

Locality in distributed quantum and online graph algorithms

Henrik Lievonen



Aalto University

Aalto University publication series
Doctoral Theses 162/2026

Locality in distributed quantum and online graph algorithms

Henrik Lievonen

A doctoral thesis completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall T2 of the school on September 18, 2026 at 12:00.

Aalto University
School of Science
Department of Computer Science

Supervising professor

Prof. Jukka Suomela, Aalto University, Finland

Preliminary examiners

Prof. Artur Czumaj, University of Warwick, United Kingdom

Dr. Bernhard Haeupler, INSAIT, Sofia University “St. Kliment Ohridski”, Bulgaria,
and ETH Zurich, Switzerland

Opponent

Prof. Artur Czumaj, University of Warwick, United Kingdom

Aalto University publication series

Doctoral Theses 162/2026

© Henrik Lievonen

ISBN 978-952-64-3254-0 (paperback)

ISBN 978-952-64-3253-3 (PDF)

ISSN 1799-4934 (print)

ISSN 1799-4942 (PDF)

<https://urn.fi/URN:ISBN:978-952-64-3253-3>

PunaMusta Oy

2026

Author Henrik Lievonen

Name of the doctoral thesis Locality in distributed quantum and online graph algorithms

Article-based thesis

Number of pages 163

Keywords distributed graph algorithms, distributed quantum algorithms, LCL problems

In this thesis, I study how quantum computation and quantum communication can help large networks of computers to solve computational problems faster. I also connect the notion of *locality* between various models, including distributed quantum algorithms and online graph algorithms, and show that locality in online graph algorithms can be used to provide bounds for distributed quantum advantage.

Towards proving distributed quantum advantage, I present the first family of LCL problems which shows a quantum advantage for distributed graph algorithms. Here LCLs, or Locally Checkable Labeling problems, are a family of graph problems described by a finite set of local constraints, and I mainly focus on them in this thesis. The models of distributed computing which this result separates are the classical LOCAL model and the quantum-LOCAL model. In both of these models, a network of computers exchanges messages in synchronous rounds, and the running time of the algorithm directly corresponds with locality, that is, how far the computers see in the network.

In this thesis, I also introduce new models of computing that capture locality in stronger settings. Most notably, I introduce the online-LOCAL model which captures the locality of online graph algorithms, and show that it is, asymptotically, roughly as strong as the classical LOCAL model for solving LCL problems on rooted regular trees. I provide an online-LOCAL algorithm for finding a 3-coloring of a bipartite graph with lower locality than is possible in the classical LOCAL model, and show that the algorithm is tight.

To draw things together, I show that quantum-LOCAL and online-LOCAL are indeed related to each other: any quantum-LOCAL algorithm can be simulated in the randomized online-LOCAL model. The converse does not hold in general, but an online-LOCAL algorithm solving an LCL problem in rooted, possibly non-regular, trees can be simulated in the classical LOCAL model—and thus in the quantum-LOCAL model—with an exponential cost in locality, putting indirectly a bound on the maximum possible attainable quantum advantage.

Finally, I consider the sinkless orientation problem, which holds a central role in our understanding of the locality of LCL problems in the classical LOCAL model, and provide a simpler and more direct proof for its hardness using the supported LOCAL model. I also provide a simpler algorithm solving sinkless orientation in the SLOCAL model, which is a sequential version of the LOCAL model.

Tekijä Henrik Lievonen

Väitöskirjan nimi Paikallisuus hajautetuissa kvanttialgoritmeissa sekä online-verkkoalgoritmeissa

Artikkeliväitöskirja

Sivumäärä 163

Avainsanat hajautetut algoritmit, hajautetut kvanttialgoritmit, LCL-ongelmat

Tässä väitöskirjassa tutkin, kuinka kvanttilaskenta ja -kommunikaatio voivat tehostaa laskennallisten ongelmien ratkaisemista suurissa tietokoneverkoissa. Tarkastelen erityisesti *paikallisuuden* merkitystä useissa laskennan malleissa: tutkin sekä hajautettuja kvanttialgoritmeja että online-verkkoalgoritmeja, ja osoitan, että online-verkkoalgoritmeille osoitetut paikallisuusalarajat rajoittavat myös mahdollista kvanttietua.

Väitöskirjassani keskityn erityisesti tarkastelemaan *paikallisesti tarkistettavia merkitsemisongelmia* eli LCL-ongelmia. Ne ovat laaja joukko verkko-ongelmia, jotka voidaan määrittellä luettelemalla äärellistä joukkoa paikallisia rajoitteita. Työssä esittelenkin ensimmäisen perheen LCL-ongelmia, joka todistettavasti erottaa LOCAL- ja kvantti-LOCAL-mallit toisistaan, mikä samalla osoittaa paikallisen kvanttiedun olevan saavutettavissa. Molemmat näistä malleista ovat hajautetun laskennan malleja, joissa toisiinsa kytketyt tietokoneet synkronisesti vaihtavat viestejä keskenään ja päivittävät tilojansa vastaanottamiensa viestien pohjalta. Näissä malleissa laskennan kesto vastaa paikallisuutta, eli sitä, kuinka kauas tietokoneiden on nähtävä tietokoneverkossa ratkaistakseen ongelman.

Tässä väitöskirjassa esittelen myös uusia laskennan malleja, joissa paikallisuus on merkityksellinen. Huomionarvoisin näistä malleista on online-LOCAL, joka laajentaa online-verkkoalgoritmeja paikallisuuden käsitteellä. Osoitan, että online-LOCAL-malli on asympotoottisesti suunnilleen yhtä vahva kuin LOCAL-malli juurellisissa säännöllisissä puissa. Esittelen myös uuden online-LOCAL-algoritmin, joka 3-värittää kaksijakoisen verkon huomattavasti pienemmällä paikallisuudella kuin klassiset LOCAL-algoritmit, ja osoitan, että kyseisen algoritmin paikallisuusvaativuus on tiukka.

Yhdistääkseni nämä tutkimushaarat osoitan, että kvantti-LOCAL ja online-LOCAL liittyvät toisiinsa yllättävällä tavalla: satunnaistettu online-LOCAL on tarpeeksi vahva simuloidakseen kvantti-LOCAL-algoritmeja. Simulaatio ei päde vastakkaiseen suuntaan, mutta osoitan, että LCL-ongelmille poluissa, sykleissä ja juurellisissa, mahdollisesti epäsäännöllisissä puissa klassinen LOCAL- ja siten myös kvantti-LOCAL-pystyvät simuloimaan online-LOCAL-algoritmeja käyttäen eksponentiaalisesti suurempaa paikallisuutta. Tämä epäsuorasti rajoittaa suurinta mahdollista kvanttietua näille ongelmille.

Lopuksi käynnän katseeni *nieluton suunnistus* -ongelmaan, jolla on keskeinen rooli ymmärryksessämme LCL-ongelmien paikallisuudesta klassisessa LOCAL-mallissa. Annan kyseiselle ongelmalle uuden, aiempaa yksinkertaisemman ja suoremman alarajatodistuksen klassisessa LOCAL-mallissa käyttäen tuettua LOCAL-mallia. Esittelen myös aiempaa yksinkertaisemman SLOCAL-algoritmin nieluttomalle suunnistukselle; SLOCAL on LOCAL-mallin sekventiaalinen versio.

Preface

It took—depending on how you count—either four, ten, or thirty years to complete, but now it is done: my doctoral dissertation! However, just like with everything in life, this work would not have been possible without a lot of wonderful people around me, and I would like to thank you all!

First, my deepest gratitude goes to my thesis supervisor Jukka, who has not only guided me through my initial steps as a researcher, but has actually supported me throughout my entire academic journey by letting me work as a teaching assistant on his courses. Thank you for your support and guidance through all these years!

During these past years, I have also had the pleasure of working with many awesome collaborators, without whom none of the articles presented in this dissertation would have been possible. I am grateful to have been able to work with all of you, and I am looking forward to our future collaborations!

Through these collaborations, I have also found my first postdoc position with Sebastian. Thank you for believing in me!

It has also been a joy to share my workplace with so many wonderful people. I would like to mention, in alphabetical order, the following people: Amirreza, Antonio, Augusto, Avinandan, Barbara, Darya, Erik, Esya, Francesco, Geert, Hossein, Jan, Jara, Massimo, Maxime, Mélanie, Niklas, Ronja, Rustam, Sergey, and probably many others I have missed, but who I have had the pleasure of meeting in our coffee room! Thank you for entertaining me over a coffee and making the workspace lively!

My journey with algorithms, however, began long before this formal evaluation: it started with implementing them in programming competitions. Therefore, I owe many thanks to the Finnish competitive programming community, and especially to the following people: Antti and Topi, who first introduced me to competitive programming; Juha, with whom we revived the HIIT Open programming competition and have had the joy of organizing HIIT Frontier Friday; and Roope, with whom we handled the technical organization of the Baltic Olympiad in Informatics just this spring.

There are also many people who have had a profound effect on me outside of work. My deepest thanks go to my closest friends Alpi, Perttu, and Saskia, a

group which we collectively call 4D. Even though we never implemented the four-dimensional game we envisioned nearly a decade ago, we have experienced a lot together, including a few memorable trips abroad. We have also spent many long nights expanding our horizons through movies and games, discussing everything from existential questions to the role of quantum mechanics: I still cannot decide whether yesterday exists and time is just a lie, or whether I am going to live forever.

Through my doctoral studies, I also had the opportunity to share a sailing boat, the S/Y Capsicum, with some of you. While the upkeep took a lot of work, it was also nice to work with my hands and immediately see the results of my labor. Thank you to the whole crew!

I would also like to extend my gratitude towards my other friends, including everyone I have met through Kartano! Thanks to you, for most of my doctoral studies, I have not needed to think what to do on Wednesdays, and I am already beginning to miss the discussions in the sauna.

And then there is our DnD group, which has taken me on great adventures from floating cities to the depths of the Underdark. Thank you, Olli, Otte, Mari, Matias, and all the past players!

None of this would have happened if not for the support of my family: my parents Helka and Tero, and brother Tuomas. They have always been there supporting me in all my endeavors—even during my recent move abroad, which left my apartment in a state of temporary chaos. I am eternally grateful for your support!

Last but not least, I would like to extend my gratitude towards the pre-examiners, Prof. Czumaj and Dr. Haeupler, for taking the time to read this thesis and provide their feedback. Thank you, Prof. Czumaj, for also promising to work as an opponent in my defense.

I would also like to acknowledge the Nokia Foundation for supporting my research through the Nokia Scholarship, and the Research Council of Finland for supporting me through grants 333837, 359104, and 363558.

Saarbrücken, August 14, 2026,

Henrik E. M. Lievonen

Contents

Abstract	3
Tiivistelmä	5
Preface	7
Contents	9
List of Publications	11
Author's contributions	13
1. Introduction	15
1.1 Locality	16
1.2 Distributed quantum algorithms	17
1.3 Locality of probability distributions	17
1.4 Online graph algorithms	18
1.5 Publications and highlighted contributions	19
2. Quantum advantage for local problems	23
2.1 Graphs and networks	23
2.2 Local problems	25
2.3 Classical distributed algorithms	28
2.3.1 LCL problems in the LOCAL model	30
2.4 Distributed quantum algorithms	31
2.5 Main contribution: Local quantum advantage	34
3. Locality in online algorithms	39
3.1 Local views	39
3.1.1 On the strength of the local views formalism	41
3.2 Distributed sequential algorithms	41
3.3 New model: Online-LOCAL	43
3.4 Main contribution: 3-coloring in online-LOCAL	45

3.5	New model: Dynamic-LOCAL	48
4.	Connection between quantum and online algorithms	51
4.1	New model: Randomized online-LOCAL model . . .	51
4.2	Non-signaling can be simulated in online-LOCAL . .	52
4.3	Main contribution: Symmetry breaking in fin.dep. .	54
4.4	Main contribution: From online-LOCAL to SLOCAL	55
5.	Sinkless orientation	59
5.1	Sinkless orientation	59
5.2	New lower bound via supported LOCAL	61
5.3	New upper bound via high-degree sinkless orientation	61
6.	Conclusion	63
	References	65
	Errata	75
	Publications	77

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

- I** 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, Isadora Veeren. Distributed Quantum Advantage for Local Problems. In *57th Annual ACM Symposium on Theory of Computing (STOC 2025)*, ACM, 2025, pp. 451–462, June 2025.
- II** Amirreza Akbari, Navid Eslami, Henrik Lievonen, Darya Melnyk, Joona Särkijärvi, Jukka Suomela. Locality in online, dynamic, sequential, and distributed graph algorithms. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, Leibniz International Proceedings in Informatics (LIPIcs), Volume 261, pp. 10:1–10:20, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, July 2023.
- III** Amirreza Akbari, Xavier Coiteux-Roy, Francesco d’Amore, François Le Gall, Henrik Lievonen, Darya Melnyk, Augusto Modanese, Shreyas Pai, Marc-Olivier Renou, Václav Rozhoň, Jukka Suomela. Online Locality Meets Distributed Quantum Computing. In *57th Annual ACM Symposium on Theory of Computing (STOC 2025)*, ACM, 2025, pp. 1295–1306, June 2025.
- IV** Thomas Boudier, Filippo Casagrande, Avinandan Das, Massimo Equi, Henrik Lievonen, Augusto Modanese, Ronja Stimpert. Orientation Does Not Help with 3-Coloring a Grid in Online-LOCAL. In *29th International Conference on Principles of Distributed Systems (OPODIS 2025)*, Leibniz International Proceedings in Informatics (LIPIcs), Volume 361, pp. 19:1–19:16, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, January 2026.
- V** Alkida Balliu, Janne H. Korhonen, Fabian Kuhn, Henrik Lievonen, Dennis Olivetti, Shreyas Pai, Ami Paz, Joel Rybicki, Stefan Schmid, Jan

Studený, Jukka Suomela, Jara Uitto. Sinkless Orientation Made Simple.
In *2023 Symposium on Simplicity in Algorithms (SOSA 2023)*, Society
for Industrial and Applied Mathematics, pp. 175–191, January 2023.

Author's contributions

Publication I: "Distributed Quantum Advantage for Local Problems"

JS came up with the research question and coordinated the project. XCR and MOR came up with using iterated quantum games for a separation. XCR, FLG, MOR, LT and IV provided the quantum-information-theoretic background needed for the separation. HL and FdA applied round elimination on the CHSH problem by hand. This later led to SB observing a pattern in the round elimination sequence, which allowed AB and DO to generalize the proof into a lower bound for the GHZ problem. AM came up with the classical upper bound for networks of non-signaling games, which was later refined by HL and FdA. ME described the quantum algorithm for GHZ problem, which was refined by HL. All authors helped with refining the text.

Publication II: "Locality in online, dynamic, sequential, and distributed graph algorithms"

DM and JuS introduced the online-LOCAL model to their summer workers AA, HL and JoS in summer 2021. JuS wrote the introduction and DM wrote definitions, and both coordinated the project. AA, HL and JoS worked together on proving the classification of LCL problems in cycles and paths, and they also developed the 3-coloring algorithm together. HL proved the classification in rooted trees. AA and JoS drew illustrations. AA and JuS proved the hardness of 3-coloring in SLOCAL. NE worked as a summer worker in 2022, and during this time, he proved separations between dynamic-LOCAL and SLOCAL. All authors helped with refining the text.

Publication III: "Online Locality Meets Distributed Quantum Computing"

JS coordinated the work. Main contributors for symmetry-breaking problems on finitely-dependent processes were HL, AA, FdA and JS. VR observed that randomized online-LOCAL can simulate non-signaling distributions, and this was proven by FdA. HL and FdA came up with turning online-LOCAL into component-wise online-LOCAL, and how component-wise algorithms can be simulated in SLOCAL on rooted trees. They also showed that an oblivious adversary is necessary in the online-LOCAL model. HL, DM, AM and SP proved hardness of 3-coloring in randomized online-LOCAL. AA provided the landscape of LCL problems on paths and cycles in randomized online-LOCAL. XCR and MOR wrote an appendix on the bounded-dependence and non-signaling arguments for the full version of the publication. All authors helped with refining the text.

Publication IV: "Orientation Does Not Help with 3-Coloring a Grid in Online-LOCAL"

The research project started in February 2025 at the RW-DIST research workshop where the authors discussed generalizing the existing 3-coloring lower bound for grids [CMN+24] to work also in the oriented case. During the workshop, TB came up with the idea of quasi-linear boosting. In May, HL came up with the idea of boosting on slopes and L-construction. FC wrote the introduction and summary of the previous lower bound. AD and ME wrote L-construction and the final contradiction. HL and AM refined and wrote the boosting construction using parallelograms. RS drew the visualizations. All authors helped with refining the text.

Publication V: "Sinkless Orientation Made Simple"

This paper is a combination of two somewhat separate research projects into simplifying the classification of sinkless orientation in two different models. The lower bound for the supported LOCAL model was introduced and written by AP, JK, JR, SS and JuS. The original idea for the upper bound algorithm came up during the AB, FK and DO's visit of HL, SP, JaS, JuS and JU at Aalto University. FK presented the idea for an SLOCAL algorithm for solving high-degree sinkless orientation. This idea was refined over the course of their week-long visit. HL wrote down the SLOCAL algorithm and proofs of its correctness. All authors helped with refining the text.

1. Introduction

Many things in the world around us can be modelled as networks: roads and streets form a *street network*, people knowing each other form a *social network*, and computers connected to each other form a *computer network*. While superficially different, these networks share many common properties, and they can be studied using the same tools. The most obvious shared property is their structure: all networks consist of interconnected entities. In graph theory, which is the study of networks, these entities are called *nodes*, and the connections between them are called *edges*.

Another example of common properties is the notion of *distance*, or *how far two nodes are in the network*. In street networks, distance corresponds to physical distance between two places, which is a good estimate of the travel time between those places. In social networks, it can help to model how fast information—or disease—spreads. And, in computer networks, distance provides an estimate of *latency* between two computers; gamers also know this as *ping*.

In all these networks, we can also imagine that nodes correspond to active entities that try to accomplish something. This can be literal computers interacting to solve a large system of equations, but it can also be people in a social network coordinating when to meet with their friends, or the traffic lights at intersections coordinating when to turn green to ensure smooth flow of traffic—without sacrificing the safety, of course. All these examples, and many more, fall under the umbrella of *distributed computing*, which studies how multiple interconnected entities work together.

