combat this, instead of adaptively choosing only the good components and discarding the rest, we sample the components from the previous phases uniformly at random without replacement. We later show that the sampled component is good with probability at least $q = q'/2$. In the worst case, each component may use up to n graph from the previous layers, each of which is sampled to be good with probability only q. As the construction is n levels deep, and this needs to succeed for all up to $g = 2^{n^2}|\Sigma_{\text{in}}|^n$ different graphs, the total probability of finding a good label in each layer is at least $q^{\Omega(gn^2)}$.

This probability is minuscule, but we can boost it higher by repeating the construction k times. As each of the constructions is independent of other, they succeed independently and the probability that all of them fail is $(1 - q^{O(gn^2)})^k$. If we can make this probability lower than $1 - 1/n$, the success probability of the randomized online-LOCAL algorithm, then the algorithm must succeed to label at least one good construction correctly. In particular, this guarantees that there exists a good label for each neighborhood which the deterministic SLOCAL algorithm can use to label the neighborhood.

We can now do the following approximation:

$$(1 - q^{O(gn^2)})^k \leq e^{-kq^{O(gn^2)}} \leq e^{-kq^{2^{n^2} n^2}} \leq e^{-kq^{2^{2^n}}} \leq 1 - \frac{1}{n}.$$

Taking natural logarithm on both sides gives us

$$-kq^{2^{2^n}} \leq \ln(1-1/n) \leq -1/n$$

and reorganizing then gives us

$$k \geq \frac{1}{nq^{2^{2^n}}}.$$

Clearly this holds if

$$k \geq \frac{1}{q^{2^{2^n}}} = (2|\Sigma_{\text{out}}|)^{2^{2^n}}$$

holds. We can now pick $k = (2|\Sigma_{\text{out}}|)^{2^{2^n}}$ to satisfy this equation. Hence, the size of our final experiment is

$$N = kO(2^{n^4}) = O((2|\Sigma_{\text{out}}|)^{2^{2^n}}).$$

Now this experiment succeeds with positive probability. In particular, there exists a run where at least one of the constructions succeed to find a good labeling for each component in each layer. We take such run and use the labeling for that component as the base for our SLOCAL algorithm. We now proceed as in the proof of Theorem 6.1 and construct an SLOCAL algorithm that has locality $T(O((2|\Sigma_{\text{out}}|)^{2^{2^n}}))$.

Finally, we show that the probability of sampling a good component is indeed at least $q = 1/(2|\Sigma_{\text{out}}|)$. Consider components in phase ℓ. For each component type, there are at least $N_\ell/|\Sigma_{\text{out}}|$ that are good. By calculations in proof of Lemma 6.5, future layers will use at most $N_\ell/(2|\Sigma_{\text{out}}|)$ of them. Hence, after each step, at least $1/(2|\Sigma_{\text{out}}|) = q$ fraction of the components are still good and unused. □

7 LOWER BOUND FOR 3-COLORING GRIDS IN RANDOMIZED ONLINE-LOCAL

In this section, we will show that 3-coloring $(\sqrt{n}\times\sqrt{n})$-grids in randomized online-LOCAL requires $\Omega(\log n)$-locality:

Theorem 7.1. *The locality of a randomized online-*LOCAL *algorithm for solving* 3*-coloring in* $(\sqrt{n}\times\sqrt{n})$*-grids is* $\Omega(\log n)$.

We will base our proof on the recent result of Chang et al. [20], where the authors show a similar lower bound for the deterministic online-LOCAL model. We will start with a brief overview of the techniques used in [20].

7.1 Relevant ideas from Chang et al. [20]

The lower bound in [20] is based on the idea that any coloring of a grid G with the three colors $\{1, 2, 3\}$ can be partitioned into *regions* with colors 1 and 2 that are separated by *boundaries* of color 3. Formally, a region is a maximal connected component that is colored only with colors 1 and 2, while a boundary is a maximal connected component in G^2 that is colored only with color 3. It is crucial to observe that the boundaries of color 3 are not always compatible and that they have parities of their own (see Fig. 4). Indeed, the core idea of the proof is to constrict many incompatible boundaries of color 3 to a small space, thus requiring a view of $\Omega(\log n)$ to resolve them.

Using the same terminology as [20], we count incompatible boundaries between two points by using so-called *a- and b-values*. The *a*-value is defined as an edge weight between any two nodes and captures the change of colors 1 and 2.

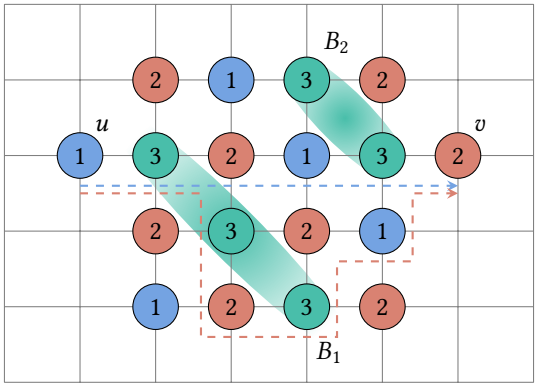


Fig. 4. Example of a 3-colored grid with two boundaries B_1 and B_2 that have different parities. As one can see, $b(u, v) = 2$: No matter if we take the blue or the red path from u to v, we always cross B_1 and then B_2.

Definition 7.2 (a-value [20]). Given a directed edge (u, v) and a 3-coloring of the nodes $c : V \mapsto \{1, 2, 3\}$, we define

$$a(u, v) = \begin{cases} c(u) - c(v), & \text{if } c(u) \neq 3 \text{ and } c(v) \neq 3 \\ 0, & \text{otherwise.} \end{cases}$$

Observe that the a-value of any directed 4-cycle in a grid is equal to 0. Using the a-value, we define the b-value of a path.

Definition 7.3 (b-value [20]). For a directed path P, its b-value is defined as

$$b(P) = \sum_{(u,v) \in H} a(u, v).$$

On a high level, the b-value of a path describes the cumulative total of incompatible boundaries along this path. Note we say "cumulative" because in this count boundaries of the same parity cancel each other out. Indeed, if we consider the b-value of a simple directed cycle in a grid, then it must be equal to 0:

Lemma 7.4 (b-value of a cycle is zero [20]). *Let C be a directed cycle in G. Then $b(C) = 0$.*

As an example, observe that a path that starts and ends with color 3 and otherwise is colored with colors 1 and 2 always has a b-value of 0 or 1. In particular, the b-value is 1 if the distance between the nodes of color 3 is even (and thus the boundaries are incompatible). Meanwhile, a path that goes through two incompatible regions has a total b-value of 2. Nevertheless, a path going through two boundaries that are compatible has a total b-value of 0.

Lemma 7.5 (Parity of the b-value [20]). *Let P denote any directed path of length ℓ that starts in node u and ends in node v in a grid. Then, the parity of $b(P)$ is*

$$b(P) \equiv \beta(u) + \beta(v) + \ell \pmod 2$$

where β is an indicator variable stating whether a node is of color 3 or not:

$$\beta(u) = \begin{cases} 1, & \text{if } c(u) = 3 \\ 0, & \text{otherwise.} \end{cases}$$

Observe that the parity of the b-value of a path is determined by the colors of the endpoints of this path.

7.2 From deterministic to randomized online-LOCAL

Suppose that we are given a randomized online-LOCAL algorithm with visibility radius $T = o(\log n)$. Our goal is to prove the algorithm fails to solve the 3-coloring problem. Using Yao's minimax principle, we show how to construct an (oblivious) adversarial distribution of inputs so that any deterministic online-LOCAL algorithm fails with noticeable probability.

As in [20], the lower bound consists of two main steps:

(1) First we show how to force paths to have an arbitrarily large b-value. Generally we cannot force a large b-value between the path's endpoints (or in fact between any two fixed nodes), but we do get the guarantee that it contains two nodes v_1 and v_2 where $b(v_1, v_2)$ is large. Note the location of v_1 and v_2 is completely unknown to us since we are working with an oblivious adversary. The construction is inductive: Given a procedure that generates a path with b-value $\geq k-1$ with probability $\geq 1/2$, we show how to generate a path with b-value $\geq k$ also with probability $\geq 1/2$. (Cf. the construction in [20] with an adaptive adversary, which succeeds every time.) Since we invoke the construction for $k-1$ a constant number of times, we are able to obtain any desired b-value of $k = o(\log n)$ with high probability. Note it is logical that we cannot do much better than this as it would contradict the existing $O(\log n)$ deterministic online-LOCAL algorithm.

(2) The second step is to actually obtain a contradiction. (See Fig. 5.) First we construct a path P_1 with large b-value, say $\gg 4T$, between nodes u_s and v_t. To get the contradiction, we wish to place two nodes w_s and w_t next to u_s and v_t, respectively, so that the four node cycle has positive b-value (which is impossible due to Lemma 7.4). If we had an adaptive adversary as in [20], this would be relatively simple: Since the adaptive adversary knows the location of u_s and v_t, it just constructs an arbitrary path P_2 with the same size as P_1, picks any two w_s and w_t that are at the same distance from each other as u_s and v_t, mirrors P_2 if needed to obtain a non-negative b-value, and then places this at minimal distance to P_1. We show that, allowing for some failure probability, we do not need any information about u_s and v_t (i.e., nor their location nor the distance between them) in order to obtain the same contradiction.

