New Limits on Distributed Quantum Advantage: Dequantizing Linear Programs

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

Corinna Coupette
Aalto University, Espoo, Finland
Max Planck Institute for Informatics, Saarbrücken, Germany

Antonio Cruciani
Aalto University, Espoo, Finland

Francesco d'Amore
Gran Sasso Science Institute, L'Aquila, Italy

Massimo Equi
Aalto University, Espoo, Finland

Henrik Lievonen
Aalto University, Espoo, Finland

Augusto Modanese
Aalto University, Espoo, Finland

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

Jukka Suomela
Aalto University, Espoo, Finland

Abstract

In this work, we give two results that put new limits on distributed quantum advantage in the context of the LOCAL model of distributed computing:

1. We show that there is no distributed quantum advantage for any linear program. Put otherwise, if there is a quantum-LOCAL algorithm $\mathcal{A}$ that finds an α-approximation of some linear optimization problem Π in T communication rounds, we can construct a classical, deterministic LOCAL algorithm $\mathcal{A}'$ that finds an α-approximation of Π in T rounds. As a corollary, all classical lower bounds for linear programs, including the KMW bound, hold verbatim in quantum-LOCAL.
2. Using the above result, we show that there exists a locally checkable labeling problem (LCL) for which quantum-LOCAL is strictly weaker than the classical deterministic SLOCAL model.

Our results extend from quantum-LOCAL to finitely dependent and non-signaling distributions, and one of the corollaries of our work is that the non-signaling model and the SLOCAL model are incomparable in the context of LCL problems: By prior work, there exists an LCL problem for which SLOCAL is strictly weaker than the non-signaling model, and our work provides a separation in the opposite direction.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Quantum computation theory

Keywords and phrases linear programming, distributed quantum advantage, quantum-LOCAL model, SLOCAL model, online-LOCAL model, non-signaling distributions, locally checkable labeling problems, dequantization

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.11

Related Version *Full Version*: `https://arxiv.org/abs/2506.07574`

Funding This work was supported in part by the MUR (Italy) Department of Excellence 2023 - 2027 for GSSI, the European Union - NextGenerationEU under the Italian MUR National Innovation Ecosystem grant ECS00000041 - VITALITY – CUP: D13C21000430001, the PNRR MIUR research project GAMING "Graph Algorithms and MinING for Green agents" (PE0000013, CUP D13C24000430001), and the Research Council of Finland, Grants 363558 and 359104.

Acknowledgements We thank the anonymous reviewers for their helpful comments and suggestions, which allowed us to improve the quality of this paper.

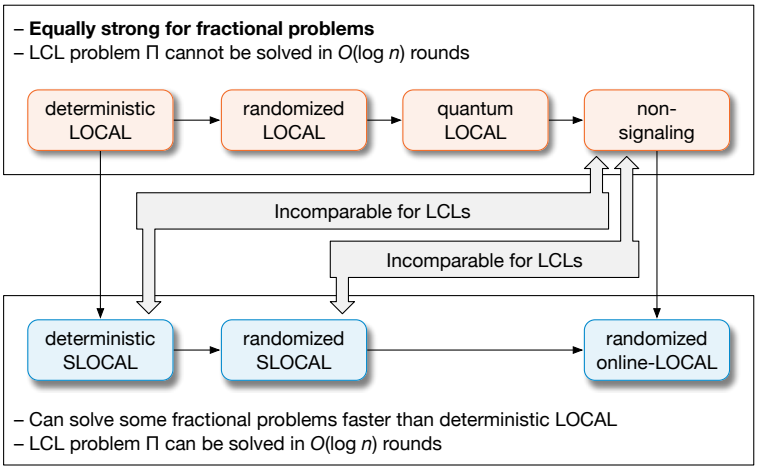

Figure 1 Overview of the results and relevant models of computing.

1 Introduction

In this work, we explore the landscape of distributed graph algorithms in two dimensions:

1. Classical distributed algorithms vs. distributed *quantum* algorithms.
2. Combinatorial graph problems (e.g., maximum independent set) vs. their *fractional* linear-programming relaxations (e.g., maximum *fractional* independent set).

We prove two results that put limits on distributed quantum advantage; see Figure 1 for a schematic overview:

1. We show that there is no distributed quantum advantage for any linear program.
2. Using the above result, we give a new separation between quantum algorithms and classical algorithms (more precisely, between quantum-LOCAL and SLOCAL models).

1.1 Contribution 1: Dequantizing Fractional Algorithms

Setting: Fractional Problems in the LOCAL Model and the Quantum-LOCAL Model. Let us first recall what fractional linear-programming relaxations of graph problems are. For example, in the maximum independent set problem, the task is to label each node $v \in V$ with a value $x_v \in \{0, 1\}$ such that for each edge $\{u, v\}$, we satisfy $x_u + x_v \leq 1$, and we are maximizing $\sum_v x_v$. Now the maximum *fractional* independent set problem is the obvious linear-programming relaxation, where the range of values is $x_v \in [0, 1]$. See Section 2.2 for more details.

In the LOCAL model of distributed computing, a set of computers (nodes) communicates via bidirectional links (edges) defined by an input graph, and computation proceeds in synchronous rounds. In each round, each computer may send a message of unlimited size to each of its neighbors and update its state based on the messages it receives, and the main complexity measure is the number of communication rounds required to solve the given problem. The quantum-LOCAL model is like the LOCAL model, except that we replace all computers with quantum computers and all communication links with quantum communication links capable of exchanging qubits. The main source of potential advantage of the quantum-LOCAL over LOCAL comes from the fact that the messages may contain qubits entangled with the state of the computer, and this has indeed been used to show an advantage for LCL problems [5].

New Result. We prove the following result that is applicable to any fractional linear-programming relaxation:

> Assume $\mathcal{A}$ is a distributed algorithm that finds an α-approximation of some linear optimization problem Π in T communication rounds in the **quantum-LOCAL model**. Then there is also an algorithm $\mathcal{A}'$ that finds an α-approximation of Π in T communication rounds in the **classical deterministic LOCAL model**.

It was already well-known that distributed graph algorithms for solving linear programs can be *derandomized* for free – without losing anything in the running time or approximation ratio. What we show is that they can also be *dequantized*. This can be pushed even further, beyond the quantum-LOCAL model: We show that it holds even if $\mathcal{A}$ is an algorithm in the *non-signaling model*, which is a model strictly stronger than quantum-LOCAL (see Section 2.3 for detailed definitions).

Technical Overview. The proof is near-trivial: $\mathcal{A}'$ outputs the *expected value* of the output of $\mathcal{A}$. A bit more precisely, let X_v be the random variable that represents the output of $\mathcal{A}$ at node v, and let $x_v = \mathbb{E}[X_v]$ be the output produced by $\mathcal{A}'$. Now if we look at the entire output vector, it holds that $x = \mathbb{E}[X]$. In particular, x is a linear combination of α-approximate solutions X to problem Π, and hence x is also an α-approximate solution to Π. We give the details in Section 3.

Implications for the KMW Bound. One of the seminal lower bounds for distributed graph algorithms is the KMW lower bound by Kuhn, Moscibroda, and Wattenhofer [16]; see Appendix C. In 2009, Gavoille, Kosowski, and Markiewicz [14] observed that this lower bound holds also for quantum-LOCAL and non-signaling models. However, to our knowledge, the proof was never published – they merely write that "by careful analysis, it is easy to prove".

However, now we no longer need to do the careful analysis. The key observation is that the KMW bound is inherently a bound for fractional problems. As we now know that there is no quantum advantage for fractional problems, we know that all the implications of the KMW bound indeed hold verbatim also in the quantum-LOCAL model.

While this is not a new result, at least now there is a proof explicitly written down, and the proof is fundamentally different from the idea of carefully inspecting the inner workings of the KMW bound and checking that the argument holds.

1.2 Contribution 2: A New Separation for SLOCAL vs. Non-Signaling Distributions

Setting: LCL Problems and SLOCAL Model. Let us now move on from fractional optimization problems to *locally checkable labeling problems* or LCLs. First defined by Naor and Stockmeyer in the 1990s [17], LCLs constitute a particularly simple family of graph problems that manages to capture many problems of interest in our field. LCL problems are graph problems that can be described by listing a *finite* set of valid labelings in local neighborhoods. There is a long line of work on understanding the landscape of LCL problems, e.g., [1,2,7–10,12,17,18], and this line of research has recently started to explore the interplay

between various models of distributed computing, including not only the classical LOCAL model, quantum-LOCAL model, and non-signaling distributions, but also models such as SLOCAL and online-LOCAL.

The SLOCAL model [15] is a sequential counterpart of the LOCAL model: An adversary queries nodes in a sequential order, and when a node v is queried, the algorithm has to choose the final label of node v. To do that, the algorithm can gather the full information on its radius-T neighborhood, store all this information at node v (for the benefit of other nodes nearby that get queried later), and use all this information to choose its label. We refer to Section 2 for precise definitions, and to Figure 1 for an overview of how SLOCAL is related to other recently-studied models of computing.

New Result. The SLOCAL model is at least as strong as the LOCAL model because with the full knowledge of the radius-T neighborhood, it can simulate any LOCAL algorithm for T steps. However, its exact relation with the quantum-LOCAL model and non-signaling distributions has been an open question – in essence, the question is whether the ability to manipulate qubits is more useful or less useful than the ability to process nodes in some sequential order. We prove the following new result:

> There exists an LCL problem Π such that Π can be solved with locality $O(\log n)$ in the SLOCAL model, but it *cannot* be solved with locality $O(\log^{1.49} n)$ in the non-signaling model or the quantum-LOCAL model.