To facilitate the study of distributed computing, we need *models of computing*, which formalize how interaction happens. For example, in a simple model, we may assume that all nodes of the graph hold some local state, and they synchronously exchange messages with their neighbors, always updating their state when they receive a message. In fact, the above model, called the LOCAL model [Lin92; Pel00], holds a central place in the study of distributed computing as it captures *locality*—the idea that nodes must make decisions based on limited, local information.

1.1 Locality

Let us take a closer look at the LOCAL model and analyze *what* an arbitrary node v knows after exchanging messages for a few rounds:

- Initially, node v knows almost nothing about the graph: it knows that it itself exists, how many neighbors it has, and how to contact them. If the node has a local input, it also knows that. However, it does not know anything about those neighbors yet. As a technical detail, the node also knows its own unique identifier, which is a natural number that no other node of the graph shares.
- After one round of communication, node v receives information from its neighbors: it knows their identifiers and how many neighbors they have—or, their *degrees*. However, it does not know anything about the neighbors' neighbors. As all communication happens synchronously, information just has not had enough time to travel that far.
- After two rounds, node v may learn something also about its neighbors' neighbors, but again, nothing about their neighbors: the information node v knows is restricted to the *radius-2 neighborhood* of the node.

The last observation generalizes to any number of rounds T : what a node knows is restricted to the radius- T neighborhood of that node, and thus its decision may not depend on the graph structure or inputs further than distance T from itself. Once a node makes a decision and announces its output, it has spent some number T of rounds communicating with its neighbors. This number T is called the *locality* of the algorithm.

The locality actually gives a sharp threshold: while, in T rounds of communication, a node cannot learn anything about the input further than distance T away, it can learn *everything* of the input within distance T . This is because every time a node sends a message to its neighbor, it can, without loss of generality, send everything it knows. This is possible as the LOCAL model does not restrict the size of the messages nor the local computation in any way.

As algorithms in the LOCAL model need to base their decisions only on the local information in the graph, the computational problems should also reflect that. A well-suited and widely-studied class of problems is Locally Checkable Labeling problems [NS95], or LCLs for short. These are problems admitting a local and finite description: The nodes and edges of the graph have local inputs on them, and the computation must produce local outputs. A labeling is correct if every constant-radius neighborhood of the graph looks like one of the neighborhoods from a finite set of allowed neighborhoods given by the LCL problem.

1.2 Distributed quantum algorithms

The recent rise of research into quantum computing raises an interesting question: does adding quantum resources to distributed systems help with the locality of solving problems? The model of distributed quantum computing that I use in this thesis is called quantum-LOCAL [GKM09]. In this model, as in the LOCAL model, the nodes of the network hold state registers, but this time they are quantum registers holding quantum state. In addition to this, the nodes exchange quantum messages carrying qubits that can potentially be entangled with the state of the sender. When a node receives a message, it may entangle its own state with the message, which further can entangle its state with the states of its neighbors.

It is essential for the model that both the states of the nodes and the messages they exchange are quantum; if either of them were classical, then the model would collapse back to LOCAL. If the model used only quantum communication but classical states, then the communication could possibly be sped up with the help of superdense codings [BW92]. However, already in the LOCAL model, we do not assume there to be any limit on the amount of communication per round, and thus, quantum communication alone would not provide distributed quantum advantage for the LOCAL model.

On the other hand, the quantum-LOCAL model could be defined to have quantum states but only classical communication. In centralized setting, it is known that quantum computers can speed up some computations polynomially [Gro96] or even exponentially [Sho97]. However, again, in the LOCAL model, we impose no limit on the amount of local computation the nodes may perform each round, and thus this would not be enough for quantum advantage.

Just as the LOCAL model captures locality of distributed computation, the quantum-LOCAL model captures locality of distributed quantum computation. And just like in LOCAL, the output of a node after T rounds of communication is dependent only on the radius- T neighborhood of the node. The node can still collect the whole structure of the input in this neighborhood. However, the node may also entangle itself with some or all of the nodes in its radius- T neighborhood, and may use that when solving a problem.

1.3 Locality of probability distributions

The output of a quantum algorithm may also be seen as a probability distribution over outputs; let us call that an *outcome*. The outcomes of quantum-LOCAL algorithms are not arbitrary; instead, they have some local properties due to the locality of the algorithm: the output, and thus the outcome, of a node does not depend on the input graph far away from the node. This observation actually implies two slightly different properties: *bounded-dependence* and *non-signaling*.

The bounded dependence of an outcome means that the output distributions of two far-away nodes are independent. The locality of dependence is the distance T at which the output distributions of any two nodes are independent. When the distance of dependence is a constant, then the distribution is said to be *finitely dependent*.

Finitely dependent distributions have been previously studied by mathematicians interested in stochastic processes [BGM93]. They have shown that there exist finitely dependent distributions for 3- and 4-coloring paths and cycles [HHL18; HL16] and 4-coloring lattices [Hol24]. However, whether finitely-dependent colorings exist for general graphs has remained open until Publication III. In parallel with our work, Timár independently found a finitely-dependent coloring of general graphs [Tim24]. However, they did not extend the result to all LCLs with locality $O(\log^* n)$ in the LOCAL model.

The non-signaling property states that the outcome of a node is independent of the input or the graph structure beyond its radius- T neighborhood. In particular, if the radius- T neighborhoods of two nodes in two different graphs look isomorphic, then their outcomes should also coincide.

1.4 Online graph algorithms

In addition to considering distributed algorithms, there are other families of graph algorithms that may benefit from the concept of locality. One of those families is *online graph algorithms*. In traditional online graph algorithms, there is an underlying input graph that is revealed to a centralized algorithm one node at a time. When a node is revealed to the algorithm, all edges connecting that node to previously revealed nodes are also shown. The algorithm must then process the node and make an irrevocable decision on the output of that node before the next node is revealed.

In a sense, these types of algorithms have locality somewhere between 0 and 1: when a node is processed, the algorithm may inspect all the previously revealed nodes in the neighborhood, but none of the future nodes. This gives a natural place to introduce locality into the model: when the algorithm processes a node, in addition to seeing all previously processed nodes, it gets to inspect all nodes in the radius- T neighborhood of the processed nodes—including nodes that have not been processed yet. We call this model the online-LOCAL model.

This model is closely related to the SLOCAL model [GKM17]: In the SLOCAL model, as in the online-LOCAL model, the algorithm processes nodes one-by-one and must make irrevocable decisions on the output of the node based on its radius- T neighborhood. However, unlike in an online-LOCAL algorithm, an SLOCAL algorithm does not remember all previously processed nodes. Instead, it can only inspect its radius- T neighborhood, including the outputs of previously processed nodes. In fact, the algorithm can store arbitrary auxiliary information at the nodes when processing them, and it can use this information

when processing future nodes in the neighborhood.

An online-LOCAL algorithm may thus be interpreted as an SLOCAL algorithm with access to *global* memory. On the other hand, when processing a node, an SLOCAL algorithm may simulate a LOCAL algorithm in its neighborhood. In this way, SLOCAL is sandwiched between LOCAL and online-LOCAL.

Finally, locality may also be introduced into dynamic graph algorithms. These are algorithms which need to maintain a solution to a problem while the graph changes. These changes may be insertions or removals of edges or nodes, but for simplicity, I am assuming that initially all nodes exist, and edges are only added, never removed. Typically, a dynamic algorithm is allowed to change the solution anywhere in the graph, but it needs to minimize the total number of label changes it performs or the total computation time it uses. However, in the dynamic-LOCAL model, the output label changes are restricted to be in the local radius- T neighborhood of the modified edge; the computation time is not restricted. If the locality T is small enough, the total number of changes the algorithm performs is also small.

Like the SLOCAL model, the dynamic-LOCAL model is also sandwiched between LOCAL and online-LOCAL: A dynamic-LOCAL algorithm can just simulate a LOCAL algorithm everywhere, and the locality of the simulated algorithm guarantees that the output never changes far from the modification. On the other hand, when an online-LOCAL algorithm processes a node v , it may feed the radius- T neighborhood of v to the dynamic-LOCAL algorithm; after this, no edges can be inserted into the neighborhood of node v , and thus the dynamic-LOCAL algorithm cannot modify the output of node v anymore.

1.5 Publications and highlighted contributions

My main contribution in this thesis is to study the notion of locality in various computational settings. Figure 1.1 summarizes the relationship between all the above-mentioned models and their relations, and it highlights some of the key contributions of this thesis.

In Publication I, I introduce the first local problem that admits a distributed quantum advantage. While the local problem I present is a *family* of LCL problems, not a single LCL problem, it still admits a local description:

Key contribution 1.1. *There exists a family of LCL problems which separates the LOCAL and the quantum-LOCAL models.*

Moreover, in a follow-up work, we build on Publication I and provide a genuine LCL problem showing quantum advantage [BCd+26]. In Chapter 2, I discuss this result. I also provide formal definitions of LCL problems and both the LOCAL and quantum-LOCAL models.

In Publication II, I introduce the online-LOCAL model which brings the concept of locality into online graph algorithms. I connect the locality of

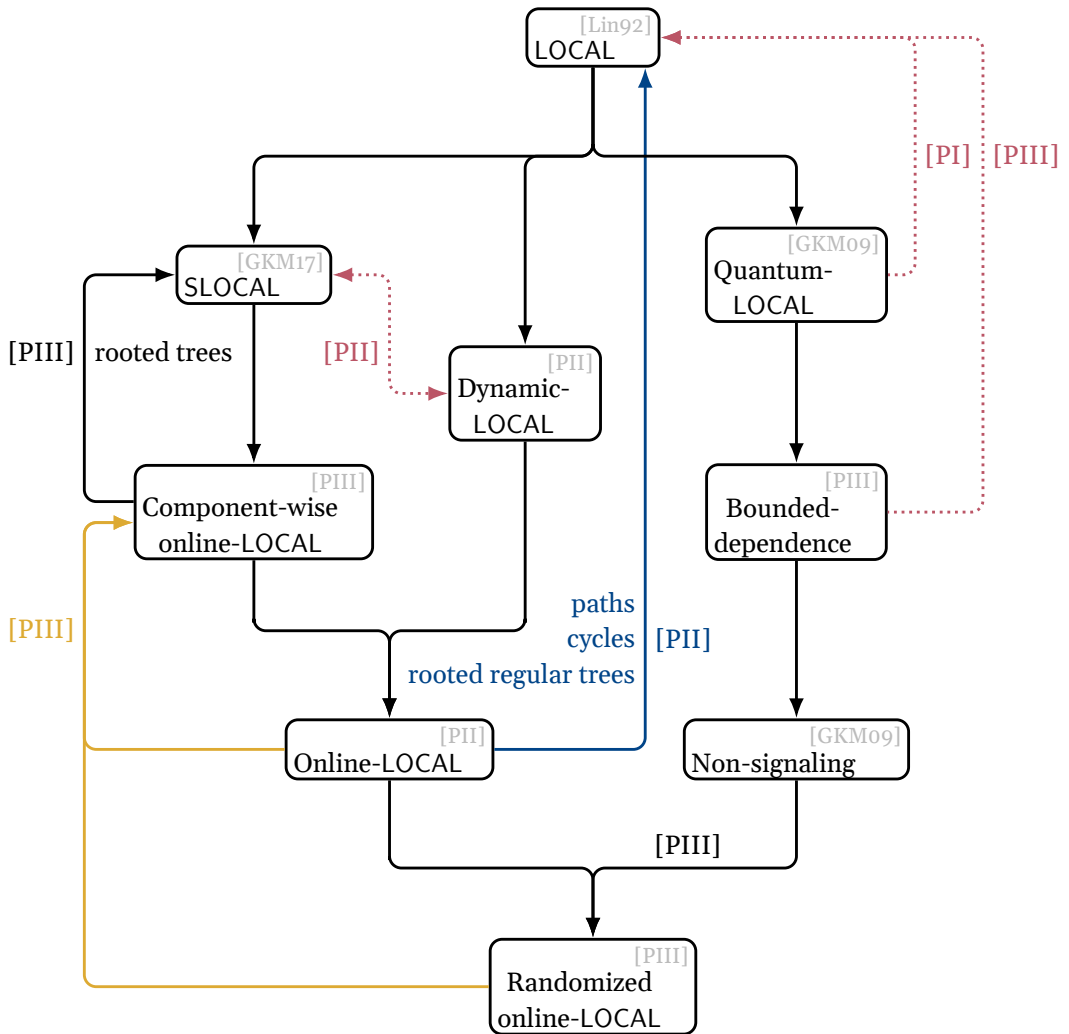


Figure 1.1. Summary of the results of Publications I to III. A black arrow $A \rightarrow B$ means that model B is asymptotically at least as strong as model A for LCL problems, that is, if an LCL problem can be solved with locality T in model A , then it can be solved with locality $O(T)$ in model B . A red dotted arrow $A \cdots \rightarrow B$ indicates lack of a black arrow, that is, model B is weaker than model A . A yellow arrow $A \rightarrow_{\text{exp}} B$ indicates a simulation in a weaker model with an exponential cost in the locality. A blue arrow $A \rightarrow_{\text{poly}} B$ indicates a simulation with additive $O(\log^* n)$ cost.

LCL problems in cycles, paths and rooted regular trees with their locality in the LOCAL model. Additionally, I provide a novel 3-coloring online-LOCAL algorithm that works on bipartite graphs. I provide more details on the online-LOCAL model in Chapter 3.

Key contribution 1.2. *There exists an online-LOCAL algorithm that finds a 3-coloring of a bipartite graph with locality $O(\log n)$.*

Since its publication in 2023, Publication II has provoked follow-up research, most notably showing connections to quantum algorithms [Publication III], that the 3-coloring algorithm is tight in grids [CMN+24, Publication IV], partial classification of LCLs in trees [DKL+24], and a novel lower bound for sinkless orientation [BG26].

In Publication III, I connect the complexity of solving LCL problems in the quantum-LOCAL and online-LOCAL models: I show that algorithms in the quantum-LOCAL model imply probability distributions that both have a bounded dependence and are non-signaling. I then continue to show that such distributions can be sampled in the online-LOCAL model, proving that the online-LOCAL model is at least as strong as the quantum-LOCAL model. In addition, I show that there exist finitely dependent distributions for $(\Delta + 1)$ -coloring, which leaves open the possibility that coloring would have locality $o(\log^* n)$ in the quantum-LOCAL model. Finally, I show that sufficiently-low-locality online-LOCAL algorithms imply the existence of low-locality SLOCAL algorithms in rooted trees.

Key contribution 1.3. *The online-LOCAL model is at least as strong as the quantum-LOCAL model for LCL problems.*

In Chapter 4, I present these contributions in more detail.

In Publication IV, I continue the analysis of the 3-coloring algorithm introduced in Publication II and show that the locality of the algorithm is already tight in oriented grids. This improves on the previous bound which was provided for unoriented grids [CMN+24]:

Key contribution 1.4. *The locality of solving 3-coloring in oriented grids in online-LOCAL is $\Theta(\log n)$.*

I introduce the key ideas of the proof in Section 3.4.

Finally, in Publication V, I inspect one of the most fundamental problems in the research of LCLs in the LOCAL model: the sinkless orientation. This is the problem of orienting the edges of a graph such that every node has at least one adjacent edge oriented outwards. It is known that the locality of sinkless orientation is $\Theta(\log n)$ in the deterministic LOCAL model and $\Theta(\log \log n)$ in the SLOCAL model. However, the previous bounds have relied on the fact that the complexity of the problem in *randomized* LOCAL is $\Theta(\log \log n)$ [BFH+16; GS17], and have then used either a gap result [CP19] or derandomization [GHK18] to get the bounds for deterministic LOCAL and SLOCAL, respectively.

In Chapter 5, I demonstrate a more direct lower bound for the complexity of sinkless orientation in the deterministic LOCAL model via the supported LOCAL model, and provide a self-contained SLOCAL algorithm working with locality $O(\log \log n)$ based on high-degree sinkless orientation.

Key contribution 1.5. *There exists a self-contained SLOCAL algorithm solving sinkless orientation with locality $O(\log \log n)$.*

2. Quantum advantage for local problems

One of the central questions in the study of distributed computing is whether quantum resources help with solving distributed problems. In general, we know that they do help: for example Le Gall, Nishimura, and Rosmanis provided a graph problem which has locality $O(1)$ in quantum-LOCAL, but requires $\Omega(n)$ locality in the classical LOCAL model [LNR19]. However, to verify the correctness of an output to their problem, one needs to see the whole graph. In other words, their problem is *global*.

In this thesis, I present the first known *local* problem that shows asymptotic quantum advantage. I start by defining a few graph concepts and what exactly I mean by local problems. I then proceed to define the LOCAL and the quantum-LOCAL models of distributed computing. Finally, I define the local problem that shows a distributed quantum advantage. I give some intuition on why this problem shows quantum advantage, but I defer the details to Publication I.

2.1 Graphs and networks

Computer networks are most commonly modelled as graphs, and hence I will also use graphs a lot in this thesis. I now define the core graph concepts and conventions used in this thesis.

Definition 2.1 (Graph, degree and distance).

A graph G consists of a pair (V, E) of sets of vertices, which are also called *nodes*, and edges. The edge set E consists of unordered pairs $\{u, v\}$ of vertices of set V . When the graph in question is ambiguous, the vertex set of graph G can be denoted by $V(G)$ and the edge set by $E(G)$. On the other hand, when the graph is clear, the subscript G may be dropped from following definitions.

Degree $\deg_G(v)$ of node v denotes the number of edges adjacent to v :

$$\deg_G(v) = |\{\{v, u\} \in E\}|.$$

The degree of a graph is the maximum of the degrees of its nodes:

$$\deg(G) = \max_{v \in V(G)} \{ \deg_G(v) \}.$$

All graphs considered in this thesis have a finite degree. I use Δ to refer to the degree of a graph, and n to the number of vertices of a graph.

A walk is a sequence $v_0 v_1 v_2 \dots v_k$ of vertices connected by edges, that is $\{v_i, v_{i+1}\} \in E$ for each $i \in [0, k)$. The length of a walk is the number of edges in it, that is the above walk has length k . Distance between nodes v and u , denoted by $\text{dist}_G(v, u)$, is the minimum length of all walks connecting v and u ; by convention, if no such walk exists, the distance is ∞ .