We now proceed with the proof as outlined above. Accordingly, the first step is the following:

Lemma 7.6. *Given any $k = o(\log n)$, there is an adversarial strategy to construct a directed path of length $n^{o(1)}$ with a b-value of at least k against any online-*LOCAL *algorithm with locality $T = o(\log n)$. Moreover, this strategy succeeds with high probability.*

Proof. Consider the following recursive construction:

- If $k = 0$, create a single node with previously unrevealed nodes all around it (inside the visibility radius T).
- Otherwise, repeat the following steps four times in total:
  - Create four distinct paths P_1, P_2, P_3, P_4 by following the procedure for $k-1$.
  - Toss independent fair coins $c_1, c_2, c_3 \in \{0, 1\}$.
  - Connect the P_i's horizontally aligned and in order while placing $c_i + 1$ nodes between paths P_i and P_{i+1}.

  Let Q_1, Q_2, Q_3, Q_4 be the four paths created by this procedure (each having their own four P_i's coming from the procedure in the previous iteration $k-1$). Next, we connect the Q_i's horizontally aligned with each other and in arbitrary order. Note that we need to place an additional node between each pair of Q_i's. Otherwise, we would have to have revealed the edge between the two endpoints too early to the algorithm.

Thus, for $k \geq 1$ we have 16 invocations of the procedure for $k-1$ in total. We argue that, with probability at least $1/2$, the path yielded by this procedure contains a segment with b-value at least

k. By repeating this procedure $O(\log n)$ times independently, we obtain at least one path with the desired property with high probability.

To prove that the recursive construction works, we proceed by induction. Fix $k \geq 1$ and suppose that the procedure for $k-1$ succeeds with probability $p \geq 1/2$ at yielding a segment with b-value at least $k-1$. Let us first consider Q_1. Let X_i be a random variable that is 1 if this occurs for path P_i and 0 otherwise. Then we have

$$\begin{aligned}\Pr\left[\sum_{i=1}^{4} X_i < 2\right] &= \Pr\left[\sum_{i=1}^{4} X_i = 0\right] + \Pr\left[\sum_{i=1}^{4} X_i = 1\right] \\ &= (1-p)^4 + 4p(1-p)^3 \\ &= (1-p)^3(1+3p) \\ &\leq \frac{1}{2}.\end{aligned}$$

Hence, with probability at least $1/2$ there are at least two paths P_i and P_j, $i < j$, for which the construction succeeds.

Since we are dealing with paths, we may simplify the notation and write $b(u,v)$ for the b-value of the (unique) segment that starts at u and ends at v. Let thus u_i, v_i, u_j, v_j appear in this order in Q_1 and $|b(u_i,v_i)|, \left|b(u_j,v_j)\right| \geq k-1$. Next we will show that, conditioned on this assumption, the probability that Q_1 contains a segment with b-value at least k is at least $1/2$. Hence, *a priori*, Q_1 contains such a segment with probability at least $1/4$. Since all Q_i's are constructed the in the same way and independently of one another, the probability that at least one of the Q_i's contains such a segment is at least $1-(1-1/4)^4 > 1-1/e > 1/2$.

Let us write $\sigma(x)$ for the sign function of x (i.e., $\sigma(x) = 1$ if $x > 0$, $\sigma(x) = -1$ if $x < 0$, and $\sigma(0) = 0$). To see why the above holds for Q_1, consider the two following cases:

$\sigma(b(u_i,v_i)) = \sigma(b(u_j,v_j))$. Using Lemma 7.5, we have that $b(v_i,u_j) \equiv \beta(v_i) + \beta(u_j) + c_i + \cdots + c_{j-1} \pmod 2$. Since $\beta(v_i)$ and $\beta(u_j)$ are fixed, and the coin tosses are independent, we have $\left|b(v_i,u_j)\right| \not\equiv k-1 \pmod 2$ with probability $1/2$. Assuming this holds, we have either $\left|b(v_i,u_j)\right| \geq k$, in which case we are done, or $\left|b(v_i,u_j)\right| \leq k-2$. Since $b(u_i,v_i)$ and $b(u_j,v_j)$ have the same sign, we have

$$\begin{aligned}\left|b(u_i,v_j)\right| &= \left|b(u_i,v_i) + b(v_i,u_j) + b(u_j,v_j)\right| \\ &\geq \left|b(u_i,v_i) + b(u_j,v_j)\right| - \left|b(v_i,u_j)\right| \\ &\geq 2(k-1) - (k-2) \\ &= k.\end{aligned}$$

$\sigma(b(u_i,v_i)) \neq \sigma(b(u_j,v_j))$. Arguing by using Lemma 7.5 as before, we obtain that $\left|b(v_i,u_j)\right| \not\equiv 0 \pmod 2$ holds with probability $1/2$. Assuming this is the case, we have thus $\left|b(v_i,u_j)\right| \geq 1$. Without restriction, let $\sigma(b(u_i,v_i)) = \sigma(b(v_i,u_j))$. Then

$$\left|b(u_i,u_j)\right| = \left|b(u_i,v_i) + b(v_i,u_j)\right| \geq k-1+1 = k.$$

Finally, let us confirm that the construction fits into the $(\sqrt{n} \times \sqrt{n})$-grid. We only reveal at most $2T+1 = o(\sqrt{n})$ in a column, so we need to only consider nodes along a row. The initial path contains $m_0 = 2T+1$ visible nodes and in the i-th recursive step we have $m_i \leq 16p_{i-1} + 27$ visible nodes in total for $i \geq 1$. (Inside each Q_i we need at most $2(4-1) = 6$ additional nodes to join the four P_i's, and to join the four Q_i's we need an additional 3 nodes.) Solving the recursion, in the

k-th step we have thus

$$m_k = \frac{1}{5}\left(2^{4k+1}(5T+7) - 9\right) = n^{o(1)}$$

visible nodes along the path since $k, T = o(\log n)$. □

We now illustrate the idea for getting the contradiction previously described. (See Fig. 5.) Invoking Lemma 7.6, we obtain a path P_1 with a b-value of $4T + 4$ between two nodes u_s and v_t (whose positions are unknown to us). Letting L be the length of P_1, we (arbitrarily) create a second path P_2 of length $10L$ and then randomly choose how to align P_1 and P_2. More specifically, letting u be the first node in P_1, we choose some node w of P_2 uniformly at random and align u and w; then we mirror P_2 with probability $1/2$ and reveal it at distance $2T + 2$ to P_1 (which is consistent with all previously revealed nodes).

The reason why this works is the following: Since P_1 has length L and P_2 length $10L$, with at least $4/5$ we align the paths so that each node in P_1 has a matching node underneath it in P_2. Let w_s and w_t be the nodes matching u_s and v_t, respectively. Then because we mirror P_2 with probability $1/2$, we get that $b(w_s, \ldots, w_t)$ is at least zero in expectation (conditioned on having properly aligned the two paths). Hence, with probability at least $2/5$ we obtain a cycle $(u_s, \ldots, v_t, \ldots, w_t, \ldots, w_s, \ldots, u_s)$ where $b(u_s, \ldots, v_t) > 4T + 4$, $b(w_t, \ldots, w_s) \geq 0$, and $b(v_t, \ldots, w_t), b(w_s, \ldots, u_s) \geq -2T - 2$ (due to the distance between the two paths).

Proof of Theorem 7.1. Observe that, using Lemma 7.6, we can construct a path $P_1 = (u_0, \ldots, u_L)$ of length $L = n^{o(1)}$ where some segment $(u_s, \ldots, u_t)$ of P_1 has a b-value of $k > 4T + 4$. Next we construct a path $P_2 = (v_0, \ldots, v_{10L})$ of length $10L$ that will be placed below P_1. We assume that the points of the path are revealed to the algorithm in some predefined order.

The position and orientation of P_2 are chosen as follows:

(1) Choose a node $v_r \in [L, 9L]$ of P_2 uniformly at random. This node is placed below the node u_0 of P_1 at $2T + 2$ distance from it.
(2) Throw a fair coin $m \in \{0, 1\}$. If $m = 1$, mirror P_2 along the vertical axis that goes through u_0 and v_r.

The nodes of P_2 are revealed to the algorithm in the same predefined and possibly mirrored order. See Fig. 5 for an example.

We next prove that we obtain the desired lower bound. First notice that, since we pick $v_r \in [L, 9L]$, every node in P_1 has a counterpart in P_2 whether we mirror P_2 or not. Let $(w_s, \ldots, w_t)$ denote the segment matched to $(u_s, \ldots, u_t)$. Consider the case where either of the following is true:

- $m = 0$ and $r \in [L, 9L - s - t]$
- $m = 1$ and $r \in [L + s + t, 9L]$

Denoting by E the event in which this occurs, note we have

$$\Pr[E] = \frac{1}{2}\Pr[r \in [L, 9L - s - t]] + \frac{1}{2}\Pr[r \in [L + s + t, 9L]] = \frac{8L - s - t + 1}{9L + 1} \geq \frac{6L + 1}{9L + 1} > \frac{2}{3}.$$

Now conditioned on E, notice that every segment $(v_x, \ldots, v_y)$ where $x \in [L + s, 9L - s]$ and $y \in [L + t, 9L - t]$ has equal probability of being matched with $(u_s, \ldots, u_t)$ either in the same direction (i.e., $w_s = v_x$ and $u_t = v_y$) or reversed (i.e., $w_s = v_y$ and $w_t = v_x$). Hence,

$$\Pr[b(w_t, \ldots, w_s) \geq 0 \mid E] \geq \frac{1}{2}.$$

(In fact, the probability is exactly $1/2$ if $b(w_t, \ldots, w_s) > 0$ and 1 if $b(w_t, \ldots, w_s) = 0$.) Thus, the probability that $b(w_t, \ldots, w_s) \geq 0$ is $> (2/3) \cdot (1/2) = 1/3$.