Technical Overview. While this may seem disconnected from the results in Section 1.1, it is, in a sense, a direct corollary. One implication of the KMW bound is that the problem of finding a maximal matching in a bipartite graph of degree at most Δ cannot be solved in $o(\log \Delta / \log \log \Delta)$ rounds. On the other hand, this is a problem that is trivial to solve in the SLOCAL model in $O(1)$ rounds. So at this point, we have a *family* of LCL problems $\Pi'(\Delta)$ parameterized by Δ that is trivial in SLOCAL but nontrivial in the non-signaling model. We can now plug this family of problems into the construction of [5], as maximal matchings satisfy their key technical requirement of *linearizability*. Deploying this machinery yields a single genuine LCL problem that is strictly easier to solve in SLOCAL than in the non-signaling and quantum-LOCAL models. We give the details in Section 4.

Implications for the Landscape of Models. By recent work [6], there is an LCL problem that is strictly easier to solve in the non-signaling model than in the SLOCAL model. Here, we have obtained a separation in the converse direction. Therefore, in particular:

> The SLOCAL model and the non-signaling model are incomparable in the context of LCL problems; neither is able to simulate the other with constant overhead.

We contrast this with, e.g., the situation for the randomized online-LOCAL model, i.e., the SLOCAL model augmented with a global memory that all nodes can access [2]. The randomized online-LOCAL model *can* simulate both the SLOCAL model and the non-signaling model [1].

2 Preliminaries

Natural numbers are denoted as $\mathbb{N}$ and include 0, while we use $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$ for natural numbers without 0. Moreover, we use the notation $[n] = \{1, \ldots, n\}$ for any $n \in \mathbb{N}_+$. Throughout this work, we consider only undirected graphs. A graph $G = (V, E)$ consists of a set of nodes V and a set of edges E, and we use the notation $V(G)$ and $E(G)$, respectively, if we need to specify which graph we refer to. Given a subset of nodes A, $G[A]$ is the subgraph induced by A, that is, $G[A] = (A, E_A)$, where $(u, v) \in E_A$ if and only if $u, v \in A$ and $(u, v) \in E$.

For any two nodes $u, v \in V$ in graph $G = (V, E)$, $\mathrm{dist}_G(u, v)$ is the length of a shortest path starting from u and ending at v. If there is no ambiguity, we write $\mathrm{dist}(u, v)$ without the graph subscript. For subsets of nodes $A, B \subseteq V$, we can then define $\mathrm{dist}(A, B) = \min_{(u,v)\in A\times B} \mathrm{dist}(u, v)$. For $T \in \mathbb{N}$, we define the radius-T neighborhood of a node u as $\mathcal{N}_T[u] = \{v \mid \mathrm{dist}(u, v) \leq T\}$, i.e., the set of nodes at distance at most T from u. This can be extended to a subset of nodes $A \subseteq V$ as $\mathcal{N}_T[A] = \cup_{u\in A}\mathcal{N}_T[u]$. All the above notions of distances and neighborhoods can be easily generalized to the case where we consider the edges instead of the nodes of a graph.

If v is a node and $e = \{u, v\}$ is an edge, then the pair (v, e) is a *half-edge*, and $\mathcal{H}(G)$ is the set of all pairs (v, e) where $v \in V(G)$ and (v, e) is a half-edge. A *centered graph* is a pair (G, c), where G is a graph and $c \in V(G)$ is one of its nodes, called the *center*. The *eccentricity* of (G, c) is the maximum distance $\max_{u\in V(G)} \mathrm{dist}_G(c, u)$ of a node from the center.

We now introduce the notion of *labeled graph*.

▶ **Definition 1** (Labeled graph [5]). *Suppose that $\mathcal{V}$ and $\mathcal{E}$ are sets of labels. A graph $G = (V, E)$ is said to be $(\mathcal{V}, \mathcal{E})$-*labeled *if the following holds:*

1. *Each node $v \in V$ is assigned a label from $\mathcal{V}$;*
2. *Each half-edge $(v, e) \in V \times E$ satisfying $v \in e$ is assigned a label from $\mathcal{E}$.*

We will consider problems that, in general, take as input a labeled graph and require as output a labeling such that some constraints are satisfied. To formalize the class of problems we consider, for a $(\mathcal{V}, \mathcal{E})$-labeled graph G, let $\ell(v) \in \mathcal{V}$ be the label assigned to node v and $\ell((v, e)) \in \mathcal{E}$ be the label assigned to half-edge (v, e).

To be able to consider labelings restricted to a subset of nodes, we say that, if we are given sets A and B, a subset $A' \subseteq A$, and a function $f : A \to B$, then $f \restriction_{A'}$ is the function $g : A' \to B$ such that $f(x) = g(x)$ for all $x \in A'$, and $\restriction$ is called the restriction operator. Given a $(\mathcal{V}, \mathcal{E})$-labeled graph G with labeling function ℓ, and another $(\mathcal{V}, \mathcal{E})$-labeled graph G' with labeling ℓ', suppose that $\varphi : V(G) \to V(G')$ is an isomorphism between G and G'. We say that ℓ and ℓ' are isomorphic if the labeling is preserved under φ.

2.1 Locally Checkable Labeling (LCL) Problems

In the distributed setting, the well-studied class of *locally checkable labeling (LCL) problems* [17] plays a central role, as they are those problems where the validity of a solution can be checked *locally*, that is, within a constant radius r. The intuition is that a node can gather its radius-r neighborhood to check whether its output satisfies the constraints of the LCL.

To formally define what it means to satisfy an LCL, we first introduce the notion of a set of constraints. This is the set of all valid labelings of a neighborhood of radius r and maximum degree Δ.

▶ **Definition 2** (Set of constraints). *Let $r, \Delta \in \mathbb{N}$ be constants. Consider two finite label sets $\mathcal{V}$ and $\mathcal{E}$. Let $\mathcal{C}$ be a finite set of pairs (H, v_H), where (H, v_H) is a $(\mathcal{V}, \mathcal{E})$-labeled centered graph such that the eccentricity of v_H is at most r and the degree of H is at most Δ. We say that $\mathcal{C}$ is an (r, Δ)-set of constraints over $(\mathcal{V}, \mathcal{E})$.*

After defining the notion of constraint, we can now define the notion of *constraint satisfaction*. Namely, we say that a graph satisfies a set of constraints if all of its radius-r neighborhoods belong to that set.

▶ **Definition 3** (Satisfying a set of constraints). *Let G be a $(\mathcal{V}, \mathcal{E})$-labeled graph, and let $\mathcal{C}$ be an (r, Δ)-set of constraints over $(\mathcal{V}, \mathcal{E})$, for some finite set of labels $\mathcal{V}, \mathcal{E}$. The graph G satisfies $\mathcal{C}$ if the following holds:*

- *For every node $u \in V(G)$, the $(\mathcal{V}, \mathcal{E})$-labeled graph $G[\mathcal{N}_r[u]]$ is such that the centered graph $(G[\mathcal{N}_r[u]], u)$ belongs to $\mathcal{C}$.*

We can now define the notion of *locally checkable labeling (LCL) problems*. An LCL can be seen as a relation that specifies which pairs of input and output labelings are valid.

▶ **Definition 4** (Locally Checkable Labeling (LCL) problems). *Let $r, \Delta \in \mathbb{N}$ be constants, and let $\mathcal{V}_{in}$, $\mathcal{E}_{in}$, $\mathcal{V}_{out}$, and $\mathcal{E}_{out}$ be finite sets of labels. A locally checkable labeling (LCL) problem Π is a tuple $(\mathcal{V}_{in}, \mathcal{E}_{in}, \mathcal{V}_{out}, \mathcal{E}_{out}, \mathcal{C})$ such that the following holds:*

- *$\mathcal{C}$ is an (r, Δ)-set of constraints over $(\mathcal{V}_{in} \times \mathcal{V}_{out}, \mathcal{E}_{in} \times \mathcal{E}_{out})$.*

Suppose that we are given as input a $(\mathcal{V}_{\text{in}}, \mathcal{E}_{\text{in}})$-labeled graph G, and let ℓ_{in} be the input labeling function of G. Solving an LCL problem $\Pi = (\mathcal{V}_{\text{in}}, \mathcal{E}_{\text{in}}, \mathcal{V}_{\text{out}}, \mathcal{E}_{\text{out}}, \mathcal{C})$ on G means to find a labeling function ℓ_{out} that produces an output labeling on G, turning G into a $(\mathcal{V}_{\text{out}}, \mathcal{E}_{\text{out}})$-labeled graph, such that the following holds:

- For each node $v \in G$, let $\ell(v) = (\ell_{\text{in}}(v), \ell_{\text{out}}(v))$. For each half-edge (v, e) in G, let $\ell((v, e)) = (\ell_{\text{in}}((v, e)), \ell_{\text{out}}((v, e)))$. Then the labeling function ℓ turns G into a $(\mathcal{V}_{\text{in}} \times \mathcal{V}_{\text{out}}, \mathcal{E}_{\text{in}} \times \mathcal{E}_{\text{out}})$-labeled graph that satisfies $\mathcal{C}$ according to Definition 3.

In other words, if we are given a graph and an input labeling, solving an LCL means finding an output labeling under which the graph satisfies the set of constraints of the LCL.