The distance between a vertex and an edge is defined to be

$$\text{dist}_G(v, \{u, w\}) = \min \{ \text{dist}_G(v, u), \text{dist}_G(v, w) \} + \frac{1}{2}.$$

Finally, the distance between two sets of vertices is the minimum distance between any two vertices of the set:

$$\text{dist}_G(V, U) = \min \{ \text{dist}_G(v, u) \mid v \in V, u \in U \}.$$

Distance function dist provides a metric over the vertices of the graph. This enables defining neighborhoods of vertices:

Definition 2.2 (Neighborhood).

Let $G = (V, E)$ be a graph, let $v \in V$ be one of its vertices, and let r be a number of the form $k/2$ for a natural number k , that is, either nonnegative integer or half-integer. Then *the radius- r neighborhood of vertex v* , denoted by $B_G(v, r)$, is the subgraph H of G consisting of all vertices and edges that are within distance r of v , that is

$$V(H) = \{ u \in V(G) : \text{dist}(v, u) \leq r \}$$

$$E(H) = \{ e \in E(G) : \text{dist}(v, e) \leq r \}.$$

For a set $S \subseteq V$, I overload the notation $B_G(S, r)$ to mean the graph

$$B_G(S, r) = \bigcup_{v \in S} B_G(v, r).$$

A related concept that arises naturally out of neighborhoods is *centered graphs*. These are just pairs of a graph and a distinguished node called a center:

Definition 2.3 (Centered graph).

Let $G = (V, E)$ be a graph, and let $v \in V$ be one of its vertices. Then the pair (G, v) is a *graph centered at v* . The radius of centered graph (G, v) is the maximum distance from v to any vertex or edge of G .

When discussing distributed algorithms, it is often useful to be able to distin-

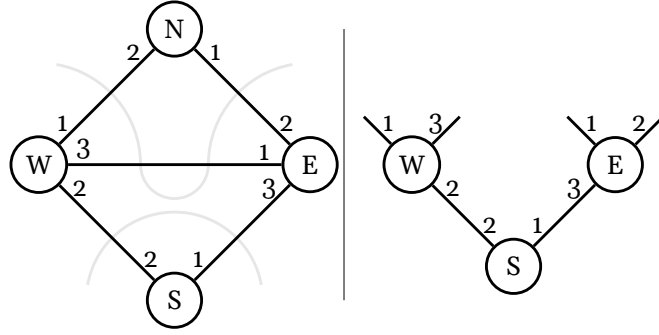


Figure 2.1. A port-numbered network is shown on the left: each node has a unique port number for each one of its adjacent edges. The gray lines show the radius-0 and radius-1 neighborhoods of the node S. On the right, the radius-1 neighborhood of node S is visualized without the rest of the graph. Note that the node does not know that the port 3 of node W and the port 1 of node E are connected; it would, however, see the connection in the radius- $\frac{3}{2}$ neighborhood.

guish between the edges neighboring a vertex by an *index*. If the distributed system is thought of a network of interconnected computers, then these indices would correspond to physical ports on the computers, and hence they are also called *port numbers*:

Definition 2.4 (Port-numbered network [Ang80]).

Let $G = (V, E)$ be a graph, and let a *port numbering* $\{p_v\}_{v \in V}$ be a family of maps such that for each node $v \in V$, function p_v is a bijection from edges neighboring v to numbers $[1, \deg_G(v)]$. For notational convenience, I will be referring to this family of maps by $p = \{p_v\}_{v \in V}$. A *port-numbered network* N is the pair (G, p) of a graph and its port-numbering. I also overload p to mean the set of ports on N —that is, the pairs of a vertex and an adjacent edge.

The concept of port-numbered networks extends naturally to neighborhoods and centered graphs: a radius- r neighborhood of node v includes the degrees and ports of all nodes with distance at most r to node v , but it does not include the connections between nodes at exactly distance r .

Figure 2.1 shows an example of a port-numbered network and the radius-1 neighborhood of one of its nodes.

2.2 Local problems

The main contribution of this thesis is to study the concept of *locality of computation* in various settings. However, to make this fair, we need to restrict our attention to *local problems*—after all, it would be unfair to ask a computer with only access to its local neighborhood to, for example, count the number of computers in the whole network; there is just no way to do that without

gathering information from every other node of the network.

A natural way to define local problems is that each vertex or edge of the graph needs to be assigned a label such that the labeling satisfies local constraints. These local constraints are usually of the form

The label of a vertex should come from set $\{1, 2, \dots, c\}$ and be distinct from the labels of all of its neighbors.

or

Edges should be labeled with either 0 or 1 such that edges labeled with 1 are not adjacent to other edges with label 1, and edges labeled with 0 are adjacent to at least one edge with label 1.

These problems are known as the *c-coloring problem* and the *maximal matching problem*, respectively.

These problems, and many more, can be unified under the class of *Locally Checkable Labeling problems*, or *LCLs* for short, introduced by Naor and Stockmeyer in early 1990s [NS93; NS95]. This is the class of problems where each vertex and edge has an input label, and the task is to assign output labels for them subject to local constraints. Intuitively, a labeling is correct if every node can inspect its local, constant-radius, neighborhood, and verify that the labeling is locally correct. Conversely, if the labeling is not correct, then at least one node “sees” an error in the labeling in its local neighborhood. To allow for a unified description of both vertex and edge labeling problems, I define the labels to be on ports:

Definition 2.5 (Locally Checkable Labeling problem, LCLs [NS95]).

An LCL problem Π consists of an *input label set* Σ , an *output label set* Γ , a positive integer *checking radius* r , and a finite set $\mathcal{C}$ of *local labeling constraints*. Each constraint of $\mathcal{C}$ is a centered port-numbered network of radius at most r with each port labeled from set $\Sigma \times \Gamma$.

Given a network $N = (G, p)$ and a port labeling $\lambda: p \rightarrow \Sigma \times \Gamma$, the labeling λ is *valid* if for each vertex v , the radius- r neighborhood of v is isomorphic to some element of $\mathcal{C}$, including the labeling but excluding the port numbering.

Note that, without loss of generality, we may assume that also the input and output label sets Σ and Γ are finite. This is because the set of local labeling constraints $\mathcal{C}$ contains only finitely many networks with finitely many ports (recall that all graphs and networks in this thesis have finite degree), and hence in total they also use only finitely many different labels. Any label that is not present in any network of $\mathcal{C}$ cannot be used in any valid labeling of an input network, and hence we can prune them out of Σ and Γ without affecting the

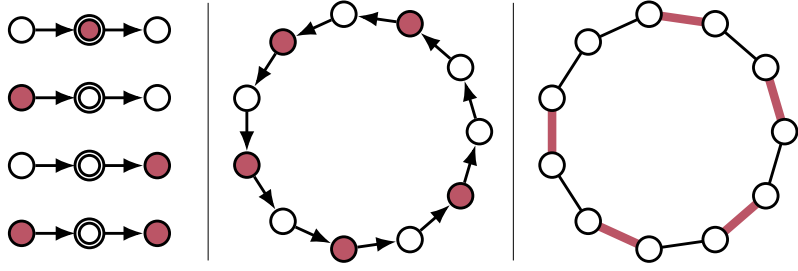


Figure 2.2. Examples of LCL problems. The left picture represents the set of valid centered neighborhoods of the maximal independent set problem in oriented cycles; the nodes of the independent set are shaded, and the node with double border is the center of the neighborhood. In the middle, a proper solution for maximal independent set is visualized. On the right, a maximal matching is shown; here selected edges are shaded.

set of valid labelings.

While this finiteness in the definition of LCL problems may seem arbitrary at first, it is exactly what makes them so useful and allows us to develop strong complexity theory for them in the LOCAL model; I will discuss this more in Section 2.3.1. In fact, in 2025, the article by Naor and Stockmeyer was awarded Edsger W. Dijkstra Prize in Distributed Computing in part for their definition of LCL problems [ACK+25].

The above definition of LCL problems is quite generic, but it also captures many natural subclasses of LCL problems, for example problems without inputs, and problems with labels defined only on vertices or edges: Input-free problems are just the special case with $|\Sigma| = 1$, that is the input labeling does not convey any information to an algorithm. Vertex-labeling problems can be encoded by requiring that all ports around a node have the same label. Similarly, edge-labeling problems can be encoded by requiring that both endpoints of every edge share the same label. Both of these can be encoded in the neighborhoods in $\mathcal{C}$.

The class of LCL problems is rich and includes many problems traditionally studied in graph theory. In addition to the vertex coloring and maximal matching problems on bounded-degree graphs presented at the start of this section, LCLs also include problems like

- maximal independent set,
- minimal dominating set,
- perfect matching,
- c -edge coloring,
- distance- k coloring with c colors.

Since the constraint set $\mathcal{C}$ is required to be finite, all of these problems are defined on bounded-degree graphs. Figure 2.2 shows a few examples of LCL problems.

One particularly interesting LCL problem is *sinkless orientation*, that is the problem of orienting edges such that every node (of degree at least 3) has at

least one outgoing edge. This is a problem that I will discuss more in Chapter 5.

2.3 Classical distributed algorithms

A distributed system, in its simplest form, consists of a network of interconnected computers that execute the same program. The computers can communicate with each other over their communication links, and—for simplicity—we may assume that the communication is synchronous—that is, every computer in the network exchanges messages at the same time. Moreover, we may assume that the computers and communication links experience no faults—handling those is a whole different field of research [HKR14].

The model I have just described—with the addition of unique identifiers for each computer—is the LOCAL model of distributed computing, first introduced by Linial in the late 1980s [Lin87; Lin92]. Formally, the model is defined as follows:

Definition 2.6 (LOCAL model, [Lin92; Pel00]).

Let Σ and Γ be the input and output label sets, respectively. An algorithm $\mathcal{A}$ in the LOCAL model now consists of

- a set of states S and a set of messages M ,
- a subset of halting states $H \subseteq S$,

and, for each degree $d \in \mathbb{N}$,

- an initialization function $\text{init}_d: \mathbb{N} \rightarrow \Sigma^d \rightarrow S$,
- a sending function $\text{send}_d: S \rightarrow M^d$,
- a receiving function $\text{recv}_d: S \rightarrow M^d \rightarrow S$, and
- an output extraction function $\text{out}_d: H \rightarrow \Gamma^d$.

An input instance for LOCAL algorithm $\mathcal{A}$ consists of a port-numbered network (G, p) , an injective function $\text{id}(v)$ giving each vertex of G a unique integer identifier from a polynomial space in $|V(G)|$, and an input labeling $\sigma: p \rightarrow \Sigma$ on the ports of G . By polynomial space, I mean that, for each node $v \in V(G)$, it must hold that

$$\text{id}(v) \leq \text{poly}(|V(G)|).$$

The execution of algorithm $\mathcal{A}$ now proceeds as follows: Each node $v \in V(G)$ starts by initializing its state:

$$S_v^0 = \text{init}_{\deg_G(v)}(\text{id}(v), \sigma_v),$$

where σ_v is the vector of labels of ports around v , that is

$$\sigma_v = [\sigma(v, p_v^{-1}(1)), \dots, \sigma(v, p_v^{-1}(\deg_G(v)))].$$

In effect, every node gets to know its unique identifier and local input, and stores it into its local state.

The execution proceeds in synchronous rounds. In round t , every node first forms a vector of messages it sends to its neighbors using `send` function based on its current state:

$$m_{\text{send}}^t[v] = \text{send}_{\deg_G(v)}(S_v^t).$$

Port-numbering is then used to route the messages to adjacent nodes:

$$m_{\text{recv}}^t[v, p_v(e)] = m_{\text{send}}^t[u, p_u(e)]$$

for each edge $e \in E(G)$ and vertices $u, v \in e$. Finally, each node updates its state using function `recv` based on the messages it receives:

$$S_v^{t+1} = \text{recv}_{\deg_G(v)}(S_v^t, m_{\text{recv}}^t[v]).$$

States in set H are halting states. For them, it is required that `recv` does not change the state, that is, it is idempotent in its first argument. Once the state of a vertex v reaches H , the computation at vertex v is said to *halt*. Once all vertices of the graph halt, the computation for the whole graph is said to have halted, and the output labeling can be extracted as follows:

$$\gamma: p \rightarrow \Gamma = (v, e) \mapsto \text{out}_{\deg_G(v)}(S_v^T)[p_v(e)],$$

that is, function `out` is used to convert the state of a vertex into local output on the ports surrounding it. The smallest integer $T \in \mathbb{N}$ such that all states in S^T are halting states is the *running time* or *locality* of the algorithm.

The primary complexity measure in the LOCAL model is the number T of synchronous communication rounds needed for all nodes to halt, and this is also the only complexity measure that I consider in this thesis. Usually, the running time T is written as a function of the size of the graph $n = |V|$, but it may also depend on other parameters like the maximum degree of the nodes Δ . However, it is worth noting that if the size of the message set M is restricted to be $\text{poly}(|V|)$, then this model is called the CONGEST model.

In the LOCAL model, a node cannot learn anything about the graph further than T hops away from itself in only T rounds of communication. To see this, consider that in the first round of communication, a node v can only receive messages from its neighbors. This holds also for the neighbors of v which only learn information from their neighbors. In the second round, node v can again only learn what its neighbors know, which is restricted by the radius-2 neighborhood of v . Iterating this inductively for T rounds, we can see that a node can only receive information from nodes in its T -hop neighborhood, and hence the output of any algorithm is limited to only local information around any node. Consequently, the running time of a LOCAL algorithm is also called

its *locality*.

On the other hand, one can also prove that nodes can learn *everything* about their radius- T neighborhood by always sending their whole state to all of their neighbors. For this to work, however, the nodes need to have unique identifiers as, without them, distinguishing even between a 3-cycle and a long path is impossible, given a suitable port numbering [Ang80; BV01; YK96]. Indeed, this is the main reason for the LOCAL model including unique identifiers, the other reason being that they facilitate breaking symmetry.

Luckily, the assumption of unique identifiers is practical as often computers in a computer network have unique IP addresses or MAC addresses pre-assigned. If this is not the case, then randomness can also be used to produce unique identifiers with high probability.

Indeed, the LOCAL model could also be defined to employ randomness, and—in this case—it would be called the *randomized* LOCAL model, as opposed to the *deterministic* LOCAL model I defined above. When a LOCAL algorithm uses randomness, its behavior becomes probabilistic, and—in particular—the running time may become probabilistic. In this thesis, I adopt the following convention: A randomized LOCAL algorithm solves problem Π with locality T and probability p if, after T rounds of communication, with probability p all nodes of the graph have halted, and they solve problem Π . Usually, with LOCAL algorithms, we want an algorithm to work *with high probability*. This means that the failure probability is polynomially small in the size of the input graph:

$$p = 1 - \frac{1}{n^c}$$

for a constant $c > 0$ chosen by the algorithm user.

Note that the above definition of randomized LOCAL also implies a success probability: If a LOCAL algorithm is executed for a fixed number T of rounds, then all nodes have produced a valid output with probability p . If a node has not halted by time T , then the output is incomplete, and thus the algorithm has failed. However, this can only occur with probability at most $1 - p$.

2.3.1 LCL problems in the LOCAL model

In the last decade, there has been a great research effort into understanding the locality of solving LCL problems in the LOCAL model, and thanks to this effort, we know the tight complexity of many LCL problems. Moreover, we know that *all* LCL problems in a wide class of graphs fall into one of a few complexity classes: For example, in paths, cycles, and n -by- n grids, the only possible complexities for LCLs are $O(1)$, $\Theta(\log^* n)$, and $\Omega(n)$ [BBC+19; BHK+17; CSS23; NS95]. In trees, on the other hand, the deterministic complexity classes are $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, and $\Theta(n^{1/k})$ for any $k \in \mathbb{Z}_+$, with the addition of $\Theta(\log \log n)$ for the randomized LOCAL model [BBO+21; BFH+16; Cha20; CHL+18; CKP19; CP19; GRB22; NS95]. Moreover, we know that, for many—

but not all—of these complexity classes, it is *decidable* to which complexity class a problem belongs [BBC+19; BBC+22; BHK+17; CSS23; NS95].

In general graphs, it is known that there are no LCL problems with deterministic complexity between $\omega(1)$ and $o(\log \log^* n)$ [CP19], or between $\omega(\log^* n)$ and $o(\log n)$ [CKP19]. However, it is also known that there are LCL problems with localities like $\Theta(\log^\alpha n)$, $2^{\Theta(\log^{1/\alpha} n)}$ and $\Theta(n^{1/\alpha})$ for any rational $\alpha \geq 1$ [BBO+21; BHK+18].

2.4 Distributed quantum algorithms

Distributed quantum algorithms are closely related to classical distributed algorithms: we can just replace the classical computers of a distributed system with quantum computers, and the classical communication links with quantum communication links. Although the transition from classical to quantum computing may seem straightforward, it can have profound effects on algorithms and how they can be analyzed.

The most notable of the changes relates to how the state of the algorithm can be represented: In classical distributed systems, the state of the whole system is the product of the states of individual computers of the network. In particular, we can analyze the states of individual computers in isolation. However, in distributed quantum algorithms, this no longer holds, as the states of computers may be *entangled*, which means that the state of the whole system cannot be written as a product of the states of individual computers. This complicates the analysis of algorithms, as we now need to reason about the state of the whole system at the same time. So far we have only a few tools at our disposal to limit what can be done with distributed quantum algorithms; I will discuss some of these tools more in Chapter 4.

In the following definition of quantum-LOCAL, I will use the term “observable F from $\mathcal{H}$ to S ” to indicate a projection-valued measure that produces an element in a finite set S . Formally, a projection-valued measure operates on a Hilbert space $\mathcal{H}$ representing the state space of the system and consists of a projection π from subsets of S to the set of bounded self-adjoint operators on $\mathcal{H}$, satisfying

- $\pi(E)$ is an orthogonal projection for every $E \subseteq S$,
- $\pi(\emptyset) = 0$ and $\pi(S) = I$ where I denotes the identity operator on $\mathcal{H}$, and
- for disjoint events $E_1, E_2, \dots, E_k \subseteq S$

$$\pi\left(\bigcup_{j=1}^k E_j\right)v = \sum_{j=1}^k \pi(E_j)v$$

for all $v \in \mathcal{H}$.