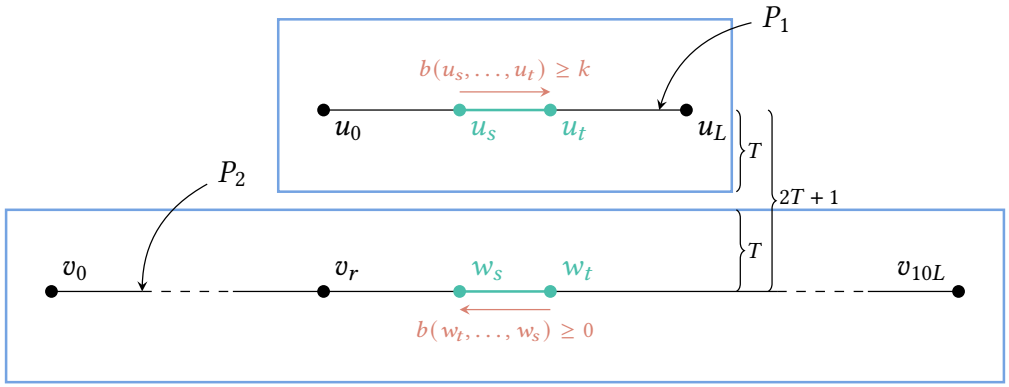


Fig. 5. How to turn paths with large b-value into a contradiction. Here the blue area includes the nodes revealed so far around the path segments $(u_0, \ldots, u_L)$ and the corresponding part of P_2 underneath it. The green segments are the two segments we consider in the proof. A cycle going through both paths leads to a contradiction.

From here on, we proceed as in [20]. By Lemma 7.4, the cycle $(u_s, \ldots, u_t, \ldots, w_t, \ldots, w_s, \ldots, u_s)$ must have a b-value of 0. However, recall that the b-value of a path is bounded by its length by definition. In our case, $-2T - 2 < b(u_t, \ldots, w_t) < 2T + 2$, $-2T - 2 < b(w_s, \ldots, u_s) < 2T + 2$, $b(u_s, \ldots, u_t) \geq k$, and $b(w_t, \ldots, w_s) \geq 0$. In order for the b-value of the cycle to be 0, we would need to have $2(2T + 2) \geq k$, which is a contradiction. Since this occurs with noticeable probability (i.e., $> 1/3$), the claim follows. □

8 DYNAMIC-LOCAL DERANDOMIZES LOCAL AND BREAKS SYMMETRY

This section presents a derandomization result for the dynamic-LOCAL model: we show that not only dynamic-LOCAL can simulate randomized LOCAL with no overhead in the locality, but actually it brings the complexity class $O(\log^* n)$ in randomized LOCAL down to $O(1)$.

We make use of the method of conditional expectations from [34] to derandomize the model of dynamic-LOCAL computation. We note that while the following theorems are proven for the class of all graphs, they can be extended to any subclass of graphs.

Theorem 8.1. *Let Π be an LCL problem, and let $\mathcal{A}$ be a randomized* LOCAL *algorithm solving Π with locality $T(n)$ and probability $p > 1 - 1/n$. Then there exists a deterministic dynamic-*LOCAL *algorithm $\mathcal{A}'$ solving Π with locality $O(T(n))$.*

Proof. The randomized LOCAL algorithm $\mathcal{A}$ can be viewed as a deterministic LOCAL algorithm which is given i.i.d. random variables R_v locally for each node v. We show that there exists a dynamic-LOCAL algorithm with locality $O(T(n))$ that assigns fixed values ρ_v for each R_v such that when $\mathcal{A}$ is run on these values as R_v, it produces a correct output for problem Π. Such dynamic-LOCAL clearly implies an $O(T(n))$-locality dynamic-LOCAL algorithm for problem Π as one can compose it with $\mathcal{A}$. The proof uses a technique similar to that of derandomizing LOCAL algorithms in SLOCAL [34].

We use the method of conditional expectation. Let F_v be a flag that is 1 when the output around node v is invalid, and 0 otherwise. Note that F_v depends only on the radius-r neighborhood of node v. Let $F = \sum_{v \in V} F_v$. As Π is an LCL problem, $F = 0$ iff the labeling is correct for the whole graph, and otherwise $F \geq 1$. By definition, $\mathbb{E}[F] = \sum_{v \in V} \mathbb{E}[F_v] \leq n(1 - p) < 1$.

Let $\sigma = e_1, e_2, \ldots, e_m$ be the order of edges in which the adversary dynamically modifies the graph, and let G_i be the graph after modification i. Note that any edge can appear multiple times in the sequence if the adversary also removes edges.

Now algorithm $\mathcal{A}'$ sets the randomness as follows: In step i, algorithm $\mathcal{A}'$ first clears ρ_u for all nodes u with $\operatorname{dist}(u, e_i) \leq T(n) + r$. Then

$$\mathbb{E}[F \mid \bigwedge_{u,\operatorname{dist}(u,e_i)>T(n)+r} R_u = \rho_u] < 1$$

as

$$\mathbb{E}[F \mid \bigwedge_{u} R_u = \rho_u] < 1$$

in the previous step. Hence, there exists some concrete values for ρ_u for $\operatorname{dist}(u, e_i) \leq T(n) + r$ such that $\mathbb{E}[F \mid \bigwedge_u R_u = \rho_u] < 1$. Algorithm $\mathcal{A}'$ sets all ρ_u to such values. Finally, algorithm $\mathcal{A}'$ simulates $\mathcal{A}$ for all nodes within radius $T(n) + r$ from e_i with randomness fixed at ρ.

This produces a correct labeling for all nodes as $\mathbb{E}[F \mid \bigwedge_u R_u = \rho_u] < 1$, and hence $\mathbb{E}[F \mid \bigwedge_u R_u = \rho_u] = 0$. □

Theorem 8.2. *Let Π be an LCL problem, and let there exist a randomized* LOCAL *algorithm solving it with locality $O(\log^* n)$. Then there exists a dynamic-*LOCAL *algorithm solving Π with locality $O(1)$.*

Proof. Using [19], a randomized LOCAL algorithm for solving Π with locality $O(\log^* n)$ implies the existence of a deterministic LOCAL algorithm solving Π with locality $O(\log^* n)$. Moreover, the deterministic LOCAL algorithm can be normalized into an algorithm that first finds a distance-k coloring with $O(\Delta^k)$ colors for some constant k, and then applies an $O(k)$-locality LOCAL algorithm $\mathcal{A}$ on this coloring.

The dynamic-LOCAL algorithm for solving Π maintains a distance-k coloring with $O(\Delta^k)$ colors on the graph, and then applies $\mathcal{A}$ on these colors. When the dynamic-LOCAL algorithm encounters an update, it first clears the colors of all nodes within distance k from the update, and greedily assigns new colors for those nodes. The greedy assignment always succeeds since each node can have at most Δ^k neighbors within distance k hence they can use at most Δ^k of the available colors. Finally, the algorithm applies $\mathcal{A}$ on all nodes of the graph. As $\mathcal{A}$ is a deterministic LOCAL algorithm with locality $O(k)$, and only the colors of nodes within distance k of the update have changed, we have that only the output of nodes within distance $O(k) = O(1)$ from the update change. □

9 RANDOMIZED ONLINE-LOCAL WITH AN ADAPTIVE ADVERSARY EQUALS DETERMINISTIC ONLINE-LOCAL

This section aims at showing that randomized online-LOCAL with an adaptive adversary is as strong as its deterministic counterpart. The proof is very similar to the proof of Theorem 8.1 and uses, again, the method of conditional expectations. Also, as in the previous section, the results can be extended to any subclass of graphs.

Theorem 9.1. *Let Π be an LCL problem, and let $\mathcal{A}$ be an online-*LOCAL *algorithm with an adaptive adversary solving Π with locality $T(n)$ and probability $p > 1 - 1/n$. Then there exists a deterministic online-*LOCAL *algorithm $\mathcal{A}'$ solving Π with locality $T(n)$.*

Proof. Let $\mathcal{A}$ be an algorithm in the online-LOCAL model with an adaptive adversary and with round complexity $T(n)$. We construct a deterministic online-LOCAL algorithm $\mathcal{A}'$ with the same round complexity $T(n)$ as follows: The algorithm $\mathcal{A}'$ simulates $\mathcal{A}$ with certain fixed random-bit-strings. In particular, whenever new nodes arrive at the input, the algorithm $\mathcal{A}'$ fixes the random bits in all newly arrived nodes in the way we describe below.

We view a run of an online-LOCAL algorithm with an adaptive adversary as a game between an algorithm $\mathcal{A}$ and an adversary. In each step of that game, first the adversary chooses a set of nodes that arrive at the input (one special node and its neighborhood). Then the algorithm $\mathcal{A}$ samples the random bits of the newly arrived nodes and fixes the label of the arrived node. After i steps of this game, we use $\mathcal{F}$ to denote the event that the adversary wins the game, that is algorithm $\mathcal{A}$ produces an invalid output according to problem Π. By definition, we have $\Pr[\mathcal{F}] \leq 1/n$. Here and from now on, the probability expressions assume that the adversary maximizes the probability of $\mathcal{F}$ happening.