2.2 Distributed Linear Programming (LP) Problems

Next, we define distributed LP problems. We consider the following distributed setting: We are given a communication graph $G = (V, E)$ and a linear program bound to G of the form

$$\begin{aligned} \text{optimize} \quad & \sum_{i \in \mathcal{F}} c_i \cdot x_i \\ \text{subject to} \quad & \sum_{i \in \mathcal{F}} A_{j,i} \cdot x_i \trianglelefteq b_j \quad \forall j \in \mathcal{C} \\ & x_i \geq 0 \quad \forall i \in \mathcal{F} \,, \end{aligned}$$

where $\mathcal{F}$ is the set of variables, $\mathcal{C}$ is the set of constraints, coefficients $A_{j,i}, b_j, c_i$ are known locally, and the inequality $\trianglelefteq$ can be $\leq$, $=$, or $\geq$, depending on the LP formulation. In the distributed setting, each node $v \in V$ in the network "owns" one or more variables $x_i \in \mathcal{F}$. Furthermore, in the LOCAL (resp. SLOCAL) model, each node $v \in V$ knows the local constraints and variables involving nodes within its radius-T neighborhood $\mathcal{N}_T[v]$.

Types of Distributed LPs. There are three classes of distributed LP formulations: node-based, edge-based, and node-edge-based. In the first one, each node $v \in V$ is associated with a variable x_v. In the second, each node has a variable $x_{(v,u)}$ for each $u \in \mathcal{N}[v]$. Since an edge $(u, v) \in E$ is shared by both endpoints, we require that the two local copies coincide, i.e., $x_{(v,u)} = x_{(u,v)}$. In other words, v and u must agree on a common value for their shared edge variable. In the latter formulation, each node is associated with both a variable x_v and a set of variables $x_{(v,u)}$ shared with its neighborhood.

There are different types of local outputs for each class of distributed LP. For node-based LPs, the local output is a real value $x_v \in \mathbb{R}^{\geq 0}$ for each node $v \in V$, whereas for edge-based LPs, each node pair $(v, u) \in E$ agrees on and outputs a real value $x_{(u,v)}$. Consequently, for node-edge LPs, each node outputs both x_v and the incident edge values $x_{(v,u)}$. These local outputs must collectively satisfy the constraint set of the LP. That is, the union of all local outputs across the network must form a globally feasible solution to the LP, meaning that the variable assignments computed and output by the nodes must jointly satisfy all constraints of the LP formulation.

We remark that, in general, distributed LPs are not LCLs. The reason is that LPs involve continuous variables and global feasibility constraints, which cannot be verified using only local information and a finite label set.

Approximation Factor for Distributed LPs. Let $\mathcal{P}$ be a linear program defined over a communication network $G = (V, E)$ with variable set $\mathcal{F}$, and denote the optimal objective value by OPT. Let $\hat{x}$ be a solution produced by a distributed algorithm after a bounded number of synchronous communication rounds. We say that the distributed algorithm achieves an α-*approximation* to $\mathcal{P}$, for some $\alpha \geq 1$, if (1) $\hat{x}$ is a feasible solution to the LP, and (2) $\sum_{i \in \mathcal{F}} c_i \cdot \hat{x}_i \leq \alpha \cdot \text{OPT}$ or $\text{OPT} \leq \alpha \cdot \sum_{i \in \mathcal{F}} c_i \cdot \hat{x}_i$ for minimization or maximization problems, respectively. In other words, the distributed solution is within a factor α of the global optimum, even though each node operates with only local information. If we are dealing with a probabilistic model of computation (e.g., rand-LOCAL or non-signaling; see Section 2.3 below), then we require that the respective algorithm outputs an α-approximation *in expectation*.

Fractional Maximum Matching. In this work, we consider the fractional maximum matching problem formulated as the following LP:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} x_e \\ \text{subject to} \quad & \sum_{(v,u) \in E} x_{(v,u)} \leq 1 \quad \forall v \in V \\ & 0 \leq x_e \leq 1 \quad \forall e \in E \end{aligned}$$

Here, each variable x_e corresponds to an edge $e = (v, u) \in E$ and it is "owned" by both endpoints v and u in G. Each node $v \in V$ is responsible for the constraint $\sum_{u \in \mathcal{N}[v]} x_{(v,u)} \leq 1$, which involves all variables corresponding to the edges incident to v.

Now consider any *maximal matching* in the graph. By definition, a maximal matching is a matching where no additional edge can be added without violating the matching property. It is a standard result in approximation algorithms that any maximal matching is a 2-approximation to the maximum integral matching. Furthermore, the size of a maximum matching can be up to a factor $2/3$ smaller than the fractional maximum matching. Combining these bounds, we obtain that any maximal matching gives a feasible solution to the above LP and a solution value within a factor 3 of the optimum.

2.3 Models

In this section, we define all our computational models of interest.

The LOCAL Model. In the LOCAL model of computing, we are given a distributed system of n processors (or nodes) connected through a communication network represented as a graph $G = (V, E)$, along with an input function x. Every node $v \in V(G)$ has input data $\mathrm{x}(v)$, which encodes the number n of nodes in the network, a unique identifier from the set $[n^c] = \{1, 2, \ldots, n^c\}$, where $c \geq 1$ is a fixed constant, and possible inputs defined by the problem of interest (we assume nodes store both input node labels and input half-edge labels). If computation is randomized, we call the model *randomized LOCAL* (or rand-LOCAL), which means that $\mathrm{x}(v)$ additionally encodes an infinite string of bits that are uniformly and independently sampled for each node, and not shared with the other nodes. If this is not the case and computation is deterministic, we call the model *deterministic LOCAL* (or det-LOCAL). Computation is performed by synchronous rounds of communication. In each round, nodes can exchange messages of unbounded (but finite) size with their neighbors, and then perform an arbitrarily long (but terminating) local computation. Errors occur neither in sending messages nor during local computation. Computation terminates when every node v outputs a label $\ell_{\mathrm{out}}(v)$. The running time of an algorithm is the number of communication rounds, given as a function of n, that are needed to output a labeling that solves the problem of interest. In rand-LOCAL, we also ask that the algorithm solves the problem of interest with probability at least $1 - 1/\operatorname{poly}(n)$, where $\operatorname{poly}(n)$ is any polynomial function in n. If an algorithm runs in T rounds and both communication and computation are unbounded, we can look at it as a function mapping radius-T neighborhoods to output labels in the deterministic case, or to a distribution of output labels in the randomized case. Thus, we say that T is the *locality* of the algorithm.

Depending on the context, we may assume that the computing units are actually the *edges* of the graph, and the local variable $\mathrm{x}(v)$ is stored inside all edges that are incident to v.

The quantum-LOCAL Model. The quantum-LOCAL model is defined like the LOCAL model introduced above, with the following differences. Every processor (node) can locally operate on an unbounded (but finite) number of qubits, applying any unitary transformations, and quantum measurements can be locally performed by nodes at any time. In each communication round, nodes can send an unbounded (but finite) number of qubits to their neighbors. The local output of a node still needs to be an output label encoded in classical bits. As in rand-LOCAL, we ask that an algorithm solves a problem with probability at least $1 - 1/\operatorname{poly}(n)$. A more formal definition of the model can be found in [14].

The SLOCAL Model. The SLOCAL model of computing [15] is a sequential counterpart of the LOCAL model: An algorithm $\mathcal{A}$ processes the nodes sequentially in an order $p = v_1, v_2, \ldots, v_n$. The algorithm must work for any given order p. When processing a node v, the algorithm can query $\mathcal{N}_T[v]$, and $\mathcal{A}$ can read u's state for all nodes $u \in \mathcal{N}_T[v]$. Based on this information, node v updates its own state and computes its output $y(v)$. In doing so, node v can perform unbounded computation, i.e., v's new state can be an arbitrary function of the queried $\mathcal{N}_T[v]$. The output $y(v)$ can be remembered as a part of v's state. The *time complexity* $T_{\mathcal{A},p}(G, \boldsymbol{x})$ of the algorithm on graph G and inputs $\boldsymbol{x} = (x(v_1), x(v_2), \ldots, x(v_n))$ with respect to order p is defined as the maximum T over all nodes v for which the algorithm queries a radius-T neighborhood of v. The *time complexity* $T_{\mathcal{A}}$ of algorithm $\mathcal{A}$ on graph G and inputs $\boldsymbol{x}$ is the maximum $T_{\mathcal{A},p}(G, \boldsymbol{x})$ over all orders p.

The non-signaling Model. The non-signaling model is a model of computing that abstracts from how the actual computation is happening in the network, focusing on a probabilistic description of the valid output labelings. In this model, rather than given algorithms, we are asked to produce *outcomes* (also called strategies) that are functions mapping the input to a probability distribution over output labelings. In other words, given a graph and its input, a probability distribution over output labelings is assigned to the graph such that a sample from the distribution will produce a valid output labeling with high probability. The complexity of the outcome is given by its dependency radius T. This means that an outcome is non-signaling beyond distance T if, for any subset A of the nodes of the graph, modifying the graph or its input at distance greater than T from A does not change the output distribution over A. We proceed to give all the formal details needed to properly define the non-signaling model.

We first formally introduce the concept of *outcome*. For a network G, let x represent the function that maps every node to its input, which includes the input labeling function, port numbers, and unique identifiers. An outcome, then, is a function mapping a network and an input (G, x) to a probability distribution over output labelings.