A measurement of a given normalized state vector $|\phi\rangle \in \mathcal{H}$ is done as follows:

Function

$$E \mapsto \langle \phi | \pi(E) | \phi \rangle$$

provides a probability measure for the values in S , and an outcome $s \in S$ is sampled according to this probability measure. As measurements in quantum mechanics collapse the state, the state $|\phi\rangle$ is projected onto the subspace associated with s :

$$|\phi'\rangle = \frac{\pi(\{s\})|\phi\rangle}{\sqrt{\langle \phi | \pi(\{s\}) | \phi \rangle}}.$$

For convenience of notation, I will denote the measurement of an observable F by the randomized function

$$F: \mathcal{H} \rightarrow \mathcal{H} \times S = |\phi\rangle \mapsto (|\phi'\rangle, s)$$

which maps a quantum state $|\phi\rangle$ into a random measurement result s and a posterior state $|\phi'\rangle$ according to the above process.

Definition 2.7 (Quantum-LOCAL model, $\mathcal{LOCAL}^+Q$ [GKM09]).

Let Σ and Γ be the input and output label sets. A quantum-LOCAL algorithm $\mathcal{A}$ consists of Hilbert spaces $\mathcal{Q}$ and $\mathcal{IO}$ for state and communication registers, and, for each degree $d \in \mathbb{N}$,

- an initialization function $\text{init}_d: \mathbb{N} \rightarrow \Sigma^d \rightarrow \mathcal{Q} \otimes \mathcal{IO}^d$ producing unit vectors,
- a unitary computation operator $\varphi_d: \mathcal{Q} \otimes \mathcal{IO}^d \rightarrow \mathcal{Q} \otimes \mathcal{IO}^d$,
- a halting observable F_d^{halt} from $\mathcal{Q} \otimes \mathcal{IO}^d$ to $\{0, 1\}$, and
- an output observable F_d^{out} from $\mathcal{Q} \otimes \mathcal{IO}^d$ to Γ^d .

An input instance for algorithm $\mathcal{A}$ consists of a port-numbered network (G, p) , an injective function $\text{id}(v)$ giving each vertex of G a unique integer identifier from a polynomial space in $|V(G)|$, and an input labeling $\sigma: p \rightarrow \Sigma$ on ports. During the execution of the algorithm, each node $v \in V(G)$ holds a bit r_v initialized to 1 indicating whether the node is still running.

As I noted at the start of this section, the states of individual nodes may be entangled, and thus the state of a quantum algorithm cannot be described as a product of the states of individual nodes. However, the *state space* of the system can still be represented as a product of the state spaces of individual nodes. I denote the subspace localized at node v by $\mathcal{A}_v = \mathcal{Q} \otimes \mathcal{IO}^{\deg_G(v)}$, where $\mathcal{IO}^{\deg_G(v)}$ holds a communication register for each of the $\deg_G(v)$ ports adjacent to v . The state space of the whole system is the product of these individual state spaces:

$$\mathcal{A}^{\otimes} = \bigotimes_{v \in V} \mathcal{A}_v = \bigotimes_{v \in V} \mathcal{Q} \otimes \mathcal{IO}^{\deg_G(v)}.$$

Initially, the state of the system is the product of localized states, as no communication has yet occurred: the local state of each vertex $v \in V(G)$ is

initialized with the initialization function:

$$|S_v^0\rangle = \text{init}_{\deg_G(v)}(\text{id}(v), \sigma_v)$$

where σ_v is the vector of labels of ports around v , that is

$$\sigma_v = [\sigma(v, p_v^{-1}(1)), \dots, \sigma(v, p_v^{-1}(\deg_G(v)))],$$

and the whole initial state is the product of these states:

$$|S^0\rangle = \bigotimes_{v \in V} |S_v^0\rangle.$$

To describe the computation, I define a computation operator φ and a communication operator ψ . The computation operator is defined as the product of individual local computation operators for nodes that are still running, based on register r_v ; for each node v , the operator acts on the corresponding subspace $\mathcal{A}_v$. As all of these operators operate on a disjoint subspace $\mathcal{A}_v$ of $\mathcal{A}^\otimes$, they clearly commute and thus the operator is well-defined:

$$\varphi = \bigotimes_{\substack{v \in V(G) \\ r_v=1}} \varphi_{\deg_G(v)}.$$

The communication operator ψ exchanges the communication registers of each node according to the port numbering: for each edge $\{u, v\} = e \in E(G)$, operator ψ exchanges the states of communication registers $p_v(e)$ and $p_u(e)$ of nodes v and u , respectively. Again, as the communication operator exchanges a disjoint set of communication registers, it is well-defined.

The computation now proceeds as follows: In round t , the nodes exchange messages through the communication operator and then perform local computation via the computation operator:

$$|L^{t+1}\rangle = \varphi \psi |S^t\rangle.$$

Then, the halting observable F^{halt} is measured for each running node, producing a bit for each of those nodes. Again, these measurements are performed on disjoint subspaces of $\mathcal{A}^\otimes$, and thus the measurement is well-defined. With a slight abuse of notation, this produces the posterior state and a bit vector h :

$$(|P^{t+1}\rangle, h_{v_1} h_{v_2} \dots) = \left(\bigotimes_{\substack{v \in V(G) \\ r_v=1}} F_{\deg_G(v)}^{\text{halt}} \right) |L^{t+1}\rangle.$$

If the bit h_v for node v is 0, then the node has not halted yet and running register r_v remains 1. Otherwise, the node halts and the corresponding running register r_v is set to 0. Finally, the nodes that just halted are measured for their output, again with a slight abuse of notation:

$$(|S^{t+1}\rangle, \gamma_{v_1} \gamma_{v_2} \dots) = \left(\bigotimes_{\substack{v \in V(G) \\ h_v=1}} F_{\deg_G(v)}^{\text{out}} \right) |P^{t+1}\rangle.$$

Once all nodes have halted, the output of the algorithm is $\gamma: V(G) \rightarrow \Gamma$ which maps a vertex to the locally-measured outcome.

The running time or locality of an execution of a quantum-LOCAL algorithm is the smallest T such that after iteration T , all the nodes have halted and their output has been measured. As the measurement process is randomized, the running time may also be randomized. I say that a quantum-LOCAL algorithm *has locality T with probability p* if, after T rounds, all nodes have halted with probability at least p . The algorithm solves a problem with this locality and probability if the input σ and output γ solve the problem after the algorithm has halted.

Above, I have defined the quantum-LOCAL model in a very general way: for example, the algorithm designer can choose the Hilbert spaces $\mathcal{Q}$ and $\mathcal{IO}$ arbitrarily—they do not even need to be realizable with a finite number of qubits. In this sense, the above model—and actually *all* models in this thesis—are *lower-bound models*, that is, they are best suited for proving lower bounds. This is convenient, as lower bounds proved in these strong models extend to any physically realizable model for free. In practice, this does not cause problems for upper bounds either: many algorithms provided for these models use only a finite amount of memory or qubits and perform only finite computation.

2.5 Main contribution: Local quantum advantage

The inclusion of quantum resources in the LOCAL model raises key questions:

1. Does quantum computing help with the locality of solving graph problems?
2. What happens when we restrict our attention to LCL problems?
3. Can we bound the quantum advantage?

The first question was answered by Le Gall, Nishimura, and Rosmanis in 2019 in the affirmative: they showed that there exists a graph problem that can be solved with locality $O(1)$ in the quantum-LOCAL model but requires $\Omega(n)$ locality in the classical LOCAL model [LNR19]. However, the problem they provided is a very global one. In fact, to *verify* whether the solution is correct, one also needs locality $\Omega(n)$ both classically and in quantum settings, and therefore the problem they described is not a local problem, let alone an LCL problem.

In Publication I, I provide a partial answer to the second question by providing a family $\mathcal{P}_\Delta$ of LCL problems that shows a quantum advantage as a function of the maximum degree Δ of the graph. The problem family $\mathcal{P}_\Delta$, called *iterated GHZ*, is constructed such that

1. the checking radius of each problem in $\mathcal{P}_\Delta$ is $O(1)$, independent of Δ , and

2. each problem in $\mathcal{P}_\Delta$ can be solved with locality $O(1)$ in the quantum-LOCAL independent of Δ , while
3. every problem in $\mathcal{P}_\Delta$ on degree- Δ graphs requires locality $\Omega(\Delta)$ in the classical LOCAL model.

A key concept in the construction of the family $\mathcal{P}_\Delta$ is *games*, and particularly *non-signaling* and *quantum* games. A game for k players is described by a relation $R \subseteq X^k \times A^k$ between input and output sets X and A . When playing the game, the players may agree on a strategy; in particular, they may generate shared randomness and—in the case of quantum strategies—share an entangled quantum state. After the players receive their inputs $x_i \in X$, they need to produce their outputs $a_i \in A$ without further communication. The players win the game if the input-output tuple is in the game relation R , that is

$$((x_1, \dots, x_k), (a_1, \dots, a_k)) \in R.$$

A particularly interesting game for the purposes of this thesis is the GHZ game [GHZ89] between three players who are traditionally called Alice, Bob and Charlie. In this game, Alice, Bob, and Charlie are given bits x , y , and z as inputs, respectively, and they need to produce output bits a , b , and c . Traditionally, it is promised that $x \oplus y \oplus z = 0$. In this case, the players need to produce outputs such that $a \oplus b \oplus c = 0$ if and only if $x = y = z = 0$; the operator $\oplus$ denotes the XOR operation on bits or, equivalently, addition modulo 2. In our case, if the input condition is not satisfied, we allow the outputs of the players to be anything.

Definition 2.8 (GHZ game [GHZ89]).

In the GHZ game, three players are given input bits x, y and z , and they need to produce output bits a, b and c such that one of the following conditions hold:

1. If $x + y + z = 0$, then $a \oplus b \oplus c = 0$.
2. If $x + y + z = 2$, then $a \oplus b \oplus c = 1$.
3. If $x + y + z \in \{1, 3\}$, then any combination of output bits a, b and c is allowed.

The GHZ game is notable for admitting a quantum strategy, which allows the players to win the game with probability 1, while classically the game cannot be won with probability higher than 0.75, even with access to shared randomness. The quantum strategy works by pre-sharing a so-called GHZ state of three entangled qubits among the three players:

$$|\text{GHZ}\rangle = \frac{1}{\sqrt{2}}(|0\rangle_a|0\rangle_b|0\rangle_c + |1\rangle_a|1\rangle_b|1\rangle_c).$$

The players then perform a local measurement on their local qubit, selecting the direction of the measurement based on the input they receive. It is well-known that by selecting the measurement axis correctly, the outputs of these measurements win the game with probability 1 [BBT05; Mer90].

To understand why no classical strategy can win the GHZ game with certainty, note that after the players have drawn their randomness and agreed on their strategy, the strategy can be represented as six bits a_0, a_1, b_0, b_1, c_0 , and c_1 . This strategy is interpreted such that when Alice receives her input bit x , she locally outputs a_x , and similarly Bob outputs b_y and Charlie outputs c_z . Writing down the four non-trivial victory conditions

$$\begin{aligned} a_0 + b_0 + c_0 &\equiv 0 \pmod{2}, \\ a_1 + b_1 + c_0 &\equiv 1 \pmod{2}, \\ a_1 + b_0 + c_1 &\equiv 1 \pmod{2}, \text{ and} \\ a_0 + b_1 + c_1 &\equiv 1 \pmod{2}, \end{aligned}$$

we can notice that the sum on the left side is 0 while the sum on the right side is 1. This shows that there does not exist an assignment of these bits that wins the game no matter what is the input, and hence no perfect classical strategy exists [BBT05].

Playing a single GHZ alone is not yet enough to separate the LOCAL and quantum-LOCAL models. This is because, in the classical case, the algorithm can win the game by spending one communication round sharing the inputs with all players, allowing them to pick a winning output. On the other hand, even the quantum algorithm requires one round to establish the shared GHZ state among the players. Instead, to separate the models, we engineer a problem that we call the *iterated GHZ problem*. In this problem, each player plays multiple games in a sequence such that the output of one game becomes the input of the next game.

Definition 2.9 (Iterated GHZ problem [Publication I]).

Fix a degree Δ . The iterated GHZ game $\mathcal{P}_\Delta$ is defined for $(\Delta, 3)$ -biregular graphs: nodes with degree Δ represent players, and nodes with degree 3 represent games. The input assigns a label from $[\Delta]$ to each game node such that every player node is adjacent to exactly one game of each label.

Every player node needs to output a bit vector $g = (g_1, \dots, g_\Delta)$. The correctness is defined subject to rules on the game nodes. For convenience, let me denote the output bit vectors of the adjacent player nodes by g^a, g^b and g^c :

- If the game node has input label 1, then $g_1^a + g_1^b + g_1^c = 1$.
- If the game node has input label $i \geq 2$, then $(g_{i-1}^a, g_{i-1}^b, g_{i-1}^c)$ and (g_i^a, g_i^b, g_i^c) must satisfy the GHZ game rules, with the former as the input and the latter as the output.

The first game is different from the rest. This is to bootstrap the rest of the games with non-trivial inputs; we cannot simply hard-code the inputs for the first game as 0 or 1, as this would allow for a trivial solution for all the following GHZ games. The rest of the games are played according to GHZ rules. Note

that for a fixed degree Δ , this problem is an LCL problem, as both the input and output label sets are finite, and the correctness constraints are defined with respect to radius-1 neighborhoods around game nodes.

Theorem 2.10 (Local quantum advantage [Publication I]).

The iterated GHZ problem for degree Δ can be solved with locality $O(1)$ in the quantum-LOCAL model but requires locality $\Omega(\Delta)$ in the classical LOCAL model.

The quantum-LOCAL algorithm for the iterated GHZ game is straightforward: In the first and only round of communication, the game nodes with label 1 produce bit strings $(1, 0, 0)$ and distribute these bits to their neighbors to become bits g_1 . The rest of the game nodes prepare three qubits in the GHZ state and distribute these, along with the game number, to adjacent players. After this round, all player nodes have their bits g_1 fixed and have pre-shared entangled qubits for each of the games they need to play. Hence, without further communication, they can play according to the quantum strategy for the GHZ game, routing the output of one game to become the input of the next game locally. They then collect the outputs of each game into a vector and produce that as their output label.

To establish classical hardness of the iterated GHZ game, we employ the *round elimination* technique [BFH+16], which is widely used to show lower bounds for LCL problems in the LOCAL model [BBG+26; BBH+21; BBK+21; BBK+22; BBK+25; BO20; Bra19]. At the core of the round elimination technique is the round elimination function $\mathcal{R}$ that transforms an LCL problem into another LCL problem. This function has the property that if an LCL problem Π can be solved with locality $T > 0$ in the LOCAL model, then problem $\mathcal{R}(\Pi)$ can be solved with locality $T - 1$. By iterating the $\mathcal{R}$ function on an LCL problem Π for t iterations and showing that $\mathcal{R}^t(\Pi)$ cannot be solved with 0 rounds, we get that Π requires at least $t + 1$ rounds to solve in the LOCAL model.

In Publication I, we actually show a slightly stronger result by considering a variant of the iterated GHZ problem where the algorithm is allowed to permute input labels on the games, provided that after the permutation each player still has exactly one game of each color neighboring it. To complicate matters, the round elimination function $\mathcal{R}$ increases the size of the description of the LCL problem exponentially. To combat this, we come up with new systematic ways to reduce the size of the problem after each iteration of round elimination while still keeping the intermediate problems hard enough for us to get the lower bound. After $\Omega(\Delta)$ iterations of the round elimination function $\mathcal{R}$, we show that the resulting problem still cannot be solved without communication in the classical LOCAL model, which gives us the desired $\Omega(\Delta)$ lower bound.

Finally, it is worth noting that, while we provide a family of LCL problems, it is possible to turn this construction into a single genuine LCL problem [BCd+26]. It can be done by noticing that the problem family in question is *linearizable*, which allows us to embed a high-degree version of the problem into a low-

degree graph. However, the exact construction is outside the scope of this thesis.

3. Locality in online algorithms

The concept of locality is not restricted to the study of distributed algorithms—whether classical or quantum—but can also be extended to other settings. In particular, in this thesis, I study locality also from the perspective of online algorithms. Distributed and online settings may seem unrelated at first: in distributed computing, multiple computers in a fixed network exchange messages to solve a problem, while in an online setting, a *centralized* algorithm solves a problem on a graph that is constructed one node at a time. Yet, in this thesis, I show that these settings are related. Towards that end, I start by revising the classical LOCAL model and present an alternative formalism for it from the point of view of *local views*. I then make my way towards online algorithms by first defining *sequential* LOCAL algorithms, or SLOCAL for short, and then continue to define the online-LOCAL model, which was first introduced in Publication II. In the same publication, we provide a classification of LCL problems on paths, cycles and rooted regular trees in the online-LOCAL model. We also provide an efficient online-LOCAL algorithm for 3-coloring bipartite graphs, the tightness of which we prove in Publications III and IV. Finally, I take a step back from online-LOCAL to also consider locality in dynamic graph algorithms, and introduce the dynamic-LOCAL model.

3.1 Local views

Let us recall the definition of the LOCAL model (Definition 2.6): Each node of the network has a computer—or, more formally, a state machine—attached to it, and these computers exchange messages in synchronous rounds using the `send` and `recv` functions. However, we make no assumptions about the computational complexity of these functions, and in particular, they do not need to be efficiently computable¹. In fact, we could just as well fix the sending function `send` to always send the whole state of the computer, along with the number of the port the message is sent over, to the neighboring computers.

¹Whether the update functions need to be computable at all is a question for follow-up work [CDE+26].

The computers could still update their state using the receiving function recv by locally recovering the original message using the original send function. This allows us to defer computation for later.

The same idea generalizes to more than one round of communication: in one round, the nodes can propagate their state to their neighbors, and in two rounds to their neighbors' neighbors. The nodes can then locally use this information to compute the messages that the nodes in their radius-2 neighborhood would have sent, how the nodes in their radius-1 neighborhood would have reacted to these messages, what they would send in the second round, and finally, their own state after two rounds of communication.

Iterating on this idea further, it is easy to see that in T rounds of communication, each node can collect the states of all nodes in its radius- T neighborhood along with the local topology. By deferring even the local initialization to this simulation, any LOCAL algorithm can be normalized such that it first spends some number of rounds collecting its *local view* of the graph before computing its local output. In effect, we have constructed a *universal LOCAL algorithm* that can simulate *any* LOCAL algorithm by first collecting the radius- T neighborhood and then mapping it to a local output. In fact, this is an alternative way to characterize the LOCAL model using local views:

Definition 3.1 (LOCAL model—view perspective [Lin92]).