Our algorithm $\mathcal{A}'$ in each i-th step fixes the random bits of the newly arrived nodes in a way that guarantees that

$$\Pr[\mathcal{F}|B_1 = b_1, \ldots, B_i = b_i] \leq \Pr[\mathcal{F}|B_1 = b_1, \ldots, B_{i-1} = b_{i-1}].$$

Here, $B_j = b_j$ stands for fixing the values of random bits of nodes that arrived in the j-th step to b_j. We note that this can always be done since, by definition, we have

$$\Pr[\mathcal{F}|B_1 = b_1, \ldots, B_{i-1} = b_{i-1}] = \mathbb{E}_{B_i}[\Pr[\mathcal{F}|B_1 = b_1, \ldots, B_i = b_i]].$$

Moreover, the online-LOCAL algorithm importantly both knows all the past arrived nodes and their randomness, and can also consider all the different choices of the adversary in the future, and hence can compute the above conditional probabilities.

At the end of the simulation of $\mathcal{A}$, we have $\Pr[\mathcal{F}|B_1 = b_1, \ldots, B_n = b_n] < 1$, and hence the bad event $\mathcal{F}$ does not occur. This implies that the algorithm $\mathcal{A}'$ is always correct. □

Remark 9.2. This proof assumes that we have white-box access to the randomized online-LOCAL algorithm and that it uses finitely many random bits. If either of these assumptions does not hold, we can use a more inefficient derandomization along the lines of [29]. Here, we just observe that an online-LOCAL algorithm working against an adaptive adversary implies the existence of a function f that maps all possible input sequences on all possible n-node graphs that can be produced by the adversary to a valid output of the problem Π. The deterministic online-LOCAL algorithm finds such a function f at the beginning of its execution and produces the output according to f.

10 COMPLEXITY OF LCL PROBLEMS IN PATHS AND CYCLES IN RANDOMIZED ONLINE-LOCAL

In this section, we show that the complexity of LCLs in paths and cycles is either $O(1)$ or $\Theta(n)$ in randomized online-LOCAL (with an oblivious adversary).

Theorem 10.1. *Let Π be an LCL problem on paths and cycles (possibly with inputs). If the locality of Π is T in the randomized online-*LOCAL *model, then its locality is $O(T + \log^* n)$ in the LOCAL model.*

This theorem is a simple corollary of the following two lemmas:

Lemma 10.2. *Let Π be an LCL problem on paths and cycles (possibly with inputs), and let $\mathcal{A}$ be a randomized online-*LOCAL *algorithm solving Π with locality $o(n)$. Then there exists a randomized online-*LOCAL *algorithm $\mathcal{A}'$ solving Π with locality $O(1)$.*

Lemma 10.3. *Let Π be an LCL problem on paths and cycles (possibly with inputs), and let $\mathcal{A}$ be a randomized online-*LOCAL *algorithm solving Π with locality $O(1)$. Then there exists a* LOCAL *algorithm $\mathcal{A}'$ solving Π with locality $O(\log^* n)$.*

By previous results [2], it is known that, in the case of paths and cycles, any (deterministic) online-LOCAL algorithm with sublinear locality can be sped up to an online-LOCAL algorithm

with constant locality. We show how to extend [2, Lemma 5.5]. We start by constructing a large virtual graph P' such that, when the original algorithm runs on the virtual graph P', the labeling produced by the algorithm is locally compatible with the labeling in the original graph P.

Proof of Lemma 10.2. Let Π be an LCL problem in paths or cycles with checking-radius r, let P be a path or a cycle with n nodes, and let $\mathcal{A}$ be a randomized online-LOCAL algorithm solving Π with locality $T(n) = o(n)$. In the proof of [2, Lemma 5.5], the authors use three phases to speed up (deterministic) online-LOCAL with sublinear locality to only constant locality. We use the same strategy to construct a randomized online-LOCAL algorithm $\mathcal{A}'$ for solving Π with constant locality. The phases for constructing randomized online-LOCAL algorithm $\mathcal{A}'$ are as follows:

(1) In the first phase, the algorithm deterministically creates an (α, α)-ruling set R for the path P, mirroring the approach outlined in the proof of [2, Lemma 5.5].
(2) In the second phase, the algorithm constructs a larger virtual path P' with N nodes and simulates algorithm $\mathcal{A}$ on P' in the neighborhoods of nodes in R. Intuitively this path P' is constructed based on the pumping-lemma-style argument on LCL problems presented by Chang and Pettie [21], and it only relies on (the definition of) problem Π, not the algorithm $\mathcal{A}$. During this simulation, the algorithm encounters a failure only if $\mathcal{A}$ fails within specific neighborhoods of P'. Consequently, the algorithm's success in this phase aligns with the success rate of $\mathcal{A}$, ensuring a high probability of success.
(3) The third phase is also as in [2, Lemma 5.5]: The algorithm extends the fixed labels around R to the entire path P by applying brute force for nodes outside the neighborhood of R. Note that this extending is feasible because of the way the path P' is created and the pumping-lemma-style argument as discussed in [2].

In the end, we compose these three phases together to obtain the randomized online-LOCAL algorithm $\mathcal{A}'$ for solving Π with constant locality. □

Proof of Lemma 10.3. This proof closely follows the argument presented in [2, Lemma 5.6]. Let Π denote an LCL problem with a constant checking-radius r, and suppose $\mathcal{A}$ is a randomized online-LOCAL algorithm solving Π with constant locality T. Define $\beta = T + r + 1$. As in [2, Lemma 5.6], we consider an input-labeled graph P formed by many copies of all feasible input neighborhoods with a radius of β. We can depict P as a collection of disjoint path fragments that we later connect to each other and create a long path with. Each of these path fragments has size $2\beta + 1$, so the node in the center of each segment (i.e., the $(\beta + 1)$-th node in the segment) has the same view (up to radius β) as in the final path. Let V be the set of central nodes within these fragments.

We apply $\mathcal{A}$ to each node within the radius-r neighborhood of the nodes in V following an arbitrary order and then terminate. These nodes have the same view as in the final path because their distance to the endpoints is at most $\beta - r = T + 1$. We iterate the process multiple times, yielding a distribution of output labels around the central nodes. Given that the size of P is constant, there exists an output labeling L of the nodes occurring with probability $p = \Omega(1)$. If needed, we can augment P with (constantly many) additional nodes such that $|P| > 1/p$.

We set this labeling L as the deterministic output for graph P and proceed similarly to the deterministic case by constructing the canonical labeling f from L. Then, we use the function f to construct a LOCAL algorithm with $O(\log^* n)$ locality following the same steps as in the proof of [2, Lemma 5.6]. It is important to highlight that it is feasible to fill any gap of sufficient length between parts labeled with the canonical labeling: If the algorithm $\mathcal{A}$ would never produce a valid labeling for this gap, then the algorithm would fail to label P with probability at least p and, since $p \geq 1/|P|$, the algorithm $\mathcal{A}$ would not succeed with high probability. □

ACKNOWLEDGMENTS

This work was supported in part by the Research Council of Finland, Grant 333837. Xavier Coiteux-Roy is supported by the Swiss National Science Foundation. Francesco d'Amore is supported by MUR FARE 2020 - Project PAReCoDi CUP J43C22000970001. Darya Melnyk is supported by the European Research Council (ERC), grant agreement No. 864228 (AdjustNet), Horizon 2020, 2020-2025. Augusto Modanese and Shreyas Pai are supported by the Helsinki Institute for Information Technology (HIIT). Marc-Olivier Renou acknowledges funding by INRIA through the Action Exploratoire project DEPARTURE, and by the ANR for the JCJC grant LINKS (ANR-23-CE47-0003).

REFERENCES

[1] Jon Aaronson, David Gilat, Michael Keane, and Vincent de Valk. 1989. An algebraic construction of a class of one-dependent processes. *The Annals of Probability* 17, 1 (1989), 128–143.