▶ **Definition 5** (Outcome). *Let $\mathcal{V}, \mathcal{E}$ be sets of labels, and let $\mathcal{F}$ be the family of all input networks (G, x). An outcome O is a function that maps an input network $(G, \mathrm{x}) \in \mathcal{F}$ to a probability distribution $\mathrm{O}(G, \mathrm{x}) = \{(\mathrm{out}_i, p_i)\}_{i \in I}$ defined as follows:*

- *The set I is a set of indices.*
- *The function out_i is a labeling function that maps half-edges and nodes of G to labels in $\mathcal{V}$ and $\mathcal{E}$, respectively, making G a $(\mathcal{V}, \mathcal{E})$-labeled graph.*
- *Each p_i is a non-negative probability and $\sum_{i \in I} p_i = 1$.*

We say that an outcome O *solves* an LCL problem Π over a family of graphs $\mathcal{F}$ with probability $q > 0$ if, for every $G \in \mathcal{F}$ and every input data x, it holds that

$$\sum_{\substack{\mathrm{out}_i \in \mathrm{O}(G,\mathrm{x}): \\ \mathrm{out}_i \text{ solves } \Pi \text{ on } G}} p_i \geq q.$$

Let (G, x) be an input network, and consider any subset of nodes $S \subseteq V(G)$. Let $\mathcal{H}(G)[S]$ be the subset of $\mathcal{H}(G)$ that contains half-edges (v, e) for $v \in S$. The restriction of the output distribution $\mathrm{O}(G, \mathrm{x}) = \{(\mathrm{out}_i, p_i)\}_{i \in I}$ to S is the distribution $\mathrm{O}(G, \mathrm{x})[S] = \{(\mathrm{out}_j, p'_j)\}_{j \in J}$, where the output-labeling functions $\{\mathrm{out}_j\}$ assign labels only on nodes of S and on half-edges of $\mathcal{H}(G)[S]$, and the probability p'_j satisfies the following condition:

$$p'_j = \sum_{\substack{\mathrm{out}_i \in \mathrm{O}(G,\mathrm{x}): \\ \mathrm{out}_i \text{ coincides with } \mathrm{out}_j \\ \text{on } S \text{ and } \mathcal{H}(G)}} p_i.$$

We now define the notion of isomorphic output distributions. Consider two graphs G and G' such that $\varphi : V(G) \to V(G')$ is an isomorphism. A probability distribution $\{(\mathrm{out}_i, p_i)\}_{i \in I}$ over output labelings for G is isomorphic to a probability distribution $\{(\mathrm{out}_j, p'_j)\}_{j \in J}$ over output labelings for G' if they are preserved under the action of φ.

We would now like to define a special type of outcome called *non-signaling outcome*. To this end, we first need to define the concept of *view* up to distance T. Given an input network (G, x) and a subset of its nodes $A \subseteq V(G)$, consider the subgraph $G[\mathcal{N}_T[A]]$ induced by $\mathcal{N}_T[A]$. The view up to distance T of A is the pair $\mathcal{V}_T(A) = (G_A, \mathrm{x}_A)$, where G_A is the graph defined as $V(G_A) = V(G[\mathcal{N}_T[A]]) = \mathcal{N}_T[A]$ and $E(G_A) = \{(u, v) \mid (u, v) \in$

$\mathrm{E}(G[\mathcal{N}_T[A]]), \mathrm{dist}_G(u, A) < T$ or $\mathrm{dist}_G(v, A) < T\}$, and $\mathrm{x}_A = \mathrm{x} \restriction \mathcal{N}_T[A]$. Intuitively, nodes in A see everything up to distance T except for the edges among the bordering nodes of $G[\mathcal{N}_T[A]]$ (but they can see the labels of the half-edges incident to them). In general, for two arbitrary graphs G and H and subsets of nodes $A \subseteq V(G), B \subseteq V(H)$, we say that a function $\varphi : V(G) \to V(H)$ is an isomorphism between $\mathcal{V}_T(A) = (G_A, \mathrm{x}_A)$ and $\mathcal{V}_T(B) = (G_B, \mathrm{x}_B)$ if φ is an isomorphism between $V(G_A)$ and $V(G_B)$ and $\mathrm{x}_A = \mathrm{x}_B \circ \varphi$. Now, a non-signaling outcome is defined as follows.

▶ **Definition 6** (Non-signaling outcome). *Let* O *be an outcome, G and H be graphs, $\varphi : V(G) \to V(H)$ be a function, and $T \in \mathbb{N}$. Outcome* O *is* non-signaling beyond distance T *if, for any two subsets of nodes $A_G \subseteq V(G), A_H \subseteq V(H)$ such that φ is an isomorphism between $\mathcal{V}_T(A_G)$ and $\mathcal{V}_T(A_H)$, the restricted distributions $\mathrm{O}(G, \mathrm{x}_G)[A_G]$ and $\mathrm{O}(H, \mathrm{x}_H)[A_H]$ are isomorphic under φ.*

Alternatively, we can also say that O has locality T. Running T-round classical or quantum-LOCAL algorithms, both with or without shared resources, yields output-labeling distributions that are non-signaling outcomes with locality T.

The *non-signaling model* is, thus, a computational model where the input is a network with input (G, x), and an LCL problem Π is solved if there exists a non-signaling outcome O that solves Π with success probability at least $1 - 1/\operatorname{poly}(n)$, where $n = |V(G)|$.

When the input of a problem is clear from the context, we will omit the input network (G, x), writing $\mathrm{O}(G)$. Observe that all the concepts of views and of restrictions of outcomes can be naturally defined via half-edges instead of nodes, especially when dealing with problems that only ask us to label half-edges (such definitions will be used later in Section 4.5).

We further assume that all probability distributions and output labelings that define a non-signaling O are *computable*. This is because a proper quantum-LOCAL algorithm can be implemented by a quantum circuit, which can be simulated in a classical computer (with costly computation). Hence, its output distribution is computable, and we can restrict ourselves to computable output-labeling distributions.

3 Dequantization for Distributed Linear Programming Problems

In this section, we prove our first result, i.e., that distributed non-signaling (and in particular also quantum-LOCAL) has no advantage over det-LOCAL for distributed linear programming problems.

▶ **Theorem 7.** *Let $\mathcal{P}$ be a distributed linear programming problem that admits a non-signaling distribution over α-approximations with locality T. Then there exists a deterministic LOCAL algorithm that finds an α-approximation of $\mathcal{P}$ with locality T.*

For simplicity, we will consider only the case where $\mathcal{P}$ is a node-based problem. It is clear how to extend the proof to the other classes of distributed LPs. The idea of the proof is relatively simple: We first note that for any distribution of α-approximations of a linear program $\mathcal{P}$, the *expectation* is also an α-approximation; this follows directly from the convexity of a linear program and the linearity of expectation. A detailed proof can be found in Appendix B.

▶ **Lemma 8.** *Let $\mathcal{P}$ be a linear program, and let* O *be a distribution over α-approximations of $\mathcal{P}$. Then $\hat{x} = \mathbb{E}[\mathrm{O}]$ is also an α-approximation of $\mathcal{P}$.*

Then we show that there exists a LOCAL algorithm that can locally compute this expectation, given access to the distribution. Intuitively, the algorithm gathers its radius-T neighborhood, where T is the locality of the non-signaling distribution, then calls the outcome

function O on its neighborhood to obtain a distribution of output labelings, and finally outputs the expected value of such a distribution. Again, the technical details are deferred to Appendix B.

▶ **Lemma 9.** *Let* O *be a computable non-signaling distribution with locality* T *over graph family* $\mathcal{F}$*. Then there exists a LOCAL algorithm with locality* T *that computes the expected outcome of this distribution everywhere.*

We are now ready to state the proof of Theorem 7.

Proof of Theorem 7. Let $\mathcal{P}$ be a distributed linear programming problem and let O be computable non-signaling distribution with locality T over α-approximations of $\mathcal{P}$. By Lemma 9, we have a LOCAL algorithm $\mathcal{A}$ that computes the expectation of O locally everywhere. As the outcome is a vector over local elements, its expectation is a vector over the expectations of the local elements. Hence, $\mathcal{A}$ computes the expectation of O. By Lemma 8, this is an α-approximation of $\mathcal{P}$. ◀

4 Separation Between SLOCAL and non-signaling

In this section, we prove that there exists an LCL problem Π that has complexity $O(\log n)$ in the SLOCAL model and complexity $\omega(\log n)$ in the non-signaling model.

▶ **Theorem 10.** *There exists an LCL problem* Π *that has complexity* $O(\log n)$ *in the deterministic SLOCAL model and* $\Omega\left(\log n \cdot \sqrt{\frac{\log n}{\log\log n}}\right)$ *in the non-signaling model.*

We devote the rest of this section to proving Theorem 10.

4.1 Overview

In order to define the problem Π, we borrow ideas from [5]. We start by giving a recap of the main ideas presented in [5].

Recap of Previous Results. The authors of [5] introduced the notion of *linearizable problems*, which are locally checkable problems that are not necessarily LCLs. They proved that, if there exists some linearizable problem P with some complexity $f(n)$ for some function f, then there exists some LCL problem $\Pi = \text{lift}(P)$ with some complexity $f'(n)$, where f' depends on f. For small-enough f, the function f' is a multiplicative factor $\Theta(\log n)$ larger than f. Interestingly, both the quantum complexity and the standard complexity are increased by this $\Theta(\log n)$ factor. In more detail, the authors of [5] proved the following:

1. In [3], it has been shown that there exists a problem P with quantum complexity $O(1)$ and randomized LOCAL complexity $\Omega(\min\{\Delta, \log_\Delta \log n\})$. By taking a suitable value of Δ, this result implies a lower bound of $\Omega(\frac{\log\log n}{\log\log\log n})$.
2. The problem P can be expressed as a linearizable problem.
3. The authors defined a function lift that takes as input a linearizable problem P and returns an LCL problem $\Pi = \text{lift}(P)$.
4. The authors showed that $\Pi = \text{lift}(P)$ has the following complexities:
   - $O(\log n)$ in quantum-LOCAL.
   - $\Omega(\log n \cdot \frac{\log\log n}{\log\log\log n})$ in rand-LOCAL.

Note that, while the problem P itself does not have any super-constant lower bound when $\Delta = O(1)$, this construction allows us to nevertheless obtain a problem $\Pi = \text{lift}(P)$ with a super-constant lower bound as a function of n on graphs in which $\Delta = O(1)$.