Let Σ and Γ be the input and output label sets, respectively. An algorithm $\mathcal{A}$ in the LOCAL model is a family of functions parametrized over the degree d of nodes

$$\mathcal{A}_d: B \rightarrow \Gamma^d \cup \{\perp\}$$

from the set of centered neighborhoods B with inputs and identifiers to output label set Γ^d for adjacent ports or the empty label $\perp$.

An input instance for LOCAL algorithm $\mathcal{A}$ consists of a port-numbered network (G, p) , an injective function $\text{id}(v)$ giving each vertex of G a unique integer identifier from a polynomial space in $|V(G)|$, and an input labeling $\sigma: p \rightarrow \Sigma$ on the ports of G .

The execution of a LOCAL algorithm $\mathcal{A}$ works as follows: For each node v , a radius r is initialized at 0. The algorithm is then invoked with the radius- r neighborhood of v , including all the identifiers, port numbers and inputs in the neighborhood. If it produces a label in Γ^d , then these labels correspond to the labels of ports adjacent to v , and the algorithm halts for node v . Otherwise, if the algorithm produces $\perp$, the radius r is increased by one and the process is repeated until the algorithm halts.

The locality T of a LOCAL algorithm is the minimum radius r for which the algorithm produces an output for every vertex of G .

One thing to note about the local-views perspective is that it is genuinely a classical model and does not work with quantum-LOCAL. This is because the non-cloning theorem prohibits us from creating a copy of the quantum

state of a node. In particular, there does not exist a unitary local computation operator φ that copies the state from the state register to all the communication registers. However, this way of analyzing the classical LOCAL model is still useful, as it allows us to define the SLOCAL model and the online-LOCAL model, which I will present next. In Chapter 4, I will show a connection between the randomized variant of the online-LOCAL model and the quantum-LOCAL model.

3.1.1 On the strength of the local views formalism

As motivated before the above definition, it is easy to believe that this model is at least as strong as the message-passing version of the LOCAL model. Whether the message-passing LOCAL model can simulate the local-views LOCAL model is not clear: Consider an (artificial) graph problem where every node is given a natural number T_v as an input, and the output at every node should contain the radius- T_v neighborhood of the node. This problem is trivial to solve in the local-views LOCAL model: each node just outputs $\perp$ until it sees the full neighborhood it needs to output. The locality of this algorithm is the maximum T_v that appears in the input.

However, this problem is not as easy to solve in the message-passing LOCAL model: Consider two graphs G_1 and G_2 that differ only in the input of one node: in both graphs, the input of all nodes is 1 except for a distinguished node which has input 1 in G_1 but input n in G_2 . In graph G_1 , the optimum locality is 1, while in G_2 it is n . However, the neighborhoods around almost all nodes are isomorphic in both graphs, and hence they cannot distinguish in which of these two graphs they are. To achieve even asymptotically optimal locality, all nodes in G_1 need to decide within $O(1)$ rounds to stop, while nodes in G_2 need to remain active for n rounds to relay the neighborhood information to the distinguished node. This contradicts the optimality of the message-passing algorithm.

The problem presented above is far from being an LCL problem, as it requires unbounded input and output label sets, but it shows that direct simulation of the local-views LOCAL model is not possible in the message-passing LOCAL model. However, if we restrict our attention to LCL problems, it might still be the case that the structure of the problem allows for a general simulation. This would be a good question for follow-up research.

3.2 Distributed sequential algorithms

The LOCAL model can be seen as both a model of distributed computing and as a model of *parallel* computing: It is distributed in the sense that multiple networked computers interact to reach a common target. It is parallel in the sense that—taking the local-views definition—every node of the graph sees

its local neighborhood and produces a local output *simultaneously* and *in parallel*.

This raises a question: could we consider a model that is both distributed and *not* parallel, that is, *sequential*? In 2017, Ghaffari, Kuhn, and Maus did exactly that when they defined the sequential-LOCAL model, often denoted as SLOCAL [GKM17]. In the SLOCAL model, the nodes are processed sequentially in an adversarial order. When a node is processed, it sees its radius- T neighborhood of the graph, along with the outputs and auxiliary data produced by the previously processed nodes. It then produces its own local output and the auxiliary data for future nodes to use.

Definition 3.2 (SLOCAL model [GKM17]).

Let Σ and Γ be the input and output label sets. An algorithm $\mathcal{A}$ in the SLOCAL model consists of

- a state space S ,
- an update function step that maps local neighborhoods, including their states, to a state in S or the empty symbol $\perp$, and
- an output extractor function $\text{out}_d: S \rightarrow \Gamma^d$ for each degree d .

An input instance for $\mathcal{A}$ consists of a port-numbered network (G, p) , an injective function $\text{id}(v)$ giving each vertex of G a unique integer identifier, an input labeling $\sigma: p \rightarrow \Sigma$ on the ports of G , and a permutation $\pi = v_1 v_2 \dots v_n$ of the nodes of G . It must hold that the input identifiers come from a polynomial space, that is

$$\text{id}(v) \leq \text{poly}(|V(G)|) \text{ for each } v \in V(G).$$

We associate a state register S_v from state space S for each node v . Register S_v is initialized with the input on the ports adjacent to the corresponding node, that is

$$S_v = [\sigma(v, p_v^{-1}(1)), \dots, \sigma(v, p_v^{-1}(\deg_G(v)))].$$

The algorithm $\mathcal{A}$ now processes nodes of G individually in the order given by π : When node v_i is processed, a radius r is initialized at zero. Then function step is invoked with the radius- r neighborhood of v_i , including all the states S_u in this neighborhood. If step returns a new state, then S_{v_i} is replaced by that state, and the algorithm proceeds to process the next node. Otherwise, if step returns $\perp$, radius r is increased by one, and function step is invoked again until it returns a value.

Once all nodes have been processed, the output of the algorithm for node v can be extracted using the output extraction function $\text{out}_{\deg_G(v)}$ on state S_v , which gives an output label for each port adjacent to v . The locality of an SLOCAL algorithm is the maximum radius r used for any of the processed nodes.

It is easy to see that this model is at least as strong as the LOCAL model: when a node is processed, it can ignore all the information stored in the local state registers that is not part of the input and can therefore simulate a LOCAL algorithm defined in the view perspective directly.

In the SLOCAL model, problems of the symmetry-breaking flavor vanish and merge with constant-locality-solvable problems. This is because any $\Theta(\log^* n)$ -locality LOCAL algorithm can be normalized so that it first computes a distance- k coloring for some constant k , using locality $\Theta(\log^* n)$ in the process, and then applies a mapping from this colored neighborhood to the local output [CKP19]. Since finding such a coloring is trivial in the SLOCAL model with locality k , all problems in the $\Theta(\log^* n)$ class in the LOCAL model have locality $O(1)$ in the SLOCAL model [GKM17].

3.3 New model: Online-LOCAL

Now that we have extended our study from purely parallel and distributed algorithms to include sequential algorithms, we can further extend our consideration to *online graph algorithms*. In online graph algorithms, an input graph is fed to a single, centralized algorithm one node at a time. When a node is processed, the algorithm sees all edges connecting the new node to previously processed nodes and must commit to an output for the new node.

Many problems are hard to solve using pure online graph algorithms. For example, properly coloring a tree with n nodes requires $\Omega(\log n)$ colors, while graph-theoretically it is possible with just 2 colors [Bea76]. To get better results, online graph algorithms are sometimes augmented with advice. In Publication II, we present a new type of spatial advice: when a node is processed, the algorithm also sees the radius- T neighborhood of the node. Compared to the no-advice setting, the algorithm sees the future evolution of the graph in the local neighborhood of the new node. We call this model the online-LOCAL model.

It turns out that this model is essentially the SLOCAL model augmented with a *global* memory: in addition to seeing the data the algorithm has stored in the nodes locally, the algorithm also sees all previously processed nodes.

Definition 3.3 (Online-LOCAL model [Publication II]).

Let Σ and Γ be the input and output label sets. An algorithm $\mathcal{A}$ in the online-LOCAL model consists of

- a global memory space M , and
- an update function step that maps local neighborhoods and memory to either a local output and new memory state, or to a distinct empty element $\perp$.

An input instance for $\mathcal{A}$ consists of a port-numbered network (G, p) , an

input label $\sigma: p \rightarrow \Sigma$ on the ports of G , and a permutation $\pi = v_1 v_2 \dots v_n$ of the nodes of G . The state register M^0 is initialized to a distinguished initialization element $\perp \in M$. The algorithm $\mathcal{A}$ now processes nodes in the order given by π . When node v_i is processed, a radius r is initialized at zero. Then function `step` is invoked with the current state M^{i-1} and the local neighborhood $B(v_i, r)$, including the local input from σ . If the function returns $\perp$, the radius r is increased by one, and the function is invoked again. Otherwise, the returned state is stored at M^i , and the returned output is used as the local output at v_i .

The locality of an online-LOCAL algorithm is the maximum radius r used for any of the processed nodes. For simplicity, I will assume that the online-LOCAL algorithm also knows a polynomial upper bound on the size of the graph.

When describing an online-LOCAL algorithm, it is often useful to consider the process as follows: An adversary chooses the order in which nodes are processed. However, when a node is processed, the adversary must also *reveal* the neighborhood of the node to the algorithm. The algorithm can remember all nodes that have been revealed to it in its memory M .

When Ghaffari, Kuhn, and Maus introduced the SLOCAL model, they actually briefly considered giving the SLOCAL model global memory. However, they noted

We do not allow any globally shared state as this would make the model too powerful.

adding in the footnote

E.g., even one bit of global memory would allow to solve *leader election*, which clearly cannot be solved locally.

Somewhat surprisingly, it turns out that, in many cases, the online-LOCAL model is not “too powerful”. Indeed, as we show in Publication II, on paths, cycles, and rooted regular trees, solving LCL problems is essentially as hard as in the LOCAL model. We show that on paths and cycles, the only possible complexity classes for LCL problems are $O(1)$ and $\Theta(n)$ in the online-LOCAL model, and that these correspond to the complexity classes $O(\log^* n)$ and $\Theta(n)$ in the LOCAL model, respectively. Similarly, in rooted regular trees, the only possible complexity classes for LCL problems are $O(1)$, $\Theta(\log n)$, and $n^{\Theta(1)}$ in the online-LOCAL model, and these correspond to $O(\log^* n)$, $\Theta(\log n)$, and $n^{\Theta(1)}$ in the LOCAL model.

Finally, I remark that the online-LOCAL model also admits another interpretation: When a node is processed, an online graph algorithm is given the possibility to *delay* the processing of that node until it has seen the whole radius- T neighborhood of the node. This may sometimes be a more natural

setting to consider than giving the algorithm information about the future changes to the graph.

3.4 Main contribution: 3-coloring in online-LOCAL

One of the key questions we analyze in Publication II is the complexity of 3-coloring a 2-dimensional grid in the online-LOCAL model. It is easy to see that, even in the online-LOCAL model, finding a proper 2-coloring of a grid requires order-of-diameter locality—fixing the color of one node effectively fixes the colors of all other nodes in the graph; this corresponds to a lower bound of $\Omega(n)$. On the other hand, finding a 4-coloring is known to be easy already in the LOCAL model, as it is possible with locality $O(\log^* n)$ [BHK+17; BHM+22]. Hence, both the complexity of 2-coloring and the complexity of 4-coloring a 2-dimensional grid remain the same across a broad range of models, including LOCAL, SLOCAL, and online-LOCAL.

However, when we shift our attention to 3-coloring grids, things change. It is known that finding a 3-coloring requires locality $\Omega(\sqrt{n})$ in the LOCAL model [BHK+17], and, in Publication II, we extend this to an $\Omega(n^{1/10})$ -locality lower bound for the SLOCAL model, which—while likely not tight—is still polynomial in n . However, to our surprise, we show that there exists an $O(\log n)$ -locality online-LOCAL algorithm for finding a 3-coloring on grids. In a nutshell, the algorithm tries to 2-color the grid using only colors 1 and 2 for as long as possible and only resorts to color 3 to change the parity of a component.

Theorem 3.4 (3-coloring online-LOCAL algorithm [Publication II]).

There exists an online-LOCAL algorithm with locality $O(\log n)$ that finds a 3-coloring of a bipartite graph.

The key observation of our 3-coloring algorithm is that the underlying grid admits only 2 different 2-colorings—fixing the color of one node fixes the coloring everywhere; we call this *parity*. It is therefore enough for the algorithm to locally guess the parity and, when two local guesses turn out to be incompatible, be able to change our guess. By always choosing to change the guess of the component whose guess has been changed the least times, and by the fact that the number of components is upper bounded by n , it is evident that the guess of any component may change at most $\log n$ times.

Another key tool we use is the fact that an online-LOCAL algorithm may *pre-commit* to a solution for a node it has seen but that has not been processed yet: the algorithm can decide and store in its state what the output for that node is going to be and act as if it had been processed already. Once a node with a pre-commitment gets actually processed, the algorithm just recalls the output for that node from its state.

Our online-LOCAL algorithm uses both of these observations for 3-coloring a grid: in its state, the algorithm keeps track of a list of connected components of

nodes it has seen so far, how many times the guess has been changed for each component, and uses pre-commitments to change the guess for a component. When the algorithm processes a node, it first checks if the local neighborhood of the node connects to an existing component. If not, the algorithm creates a new component for the neighborhood of the processed node and guesses the parity for that component; without loss of generality, it may assume that the color of the processed node is 1. If the neighborhood touches exactly one existing component (or multiple components, but their parities are compatible), the newly revealed nodes join that component, and the node gets a color that is consistent with the parity of the component. If the neighborhood touches two or more existing components and their parities are incompatible, the algorithm first changes the parities of existing components to be compatible. The algorithm then merges all these components together, including the newly revealed neighborhood.

It just remains to show that the online-LOCAL algorithm can change the guess of parity by pre-committing the color only for nodes it has already seen. To change the parity, the algorithm essentially builds a border around all nodes of the component that have already been colored using the third color, after which it can change the parity to be opposite of what it used to be. This building of a border only requires pre-committing the color of nodes within distance 3 from already-colored nodes. By ensuring that the algorithm always sees at least radius- $(3 \log n)$ neighborhoods of all processed nodes, and by the fact that the parity of a component needs to be changed at most $\log n$ times, it is easy to see that the algorithm only pre-commits color for nodes it has already seen.

It is worth noting that our algorithm actually works for a broader class of graphs than just 2-dimensional grids. In fact, it works for all *bipartite* graphs, that is graphs that are 2-colorable. However, grids remain an interesting special case of bipartite graphs, and for the rest of this section, I am going to discuss the optimality of our algorithm on grids.

The tightness of this algorithm was shown by Chang, Mishra, Nguyen, Yang, and Yeh for grids [CMN+24]. Their lower bound relies on an argument based on a *potential*—which they call the *b*-value—defined over walks in a 3-colored graph. They define the *b*-value of a walk to count the difference between the number of times a walk steps from a node with color 2 to a node with color 1, and the number of times the walk steps from color 1 to color 2. This *b*-value has some nice properties:

1. The *b*-value of any closed walk is 0, and
2. the parity of the *b*-value is uniquely determined by the parity of the length of the walk and the colors at the endpoints of the walk.

Chang, Mishra, Nguyen, Yang, and Yeh provide an adversarial strategy to force any online-LOCAL algorithm with locality $T = o(\log n)$ to construct a path with potential $\Omega(T)$. This is sufficient for the lower bound: They first construct such a path P with high potential. They then force the algorithm to label another

path P' of the same length and position it below P at distance $2T + 2$ so that the online-LOCAL algorithm does not yet see how they are connected. By choosing the orientation of path P' carefully, Chang et al. force the b -value of the cycle passing through P and P' to be much larger than T , even if the algorithm has not labeled the paths connecting P and P' yet. This contradicts the fact that the b -value of any closed walk is 0, and hence no such online-LOCAL algorithm with locality $o(\log n)$ can exist.

In Publication III—among other results—we extend this lower bound to hold also for the randomized variant of the online-LOCAL model. The general strategy of the proof is similar to that of Chang et al. The main change is adapting the potential boosting strategy to become oblivious and tolerate small failure probabilities.

In Publication IV, we extend this lower bound further to hold even in *oriented* grids—these are grids where every node has a consistent notion of north, east, south, and west.

Theorem 3.5 (Hardness of 3-coloring in oriented grids [Publication IV]).

Finding a 3-coloring of an oriented grid requires locality $\Omega(\log n)$ in oriented grids.

The lower-bound proof of Chang, Mishra, Nguyen, Yang, and Yeh breaks down in oriented grids in the last step where they need to have a possibility to *flip* the orientation of path P' . This flipping is not possible in the oriented case as it would break the *consistency* of the underlying orientation, and then some nodes would notice that their compasses point to a different direction than their neighbors'.

To obtain the $\Omega(\log n)$ lower bound to oriented grids, we present three new ideas in Publication IV:

1. We show that the potential-boosting construction of Chang et al. can be extended to give *quasilinear* boosting in potential. In particular, for a sufficiently large l , any horizontal or vertical path must have potential $\Omega(\frac{lT}{2^{10T}})$. This allows us to construct higher potentials than directly possible with the original potential-boosting construction.
2. We show that the potential-boosting construction can also be applied on sloped paths. Considering the natural embedding of a grid on $\mathbb{Z} \times \mathbb{Z}$, these are paths that approximate a straight line segment drawn on the embedding.
3. We achieve the final contradiction by constructing an L-shaped labeled path with quasilinear potential on each side, and show that it must have two nodes far apart that have the same potential. We connect these nodes together with a path having slope s , and then we use the sloped boosting construction to construct a path with high potential and slope s . Finally, we show that this high-potential path is close to the original sloped path,

which leads to a contradiction in the same way as Chang et al. do.

I leave more a detailed explanation of these ideas to Publication IV.

3.5 New model: Dynamic-LOCAL

In addition to online graph algorithms, there are other families of graph algorithms that can benefit from the concept of locality. One such family is *dynamic graph algorithms*. These are algorithms where the input graph is modified over time, and the task of the algorithm is to maintain a proper solution after each change. Usually, these modifications are insertions and removals of nodes and edges. In this thesis, I only discuss the case where initially all nodes of the graph exist and edges are inserted one at a time.

Usually, in the study of dynamic graph algorithms, the research has concentrated both on minimizing the amount of computation and memory needed to maintain a proper solution, and on minimizing the number of times the algorithm needs to modify the output of previously-labeled nodes or edges, often called the *recourse* of an algorithm. In this thesis, the focus is on understanding locality, and recourse is very close to that. In fact, if a dynamic graph algorithm is only allowed to modify the output near the changes to the input, it also maintains a good recourse.