[2] Amirreza Akbari, Navid Eslami, Henrik Lievonen, Darya Melnyk, Joona Särkijärvi, and Jukka Suomela. 2023. Locality in Online, Dynamic, Sequential, and Distributed Graph Algorithms. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany (LIPIcs, Vol. 261)*, Kousha Etessami, Uriel Feige, and Gabriele Puppis (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 10:1–10:20. https://doi.org/10.4230/LIPICS.ICALP.2023.10

[3] Heger Arfaoui and Pierre Fraigniaud. 2014. What can be computed without communications? *SIGACT News* 45, 3 (2014), 82–104. https://doi.org/10.1145/2670418.2670440

[4] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. 2019. The Distributed Complexity of Locally Checkable Problems on Paths is Decidable. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, Peter Robinson and Faith Ellen (Eds.). ACM, 262–271. https://doi.org/10.1145/3293611.3331606

[5] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Jan Studený, and Jukka Suomela. 2022. Efficient Classification of Locally Checkable Problems in Regular Trees. In *36th International Symposium on Distributed Computing, DISC 2022, October 25-27, 2022, Augusta, Georgia, USA (LIPIcs, Vol. 246)*, Christian Scheideler (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 8:1–8:19. https://doi.org/10.4230/LIPICS.DISC.2022.8

[6] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Jan Studený, Jukka Suomela, and Aleksandr Tereshchenko. 2023. Locally checkable problems in rooted trees. *Distributed Computing* 36, 3 (2023), 277–311. https://doi.org/10.1007/S00446-022-00435-9

[7] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. 2021. Lower Bounds for Maximal Matchings and Maximal Independent Sets. *J. ACM* 68, 5 (2021), 39:1–39:30. https://doi.org/10.1145/3461458

[8] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. 2020. How much does randomness help with locally checkable problems?. In *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, Yuval Emek and Christian Cachin (Eds.). ACM, 299–308. https://doi.org/10.1145/3382734.3405715

[9] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. 2021. Almost global problems in the LOCAL model. *Distributed Computing* 34, 4 (2021), 259–281. https://doi.org/10.1007/S00446-020-00375-2

[10] Alkida Balliu, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. 2018. New classes of distributed time complexity. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, Ilias Diakonikolas, David Kempe, and Monika Henzinger (Eds.). ACM, 1307–1318. https://doi.org/10.1145/3188745.3188860

[11] Alkida Balliu, Juho Hirvonen, Darya Melnyk, Dennis Olivetti, Joel Rybicki, and Jukka Suomela. 2022. Local Mending. In *Structural Information and Communication Complexity - 29th International Colloquium, SIROCCO 2022, Paderborn, Germany, June 27-29, 2022, Proceedings (Lecture Notes in Computer Science, Vol. 13298)*, Merav Parter (Ed.). Springer, 1–20. https://doi.org/10.1007/978-3-031-09993-9_1

[12] Alkida Balliu, Janne H. Korhonen, Fabian Kuhn, Henrik Lievonen, Dennis Olivetti, Shreyas Pai, Ami Paz, Joel Rybicki, Stefan Schmid, Jan Studený, Jukka Suomela, and Jara Uitto. 2023. Sinkless Orientation Made Simple. In *2023 Symposium on Simplicity in Algorithms, SOSA 2023, Florence, Italy, January 23-25, 2023*, Telikepalli Kavitha and Kurt Mehlhorn (Eds.). SIAM, 175–191. https://doi.org/10.1137/1.9781611977585.CH17

[13] Salman Beigi and Marc-Olivier Renou. 2021. Covariance decomposition as a universal limit on correlations in networks. *IEEE Transactions on Information Theory* 68, 1 (2021), 384–394.

[14] Max Born. 1926. Quantenmechanik der Stoßvorgänge. *Zeitschrift für Physik* 38, 11-12 (1926), 803–827. https://doi.org/10.1007/bf01397184

[15] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. 2016. A lower bound for the distributed Lovász local lemma. In *Proceedings of the 48th Annual ACM SIGACT*

Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016, Daniel Wichs and Yishay Mansour (Eds.). ACM, 479–488. https://doi.org/10.1145/2897518.2897570

[16] Sebastian Brandt, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Patric R. J. Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemyslaw Uznanski. 2017. LCL Problems on Grids. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, Elad Michael Schiller and Alexander A. Schwarzmann (Eds.). ACM, 101–110. https://doi.org/10.1145/3087801.3087833

[17] Keren Censor-Hillel, Orr Fischer, François Le Gall, Dean Leitersdorf, and Rotem Oshman. 2022. Quantum Distributed Algorithms for Detection of Cliques. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA (LIPIcs, Vol. 215)*, Mark Braverman (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 35:1–35:25. https://doi.org/10.4230/LIPICS.ITCS.2022.35

[18] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. 2016. An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, Irit Dinur (Ed.). IEEE Computer Society, 615–624. https://doi.org/10.1109/FOCS.2016.72

[19] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. 2019. An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model. *SIAM J. Comput.* 48, 1 (2019), 122–143. https://doi.org/10.1137/17M1117537

[20] Yi-Jun Chang, Gopinath Mishra, Hung Thuan Nguyen, Mingyang Yang, and Yu-Cheng Yeh. 2023. A Tight Lower Bound for 3-Coloring Grids in the Online-LOCAL Model. *CoRR* abs/2312.01384 (2023). https://doi.org/10.48550/ARXIV.2312.01384 arXiv:2312.01384

[21] Yi-Jun Chang and Seth Pettie. 2019. A Time Hierarchy Theorem for the LOCAL Model. *SIAM J. Comput.* 48, 1 (2019), 33–69. https://doi.org/10.1137/17M1157957

[22] Yi-Jun Chang, Jan Studený, and Jukka Suomela. 2023. Distributed graph problems through an automata-theoretic lens. *Theoretical Computer Science* 951 (2023), 113710. https://doi.org/10.1016/J.TCS.2023.113710

[23] Giulio Chiribella and Robert W. Spekkens (Eds.). 2016. *Quantum Theory: Informational Foundations and Foils.* Springer Netherlands. https://doi.org/10.1007/978-94-017-7303-4

[24] Xavier Coiteux-Roy, Francesco d'Amore, Rishikesh Gajjala, Fabian Kuhn, François Le Gall, Henrik Lievonen, Augusto Modanese, Marc-Olivier Renou, Gustav Schmid, and Jukka Suomela. 2024. No distributed quantum advantage for approximate graph coloring. In *Proc. 56th Annual ACM Symposium on Theory of Computing (STOC 2024).* ACM Press.

[25] Xavier Coiteux-Roy, Owidiusz Makuta, Fionnuala Curran, Remigiusz Augusiak, and Marc-Olivier Renou. 2024. The genuinely multipartite nonlocality of graph states is model-dependent. In preparation.

[26] Xavier Coiteux-Roy, Elie Wolfe, and Marc-Olivier Renou. 2021. Any physical theory of nature must be boundlessly multipartite nonlocal. *Physical Review A* 104, 5 (2021), 052207.

[27] Xavier Coiteux-Roy, Elie Wolfe, and Marc-Olivier Renou. 2021. No Bipartite-Nonlocal Causal Theory Can Explain Nature's Correlations. *Physical Review Letters* 127, 20, Article 200401 (2021). https://doi.org/10.1103/physrevlett.127.200401

[28] Richard Cole and Uzi Vishkin. 1986. Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking. *Information and Control* 70, 1 (1986), 32–53. https://doi.org/10.1016/S0019-9958(86)80023-7

[29] Sameep Dahal, Francesco D'Amore, Henrik Lievonen, Timothé Picavet, and Jukka Suomela. 2023. Brief Announcement: Distributed Derandomization Revisited. In *37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L'Aquila, Italy (LIPIcs, Vol. 281)*, Rotem Oshman (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 40:1–40:5. https://doi.org/10.4230/LIPICS.DISC.2023.40

[30] Giacomo Mauro D'Ariano, Giulio Chiribella, and Paolo Perinotti. 2016. *Quantum Theory from First Principles: An Informational Approach.* Cambridge University Press. https://doi.org/10.1017/9781107338340

[31] Michael Elkin, Hartmut Klauck, Danupon Nanongkai, and Gopal Pandurangan. 2014. Can quantum communication speed up distributed computation?. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, Magnús M. Halldórsson and Shlomi Dolev (Eds.). ACM, 166–175. https://doi.org/10.1145/2611462.2611488

[32] Manuela Fischer and Mohsen Ghaffari. 2017. Sublogarithmic Distributed Algorithms for Lovász Local Lemma, and the Complexity Hierarchy. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria (LIPIcs, Vol. 91)*, Andréa W. Richa (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 18:1–18:16. https://doi.org/10.4230/LIPICS.DISC.2017.18

[33] Cyril Gavoille, Adrian Kosowski, and Marcin Markiewicz. 2009. What Can Be Observed Locally?. In *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5805)*, Idit Keidar (Ed.). Springer, 243–257. https://doi.org/10.1007/978-3-642-04355-0_26

[34] Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. 2018. On Derandomizing Local Distributed Algorithms. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, Mikkel Thorup (Ed.). IEEE Computer Society, 662–673. https://doi.org/10.1109/FOCS.2018.00069

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

[36] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. 2017. On the complexity of local distributed graph problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, Hamed Hatami, Pierre McKenzie, and Valerie King (Eds.). ACM, 784–797. https://doi.org/10.1145/3055399.3055471

[37] Mohsen Ghaffari and Hsin-Hao Su. 2017. Distributed Degree Splitting, Edge Coloring, and Orientations. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, Philip N. Klein (Ed.). SIAM, 2505–2523. https://doi.org/10.1137/1.9781611974782.166

[38] Nicolas Gisin, Jean-Daniel Bancal, Yu Cai, Patrick Remy, Armin Tavakoli, Emmanuel Zambrini Cruzeiro, Sandu Popescu, and Nicolas Brunner. 2020. Constraints on nonlocality in networks from no-signaling and independence. *Nature Communications* 11, 1, Article 2378 (2020). https://doi.org/10.1038/s41467-020-16137-4

[39] Andrew V. Goldberg, Serge A. Plotkin, and Gregory E. Shannon. 1988. Parallel Symmetry-Breaking in Sparse Graphs. *SIAM Journal on Discrete Mathematics* 1, 4 (1988), 434–446. https://doi.org/10.1137/0401044

[40] Alexander E. Holroyd. 2023. Symmetrization for finitely dependent colouring. *CoRR* abs/2305.13980 (2023). https://doi.org/10.48550/arXiv.2305.13980 arXiv:2305.13980