Our Approach. In essence, we show that the construction of [5] also preserves complexities in SLOCAL and in non-signaling, and we will use the maximal matching problem, phrased as a linearizable problem, as P. We will obtain the following:

1. Our problem P will be the maximal matching problem, phrased as a linearizable one.
2. The problem P has complexity $O(1)$ in SLOCAL, even on graphs of unbounded degree.
3. The problem $\Pi = \text{lift}(P)$ has complexity $O(\log n)$ in SLOCAL.
4. Maximal matching has a rand-LOCAL lower bound of $\Omega(\sqrt{\frac{\log n}{\log\log n}})$, proved as part of the KMW lower bound [16].
5. Maximal matching is a 3-approximation of fractional maximum matching, and the KMW bound, which is more general, holds for this latter problem as well. Thus, by Theorem 7, the lower bound for maximal matching also holds in non-signaling.
6. We prove that, in non-signaling, an upper bound of $o(\log n \cdot \sqrt{\frac{\log n}{\log\log n}})$ for Π would imply an upper bound of $o(\sqrt{\frac{\log n}{\log\log n}})$ for P, contradicting the KMW lower bound.

In order to achieve this, we prove a result similar to the one of [5], with the difference that we consider SLOCAL upper bounds, and non-signaling lower bounds.

4.2 The Definition of $\Pi = \text{lift}(P)$

We summarize the definition of $\Pi = \text{lift}(P)$ that appeared in [5]. At a high level, the problem Π is an LCL with inputs (i.e., nodes and node-edge pairs have input labels that come from a finite set) that is defined as a combination of two LCL problems:

- The LCL problem Π^{badGraph}, which is a problem defined on any graph.
- The LCL problem Π^{promise}, which is a problem defined on some specific class $\mathcal{G}$ of graphs labeled with some input (which is the input of Π). Note that, in order for a graph G to be in $\mathcal{G}$, the input given to the nodes of G must satisfy some specific local constraints.

In particular, Π^{badGraph} asks us to produce some output that satisfies, among others, the following properties:

- Each node is either marked (labeled with some specific output labels) or unmarked (labeled $\perp$).
- If $G \in \mathcal{G}$, then no node of G is marked.

Moreover, it is shown that there exists a deterministic $O(\log n)$ LOCAL algorithm $\mathcal{A}$ solving Π^{badGraph} on any graph G, such that the output of $\mathcal{A}$ satisfies that each connected component induced by unmarked nodes is in $\mathcal{G}$. Specifically, the authors of [5] proved the following.

▶ **Lemma 11** ([5]). *Let $G \in \mathcal{G}$. Then, the only valid solution for Π^{badGraph} on G is the one assigning $\perp$ to all nodes.*

▶ **Lemma 12** ([5]). *Let G be any graph. There exists a solution for Π^{badGraph} where each connected component induced by nodes outputting $\perp$ is a graph in $\mathcal{G}$. Moreover, such a solution can be computed in $O(\log n)$ deterministic rounds in the LOCAL model.*

The problem Π is defined such that it is first required to solve Π^{badGraph}, and then, on each connected component induced by unmarked nodes, it is required to solve Π^{promise}. We will later describe the problem Π^{promise}, the definition of which will depend on the given linearizable problem P. Now we argue that, in order to prove our lower and upper bounds, we can restrict our attention to graphs that are in $\mathcal{G}$ and to the problem Π^{promise}.

In [5], the quantum-LOCAL upper bound for Π is obtained as follows:

- First, apply Lemma 12. That is, in $O(\log n)$ deterministic LOCAL rounds, we obtain a solution for Π^{badGraph} satisfying that each connected component induced by nodes outputting $\perp$ is a graph in $\mathcal{G}$.

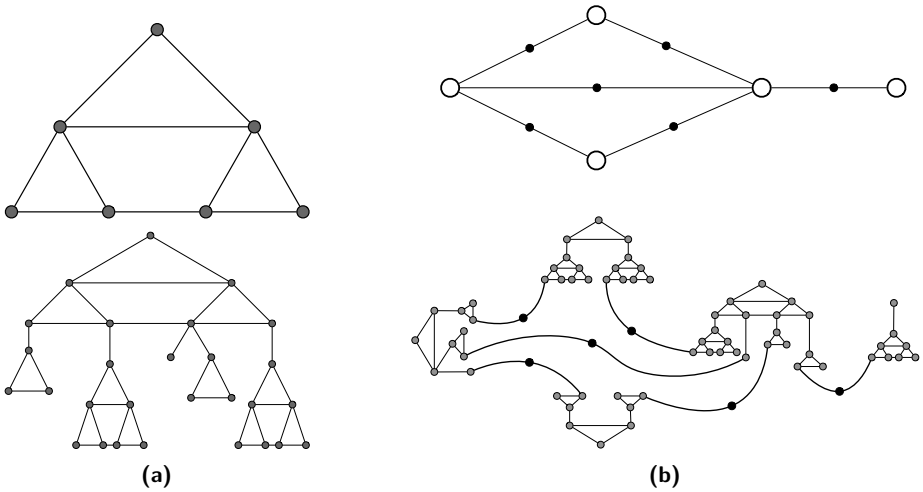

Figure 2 **(a)** A tree-like gadget at the top, and an octopus gadget at the bottom. **(b)** A proper instance at the bottom and, at the top, the graph obtained by contracting each octopus gadget into a single node.

- Then, use a quantum algorithm to solve Π^{promise} on each connected component induced by nodes outputting $\perp$. This requires $O(\log n)$ quantum rounds.

Since an $O(\log n)$ deterministic LOCAL algorithm can be directly executed in SLOCAL (i.e., SLOCAL is at least as strong as LOCAL), and since an SLOCAL algorithm obtained by composing two different SLOCAL algorithms has an asymptotic complexity equal to the sum of the complexities of the two composed algorithms, it is clear from the quantum algorithm that, in order to provide an $O(\log n)$ SLOCAL algorithm for Π, it is sufficient to provide an $O(\log n)$ SLOCAL algorithm for Π^{promise} on graphs that are in $\mathcal{G}$.

The randomized LOCAL lower bound for Π is obtained by considering graphs $G \in \mathcal{G}$. On these graphs, by Lemma 11, the only valid solution for Π^{badGraph} is the one assigning $\perp$ to all nodes. By the definition of Π, this implies that, on G, it is required to solve Π^{promise}. For our non-signaling lower bound, we will follow the exact same strategy.

4.3 The Graph Family $\mathcal{G}$

In order to define the family $\mathcal{G}$, we need to first introduce the notion of *proper instances*. At a high level, a proper instance is a graph that can be obtained by starting from some graph G' (which is not necessarily a simple graph) and replacing nodes with some *gadgets* according to some rules. Then, a graph $G \in \mathcal{G}$ will be obtained by labeling a proper instance in some specific way. In the following, we report the definition of some objects as given in [5]. The basic building block is the notion of *tree-like gadget*, of which we give an example in Figure 2a (top), while the formal definition can be found in Appendix A (Definition 23).

The next building block is called *octopus gadget*. At a high level, an octopus gadget is obtained by starting from a tree-like gadget, and connecting one or two additional tree-like gadgets to each "leaf" of the tree-like gadget. See Figure 2a (bottom) for an example and Appendix A (Definition 24) for a formal definition.

We can now define the family of proper instances. An example is shown in Figure 2b.

▶ **Definition 13** (Proper instance [5]). *Let $G = (V, E)$ be a graph. We say that G is a* proper instance *if there exists a node labeling function $\lambda : V \to \{\mathsf{intra\text{-}octopus}, \mathsf{inter\text{-}octopus}\}$ with the following properties.*

1. *Every connected component in the subgraph induced by nodes labeled* intra-octopus *is an octopus gadget (according to Definition 24).*
2. *The subgraph induced by nodes labeled* inter-octopus *does not contain any edge.*
3. *A node v labeled* intra-octopus *is connected to a node labeled* inter-octopus *if and only if v has coordinates $(w-1, 0)$ in the port gadget P containing v, where w is the height of P (that is, v is the left-most leaf of the port gadget containing v).*

The authors of [5] proved that proper instances are *locally checkable*, in the sense that there exist a set of local constraints $\mathcal{C}$ and finite sets of labels $\mathcal{V}$ and $\mathcal{E}$ for which an arbitrary graph G can be $(\mathcal{V}, \mathcal{E})$-labeled such that the constraints $\mathcal{C}$ to be satisfied on all nodes, if and only if G is a proper instance. More precisely, they proved the following statements.

▶ **Lemma 14** ([5]). *Let G be any non-empty connected graph that is $(\mathcal{V}, \mathcal{E})$-labeled such that $\mathcal{C}$ is satisfied at all nodes. Then G is a proper instance according to Definition 13.*

▶ **Lemma 15** ([5]). *Let G be a proper instance as defined in Definition 13. Then there exists a $(\mathcal{V}, \mathcal{E})$-labeling of G that satisfies the constraints in $\mathcal{C}$ at all nodes.*

We are now ready to define the family $\mathcal{G}$.

▶ **Definition 16.** *A $(\mathcal{V}, \mathcal{E})$-labeled graph G is in $\mathcal{G}$ if and only if the constraints in $\mathcal{C}$ are satisfied at all nodes.*

4.4 Linearizable Problems

In order to define Π^{promise}, we first need to introduce the notion of *linearizable problems*. A linearizable problem $\Pi^{\mathsf{linearizable}} = (\Sigma, (F, L, P), B)$ is defined as follows. Let us consider a hypergraph described as its bipartite incidence graph, where white nodes represent the original nodes, black nodes represent hyperedges, and we are also given an ordering of the edges of the white nodes. Intuitively, a problem $\Pi^{\mathsf{linearizable}} = (\Sigma, (F, L, P), B)$ is linearizable if it is possible to encode it as another LCL where the constraints for white nodes can be expressed solely in terms of consecutive edges in the ordering. More specifically, sets F and L define which labels are allowed for the first and last edges, respectively, while set P contains pairs of labels that can appear consecutively. In other words, P specifies the combinations in which labels can be assigned to a node and its successor in the ordering. We refer to Appendix A (Definition 25) for a formal definition.