In Publication II, we consider exactly this setting: when the input changes, the algorithm is allowed to modify its output in the radius- T neighborhood of the change. This also nicely corresponds to locality, as all changes to the output labeling are local to the modification of the input graph. Hence, we call this radius T of modification the *locality* of a dynamic algorithm, and call this model the dynamic-LOCAL model.

Definition 3.6 (Dynamic-LOCAL model [Publication II]).

Let Σ and Γ be the input and output label sets, respectively. An algorithm $\mathcal{A}$ in the dynamic-LOCAL model is a function which maps a sequence

$$[(G_1, p_1, \sigma_1), \dots, (G_k, p_k, \sigma_k)]$$

of labeled n -node networks into an output labeling γ for the last network of the input sequence (G_k, p_k) .

Let $k > 2$, and let $[(G_1, p_1, \sigma_1), \dots, (G_k, p_k, \sigma_k)]$ be an input sequence such that $G_i \subseteq G_{i+1}$ for all $i < k$. Let

$$\gamma_{k-1} = \mathcal{A}((G_1, p_1, \sigma_1), \dots, (G_{k-1}, p_{k-1}, \sigma_{k-1})) \text{ and}$$

$$\gamma_k = \mathcal{A}((G_1, p_1, \sigma_1), \dots, (G_{k-1}, p_{k-1}, \sigma_{k-1}), (G_k, p_k, \sigma_k))$$

be the output labelings after two consecutive steps of the algorithm. The *change-locality* of algorithm $\mathcal{A}$ for this input sequence is the minimum T such that the following holds: if the radius- T neighborhoods of any

node v in both G_{k-1} and G_k are the same, then the outputs for that node are equal: $\gamma_{k-1}(v) = \gamma_k(v)$. The *locality* of algorithm $\mathcal{A}$ is the maximum change-locality over any input sequence of n -node networks.

Many efficient dynamic graph algorithms having a low recourse fall in the class of dynamic-LOCAL algorithms [AOS+18; BCH+18; BM19; DZ18; GK21; IL93; NS16]. However, to my knowledge, no-one has studied dynamic graph algorithms exclusively from the perspective of locality before.

In Publication II, we study the relation of dynamic-LOCAL with the other models I have presented in this thesis. In particular, we bound it to be between the LOCAL and the online-LOCAL models: It is clear that any dynamic-LOCAL algorithm can just simulate a LOCAL algorithm for the last graph of the input sequence, and because the output of a LOCAL algorithm depends only on its local neighborhood, the output of the dynamic-LOCAL algorithm never changes far away from the change in the input graph.

On the other hand, an online-LOCAL algorithm can simulate a dynamic-LOCAL algorithm directly: The online-LOCAL algorithm can remember the whole graph that has been revealed to it so far. Whenever the adversary reveals a node v , the algorithm updates its view of the whole graph and invokes the dynamic-LOCAL algorithm with this updated view. As the online-LOCAL algorithm sees the whole neighborhood, it is guaranteed that the neighborhood of node v cannot change, and therefore the online-LOCAL algorithm can safely output whatever the dynamic-LOCAL algorithm outputs for v .

Moreover, we separate the dynamic-LOCAL model from the SLOCAL model by providing two problems: one that is easy in SLOCAL but hard in dynamic-LOCAL, and another that is easy in dynamic-LOCAL but hard in SLOCAL. The former problem we call *nested orientation*. In this problem, the output essentially needs to encode a possible execution of a locality-1 SLOCAL algorithm, containing all the information such algorithm has available to it when deciding the output of a node; this is clearly trivial in the SLOCAL model to do with locality 1. However, there are instances for this problem that force any dynamic-LOCAL algorithm to modify the solution at distance $\omega(1)$ from the change.

To show the separation in the other direction, we provide a problem we call *weak reconstruction*. In this problem, at least one node of each component needs to output the structure of its whole component. In the dynamic-LOCAL model, this is trivial as the algorithm has a global view of the structure of the graph after each change, and it can just store this to one of the nodes around the change. On the other hand, this requires $\Omega(n)$ in the SLOCAL model as, even on cycles, the algorithm requires linear locality for at least one node to know the length of the cycle.

One thing to note about these separations is that they are shown by using problems tailor-fit for these separations. In particular, these problems are far from being LCL problems: both problems require infinite label spaces,

and to even verify the correctness of a weak reconstruction, one needs to see the whole component. Indeed, this leaves open whether we can separate the dynamic-LOCAL model from the SLOCAL model using only LCL problems.

4. Connection between quantum and online algorithms

So far in this thesis, I have considered two distinct branches of extensions of the classical LOCAL model: one is based on augmenting the distributed network of computers with quantum capabilities (quantum-LOCAL), while the other is based on first making the algorithms less parallel (SLOCAL) and then considering locality as a restriction for centralized algorithms (dynamic-LOCAL and online-LOCAL). Initially, these settings may seem very different: in quantum-LOCAL, the nodes of the graph exchange quantum messages that they can use to form correlations with their neighbors, while in online-LOCAL, the whole algorithm is a central entity that has a global view of all previously revealed nodes. Yet, as we show in Publication III, and as I present in this section, these worlds are related in a surprising way: a randomized version of online-LOCAL is strong enough to simulate quantum-LOCAL!

In this chapter, I start by defining the randomized online-LOCAL model. I will then take a detour into analyzing the probability distributions that quantum-LOCAL algorithms can produce, and then show that these distributions can be sampled in the randomized online-LOCAL model. In particular, these output distributions both have *bounded dependence* and are *non-signaling*. I will present a new connection between the complexity of LCL problems in the LOCAL model and bounded-dependence distributions for those problems. Finally, I will discuss how randomized online-LOCAL can sample from non-signaling distributions, and how online-LOCAL algorithms can be simulated in the SLOCAL model.

4.1 New model: Randomized online-LOCAL model

Randomized online-LOCAL is a small extension of the deterministic online-LOCAL model: the function now has access to randomness that is unbeknownst to the adversary:

Definition 4.1 (Randomized online-LOCAL model [Publication III]).

A randomized online-LOCAL algorithm $\mathcal{A}$ is otherwise like a deterministic online-LOCAL algorithm (see Definition 3.3), except that its update function step gets an additional parameter: a real number in the range $[0, 1]$. This encodes the randomness that the function can utilize.

The execution of algorithm $\mathcal{A}$ proceeds as in the case of deterministic online-LOCAL algorithms, except that for each processed node v_i , a uniformly random real number x_i in the range $[0, 1]$ is drawn. This number x_i is given to the update function step and remains constant for the processing of node v_i . The input instance is chosen by an adversary before the randomness is drawn.

Note that the algorithm being parametrized by a real number is equivalent to it being parametrized over an infinite bit sequence: a real number can be written in binary and thus interpreted as a bit sequence, and an infinite bit sequence can be interpreted as a real number in binary representation.

The last part of the definition means that the randomized online-LOCAL adversary is *oblivious* to the randomness that the algorithm uses. In Publication III, we show that this is indeed necessary for algorithms solving LCL problems. Otherwise, we could *derandomize* the online-LOCAL algorithm and obtain a deterministic online-LOCAL algorithm with the same locality. The proof is based on the idea of tracking the conditional expectation of a failure and always fixing the randomness so that it minimizes the expected number of future failures, similar to what Ghaffari, Harris, and Kuhn did when they proved that SLOCAL can derandomize LOCAL [GHK18].

4.2 Non-signaling can be simulated in online-LOCAL

It is clear that the randomized online-LOCAL model is at least as strong as deterministic online-LOCAL, and hence it also dominates the SLOCAL and dynamic-LOCAL models. In Publication III, we show that it is also at least as strong as the quantum-LOCAL model. We do this by first considering quantum-LOCAL algorithms as sampling processes that produce *outcomes*; these are probability distributions over output labelings of the graph:

Definition 4.2 (Outcome [Publication III]).

Let Σ and Γ be input and output label sets, respectively, and let $\mathcal{G}$ be a family of networks. An *outcome* O over $\mathcal{G}$ is a mapping from a labeled network to a probability distribution over output labelings:

$$O = (G, q, \sigma) \mapsto \{(\gamma_i, p_i)\}_{i \in I}.$$

Here I denotes a countable set of indices of output labelings. For this to be a probability distribution, it must hold that

1. for all $i \in I$, $p_i > 0$,
2. $\sum_{i \in I} p_i = 1$, and
3. p_i represents the probability of obtaining γ_i as the output labeling for the given network.

Usually, we want to consider outcomes that *solve* a graph problem. We say that an outcome O *solves problem Π over family $\mathcal{G}$ with probability p* if for every network (G, q) and every input labeling σ , it holds that

$$\sum_{\substack{(\gamma_i, p_i) \in O \\ (\sigma, \gamma_i) \text{ is valid for } \Pi}} p_i \geq p.$$

If this holds with $p = 1$, then O *solves problem Π over family $\mathcal{G}$* .

It is clear that a quantum-LOCAL algorithm produces an outcome. However, the outcome it produces is not arbitrary: For example, the output distribution of a node after T rounds of communication cannot depend on the input of a node at a distance larger than T ; if it could, this would correspond to the algorithm breaking *causality*. Causality here refers to the physical principle that current events can only depend on past events, not on future events. In the quantum-LOCAL model, the radius of causal dependence of a node expands by one on each communication step, which directly corresponds to the locality of the algorithm, and therefore the output of a node cannot be causally dependent on the input at a distance greater than T . In other words, a T -round quantum-LOCAL algorithm cannot be used to send a *signal* further than distance T , and thus the outcome it produces must also be *non-signaling*. This is formalized by the following definition:

Definition 4.3 (Non-signaling [GKM09]).

Let $\mathcal{G}$ be a family of networks, and let O be an outcome over the family. Outcome O is *non-signaling over $\mathcal{G}$ with locality T* if, for any two labeled networks (G_1, q_1, σ_1) and (G_2, q_2, σ_2) from family $\mathcal{G}$ with the same number of nodes, and any set S_1 of nodes in G_1 such that there exists a set S_2 in G_2 such that $B_{G_1}(S_1, T)$ is isomorphic to $B_{G_2}(S_2, T)$, the output distributions of outcome O over (G_1, q_1, σ_1) and (G_2, q_2, σ_2) have identical marginal distributions on sets S_1 and S_2 , respectively.

The non-signaling property of an outcome is exactly what is needed to perform sampling from a distribution in the randomized online-LOCAL model: it allows sampling from the global distribution given only a localized view of the input instance. When an adversary reveals a node and its neighborhood to the online-LOCAL algorithm, the algorithm can imagine *any* extension of the local view to a full input instance that is consistent with what it has seen so far, and the non-signaling property ensures that the outcome restricted to this local view is independent of the choice of the extension. The algorithm can then sample from this distribution, conditioned on what it has sampled before.

This ensures that the outcome distribution produced by the online-LOCAL algorithm matches the original distribution. In Publication III, we prove the following theorem:

Theorem 4.4 (From non-signaling to online-LOCAL [Publication III]).

Let Π be any LCL over any family of networks $\mathcal{G}$. Let O be an outcome over $\mathcal{G}$ which solves Π with failure probability at most ϵ and is non-signaling with locality T . Then there exists a randomized online-LOCAL algorithm $\mathcal{A}$ with locality T that solves Π over $\mathcal{G}$ and has failure probability at most ϵ .

4.3 Main contribution: Symmetry breaking in finitely dependent distributions

The output distributions of quantum-LOCAL algorithms actually satisfy an even stronger condition than the non-signaling property: in addition to the local output depending only on the local neighborhood of the nodes, if we consider two sets of nodes that are far apart, their outcomes are also *independent* of each other. If an outcome has this property, it is said to have *bounded dependence*:

Definition 4.5 (Bounded dependence [Publication III]).

*Let $\mathcal{G}$ be a family of networks, let O be a non-signaling outcome over this family, and let T be a radius. Let (G, q, σ) be a labeled network. Outcome O is *T -dependent on graph G* if, for any two sets $A \subseteq V(G)$ and $B \subseteq V(G)$ of vertices of G such that $\text{dist}_G(A, B) > T$, the output distributions of O marginalized on A and B are independent. If this holds for any graph in $\mathcal{G}$, then outcome O is *T -dependent on family $\mathcal{G}$* , or just *T -dependent*. If an outcome O is T -dependent for some T , then it has *bounded dependence*. Moreover, if T is a constant independent of the size of the input, then the outcome is said to be *finitely dependent*¹.*

Note that the bounded-dependence property is subtly different from the non-signaling property. Consider, for example, the following distribution: a global coin is tossed, and every node of the graph outputs the result of the coin toss. This is non-signaling, as the output distribution of any set of nodes matches the output distribution of the coin and, in particular, does not depend on the structure or the input of the graph far from the nodes. On the other hand, the outputs of all nodes of the graph are highly correlated, and hence the distribution does not have bounded dependence.

An important property of both bounded-dependence and non-signaling distributions is that they are purely classical and mathematical concepts with

¹The name comes from stochastic processes where mathematicians consider labeling *infinite* graphs, usually the graph of all integers $\mathbb{Z}$. In such graphs, the dependence being finite immediately implies that the radius of dependence is constant independent of the size of the graph.

no references to quantum. This gives hope that maybe they could be used to prove tight lower bounds for quantum-LOCAL as we currently lack generally-applicable lower-bound tools for quantum-LOCAL—tools like round elimination for classical LOCAL. In particular, we do not even know the locality of finding a 3-coloring of a cycle in quantum-LOCAL. Perhaps bounded-dependence distributions could be used to provide a tight lower bound for this problem?

In Publication III, we answer this question in the negative: We show that there exists a bounded-dependence distribution with $O(1)$ locality for every LCL problem that has locality $O(\log^* n)$ in the classical LOCAL model. This demonstrates that bounded-dependence and non-signaling arguments cannot be used to exclude quantum advantage for symmetry-breaking problems. This provides a formal obstacle for proving lower bounds for the quantum-LOCAL model, and therefore we need to find other tools to determine the complexity of symmetry-breaking problems in quantum-LOCAL.

Theorem 4.6 (Finitely dependent symmetry-breaking [Publication III]).

Let $\mathcal{G}$ be a family of networks, and let Π be an LCL problem that is solvable with locality $O(\log^ n)$ in the classical LOCAL model on the family $\mathcal{G}$. There exists a finitely dependent outcome O that solves Π on family $\mathcal{G}$.*

The full proof spans a dozen pages, but the core idea is relatively straightforward and relies on the observation that we can apply constant-round LOCAL algorithms to finitely dependent distributions to obtain finitely dependent distributions: First, we recall that directed paths admit finitely dependent distributions for 3-coloring [HHL18; HL16], and use that to provide a finitely dependent 3-coloring of pseudoforests; recall that pseudoforests are directed graphs where each node has out-degree at most 1. We then note that port-numbering gives us a family of Δ overlapping pseudoforests, where Δ is the degree of the input graph; this provides a coloring of the graph with 3^Δ colors. We then apply existing LOCAL color-reduction algorithms to bring the number of colors back to $\Delta + 1$ in a number of rounds that is only dependent on the degree Δ [CV86; GPS88]. Finally, we show that we can instead run this process in the power-graph of the input graph and still get a finitely dependent distribution, which allows us to obtain a distance- k -coloring of the graph. Recall that this is sufficient, as distance- k -coloring is a complete problem for LCL problems with locality $O(\log^* n)$, as I have already alluded to on page 43.

4.4 Main contribution: From online-LOCAL to SLOCAL

Thinking of online-LOCAL algorithms as SLOCAL algorithms with global memory raises an interesting question: does there exist something between having local memory and fully global memory? In Publication III, we provide an answer by defining *component-wise*—or *amnesiac*—online-LOCAL algorithms. These are algorithms that have access to *component-wise memory*: When

a node is processed, the algorithm also sees the neighborhood of the node being processed. This neighborhood may intersect with the neighborhoods of previously-processed nodes; in this case, we say that the nodes belong to the same *component*, as the algorithm sees how these nodes are connected to each other. When the algorithm sees that a node belongs to an existing component, it also gets to observe a memory register attached to that component and may replace the value with a new value.

Being component-wise is an especially desirable property for online-LOCAL algorithms solving LCL problems, as these are problems defined using local constraints. Indeed, it is hard to imagine a situation where the output for a node might depend in a useful way on a far-away region of the input that is not even known to be connected to the current node in question, yet general online-LOCAL algorithms may exhibit such behavior. However, it is worth noting that if the algorithm in question is randomized, then having access to *global randomness* is beneficial over having access to local or even *component-wise randomness* [BGK+25].

Another reason to have an online-LOCAL algorithm be component-wise is that then it has implications on the complexity landscape of LCL problems. In particular, in Publication III, we show that in *rooted trees*, a component-wise online-LOCAL algorithm can be simulated in SLOCAL with asymptotically the same locality. The core idea of the simulation is that a component-wise online-LOCAL algorithm can process multiple far-away components simultaneously in parallel while still ensuring that the labeling can be completed on the rest of the graph.

I will now sketch an SLOCAL algorithm that simulates a component-wise online-LOCAL algorithm $\mathcal{A}$ in rooted trees. For simplicity, I assume that the SLOCAL algorithm knows the total size of the input graph and, in particular, knows the locality T of the component-wise online-LOCAL algorithm being simulated. Moreover, I assume that the LCL problem Π has checkability radius 1. The SLOCAL algorithm consists of multiple passes executed one after another, each with locality $O(T)$; it is known that such composition still produces an SLOCAL algorithm with locality $O(T)$ [GKM17].

In the first pass, the SLOCAL algorithm selects a set of nodes that cut each leaf-to-root path into $\Theta(T)$ -sized sections. The algorithm can do this by first greedily picking a maximal independent set in the $(2T)^{\text{th}}$ power graph of the tree, and then “sliding down” these nodes by $4T$ in the tree. The result is that the rooted tree is now divided into clusters of diameter $\Theta(T)$, and the leaves of each cluster are either leaves of the original tree or roots of other clusters. Such clustering has also been previously used by Chang, Studený, and Suomela [CSS23].