[41] Alexander E. Holroyd, Tom Hutchcroft, and Avi Levy. 2018. Finitely dependent cycle coloring. *Electronic Communications in Probability* 23 (2018). https://doi.org/10.1214/18-ecp118

[42] Alexander E. Holroyd and Thomas M. Liggett. 2016. Finitely Dependent Coloring. *Forum of Mathematics, Pi* 4, Article e9 (2016). https://doi.org/10.1017/fmp.2016.7

[43] Alexander E. Holroyd, Oded Schramm, and David B. Wilson. 2017. Finitary coloring. *The Annals of Probability* 45, 5 (2017), 2867–2898. https://doi.org/10.1214/16-aop1127

[44] Taisuke Izumi and François Le Gall. 2019. Quantum Distributed Algorithm for the All-Pairs Shortest Path Problem in the CONGEST-CLIQUE Model. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, Peter Robinson and Faith Ellen (Eds.). ACM, 84–93. https://doi.org/10.1145/3293611.3331628

[45] Taisuke Izumi, François Le Gall, and Frédéric Magniez. 2020. Quantum Distributed Algorithm for Triangle Finding in the CONGEST Model. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France (LIPIcs, Vol. 154)*, Christophe Paul and Markus Bläser (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 23:1–23:13. https://doi.org/10.4230/LIPICS.STACS.2020.23

[46] Amos Korman, Jean-Sébastien Sereni, and Laurent Viennot. 2013. Toward more localized local algorithms: removing assumptions concerning global knowledge. *Distributed Computing* 26, 5-6 (2013), 289–308. https://doi.org/10.1007/S00446-012-0174-8

[47] François Le Gall and Frédéric Magniez. 2018. Sublinear-Time Quantum Computation of the Diameter in CONGEST Networks. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. ACM, 337–346. https://doi.org/10.1145/3212734.3212744

[48] François Le Gall, Harumichi Nishimura, and Ansis Rosmanis. 2019. Quantum Advantage for the LOCAL Model in Distributed Computing. In *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany (LIPIcs, Vol. 126)*, Rolf Niedermeier and Christophe Paul (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 49:1–49:14. https://doi.org/10.4230/LIPICS.STACS.2019.49

[49] Nathan Linial. 1987. Distributive Graph Algorithms-Global Solutions from Local Data. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*. IEEE Computer Society, 331–335. https://doi.org/10.1109/SFCS.1987.20

[50] Nathan Linial. 1992. Locality in Distributed Graph Algorithms. *SIAM J. Comput.* 21, 1 (1992), 193–201. https://doi.org/10.1137/0221015

[51] Frédéric Magniez and Ashwin Nayak. 2022. Quantum Distributed Complexity of Set Disjointness on a Line. *ACM Transactions on Computation Theory* 14, 1 (2022), 5:1–5:22. https://doi.org/10.1145/3512751

[52] Moni Naor. 1991. A Lower Bound on Probabilistic Algorithms for Distributive Ring Coloring. *SIAM Journal on Discrete Mathematics* 4, 3 (1991), 409–412. https://doi.org/10.1137/0404036

[53] Moni Naor and Larry J. Stockmeyer. 1995. What Can be Computed Locally? *SIAM J. Comput.* 24, 6 (1995), 1259–1277. https://doi.org/10.1137/S0097539793254571

[54] Michael A. Nielsen and Isaac L. Chuang. 2012. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press. https://doi.org/10.1017/cbo9780511976667

[55] Dennis Olivetti. 2019. *Round Eliminator: a tool for automatic speedup simulation.* https://github.com/olidennis/round-eliminator

[56] Alessandro Panconesi and Romeo Rizzi. 2001. Some simple distributed algorithms for sparse networks. *Distributed Computing* 14, 2 (2001), 97–100. https://doi.org/10.1007/PL00008932

[57] David Peleg. 2000. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898719772

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

[59] Yinon Spinka. 2020. Finitely dependent processes are finitary. *The Annals of Probability* 48, 4 (2020), 2088–2117. https://doi.org/10.1214/19-aop1417

[60] Jukka Suomela. 2020. Landscape of Locality (Invited Talk). In *17th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2020, June 22-24, 2020, Tórshavn, Faroe Islands (LIPIcs, Vol. 162)*, Susanne Albers (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2:1–2:1. https://doi.org/10.4230/LIPICS.SWAT.2020.2

[61] Joran van Apeldoorn and Tijn de Vos. 2022. A Framework for Distributed Quantum Queries in the CONGEST Model. In *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, Alessia Milani and Philipp Woelfel (Eds.). ACM, 109–119. https://doi.org/10.1145/3519270.3538413

[62] ChengSheng Wang, Xudong Wu, and Penghui Yao. 2022. Complexity of Eccentricities and All-Pairs Shortest Paths in the Quantum CONGEST Model. *CoRR* abs/2206.02766 (2022). https://doi.org/10.48550/ARXIV.2206.02766 arXiv:2206.02766

[63] Xudong Wu and Penghui Yao. 2022. Quantum Complexity of Weighted Diameter and Radius in CONGEST Networks. In *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, Alessia Milani and Philipp Woelfel (Eds.). ACM, 120–130. https://doi.org/10.1145/3519270.3538441

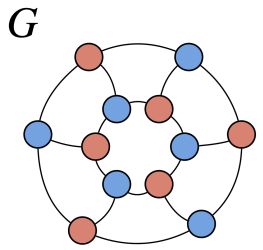


Fig. 6. An example of a *lifebuoy-shaped* graph G. The lifebuoy-shaped graphs are defined by the set $\mathcal{G}$ of graphs that are isomorphic to the above. Notice that their chromatic number is 2. In our case, the node labels required by the LOCAL model are unique and go from 1 to 12. Granting this extra power is not restrictive as we are proving a lower bound.

A THE NON-SIGNALING AND BOUNDED-DEPENDENCE MODELS OF DISTRIBUTED COMPUTING

This appendix introduces both the non-signaling model and the bounded-dependence model, which are the most powerful models that satisfy physical causality, and thus they generalize the quantum-LOCAL model. The distinction between the non-signaling model and the bounded-dependence depends on whether shared states are available (in the non-signaling model) or not (in the bounded-dependence model). We illustrate the difference between these models by analyzing the problem of $c = 2$-coloring a lifebuoy-shaped graph (see Fig. 6 and Fig. 7b) with locality $T = 2$.

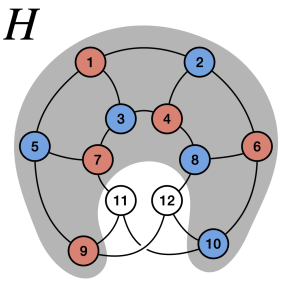
We first introduce a circuit formalism which allows us to clarify the early works of Arfaoui and Fraigniaud [3], and Gavoille et al. [33], by re-expressing the Randomized-LOCAL and Quantum-LOCAL models in this formalism (see Appendix A.1). Second, we introduce the concept of light-cones and the principle of non-signaling, explaining how (together with the symmetries of the graph) they define the non-signaling model (see Appendices A.1.3 and A.1.4). Third, we show in Appendix A.2 that lifebuoy-shaped graphs are not 2-colorable in the non-signaling model with locality $T = 2$ through a reduction from the 2-colorability of the cheating graph H shown in Fig. 7a. At last, we define the bounded-dependence model based on our circuit formalism and on the following three principles: device replication; non-signaling and independence [13, 26, 27, 38]; and invariance under symmetries of the graph (see Appendix A.3). We stress its connection to the concept of finitely dependent distributions [1, 40–42, 59].

A.1 Randomized-LOCAL, quantum-LOCAL, and non-signaling models

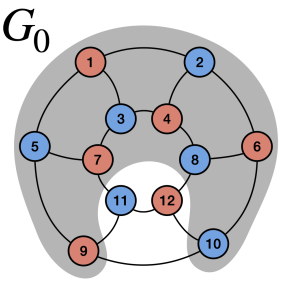
In the next two sections, we re-introduce the Randomized-LOCAL model and the Quantum-LOCAL model for coloring lifebuoy-shaped graphs in a circuit formalism. This will later enable us to clarify the definitions of the non-signaling and bounded-dependence models.

A.1.1 Randomized-LOCAL model. We consider twelve nodes with unique identifiers ranging from 1 to 12 and that are connected in a lifebuoy-shaped graph that is a priori unknown to the nodes[3] (an example of such graph is the labeled graph $G_0 \in \mathcal{G}$ illustrated in Fig. 7b). The nodes try to color this graph in the randomized-LOCAL model within $T = 2$ steps of synchronous communication. The most general ($T = 2$)-round strategy for the node 1 consists of the following procedure, which alternates between randomized processing steps and communication steps (the computational power and size of exchanged messages are unbounded):

[3]A reader used to standard quantum nonlocality should see the graph — H or $G \in \mathcal{G}$ — as an input of the problem, split and distributed among the local parties.



(a) H has chromatic number 3.



(b) A distributed algorithm that finds a 2-coloring of the lifebuoy-shaped graphs $G \in \mathcal{G}$ would by definition 2-color the particular instance $G = G_0$.

Fig. 7. The joint view of the couple of nodes $u, v = 1, 2$ after $T = 2$ rounds of communication is limited to the gray area. In this region the graphs G_0 and H are identical. The non-signaling principle implies that the outputs (c_1, c_2) must therefore be identically distributed in both G_0 and H.