We will use *maximal matching* as the running example of a linearizable problem. This is an important step to reach our goal, as we will use the fact that maximal matching can be expressed as a linearizable problem to separate SLOCAL from non-signaling. Maximal matching is a problem defined on graphs, and hence, we will describe a linearizable problem on hypergraphs of rank 2. In this case, the black constraint describes the edge constraints, and the white constraint describes the node constraints. An example of a solution to the maximal matching problem encoded as a linearizable problem is provided in Figure 3, and the following lemma formally states the existence of a linearized version of maximal matching. We defer the details of the proof to Appendix B.

▶ **Lemma 17.** *There exists a linearizable problem $\Pi^{\mathsf{linearizable}} = (\Sigma, (F, L, P), B)$ satisfying the following:*

- *A solution for $\Pi^{\mathsf{linearizable}}$ can be converted into a maximal matching in 0 deterministic LOCAL rounds.*
- *A solution for maximal matching can be converted into a solution for $\Pi^{\mathsf{linearizable}}$ in 0 deterministic LOCAL rounds.*

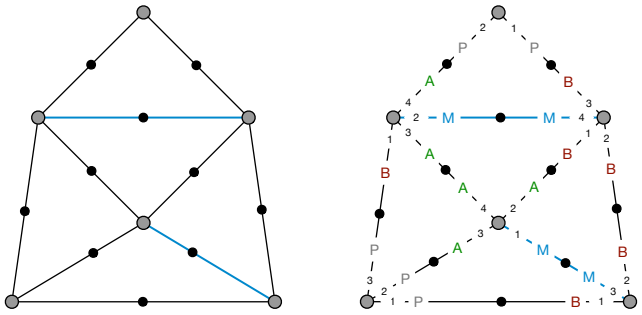

Figure 3 On the left, we show a solution to the maximal matching problem, where black nodes represent hyperedges of rank 2 and blue edges are in the matching. On the right, the same solution is encoded as a solution to $\Pi^{\text{linearizable}}$.

We now connect the notion of a linearizable problem with the problem Π^{promise} that we aim to define. The problem Π^{promise} is defined in [5] as an LCL problem (i.e., by describing its local constraints). This problem is defined as a function of a given problem $\Pi^{\text{linearizable}} = (\Sigma, (F, L, P), B)$. For our purposes, we do not need the details of the definition of Π^{promise}, and it is sufficient to state the properties that any valid solution needs to satisfy. Informally, Π^{promise} is defined such that, if we contract each octopus gadget into a single white node, and we treat each inter-cluster node as a black node, it must hold that, in the resulting graph, we get a solution for $\Pi^{\text{linearizable}}$. The formal definition of Π^{promise} can be found in Appendix A (Definition 26) or in [5].

4.5 SLOCAL Upper Bound and non-signaling Lower Bound

We now establish both an upper bound in the SLOCAL model and a lower bound in the non-signaling model, deferring the formal proofs to Appendix B. For SLOCAL, we have the following upper bound.

▶ **Lemma 18.** *Let $T(n)$ be an upper bound on the SLOCAL complexity of $\Pi^{\text{linearizable}}$ that holds also if the given graph contains parallel edges. Then the SLOCAL complexity of Π is upper bounded by $O(T(n)\log n)$.*

For non-signaling, we have the following lower bound.

▶ **Lemma 19.** *Let $T(n)$ be a lower bound on the locality of $\Pi^{\text{linearizable}}$ in non-signaling with failure probability $p(n)$, which is a non-increasing function of n bounded above by some constant $q < 1$. Then any non-signaling outcome for Π^{promise} with failure probability at most $p(n)$ requires locality $\Omega(T(n^{1/3})\log n)$.*

The following lemma is the key ingredient for the proof of Lemma 19.

▶ **Lemma 20.** *Let $\Pi^{\text{linearizable}}$ be any linearizable problem, and consider the LCL problem $\Pi = \text{lift}(\Pi^{\text{linearizable}})$. Suppose that there exists an outcome O that is non-signaling beyond distance $T(n)$ that solves Π with failure probability $p(n)$, which is a non-increasing function of n bounded above by some constant $q < 1$. Then we can construct an outcome O' that solves $\Pi^{\text{linearizable}}$ with failure probability at most $p(n)$ and is non-signaling beyond distance $T'(n) = O(T(n^3)/\log n)$.*

The proof of Lemma 19 is now straightforward.

Proof of Lemma 19. If any non-signaling outcome for Π with failure probability $p'(n) \leq p(n)$ has locality $o(T(n^{1/3}) \cdot \log n)$, then we can construct a non-signaling outcome that solves $\Pi^{\mathsf{linearizable}}$ with failure probability at most $p'(n)$ and has locality $o(T(n))$ by Lemma 20, which is a contradiction with the fact that we assumed $T(n)$ to be a lower bound on the locality of $\Pi^{\mathsf{linearizable}}$. ◀

4.6 Instantiating Our Construction

We are now ready to prove Theorem 10, referring to Appendix C for a high-level description of the KMW lower bound and to [11] for a detailed exposition. We consider the maximal matching problem, and by Lemma 17, there exists a linearizable problem P that is equivalent to maximal matching. In SLOCAL, maximal matching, and hence P, can be solved in one deterministic round by a trivial greedy algorithm. Hence, by Lemma 18, the deterministic SLOCAL complexity of $\Pi = \mathrm{lift}(P)$ is $O(\log n)$.

For a lower bound, recall that KMW gives us a lower bound of $\Omega(\sqrt{\log n / \log\log n})$ for $O(\log \Delta)$-approximation of fractional maximum matching (see Theorem 27) on graphs of degree $\Delta = 2^{\Theta(\sqrt{\log n \log\log n})}$. Combining Theorem 7 with the KMW lower bound and the fact that fractional maximum matching can be expressed as a linear program, we obtain the following corollary.

▶ **Corollary 21.** *There does not exist a non-signaling distribution for $O(\log \Delta)$-approximation of fractional maximum matching with locality $o(\sqrt{\log n / \log\log n})$ on graphs of degree $\Delta = 2^{\Theta(\sqrt{\log n \log\log n})}$.*

Since a maximal matching is a 2-approximation of a maximum matching, and a maximum matching is a $\frac{3}{2}$-approximation of a maximum fractional matching, we obtain the following.

▶ **Corollary 22.** *There does not exist a non-signaling distribution for maximal matching with locality $o(\sqrt{\log n / \log\log n})$.*

By combining Corollary 22 with Lemma 19, we obtain that in non-signaling, Π has complexity $\Omega(\log n \cdot \sqrt{\frac{\log n}{\log\log n}})$, as desired.

References

1 Amirreza Akbari, Xavier Coiteux-Roy, Francesco D'Amore, François Le Gall, Henrik Lievonen, Darya Melnyk, Augusto Modanese, Shreyas Pai, Marc-Olivier Renou, Václav Rozhon, and Jukka Suomela. Online locality meets distributed quantum computing. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 1295–1306. ACM, 2025. doi:10.1145/3717823.3718211.

2 Amirreza Akbari, Navid Eslami, Henrik Lievonen, Darya Melnyk, Joona Särkijärvi, and Jukka Suomela. Locality in online, dynamic, sequential, and distributed graph algorithms. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 10:1–10:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ICALP.2023.10.

3 Alkida Balliu, Sebastian Brandt, Xavier Coiteux-Roy, Francesco D'Amore, Massimo Equi, François Le Gall, Henrik Lievonen, Augusto Modanese, Dennis Olivetti, Marc-Olivier Renou, Jukka Suomela, Lucas Tendick, and Isadora Veeren. Distributed quantum advantage for local problems. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM*

Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025, pages 451–462. ACM, 2025. doi:10.1145/3717823.3718233.

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

5 Alkida Balliu, Filippo Casagrande, Francesco D'Amore, Massimo Equi, Barbara Keller, Henrik Lievonen, Dennis Olivetti, Gustav Schmid, and Jukka Suomela. Distributed quantum advantage in locally checkable labeling problems. *CoRR*, abs/2504.05191, 2025. doi:10.48550/arXiv.2504.05191.

6 Alkida Balliu, Mohsen Ghaffari, Fabian Kuhn, Augusto Modanese, Dennis Olivetti, Mikaël Rabie, Jukka Suomela, and Jara Uitto. Shared randomness helps with local distributed problems. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, July 8-11, 2025, Aarhus, Denmark*, volume 334 of *LIPIcs*, pages 16:1–16:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.ICALP.2025.16.

7 Sebastian Brandt. An automatic speedup theorem for distributed problems. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 379–388. ACM, 2019. doi:10.1145/3293611.3331611.

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

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

10 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. No distributed quantum advantage for approximate graph coloring. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1901–1910. ACM, 2024. doi:10.1145/3618260.3649679.

11 Corinna Coupette and Christoph Lenzen. A breezing proof of the KMW bound. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 184–195. SIAM, 2021. doi:10.1137/1.9781611976496.21.

12 Sameep Dahal, Francesco D'Amore, Henrik Lievonen, Timothé Picavet, and Jukka Suomela. Brief announcement: Distributed derandomization revisited. In Rotem Oshman, editor, *37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L'Aquila, Italy*, volume 281 of *LIPIcs*, pages 40:1–40:5. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.DISC.2023.40.

13 Kőnig Dénes. Graphok és alkalmazásuk a determinánsok és a halmazok elméletére. *Matematikai és természettudományi értes itő*, 34:104–119, 1916.