In the second pass, the SLOCAL algorithm labels the *leaves* of each cluster in an arbitrary order using the online-LOCAL algorithm $\mathcal{A}$. Since the leaves of any cluster are at least at distance $2T$ from the leaves of any other cluster, the leaves of one cluster belong to a different component than the leaves of another

cluster. This means that the outputs of $\mathcal{A}$ for one cluster are independent of the outputs on another cluster while still being able to complete the labeling for the whole graph.

In the third and final pass, the SLOCAL algorithm labels the remaining nodes of the graph. Such a labeling exists by the assumption that algorithm $\mathcal{A}$ works correctly. All clusters have radius $\Theta(T)$ and have all their boundary nodes labeled—recall that the roots of the clusters are leaves of other clusters (except for the cluster that contains the root of the whole tree). Therefore, the labeling inside each cluster is independent of the labeling inside other clusters, and the SLOCAL algorithm can find such a labeling, for example, by brute force or by invoking algorithm $\mathcal{A}$ again.

Theorem 4.7 (SLOCAL simulation [Publication III]).

Let $\mathcal{A}$ be a component-wise online-LOCAL algorithm solving an LCL problem Π in rooted trees with locality T . Then there exists an SLOCAL algorithm solving Π with locality $O(T)$.

It is worth noting that the simulation fails in unrooted trees. The exact place where it fails is in the clustering of the graph: in unrooted trees, we do not know how to find low-diameter clusters such that their interfaces also have a low diameter.

To make this simulation more applicable, in Publication III, we also provide a mechanical way to turn *any* deterministic or randomized online-LOCAL algorithm solving an LCL problem into a component-wise online-LOCAL algorithm; as a side effect, this also derandomizes the algorithm. However, the process loses exponentially in the locality of the algorithm. For lowest complexity classes, the transformation can still be used to prove lower bounds on the complexity of LCL problems in online-LOCAL.

Theorem 4.8 (From online-LOCAL to component-wise [Publication III]).

Let $\mathcal{A}$ be an online-LOCAL algorithm solving an LCL problem Π with locality $T(n)$ on networks with n nodes. Then there exists a component-wise online-LOCAL algorithm solving the same problem Π with locality

- $T(2^{O(n^3)})$ if $\mathcal{A}$ is deterministic.
- $T(2^{O(n^3)} + 2^{O(2^{n^2})} \cdot \log \frac{1}{p(n)})$ if $\mathcal{A}$ is randomized. Here $p(n)$ denotes the success probability of $\mathcal{A}$.

As the locality hints at, the proof is based on simulating the original algorithm on an exponentially large experiment graph. In this experiment graph, we embed many copies of all possible neighborhoods that may appear in an n -node graph. We then use the original online-LOCAL algorithm to label these neighborhoods. As the algorithm is solving an LCL problem—which, recall, has a finite set of output labels—at least one label must appear in a constant fraction of these neighborhoods; we pick this label as the *canonical* label for that neighborhood. We then continue to combine these canonical neighborhoods

in all possible ways to construct more complicated graphs, each time using the original algorithm to label the graphs and finding a new canonical label for these larger graphs, until we reach graphs with n nodes.

After this process, we have found a canonical label for each neighborhood, even given that some labels of that neighborhood have already been labeled with their own canonical labels. Now, when the adversary finally asks us to label a node, we find the canonical label corresponding to this neighborhood. Observe that the canonical label depends only on the current component that the node resides in, and therefore this process produces a component-wise algorithm. The algorithm also works correctly as the experiment graph contains all possible executions of the online-LOCAL algorithm on all possible inputs in all possible orders, and the construction ensures that the canonical labeling is correct for all of these instances.

This experiment can also be modified to work with randomized online-LOCAL algorithms. However, as the algorithm is now randomized, we need to construct our experiment graph in an oblivious way. In particular, we can no longer adaptively find the label that the algorithm uses most often and pick that as the canonical label. Instead, we modify the process construction to “guess” the canonical nodes at random. The guesses work with an exponentially small probability, and to amplify the success probability, we replicate the experiment graph exponentially many times. This allows us to boost the success probability of the experiment to be positive in at least one of the replicates, which ensures that a canonical labeling exists. We then apply standard derandomization arguments to find the canonical labeling for each neighborhood by just enumerating over all possible component-wise online-LOCAL algorithms to find one that works.

Finally, recall that a locality- $o(\log n)$ SLOCAL algorithm for an LCL problem in rooted trees implies that there exists a locality- $O(1)$ SLOCAL algorithm for the same problem, and this in turn implies a locality- $O(\log^* n)$ classical LOCAL algorithm [GHK18; GKM17]. This can then be combined with Theorem 4.8 to get the following corollary which connects the locality of LCL problems across all models studied in this thesis:

Corollary 4.9 (Online-LOCAL speedup [Publication III]).

If there is an online-LOCAL algorithm solving an LCL problem Π in rooted trees with locality $o(\log \log \log n)$, then there exists a classical LOCAL algorithm solving Π with locality $O(\log^ n)$. This in turn implies $O(\log^* n)$ locality in quantum-LOCAL, and $O(1)$ locality in SLOCAL, online-LOCAL, dynamic-LOCAL, and finitely dependent distributions for Π .*

5. Sinkless orientation

For the final contribution of this thesis, let me consider one LCL problem that holds a central place in the study of the complexity of LCL problems in the LOCAL model: the *sinkless orientation* problem. In this chapter, I start by introducing the problem and explaining why it holds a central place. I then present our new and simpler proofs for the complexity of sinkless orientation in both the LOCAL model and the SLOCAL model, published in Publication V. The new lower bound for the LOCAL model is actually a lower bound for the *supported* LOCAL model. For the upper bound, I provide a new and direct algorithm in the SLOCAL model that does not go through derandomization.

5.1 Sinkless orientation

Sinkless orientation is the problem of orienting all edges of the graph such that no node is a sink, that is, every node has at least one outgoing edge. As this problem is hard on paths and cycles—fixing the orientation of one edge fixes the orientation of all other edges—it is customary to either consider 3-regular graphs or to let nodes with degree less than 3 be unconstrained. In this thesis, I adopt the latter view. In fact, I only consider graphs with maximum degree 3, as these are the hardest instances [GHK+20]: If a node has a higher degree, it can be split into a virtual degree-3 node and some number of degree-1 nodes. Sinkless orientation can then be solved on this graph, and by undoing this splitting, one gets a sinkless orientation of the original graph.

Definition 5.1 (Sinkless orientation [BFH+16]).

Sinkless orientation Π_{SO} is an LCL problem on maximum-degree-3 graphs consisting of

- input label set $\Sigma = \{\perp\}$,
- output label set $\Gamma = \{I, O\}$
- checking radius $r = 1$, and
- local constraints $\mathcal{C}$.

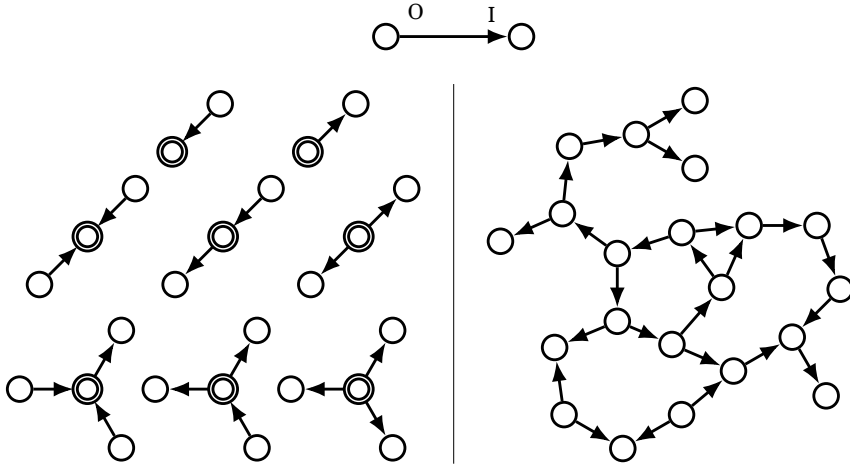


Figure 5.1. Sinkless orientation problem visualized. The output labels O and I are interpreted as arrows, as shown on the top. Valid centered neighborhoods $\mathcal{C}$ are visualized on the left with the center nodes highlighted, and an example sinkless orientation is shown on the right. Note that degree-1 and degree-2 nodes may be sinks.

Outputs on half-edges encode *orientation* of the edge, orienting the edge from the end labeled with O towards the end labeled with I; the labels at different ends of an edge must differ. The constraints $\mathcal{C}$ are so that every node of degree 3 has at least one outgoing edge.

Figure 5.1 visualizes the valid configurations of sinkless orientation and shows an example.

One interesting property of sinkless orientation is that it is an example of LCL problems for which randomness helps: sinkless orientation has locality $\Theta(\log n)$ in the deterministic LOCAL [BFH+16; CKP19; GS17] while having locality $\Theta(\log \log n)$ in the randomized LOCAL [BFH+16; GS17].

However, the main reason sinkless orientation holds such a central role in understanding the complexity of LCL problems in the LOCAL model is that it always admits a solution by the Lovász local lemma. Therefore, proving a lower bound for sinkless orientation also proves a lower bound for the distributed Lovász local lemma. The Lovász local lemma is a classic result in probability theory stating that, given a finite set of events that each happen with a sufficiently low probability and are independent of almost all other events, there exists a positive probability that none of the events occur:

Theorem 5.2 (Lovász local lemma [EL75; Spe77]).

Let $\mathcal{E} = \{E_1, E_2, \dots, E_n\}$ be a finite set of events such that each event E_i is independent of all but d other events, and the probability of each event happening individually is at most p . If, in addition, $ep(d+1) < 1$, then the probability that none of the events occur is positive; here e denotes Euler's number 2.718...

Usually, the events $\mathcal{E}$ are considered to depend only on a set

$$\mathcal{X} = \{X_1, \dots, X_k\}$$

of independent random variables; I denote by $\text{vbl}(E_i) \subseteq \mathcal{X}$ the subset of variables on which event E_i depends. These can now be interpreted as a *dependency graph* $G_{\mathcal{E}} = (\mathcal{E}, \mathcal{D})$, where the set $\mathcal{E}$ of events forms the vertex set, and the edge set is formed by events that depend on a common random variable:

$$\mathcal{D} = \{\{E_i, E_j\} \mid \text{vbl}(E_i) \cap \text{vbl}(E_j) \neq \emptyset\}.$$

In the distributed version of the Lovász local lemma, the dependency graph $G_{\mathcal{E}}$ is interpreted to also be the communication graph. The task for each node E_i is to output an assignment for all variables $\text{vbl}(E_i)$ it depends on such that the event E_i is avoided. Moreover, all nodes sharing the same random variable must agree on the assignment.

Definition 5.3 (Distributed Lovász local lemma [CPS17]).

Let $G_{\mathcal{E}} = (\mathcal{E}, \mathcal{D})$ be a dependency graph of the Lovász local lemma. Every node $E_v \in \mathcal{E}$ must output an assignment $a_v(X_i)$ for each random variable $X_i \in \text{vbl}(E_v)$ on which it depends, such that

1. for every $\{E_v, E_u\} \in \mathcal{D}$ and $X \in \text{vbl}(E_v) \cap \text{vbl}(E_u)$, it holds that $a_v(X) = a_u(X)$, and
2. the event E_v does not occur given the assignment a_v .

5.2 New lower bound via supported LOCAL

While the $\Omega(\log n)$ lower bound for sinkless orientation in the LOCAL model is known, the proof uses quite heavy machinery: it first shows an $\Omega(\log \log n)$ lower bound for the randomized LOCAL model [BFH+16] and then uses a gap result of Chang, Kopelowitz, and Pettie, which shows that there does not exist an LCL problem with locality $\omega(\log^* n)$ and $o(\log n)$ in the deterministic LOCAL model [CKP19], to arrive at the $\Omega(\log n)$ lower bound. In Publication V, we show a more straightforward lower bound proof by using the *supported* LOCAL model. This is a variant of the LOCAL model where each node knows a *support graph* G , and the input graph H is a subgraph of G . We show that, even in this setting, the sinkless orientation problem requires locality $\Omega(\log n)$, and hence it must also require this locality in the original LOCAL model.

5.3 New upper bound via high-degree sinkless orientation

Similarly, it is also known that sinkless orientation is solvable with locality $O(\log \log n)$ in the SLOCAL model, but again the proof uses some heavy ma-

chinery: it takes the randomized algorithm solving sinkless orientation with locality $O(\log \log n)$ [GS17] and derandomizes it [GHK18]. In Publication V, I provide a more direct deterministic SLOCAL algorithm.

Let me start by considering a variation of sinkless orientation called the *high-degree sinkless orientation*: in this problem, I increase the degree bound from 3 to $\delta = \lfloor \log n \rfloor + 1$, meaning that only nodes of degree at least δ need to be sinkless. I call nodes that either have an outgoing edge or have a degree less than δ *satisfied*, and the rest of the nodes *unsatisfied*. There is a simple greedy SLOCAL algorithm that solves this problem: When the algorithm processes node v , it orients each edge adjacent to v sequentially. When orienting an edge, the algorithm always orients it towards a node that is satisfied, breaking ties arbitrarily. If neither of the endpoints of the edge is satisfied, the algorithm orients the edge towards the node that has fewer of its adjacent edges oriented.

This algorithm works by the following observation: After every edge orientation operation, if we consider only the subgraph spanned by edges oriented between nodes that were both unsatisfied at the time of the orientation, the graph forms a rooted forest. Moreover, if a root of any tree has in-degree b , then the tree has at least 2^b nodes. This observation can be proven by induction, and it is sufficient to prove the correctness of the greedy algorithm: for a node of degree δ to become a sink, it must be the root of a tree with at least $2^\delta = 2^{\lfloor \log n \rfloor + 1} > n$ nodes, which contradicts the fact that the graph has only n nodes.

The next step is to turn this locality- $O(1)$ SLOCAL algorithm for high-degree sinkless orientation into an algorithm for the normal sinkless orientation problem. To do this, I construct an SLOCAL algorithm that clusters the nodes of the graph into clusters of diameter $O(\log \log n)$ such that every cluster is adjacent to either δ other clusters or contains a cycle or a node of degree less than 3. Such a clustering can be constructed in the SLOCAL model greedily with locality $O(\log \log n)$ by finding a maximal independent set in the power graph of the input graph and assigning each node to the cluster of the nearest independent-set node. The high-degree sinkless orientation algorithm can then be run on the virtual graph formed by the clusters with locality $O(\log \log n)$; this orients all inter-cluster edges. Finally, the rest of the edges inside the clusters can be oriented either towards the cycle or the low-degree node inside the cluster, or towards the inter-cluster edge that points away from the cluster. All of these steps can be implemented with locality $O(\log \log n)$ in the SLOCAL model, and thus the locality of their composition is also $O(\log \log n)$ [GKM17].

6. Conclusion

In this thesis, I have presented some of the latest developments in analyzing locality in distributed quantum algorithms and online graph algorithms. I have introduced the first local problem that shows distributed quantum advantage and presented the online-LOCAL model, which allows us to bring locality to online algorithms. The research presented in this thesis has already inspired many follow-up works.

In Chapter 2, I presented the first local problem showing quantum advantage. While the problem I presented is a family of LCL problems instead of a single LCL problem, this work has since inspired Balliu, Casagrande, d’Amore, Equi, Keller, Lievonon, Olivetti, Schmid, and Suomela to define a genuine LCL problem that separates the classical LOCAL and quantum-LOCAL models. Their LCL problem is solvable with locality $O(\log n)$ in the quantum-LOCAL model while requiring locality $\Omega(\log n \cdot \log^{0.99} \log n)$ in the classical LOCAL model [BCd+26].

Brandt and Göttlicher recently proved a $2^{\Omega(\log^* n)}$ lower bound for sinkless orientation in online-LOCAL [BG26]. This is especially noteworthy, as their lower bound—combined with the gap result of Chang and Pettie [CP19]—extends all the way to the LOCAL model. Before their work, only two techniques were known to provide lower bounds for the complexity of sinkless orientation in the LOCAL model: the round elimination technique [BFH+16] and Marks’ technique [Mar16].

While the work presented in this thesis has closed many open problems, it has left many others open.

In Publication I, I presented the iterated GHZ problem and showed that it exhibits distributed quantum advantage. However, the problem feels quite artificial and is carefully constructed to be easy to solve in the quantum-LOCAL model. This leaves the following question open:

Question 6.1. *Does there exist a natural graph problem that exhibits quantum advantage?*

As a concrete step toward answering this, it would be sufficient to study the complexity of 3-coloring cycles:

Question 6.2. *What is the locality of 3-coloring a directed cycle in the quantum-LOCAL model?*

In addition to being able to separate the quantum-LOCAL and the classical LOCAL model, we would also like to be able to separate other pairs of models. In particular, while we know some problems that separate the online-LOCAL model from the LOCAL model, it seems that many of our tools either stop working between the randomized LOCAL and quantum-LOCAL models (round elimination) or between non-signaling distributions and the randomized online-LOCAL model (linear-programming lower bounds [BCC+25]). Could there be a proof technique that provides a more fine-grained classification?