Processing 0: Sample a random real number and store it locally.

Communication 1: Send all stored information, including the sampled random number, to all neighbors. Receive information from all neighbors and store it for subsequent rounds (in G_0, the neighbors are 2, 3, 5).

Processing 1: Process all stored information (possibly in a randomized way) and store the result.[4]

Communication 2: Send all stored information, including all received messages and the outputs of processing steps, to all neighbors. Receive information from neighbors and store it.

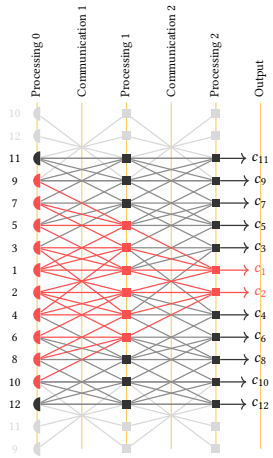
Processing 2: Process all stored information (possibly in a randomized way) to output a color.

Such T-round strategy on the graph G can be represented formally as a circuit $C_{G,T}$, such as in Fig. 8a where semicircles, line wires, and squares respectively represent the sampling of a random number, the transfer (or storage) of information, and the processing of information. Once the concrete operations performed by the nodes (i.e. randomness sampling and processing) are made explicit, it is possible to compute (using classical information theory) the exact output distribution of the strategy on graph G, that is, the probability distribution $\Pr\left[c_1, \ldots, c_n \,\middle|\, C_{G,T}\right]$ of observing that the set of nodes $\{1, \ldots, n\}$ outputs the colors $\{c_1, \ldots, c_n\}$ when connected as per one of the lifebuoy-shaped graphs $G \in \mathcal{G}$. Importantly, in our model, the operations performed by the nodes cannot depend on the connection graph G.

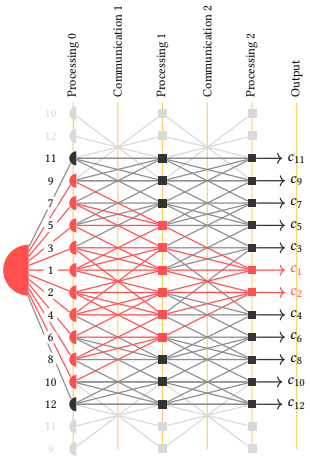
A generic classical strategy with shared randomness can be in the same way represented by the general circuit of Fig. 8b, by initializing the circuit with a source of randomness common to all nodes (the large semicircle).

*A.1.2 Quantum-*LOCAL *model.* Any quantum strategy can also be represented by the circuit formalism of Fig. 8 by using resources and processing operations that are quantum rather than classical. Quantum information theory provides a concrete mathematical formalism (based on the

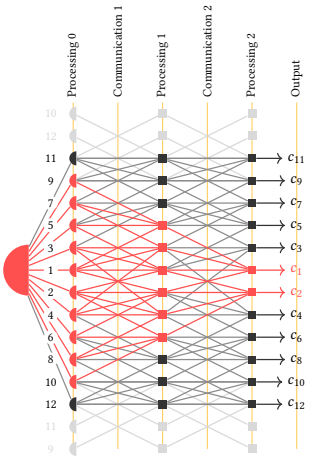
[4]In classical information theory, it is known that intermediate processing gates can be taken as identity gates, that is, any strategy can be simulated by a two-layer circuit where the parties send their first random number to all parties up to a distance T, and then make a unique processing step after all the communication has taken place. In quantum and non-signaling theories, this is not the case anymore [25].



(a) Circuit representation of a non-signaling strategy on graph G_0 of Fig. 7b, without a shared resource. Note the cyclicity of the circuit. Highlighted in red is the past-light-cone of the joint output (c_1, c_2), that is the set of gates which connects to the output gates producing (c_1, c_2).



(b) Circuit representation of a non-signaling strategy on graph G_0 of Fig. 7b, with an arbitrary shared resource. The joint output (c_1, c_2) remain, after $T = 2$ rounds of communication, independent of the graph structure around nodes 11 and 12, because the difference lies outside their joint past light-cones.



(c) Circuit representation of a non-signaling strategy on graph H of Fig. 7a, with an arbitrary shared resource. Note the difference between this circuit and the one of Fig. 8b (namely, the different connections between the top layer and bottom layer of the circuit) is only manifested outside the joint past light-cone of the outputs (c_1, c_2).

Fig. 8. The above circuits represent non-signaling strategies (it includes as special cases the classical strategies and quantum strategies) executed by the nodes 1 and 2 in various scenarios. The semicircles represent private (Fig. 8a) or shared (Fig. 8b and Fig. 8c) arbitrary-but-non-signaling resources; the wires depict communication (or storage), and the squares are local operations (using possibly private resources). The last layer of gates (or measurements) outputs the individual colors of the nodes (i.e. classical variables). Since the nodes start with no knowledge about the identity of their neighbors, the operations of the gates are a priori independent of the graph structure (as long as the nodes have the right degree). For the special cases of classical and quantum strategies, the output distribution can be computed directly from those circuits, which define it uniquely.

tensor product of Hilbert spaces) with rules to represent the potential operations performed by each of the gates of the circuit. Semicircles, line wires, and squares now respectively represent the creation of a quantum state (a density matrix), the transfer (or storage) of quantum information, and the processing of quantum information with quantum channels (or quantum measurements in the last layer) represented by completely positive operators. Once the quantum states and channels are made explicit, the Born rule of quantum information theory [14, 54] allows computing $\Pr\left[c_1, \ldots, c_n \middle| C_{G,T}\right]$, the output distribution of the strategy on graph G.

However, analyzing the quantum-LOCAL model directly is complicated. Instead, we employ the fact that it is possible to bound the time complexity of the quantum-LOCAL model by analyzing models that do not depend on the mathematical formalism of quantum information theory (and which are, therefore, arguably simpler). In the present appendix, we utilize the fact that quantum-LOCAL is sandwiched between the randomized-LOCAL model (less powerful) and the non-signaling model (the most powerful model satisfying the information-theoretic principles of non-signaling,

independence and device replication we introduce bellow). We next introduce the non-signaling model.

A.1.3 The non-signaling model (with unique identifiers). The non-signaling model is the most powerful model of synchronous distributed computing that does not violate physical causality, when a pre-established shared resource exists.

The formulation of the non-signaling model is radically different from the randomized LOCAL and quantum-LOCAL models. The randomized LOCAL and quantum-LOCAL models are based on classical (randomized) and quantum information theory which both provide a mathematical formalism to describe information processing gates and to compute the probability distribution of outputs when these gates are placed in a circuit. To show that some probability distribution is feasible in these models, one needs to propose a valid strategy in these mathematical formalisms. The non-signaling model does not rely on any underlying mathematical formalism, but on some information-theoretical principles which should not be violated. More precisely, to show that some probability distribution is feasible in the non-signaling model, one needs to explain how to obtain it, but only to make sure that this probability distribution is compatible with the *non-signaling* principle in a way we now explain.

Before introducing this principle, we state the following definition of light-cones:

Definition A.1 (Past light-cone). Consider a circuit. Consider a subset of gates R, and another gate $s \notin R$ in that circuit. We say that s is in the past light-cone of R if starting from s, one can reach a gate in R by traveling down the circuit. For instance, in Fig. 8b, the past light-cone of the second processing gates of nodes $\{1, 2\}$ is composed of all gates in red.

Remark A.2. Light-cones allow to formalize the fact that, in a circuit, the precise time ordering in which the gates process information has no influence over the result, as long as physical causality (that is, information can be transferred only through communication) is preserved. For instance, in Fig. 8a, our drawing of the circuit represents the processing-1 gate of node 8 as being in the past of the processing-2 gate of node 2. However, by stretching the communication lines, another drawing in which this time ordering is exchanged exists, and this is possible as long as one node is not in the past light cone of another. The name "light-cone" refers to relativity, where physical causality is bounded by the speed of light. Circuits provide an abstracted representation of physical causality that is not based on any notion of spacetime.

We now state the non-signaling principle[5], which is essentially the only constraint in the non-signaling model.

Definition A.3 (Non-signaling principle). Consider a set of gates, two different circuits C, C' connecting them, and corresponding distributions $\Pr[c_1, \ldots, c_n \mid C]$, $\Pr[c_1, \ldots, c_n \mid C']$. Let U be a subset of measurement gates whose past light-cones in C and C' coincide (i.e. they are composed of the same gates which are connected in the same way). Then $\Pr[\{c_i\}_{i \in U} \mid C] = \Pr[\{c_i\}_{i \in U} \mid C']$.

Implicitly, a non-signaling theory assumes that, given a set of gates and a valid way to connect them in a circuit C, one obtains a corresponding distribution $\Pr[c_1, \ldots, c_n \mid C]$[6] (here the alphabet of the outputs c_i is not limited: e.g., in case of unexpected circuits, the measurement gates can produce failure outputs).

[5] Chiribella and Spekkens [23] call this principle *no-signaling from the future*.

[6] Note that the gates have "types" and can only be connected to other gates of matching types: For instance, in our case, the inputs of gates from the second processing step must correspond to the outputs of gates from the first processing step. The theory of circuits can be formalized using category theory — see, e.g. [30].