14 Cyril Gavoille, Adrian Kosowski, and Marcin Markiewicz. What can be observed locally? In Idit Keidar, editor, *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*, volume 5805 of *Lecture Notes in Computer Science*, pages 243–257. Springer, 2009. doi:10.1007/978-3-642-04355-0_26.

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

16 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *Journal of the ACM*, 63(2):17:1–17:44, 2016. doi:10.1145/2742012.

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

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

A Omitted Definitions

▶ **Definition 23** (Tree-like gadget [4, 5]). *A graph G is a* tree-like *gadget of height ℓ if it is possible to assign coordinates (l_u, k_u) to each node $u \in G$, where*

- *$0 \leq l_u < \ell$ denotes the depth of u in the tree, and*
- *$0 \leq k_u < 2^{l_u}$ denotes the position of u (according to some order) in layer l_u,*

such that there is an edge connecting two nodes $u, v \in G$ with coordinates (l_u, k_u) and (l_v, k_v) if and only if:

- *$l_u = l_v$ and $|k_u - k_v| = 1$, or*
- *$l_v = l_u - 1$ and $k_v = \lfloor \frac{k_u}{2} \rfloor$, or*
- *$l_u = l_v - 1$ and $k_u = \lfloor \frac{k_v}{2} \rfloor$.*

▶ **Definition 24** (Octopus gadget [5]). *Let $x \geq 1$ be a natural number, and $\eta = (\eta_0, \ldots, \eta_{2^{x-1}-1})$ a vector of 2^{x-1} entries in $\{1, 2\}$. Let $W = \{w_{(i,j)}\}_{(i,j)\in I}$ be a family of positive integer weights, where I is the set containing all pairs (i, j) satisfying $(i, j) \in \{0, 1, \ldots, 2^{x-1} - 1\} \times \{1, 2\}$ and $j \leq \eta_i$.*

*A graph $G = (V, E)$ is an (x, η, W)-*octopus gadget *if there exists a labeling $\lambda : V \to \mathcal{L} = I \cup \{root\}$ of the nodes of G such that the following holds.*

1. *For each element $y \in \mathcal{L}$, let G_y be the subgraph of G induced by nodes labeled with y. Then, for all $y \in \mathcal{L}$, G_y must be a tree-like gadget according to Definition 23.*
2. *For all $y, z \in \mathcal{L}$ such that $y \neq z$, G_y and G_z must be disjoint.*
3. *G_{root} has height x and, for all $(i, j) \in I$, $G_{(i,j)}$ has height $w_{(i,j)} \in W$.*
4. *For all $(i, j) \in I$, there is an edge connecting the node of $G_{(i,j)}$ that has coordinates $(0, 0)$ with the node of G_{root} that has coordinates $(x - 1, i)$.*

G_{root} is called the head-gadget *and, for all $(i, j) \in I$, $G_{(i,j)}$ is called a* port gadget*.*

▶ **Definition 25** (Linearizable problem [5]). *Let H be a hypergraph, and let G be its bipartite incidence graph. Let the nodes of G corresponding to the nodes of H be called* white *nodes, and let the nodes of G corresponding to the hyperedges of H be called* black *nodes.*

- *The task requires to label each edge of G with a label from some finite set Σ.*
- *There is a list of allowed* black node configurations *B, which is a list of multisets of labels from Σ that describes valid labelings of edges incident on a black node. We say that a black node* satisfies the black constraint *if the multiset of labels assigned to its incident edges is in B. It is assumed that the rank of H, and hence the maximum degree of black nodes, is a constant.*
- *Constraints on white nodes are described as a triple (F, L, P), where F (which stands for* first*) and L (which stands for* last*) are finite sets of labels, and P (which stands for* pairs*) is a finite set of ordered pairs of labels. In this formalism, it is assumed that an ordering on the incident edges of a white node is given, and it is required that:*
  - *The first edge is labeled with a label from F;*
  - *The last edge is labeled with a label from L;*

- *Each pair of consecutive edges must be labeled with a pair of labels from P.*

We say that a white node satisfies the white constraint *if its incident edges are labeled in a valid way.*

When solving a linearizable problem in the distributed setting, it is assumed that each node knows whether it is white or black.

▶ **Definition 26** (The definition of Π^{promise} of [5], rephrased)**.** *Given a graph $G \in \mathcal{G}$, the problem Π^{promise} requires to label the nodes of the graph as follows:*

- *Each node that is not in a port gadget must be labeled $\perp$.*
- *Each node that is in a port gadget must be labeled with a label from Σ.*
- *Nodes that belong to the same port gadget must be assigned the same label.*
- *Let g be an octopus gadget, and let $\ell_1, \ldots, \ell_d$ be the labels assigned to the d port gadgets of g, according to the natural left-to-right order of the port gadgets of g. It must hold that $\ell_1 \in F$, $\ell_d \in L$, and $(\ell_i, \ell_{i+1}) \in P$ for all i.*
- *Let v be an inter-cluster node, and let $\ell_1, \ldots, \ell_r$ be the labels assigned to the nodes of its r incident port gadgets. It must hold that $\{\ell_1, \ldots, \ell_r\} \in B$.*

B Omitted Proofs

Proof of Lemma 8. To show that $\hat{x}$ is an approximation of $\mathcal{P}$, we need to establish (1) that $\hat{x}$ is feasible and (2) that $\hat{x}$ gives the correct approximation ratio. To see the feasibility of $\hat{x}$, observe that the feasibility constraints are of the form

$$\sum_{i \in \mathcal{F}} A_{j,i} \cdot x_i \trianglelefteq b_j \quad \forall j \in \mathcal{C} \ .$$

Plugging in $\hat{x}$ and fixing $j \in \mathcal{C}$ gives us

$$\sum_{i \in \mathcal{F}} A_{j,i} \cdot \hat{x}_i = \sum_{i \in \mathcal{F}} A_{j,i} \cdot \mathbb{E}\left[O_i\right] = \mathbb{E}\Big[\underbrace{\sum_{i \in \mathcal{F}} A_{j,i} \cdot O_i}_{\trianglelefteq b_j}\Big] \trianglelefteq b_j \ .$$

The first equality holds by definition, the second equality holds by linearity of expectation, and the conclusion holds by the fact that O is a distribution over feasible solutions and the monotonicity of expectation. Hence, $\hat{x}$ is a feasible solution for $\mathcal{P}$.

It is left to show that $\hat{x}$ is also an α-approximation. Again, we can plug $\hat{x}$ into the target function, obtaining

$$\sum_{i \in \mathcal{F}} c_i \cdot \hat{x}_i = \sum_{i \in \mathcal{F}} c_i \cdot \mathbb{E}\left[O_i\right] = \mathbb{E}\Big[\sum_{i \in \mathcal{F}} c_i \cdot O_i\Big].$$

As each outcome of O is an α-approximation, we can invoke the linearity and monotonicity of expectation and get that this new target is also an α-approximation of $\mathcal{P}$. ◀

Proof of Lemma 9. We give the description for the LOCAL algorithm $\mathcal{A}$: Node v gathers its radius-T neighborhood $\mathcal{N}_T[v]$; this can be done with locality T. It then constructs an arbitrary graph $G' \in \mathcal{F}$ such that the neighborhood of node $v' \in V(G')$ is isomorphic to $\mathcal{N}_T[v]$. Note that such a graph always exists, as the graph the algorithm is being run on is one such graph. Now v invokes the distribution O on graph G' to compute the distribution of outputs for node v' and, in particular, the outcome. Node v then outputs this expected outcome and halts.

It remains to argue that this algorithm computes the expected outcome of O everywhere. This follows directly from the definition of the non-signaling distributions as the marginals of nodes v and v' coincide, and hence their expectations must also coincide. Moreover, the choice of graph G' does not affect this marginal distribution. ◀

Proof of Lemma 17. We define the problem as follows. The label set Σ is defined as $\Sigma = \{\mathsf{M}, \mathsf{B}, \mathsf{A}, \mathsf{P}\}$, where M, B, A, and P stand for *matched*, *before*, *after*, and *pointer*, respectively. The list of allowed black configurations is defined as

$B = \{\{\mathsf{M}, \mathsf{M}\}, \{\mathsf{P}, \mathsf{B}\}, \{\mathsf{P}, \mathsf{A}\}, \{\mathsf{B}, \mathsf{B}\}, \{\mathsf{B}, \mathsf{A}\}, \{\mathsf{A}, \mathsf{A}\}\}$. Then, F is defined as $F = \{\mathsf{M}, \mathsf{B}, \mathsf{P}\}$, L is defined as $L = \{\mathsf{M}, \mathsf{A}, \mathsf{P}\}$, and P is defined as $P = \{(\mathsf{B}, \mathsf{B}), (\mathsf{B}, \mathsf{M}), (\mathsf{M}, \mathsf{A}), (\mathsf{A}, \mathsf{A}), (\mathsf{P}, \mathsf{P})\}$.

We observe that the defined problem $\Pi^{\mathsf{linearizable}}$ satisfies the following properties.

- In any valid solution, for each node v, if we treat the labels assigned to the half-edges incident to v as a string (according to the ordering assigned to the half-edges incident to v), we obtain that such a string must satisfy the regular expression $\mathsf{P}^* \mid \mathsf{B}^*\mathsf{M}\mathsf{A}^*$. We call nodes satisfying P^* *unmatched* nodes and nodes satisfying $\mathsf{B}^*\mathsf{M}\mathsf{A}^*$ *matched* nodes. Moreover, we call an edge *matched* if both its half-edges are labeled M.
- Since $\{\mathsf{P}, \mathsf{P}\} \notin B$, we get that, in any valid solution, unmatched nodes cannot be neighbors.
- Since the label M appears only in the pair $\{\mathsf{M}, \mathsf{M}\}$, and since each matched node must have exactly one incident matched edge, matched edges form an independent set.