Question 6.3. *Is there an LCL problem that separates the quantum-LOCAL model from bounded-dependence distributions?*

References

- [ACK+25] Hagit Attiya, Christian Cachin, Dariusz R. Kowalski, Fabian Kuhn, Yoram Moses, and Paul Spirakis. “2025 Edsger W. Dijkstra Prize in Distributed Computing”. In: *39th International Symposium on Distributed Computing, DISC 2025, Berlin, Germany, October 27-31, 2025*. Ed. by Dariusz R. Kowalski. Vol. 356. LIPIcs. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025, 0:xxi. ISBN: 978-3-95977-402-4. DOI: 10.4230/LIPIcs.DISC.2025.0.
- [Ang80] Dana Angluin. “Local and Global Properties in Networks of Processors (Extended Abstract)”. In: *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*. Ed. by Raymond E. Miller, Seymour Ginsburg, Walter A. Burkhard, and Richard J. Lipton. ACM, 1980, pp. 82–93. DOI: 10.1145/800141.804655.
- [AOS+18] Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. “Fully dynamic maximal independent set with sublinear update time”. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*. Ed. by Ilias Diakonikolas, David Kempe, and Monika Henzinger. ACM, 2018, pp. 815–826. DOI: 10.1145/3188745.3188922.
- [BBC+19] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. “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*. Ed. by Peter Robinson and Faith Ellen. ACM, 2019, pp. 262–271. DOI: 10.1145/3293611.3331606.
- [BBC+22] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Jan Studený, and Jukka Suomela. “Efficient Classification of Locally Checkable Problems in Regular Trees”. In: *36th Inter-*

- national Symposium on Distributed Computing, DISC 2022, Augusta, Georgia, USA, October 25-27, 2022*. Ed. by Christian Scheideler. Vol. 246. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 8:1–8:19. DOI: 10.4230/LIPICS.DISC.2022.8.
- [BBG+26] Alkida Balliu, Sebastian Brandt, Ole Gabsdil, Dennis Olivetti, and Jukka Suomela. “On the Universality of Round Elimination Fixed Points”. In: *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, BC, Canada, January 11-14, 2026*. Ed. by Kasper Green Larsen and Barna Saha. SIAM, 2026, pp. 5066–5092. DOI: 10.1137/1.9781611978971.183.
- [BBH+21] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikael Rabie, and Jukka Suomela. “Lower Bounds for Maximal Matchings and Maximal Independent Sets”. In: *Journal of the ACM* 68.5 (2021), 39:1–39:30. DOI: 10.1145/3461458.
- [BBK+21] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. “Improved Distributed Lower Bounds for MIS and Bounded (Out-)Degree Dominating Sets in Trees”. In: *PODC ’21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*. Ed. by Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen. ACM, 2021, pp. 283–293. DOI: 10.1145/3465084.3467901.
- [BBK+22] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. “Distributed Δ -coloring plays hide-and-seek”. In: *STOC ’22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. Ed. by Stefano Leonardi and Anupam Gupta. ACM, 2022, pp. 464–477. DOI: 10.1145/3519935.3520027.
- [BBK+25] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, Dennis Olivetti, and Joonatan Saarhelo. “Towards Fully Automatic Distributed Lower Bounds”. In: *39th International Symposium on Distributed Computing, DISC 2025, Berlin, Germany, October 27-31, 2025*. Ed. by Dariusz R. Kowalski. Vol. 356. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025, 13:1–13:19. DOI: 10.4230/LIPICS.DISC.2025.13.
- [BBO+21] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. “Almost global problems in the LOCAL model”. In: *Distributed Computing* 34.4 (2021), pp. 259–281. DOI: 10.1007/S00446-020-00375-2.

- [BBT05] Gilles Brassard, Anne Broadbent, and Alain Tapp. “Quantum Pseudo-Telepathy”. In: *Foundations of Physics* 35.11 (2005), pp. 1877–1907. DOI: 10.1007/s10701-005-7353-4.
- [BCC+25] Alkida Balliu, Corinna Coupette, Antonio Cruciani, Francesco d’Amore, Massimo Equi, Henrik Lievonen, Augusto Modanese, Dennis Olivetti, and Jukka Suomela. “New Limits on Distributed Quantum Advantage: Dequantizing Linear Programs”. In: *39th International Symposium on Distributed Computing, DISC 2025, Berlin, Germany, October 27-31, 2025*. Ed. by Dariusz R. Kowalski. Vol. 356. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025, 11:1–11:22. DOI: 10.4230/LIPICS.DISC.2025.11.
- [BCd+26] 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”. In: *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, BC, Canada, January 11-14, 2026*. Ed. by Kasper Green Larsen and Barna Saha. SIAM, 2026, pp. 1268–1308. DOI: 10.1137/1.9781611978971.49.
- [BCH+18] Sayan Bhattacharya, Deeparnab Chakrabarty, Monika Henzinger, and Danupon Nanongkai. “Dynamic Algorithms for Graph Coloring”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. Ed. by Artur Czumaj. SIAM, 2018, pp. 1–20. DOI: 10.1137/1.9781611975031.1.
- [Bea76] Dwight R. Bean. “Effective coloration”. In: *The Journal of Symbolic Logic* 41.2 (1976), pp. 469–480. DOI: 10.2307/2272247.
- [BFH+16] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. “A lower bound for the distributed Lovász local lemma”. In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. Ed. by Daniel Wachs and Yishay Mansour. ACM, 2016, pp. 479–488. DOI: 10.1145/2897518.2897570.
- [BG26] Sebastian Brandt and Tim Göttlicher. “A Post-Quantum Lower Bound for the Distributed Lovász Local Lemma”. In: *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, BC, Canada, January 11-14, 2026*. Ed. by Kasper Green Larsen and Barna Saha. SIAM, 2026, pp. 3298–3312. DOI: 10.1137/1.9781611978971.119.

- [BGK+25] Alkida Balliu, Mohsen Ghaffari, Fabian Kuhn, Augusto Modanese, Dennis Olivetti, Mikael Rabie, Jukka Suomela, and Jara Uitto. “Shared Randomness Helps with Local Distributed Problems”. In: *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, Aarhus, Denmark, July 8-11, 2025*. Ed. by Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis. Vol. 334. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025, 16:1–16:18. DOI: 10.4230/LIPICS.ICALP.2025.16.
- [BGM93] Robert M. Burton, Marc Goulet, and Ronald Meester. “On 1-Dependent Processes and k -Block Factors”. In: *The Annals of Probability* 21.4 (1993). DOI: 10.1214/aop/1176989014.
- [BHK+17] Sebastian Brandt, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Patric R. J. Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław Uznanski. “LCL Problems on Grids”. In: *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*. Ed. by Elad Michael Schiller and Alexander A. Schwarzmann. ACM, 2017, pp. 101–110. DOI: 10.1145/3087801.3087833.
- [BHK+18] Alkida Balliu, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. “New classes of distributed time complexity”. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*. Ed. by Ilias Diakonikolas, David Kempe, and Monika Henzinger. ACM, 2018, pp. 1307–1318. DOI: 10.1145/3188745.3188860.
- [BHM+22] Alkida Balliu, Juho Hirvonen, Darya Melnyk, Dennis Olivetti, Joel Rybicki, and Jukka Suomela. “Local Mending”. In: *Structural Information and Communication Complexity - 29th International Colloquium, SIROCCO 2022, Paderborn, Germany, June 27-29, 2022, Proceedings*. Ed. by Merav Parter. Vol. 13298. Lecture Notes in Computer Science. Springer, 2022, pp. 1–20. DOI: 10.1007/978-3-031-09993-9_1.
- [BM19] Leonid Barenboim and Tzali Maimon. “Fully Dynamic Graph Algorithms Inspired by Distributed Computing: Deterministic Maximal Matching and Edge Coloring in Sublinear Update-Time”. In: *ACM Journal of Experimental Algorithmics* 24.1 (2019), 1.14:1–1.14:24. DOI: 10.1145/3338529.
- [BO20] Sebastian Brandt and Dennis Olivetti. “Truly Tight-in- Δ Bounds for Bipartite Maximal Matching and Variants”. In: *PODC ’20: ACM Symposium on Principles of Distributed Computing, Vir-*

- tual Event, Italy, August 3-7, 2020*. Ed. by Yuval Emek and Christian Cachin. ACM, 2020, pp. 69–78. DOI: 10.1145/3382734.3405745.
- [Bra19] Sebastian Brandt. “An Automatic Speedup Theorem for Distributed Problems”. In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. Ed. by Peter Robinson and Faith Ellen. ACM, 2019, pp. 379–388. DOI: 10.1145/3293611.3331611.
- [BV01] Paolo Boldi and Sebastiano Vigna. “An Effective Characterization of Computability in Anonymous Networks”. In: *Distributed Computing, 15th International Conference, DISC 2001, Lisbon, Portugal, October 3-5, 2001, Proceedings*. Ed. by Jennifer L. Welch. Vol. 2180. Lecture Notes in Computer Science. Springer, 2001, pp. 33–47. DOI: 10.1007/3-540-45414-4_3.
- [BW92] Charles H. Bennett and Stephen J. Wiesner. “Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states”. In: *Physical Review Letters* 69 (Nov. 1992), pp. 2881–2884. DOI: 10.1103/PhysRevLett.69.2881.
- [CDE+26] Antonio Cruciani, Avinandan Das, Massimo Equi, Henrik Lievenen, Diep Luong-Le, Augusto Modanese, and Jukka Suomela. *Is a LOCAL algorithm computable?* 2026. DOI: 10.48550/arXiv.2602.21022. arXiv: 2602.21022 [cs.DC].
- [Cha20] Yi-Jun Chang. “The Complexity Landscape of Distributed Locally Checkable Problems on Trees”. In: *34th International Symposium on Distributed Computing, DISC 2020, Virtual Conference, October 12-16, 2020*. Ed. by Hagit Attiya. Vol. 179. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 18:1–18:17. DOI: 10.4230/LIPICS.DISC.2020.18.
- [CHL+18] Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. “The Complexity of Distributed Edge Coloring with Small Palettes”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. Ed. by Artur Czumaj. SIAM, 2018, pp. 2633–2652. DOI: 10.1137/1.9781611975031.168.
- [CKP19] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. “An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model”. In: *SIAM Journal on Computing* 48.1 (2019), pp. 122–143. DOI: 10.1137/17M1117537.

- [CMN+24] Yi-Jun Chang, Gopinath Mishra, Hung Thuan Nguyen, Mingyang Yang, and Yu-Cheng Yeh. “A Tight Lower Bound for 3-Coloring Grids in the Online-LOCAL Model”. In: *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing, PODC 2024, Nantes, France, June 17-21, 2024*. Ed. by Ran Gelles, Dennis Olivetti, and Petr Kuznetsov. ACM, 2024, pp. 106–116. DOI: 10.1145/3662158.3662794.
- [CP19] Yi-Jun Chang and Seth Pettie. “A Time Hierarchy Theorem for the LOCAL Model”. In: *SIAM Journal on Computing* 48.1 (2019), pp. 33–69. DOI: 10.1137/17M1157957.
- [CPS17] Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. “Distributed algorithms for the Lovász local lemma and graph coloring”. In: *Distributed Computing* 30.4 (2017), pp. 261–280. DOI: 10.1007/S00446-016-0287-6.
- [CSS23] Yi-Jun Chang, Jan Studený, and Jukka Suomela. “Distributed graph problems through an automata-theoretic lens”. In: *Theoretical Computer Science* 951 (2023), p. 113710. DOI: 10.1016/J.TCS.2023.113710.
- [CV86] Richard Cole and Uzi Vishkin. “Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking”. In: *Information and Control* 70.1 (1986), pp. 32–53. DOI: 10.1016/S0019-9958(86)80023-7.
- [DKL+24] Anubhav Dhar, Eli Kujawa, Henrik Lievonen, Augusto Modanese, Mikail Muftuoglu, Jan Studený, and Jukka Suomela. “Local Problems in Trees Across a Wide Range of Distributed Models”. In: *28th International Conference on Principles of Distributed Systems, OPODIS 2024, Lucca, Italy, December 11-13, 2024*. Ed. by Silvia Bonomi, Letterio Galletta, Etienne Rivière, and Valerio Schiavoni. Vol. 324. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 27:1–27:17. DOI: 10.4230/LIPICS.OPODIS.2024.27.
- [DZ18] Yuhao Du and Hengjie Zhang. *Improved Algorithms for Fully Dynamic Maximal Independent Set*. 2018. DOI: 10.48550/arXiv.1804.08908. arXiv: 1804.08908 [cs.DS]. URL: <https://arxiv.org/abs/1804.08908>.
- [EL75] P. Erdős and L. Lovász. “Problems and results on 3-chromatic hypergraphs and some related questions”. In: *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vols. I, II, III*. Vol. 10. Colloq. Math. Soc. János Bolyai. North-Holland, Amsterdam-London, 1975, pp. 609–627.

- [GHK+20] Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, Yannic Maus, Jukka Suomela, and Jara Uitto. “Improved distributed degree splitting and edge coloring”. In: *Distributed Computing* 33:3-4 (2020), pp. 293–310. DOI: 10.1007/S00446-018-00346-8.
- [GHK18] Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. “On Derandomizing Local Distributed Algorithms”. In: *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. Ed. by Mikkel Thorup. IEEE Computer Society, 2018, pp. 662–673. DOI: 10.1109/FOCS.2018.00069.
- [GHZ89] Daniel M. Greenberger, Michael A. Horne, and Anton Zeilinger. “Going Beyond Bell’s Theorem”. In: *Bell’s Theorem, Quantum Theory and Conceptions of the Universe*. Springer Netherlands, 1989, pp. 69–72. DOI: 10.1007/978-94-017-0849-4_10.
- [GK21] Manoj Gupta and Shahbaz Khan. “Simple dynamic algorithms for Maximal Independent Set, Maximum Flow and Maximum Matching”. In: *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*. Ed. by Hung Viet Le and Valerie King. SIAM, 2021, pp. 86–91. DOI: 10.1137/1.9781611976496.10.
- [GKM09] Cyril Gavoille, Adrian Kosowski, and Marcin Markiewicz. “What Can Be Observed Locally?” In: *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*. Ed. by Idit Keidar. Vol. 5805. Lecture Notes in Computer Science. Springer, 2009, pp. 243–257. DOI: 10.1007/978-3-642-04355-0_26.
- [GKM17] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. “On the complexity of local distributed graph problems”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. Ed. by Hamed Hatami, Pierre McKenzie, and Valerie King. ACM, 2017, pp. 784–797. DOI: 10.1145/3055399.3055471.
- [GPS88] Andrew V. Goldberg, Serge A. Plotkin, and Gregory E. Shannon. “Parallel Symmetry-Breaking in Sparse Graphs”. In: *SIAM Journal on Discrete Mathematics* 1.4 (1988), pp. 434–446. DOI: 10.1137/0401044.
- [GRB22] Christoph Grunau, Václav Rozhon, and Sebastian Brandt. “The Landscape of Distributed Complexities on Trees and Beyond”. In: *PODC ’22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*. Ed. by Alessia Milani and Philipp Woelfel. ACM, 2022, pp. 37–47. DOI: 10.1145/3519270.3538452.

- [Gro96] Lov K. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. Ed. by Gary L. Miller. ACM, 1996, pp. 212–219. DOI: 10.1145/237814.237866.
- [GS17] Mohsen Ghaffari and Hsin-Hao Su. “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*. Ed. by Philip N. Klein. SIAM, 2017, pp. 2505–2523. DOI: 10.1137/1.9781611974782.166.
- [HHL18] Alexander E. Holroyd, Tom Hutchcroft, and Avi Levy. “Finitely dependent cycle coloring”. In: *Electronic Communications in Probability* 23 (2018). DOI: 10.1214/18-ecp118.
- [HKR14] Maurice Herlihy, Dmitry Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Boston: Morgan Kaufmann, 2014. DOI: 10.1016/C2011-0-07032-1.
- [HL16] Alexander E. Holroyd and Thomas M. Liggett. “Finitely Dependent Coloring”. In: *Forum of Mathematics, Pi* 4, e9 (2016). DOI: 10.1017/fmp.2016.7.
- [Hol24] Alexander E. Holroyd. “Symmetrization for finitely dependent colouring”. In: *Electronic Communications in Probability* 29 (2024). DOI: 10.1214/24-ecp600.
- [IL93] Zoran Ivkovic and Errol L. Lloyd. “Fully Dynamic Maintenance of Vertex Cover”. In: *Graph-Theoretic Concepts in Computer Science, 19th International Workshop, WG ’93, Utrecht, The Netherlands, June 16-18, 1993, Proceedings*. Ed. by Jan van Leeuwen. Vol. 790. Lecture Notes in Computer Science. Springer, 1993, pp. 99–111. DOI: 10.1007/3-540-57899-4_44.
- [Lin87] Nathan Linial. “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, 1987, pp. 331–335. DOI: 10.1109/SFCS.1987.20.
- [Lin92] Nathan Linial. “Locality in Distributed Graph Algorithms”. In: *SIAM Journal on Computing* 21.1 (1992), pp. 193–201. DOI: 10.1137/0221015.
- [LNR19] François Le Gall, Harumichi Nishimura, and Ansis Rosmanis. “Quantum Advantage for the LOCAL Model in Distributed Computing”. In: *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, Berlin, Germany, March 13-16, 2019*. Ed. by Rolf Niedermeier and Christophe

- Paul. Vol. 126. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 49:1–49:14. DOI: 10.4230/LIPICS.STACS.2019.49.
- [Mar16] Andrew Marks. “A determinacy approach to Borel combinatorics”. In: *Journal of the American Mathematical Society* 29.2 (2016), pp. 579–600. DOI: 10.1090/jams/836.
- [Mer90] N. David Mermin. “Quantum mysteries revisited”. In: *American Journal of Physics* 58.8 (1990), pp. 731–734. DOI: 10.1119/1.16503.
- [NS16] Ofer Neiman and Shay Solomon. “Simple Deterministic Algorithms for Fully Dynamic Maximal Matching”. In: *ACM Transactions on Algorithms* 12.1 (2016), 7:1–7:15. DOI: 10.1145/2700206.
- [NS93] Moni Naor and Larry J. Stockmeyer. “What can be computed locally?” In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16–18, 1993, San Diego, CA, USA*. Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. ACM, 1993, pp. 184–193. DOI: 10.1145/167088.167149.
- [NS95] Moni Naor and Larry J. Stockmeyer. “What Can be Computed Locally?” In: *SIAM Journal on Computing* 24.6 (1995), pp. 1259–1277. DOI: 10.1137/S0097539793254571.
- [Pel00] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. DOI: 10.1137/1.9780898719772.
- [Sho97] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172.
- [Spe77] Joel Spencer. “Asymptotic lower bounds for Ramsey functions”. In: *Discrete Mathematics* 20 (1977), pp. 69–76. DOI: 10.1016/0012-365X(77)90044-9.
- [Tim24] Ádám Timár. “Finitely dependent random colorings of bounded degree graphs”. In: *CoRR* abs/2402.17068 (2024). DOI: 10.48550/arXiv.2402.17068. arXiv: 2402.17068.
- [YK96] Masafumi Yamashita and Tsunehiko Kameda. “Computing on anonymous networks: part I—characterizing the solvable cases”. In: *IEEE Transactions on Parallel and Distributed Systems* 7.1 (1996), pp. 69–89. DOI: 10.1109/71.481599.

Errata

Publication III

The proof of Theorem 3.2 in Section 3.2 is wrong. In fact, it is not known whether the theorem holds.

Business, Economy
Art, Design, Architecture
Science, Technology
Crossover

| Doctoral Theses

Aalto DT 162/2026

ISBN 978-952-64-3254-0
ISBN 978-952-64-3253-3 (pdf)

Aalto University
School of Science
Department of Computer Science
aalto.fi