We are now ready to define the non-signaling model, which is associated to circuits with a pre-shared non-signaling resource such as in Fig. 8b:

Definition A.4 (Non-signaling model, with unique identifiers). The distribution

$$\Pr\left[c_1, \ldots, c_n \,\middle|\, C_{G,T}\right]$$

is feasible in the non-signaling model (with unique identifiers) with locality T on graph G if and only if, for all possible alternative connecting graphs H of the nodes, there exists a distribution $P = \Pr\left[c_1, \ldots, c_n \,\middle|\, C_{H,T}\right]$ such that the no-signaling principle is respected.[7]

Note that H might not be a lifebuoy-shaped graph: in the case a gate 'discovers' this unexpected situation, it has the possibility to produce a new output from a set of error outputs (i.e. it crashes and no further useful constraint can be derived).

A.1.4 The non-signaling model without unique identifiers. We have up until now considered $N = 12$ nodes with unique identifiers ranging from 1 to $N = 12$. When the nodes do not start with unique identifiers, the situation is slightly more complicated. In that case, the resulting distribution should be invariant under subgraph isomorphism, as all nodes are running identical programs and the circuit is thus completely symmetric under permutations of nodes[8]. For instance, if the nodes were to color the graph of Fig. 7b without being a priori assigned unique identifiers, the resulting circuit $C_{G,T}$ would have some symmetries (as all processing gates would be the same in any layer of the respective steps 0, 1 and 2), implying that the distribution should be invariant by several non-trivial graph isomorphisms cyclically permuting the nodes, or inverting the inner and outer cycle of 6 nodes.

Definition A.5 (Non-signaling model, without unique identifiers). The distribution

$$\Pr\left[c_1, \ldots, c_n \,\middle|\, C_{G,T}\right]$$

is feasible in the non-signaling model (without unique identifiers) in T communication steps on graph G if and only if the following two conditions are respected: for all possible alternative connecting graphs H of the nodes, there exists a distribution $P = \Pr\left[c_1, \ldots, c_n \,\middle|\, C_{H,T}\right]$ such that the no-signaling principle is respected; and the distributions in G and H are invariant under subgraph isomorphism.

This definition can also be generalized to the case where several nodes with the same identifiers are present in G, or were the number of identifiers ranges from 1 to $M \neq N$.

A.2 G is not 2-colorable in $T = 2$ rounds

We now show by contradiction that the nodes $1, \ldots, 12$ cannot 2-color the set G of lifebuoy-shaped graphs with a non-signaling strategy (including pre-shared non-signaling resources) in $T = 2$ rounds of communication. More precisely, we prove that if there existed such a non-signaling algorithm, then the same algorithm would also color the cheating graph H with 2 colors, which is impossible.

[7] Note that considering scenarios with more than one copies of each node does not allow us to derive additional constraints, since the pre-shared non-signaling resource cannot by definition be cloned and extended to more than one copy of each of its original recipient.

[8] Note that while a shared classical resource is inherently symmetric, because it can be without a loss of generality taken to be distributed identically to each party, a non-signaling resource is not a priori symmetric under permutation of the parties that it connects. The absence of identifiers thus forces the non-signaling model to pre-share only a subset of all possible non-signaling resources, namely the subset that respects such symmetry.

Proof. Assume by contradiction that there exists such a non-signaling algorithm, that is, there exist gates as in Fig. 8b such that the resulting probability distribution $\Pr\left[c_1, \ldots, c_{12} \,\middle|\, C_{G,T}\right]$ is a proper coloring for all lifebuoy-shaped graphs $G \in \mathcal{G}$. Such proper coloring means that for all $G \in \mathcal{G}$, and for all nodes u, v that are neighbors in G, it holds that $\Pr\left[c_u \neq c_v \,\middle|\, C_{G,T}\right] = 1$.

We consider the same nodes performing the same gates, but we change the edges so that the connected nodes now form the graph H in Fig. 7b. Let $\Pr\left[c_1, \ldots, c_{12} \,\middle|\, C_{H,T}\right]$ be the distribution of output colors in that new configuration[9]. Consider two nodes connected by an edge in H: by symmetry of H, we can without a loss of generality assume that these are nodes 1 and 2. We introduce the lifebuoy-shaped graph G_0 of Fig. 7b ($G_0 \in G$ depends on our choice of nodes, here 1, 2, in H). Then, we observe that the common past light-cones of the two nodes is the same when connected through H or through G_0 (compare the circuit of Fig. 8b with the circuit of Fig. 8c). It hence holds that $\Pr\left[c_1 \neq c_2 \,\middle|\, C_{H,T}\right] = 1 = \Pr\left[c_1 \neq c_2 \,\middle|\, C_{G_0,T}\right]$ due to the non-signaling principle. We conclude that the non-signaling distributed algorithm outputs a 2-coloring for H, which is impossible. □

A.3 Bounded-dependence model

A.3.1 The bounded-dependence model. The bounded-dependence model is similar to the non-signaling model, but the former does not allow pre-shared non-signaling resources between the nodes. The following two new principles are needed to define feasible distributions in the bounded-dependence model.

Definition A.6 (Independence principle). Consider a set of gates connected in a circuit C, and the corresponding distribution $\Pr[c_1, \ldots, c_n \mid C]$. Let U, V be two subsets of measurement gates producing the outputs $\{c_u\}_{u \in U}$ and $\{c_v\}_{v \in V}$, respectively. If the past light-cones of U and V do not intersect, then $\Pr[\{c_w\}_{w \in U \cup V} \mid C] = \Pr[\{c_u\}_{u \in U} \mid C] \cdot \Pr[\{c_v\}_{v \in V} \mid C]$, that is their output distributions are independent.

Definition A.7 (Device-replication principle). Identical and independent copies of non-signaling gates and non-signaling resources can be prepared[10].

Then, we obtain the following definition:

Definition A.8 (Bounded-dependence model, with unique identifiers). The distribution

$$\Pr\left[c_1, \ldots, c_n \,\middle|\, C_{G,T}\right]$$

with unique identifiers has *bounded dependence* with locality T on graph G (without pre-shared non-signaling resources) if and only if for all possible alternative connecting graph H of the nodes and their replicates, there exists a distribution $\Pr\left[c_1, \ldots, c_m \,\middle|\, C_{H,T}\right]$ such as the non-signaling and independence principles are respected, and such that the distribution is invariant under subgraph isomorphisms.

Note a subtlety related to subgraph isomorphisms in the bounded-dependence model. There is the variant where the nodes have identifiers in G, and the one where they do not. When the nodes do not have any identifiers, the class of subgraph isomorphisms of all alternative graphs H created out of the nodes in G and their replicates is obviously larger than with identifiers, because

[9]While we are deceiving the nodes by promising a lifebuoy-shaped connecting graph but imposing instead H, the nodes cannot locally detect the fraud in $T = 2$ communication steps or less. (More formally, the past light-cone in H of any node is then compatible with a lifebuoy-shaped graph — an individual node cannot detect the difference and must therefore, according to the non-signaling principle, output a color as if it were in a lifebuoy-shaped graph.)

[10]Note that this principle does not imply that one can duplicate *unknown* non-signaling resources: device-replication compatible with the quantum no-cloning theorem.

the subgraph isomorphisms must respect the identifiers. However, even if the nodes of G do have distinct identifiers, as H is created out of possibly many copies of the original nodes of G, H might contain several nodes with the same identifiers. Hence, the group of subgraph isomorphism of H might be nontrivial even if all nodes in G have distinct identifiers. For instance, in lifebuoy-shaped graphs, one could consider the case represented in Fig. 9, which starts from the graph G_0 in Fig. 7b, duplicates all nodes, and constructs a new graph H of 24 nodes with identifiers ranging from 1 to 12 with one non-trivial graph isomorphism cyclically permuting the nodes.

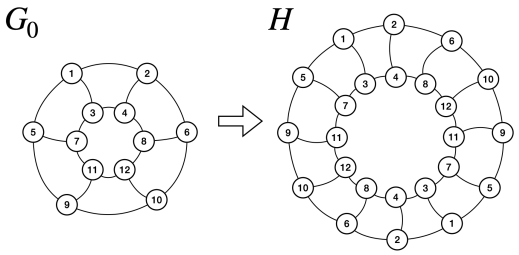


Fig. 9. In the bounded-dependence model, one can find non-trivial subgraph isomorphisms even when the nodes are provided with unique identifiers.

A.3.2 Relation with finitely dependent distributions. Our bounded-dependence model is directly connected to the concept of finitely dependent distributions introduced in mathematics [1, 40–43, 59]. In this framework, in a graph H, the color c_i produced by each node i is seen as the result of a random process C_i. The set of all random processes $\{C_i\}_i$ is said to be k-dependent in graph H if, for any two subsets U, V of the nodes of H which are at least at distance $k+1$, the two sets of processes $\{C_u\}_{u\in U}$ and $C_{vv\in V}$ are independent. In our notation, assuming even k and taking $T = k/2$, this is equivalent to asking that $\Pr\left[\{c_w\}_{w\in U\cup V} \,\middle|\, C_{H,T}\right] = \Pr\left[\{c_u\}_{u\in U} \mid C\right] \cdot \Pr\left[\{c_v\}_{v\in V} \,\middle|\, C_{H,T}\right]$.

Hence, the question of the existence of distributions with bounded dependence that solve some problem can directly be formulated in terms of the existence of finitely dependent distributions over a family of graphs that satisfy additional constraints of compatibility (to satisfy the non-signaling principle) and symmetries (to cope with subgraph isomorphisms).