This implies that a solution for $\Pi^{\mathsf{linearizable}}$ can be converted into a maximal matching in 0 deterministic LOCAL rounds.

We now show that a maximal matching can be converted into a solution for $\Pi^{\mathsf{linearizable}}$ in 0 deterministic LOCAL rounds. A solution for $\Pi^{\mathsf{linearizable}}$ can be computed as follows:

- Each unmatched node labels P all its incident half-edges.
- Each matched node labels M its incident half-edge e that is part of the matching, B all edges that come before e in the given ordering, and A all edges that come after e in the given ordering.

It is easy to see that the computed solution satisfies the constraints of $\Pi^{\mathsf{linearizable}}$. ◀

Proof of Lemma 18. Let $\mathcal{A}$ be an SLOCAL algorithm for $\Pi^{\mathsf{linearizable}}$ with complexity $T(n)$. We show how to use $\mathcal{A}$ to solve Π^{promise} with SLOCAL complexity $O(T(n) \log n)$. As argued in Section 4.2, this implies a solution for Π with the same asymptotic SLOCAL complexity.

Let $G \in \mathcal{G}$ be the graph in which we want to solve Π^{promise}. Consider the virtual bipartite graph $\widehat{G}$ obtained by contracting each octopus gadget into a single node (see Figure 2b), that is:

- For each octopus gadget g of G, there is a white node v_g in $\widehat{G}$.
- For each inter-cluster node b of G, there is a black node u_b in $\widehat{G}$.
- For each edge connecting an inter-cluster node b of G to an octopus gadget g of G, there is an edge between v_g and u_b in $\widehat{G}$.

Note that $\widehat{G}$ may contain parallel edges. Since the diameter of a valid octopus gadget is clearly upper bounded by $O(\log n)$, we get that the distances in G are at most an $O(\log n)$ factor larger than distances in $\widehat{G}$. Thus, it is possible to simulate the execution of an SLOCAL algorithm for $\widehat{G}$ with an $O(\log n)$ multiplicative overhead on G. We use $\mathcal{A}$ to solve $\Pi^{\mathsf{linearizable}}$ on $\widehat{G}$. For each octopus gadget g, we assign the solution of the i-th port of g_v to the nodes of the i-th port gadget of g, according to the natural left-to-right order of the port gadgets of g. The output clearly satisfies the constraints of Π^{promise}, and the runtime is upper bounded by $O(T(n) \log n)$. ◀

Proof of Lemma 20. By hypothesis, there exists a non-signaling outcome O that solves Π with failure probability $p(n)$. Consider any input hypergraph F for $\Pi^{\mathsf{linearizable}}$ of size n, and let G be the bipartite incidence graph of F. We construct a graph G' as a function of G as follows.

- For each white node v of degree d of G, we put an octopus gadget g_v with d port gadgets, each of height $\Theta(\log n)$, into G'.
- For each black node u of G, we put an inter-octopus node b_u in G'.
- Let v be an arbitrary white node in G, and let $\{v, u\}$ be its i-th incident edge, according to the given ordering. We put, in G', an edge connecting b_u to the left-most leaf of the i-th port gadget of g_v, according to the natural left-to-right order of the port gadgets of g_v.

By construction, G' is a proper instance, and by Lemma 15 it can be labeled such that $G' \in \mathcal{G}$. In the following, we assume that G' is labeled in such a way. Hence, G' is an input instance for Π^{promise}. Moreover, we get that if G has n nodes, then G' has $n \leq N \leq n^3$ nodes.

We define a function f_G as follows. Let v be a white node of G, and let r be the root node of the i-th port gadget of g_v. The function f_G maps r to the i-th edge incident to v, according to the given ordering. It is straightforward to see that f_G maps solutions for Π^{promise} on G' to solutions for $\Pi^{\mathsf{linearizable}}$ on G.

Let $V_r(G')$ be the domain of f_G, and let $\Sigma_{\mathsf{promise}}$ be the set of output labels for Π^{promise} for the nodes in $V_r(G')$. Let $\{(\mathrm{out}_i, p_i)\}_{i \in I}$ be the output distribution that O defines on G' for Π. As discussed in Section 4.2, this is also a valid output distribution for Π^{promise}. Note that G is a bipartite incidence graph and $\Pi^{\mathsf{linearizable}}$ asks only to label edges of G. This means that only half-edges of F are labeled. We now define an outcome O′ on G just by describing output labelings on edges of G (which correspond to half-edges of F) – more specifically, we set $\mathrm{O}'(G) = \{(\mathrm{out}_i \circ f_G^{-1}, p_i)\}_{i \in I}$.

First, it is clear that the sum of all p_i in $\mathrm{O}'(G)$ is exactly 1. Furthermore, it is straightforward to check that O′ has failure probability at most $p(n) > 0$. If not, by construction of O′, then $\mathrm{O}(G')$ has failure probability strictly greater than $p(n)$ for Π^{promise}, which is a contradiction because O has failure probability $p(N) \leq p(n)$ by monotonicity of p.

We now claim that O′ is non-signaling beyond distance $T'(n)$. Recall that each octopus gadget in G' represents a white node v of G, and neighboring octopus gadgets represent neighboring white nodes of G. Hence, for every subset of edges A of $E(G)$, the distribution $\mathrm{O}'(G)[A]$ is defined only by $\{(\mathrm{out}_i, p_i)\}_{i \in I}[f_G^{-1}(A)]$. Let $k(n)$ be the height of a port gadget, and observe that $k(n) = \Theta(\log n)$. Suppose we modify G outside the radius-$T'(n)$ view of A and obtain a graph H with a subset of edges A_H such that $\mathcal{V}_0(A)$ is isomorphic to $\mathcal{V}_0(A_H)$ and $\mathcal{V}_{T'(n)}(A)$ is isomorphic to $\mathcal{V}_{T'(n)}(A_H)$. Notice that, as before, H also defines a proper instance $H' \in \mathcal{G}$ for Π^{promise}. However, because of the isomorphic regions between G and H, we get that $\mathcal{V}_0(f_G^{-1}(A))$ is isomorphic to $\mathcal{V}_0(f_H^{-1}(A_H))$, and $\mathcal{V}_{T'(n)\cdot k(n)}(f_G^{-1}(A))$ is isomorphic to $\mathcal{V}_{T'(n)\cdot k(n)}(f_H^{-1}(A_H))$, since each port gadget has height at least $k(n)$. We impose that $T'(n) \cdot k(n) \geq T(N)$, which is equivalent to asking that $T'(n) \geq T(N)/k(n)$. Since the distribution O is non-signaling beyond distance $T(N)$, we have that $\mathrm{O}(G')[\mathcal{V}_0(f_G^{-1}(A))]$ is the same distribution as $\mathrm{O}(H')[\mathcal{V}_0(f_H^{-1}(A))]$. Hence, $\mathrm{O}'(G)[A]$ and $\mathrm{O}'(H)[A]$ are equal, and O′ is non-signaling beyond distance $T'(n)$. Note that it is sufficient to take $T'(n) = O(T(n^3)/\log n)$, since $T(N)$ is non-decreasing in N and $n \leq N \leq n^3$. ◀

C KMW in a Nutshell

To instantiate our construction, we use a lower bound for approximating maximum matchings based on the KMW bound [16]. For completeness, we now describe the high-level idea of this bound, state it formally, and sketch the key components of its construction that are relevant for our work. See Coupette and Lenzen [11] for a detailed exposition and a simplified proof.

The KMW bound establishes that there exist graphs with n nodes and maximum degree $\Delta = 2^{\Theta(\sqrt{\log n \log\log n})}$ on which $\Omega(\sqrt{\log n / \log\log n})$ (expected) communication rounds are required to obtain polylogarithmic approximations to a minimum vertex cover, minimum dominating set, or maximum matching.

▶ **Theorem 27** ([16]). *There does not exist a randomized LOCAL algorithm providing an $O(\log \Delta)$-approximation of fractional maximum matching with locality $o(\sqrt{\log n / \log\log n})$ on graphs with degree $\Delta = 2^{\Theta(\sqrt{\log n \log\log n})}$.*

The KMW bound holds under both randomization and approximation, and it extends to symmetry-breaking tasks like finding maximal independent sets or maximal matchings via straightforward reductions.

At the core of the bound lies a class of high-girth graphs constructed from a blueprint, the *Cluster Tree*, which arranges differently-sized independent sets of nodes as a tree and prescribes that node sets adjacent in the tree are connected via biregular bipartite graphs. Both blueprints and graphs are parametrized by the number of communication rounds k, and they are designed to enable an *indistinguishability argument*: For a given k, the associated Cluster Tree graph contains two independent sets of nodes, one large and one small, such that both sets of nodes have isomorphic radius-k neighborhoods, but only the small set of nodes is needed to solve a given *covering* problem. This forces any algorithm to select a large fraction of the large node set into the solution (in expectation), yielding a poor approximation ratio.

The construction extends to *packing* problems by taking two copies of a Cluster Tree and additionally prescribing that each node in the first copy is connected to its counterpart in the second copy. Importantly, the graphs arising from Cluster Trees are bipartite by design. In bipartite graphs, the optimal fractional solution and the optimal integral solution coincide for both minimum vertex cover and maximum matching, and by Kőnig's theorem [13], the solution sets to both problems have the same cardinality. Moreover, the two-copy construction of Cluster Trees for packing problems has a natural bicoloring such that the large cluster in the first copy and the small cluster in the second copy have the same color – i.e., providing the bicoloring keeps the indistinguishability argument intact. Hence, the KMW bound is inherently a bound for (bipartite) fractional problems.