Department of Computer Science

Lower bounds in distributed computing

Juho Hirvonen

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 239/2016

Lower bounds in distributed computing

Juho Hirvonen

A doctoral dissertation 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 Aalto University Otaniemi campus T-building (Konemiehentie 2) on 25 November 2016 at 10.

Aalto University
School of Science
Department of Computer Science
Distributed Algorithms

Supervising professor
Professor Jukka Suomela, Aalto University, Finland

Thesis advisor
Professor Jukka Suomela, Aalto University, Finland

Preliminary examiners
Professor Fabian Kuhn, University of Freiburg, Germany
Professor Danupon Nanongai, Kungliga Tekniska högskolan, Sweden

Opponent
Professor Michael Elkin, Ben-Gurion University of the Negev, Israel

Aalto University publication series
DOCTORAL DISSERTATIONS 239/2016

ISBN 978-952-60-7138-1 (printed)
ISBN 978-952-60-7137-4 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-7137-4

Unigrafia Oy
Helsinki 2016

Finland

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Juho Hirvonen

Name of the doctoral dissertation
Lower bounds in distributed computing

Publisher School of Science

Unit Department of Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 239/2016

Field of research Information and Computer Science

Date of the defence 25 November 2016

Permission to publish granted (date) 19 August 2016 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

In this thesis I study the complexity theory of distributed computing in synchronous message passing models. The focus is on highly local problems, that is, problems in which very little communication is required. In this setting the underlying communication network is also the input graph. The distributed system must collectively compute a solution to a problem related to the structure of this network, with each computer producing its own part of the output.

We study the LOCAL model, one of the standard models in distributed computing. It abstracts away faults, congestion, computational requirements, memory requirements, and many other challenges in distributed computing. We study this model to understand the locality aspect of distributed computing: How far does information have to propagate in distributed problem solving? How many communication rounds is required?

Many interesting problems, such as finding a spanning tree in the communication network, are inherently global. We are interested in the other extreme: problems that can be solved in time that is weakly dependent or completely independent of the size of the communication network. Typical problems include classical symmetry breaking tasks such as colouring or maximal independent set. Typically the time complexity of such problems depends on two parameters: the size and the maximum degree of the input graph. The connection with the first parameter is well understood with tight upper and lower bounds. The connection with the maximum degree is much less well understood, with an exponential gap between the upper and lower bounds. We develop a new lower bound technique to give the first lower bound for a natural graph problem that is linear in the maximum degree.

In addition we study the power of unique in several contexts. We show that while usually unique identifiers are unhelpful for constant-time algorithms, there are certain special cases in which they do help. In the context of local decision, where the task is essentially to verify a given solution instead of constructing a new one, we characterize the minimal information that is sufficient to replace unique identifiers.

Finally, we also study the power of nondeterminism in local decision. We develop a local hierarchy in a manner analogous to the polynomial hierarchy in centralized computing, and study the structure of this hierarchy.

The focus of this thesis is on structural results, especially lower bounds. The lower bounds as a function of the maximum degree of the graph employ a completely novel proof technique based on graph symmetries.

Keywords distributed computing, local algorithms, lower bounds

ISBN (printed) 978-952-60-7138-1	**ISBN (pdf)** 978-952-60-7137-4	
ISSN-L 1799-4934	**ISSN (printed)** 1799-4934	**ISSN (pdf)** 1799-4942
Location of publisher Helsinki	**Location of printing** Helsinki	**Year** 2016
Pages 152	**urn** http://urn.fi/URN:ISBN:978-952-60-7137-4	

Aalto-yliopisto

Tiivistelmä

Aalto-yliopisto, PL 11000, 00076 Aalto www.aalto.fi

Tekijä
Juho Hirvonen

Väitöskirjan nimi
Lower bounds in distributed computing

Julkaisija Perustieteiden korkeakoulu

Yksikkö Tietotekniikan laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 239/2016

Tutkimusala Tietojenkäsittelytiede

Väitöspäivä 25.11.2016

Julkaisuluvan myöntämispäivä 19.08.2016 **Kieli** Englanti

☐ **Monografia** ☒ **Artikkeliväitöskirja** ☐ **Esseeväitöskirja**

Tiivistelmä

Tutkin tässä väitöskirjatyössä hajautetun laskennan vaativuusteoriaa synkronisissa viestinvälitysmalleissa. Työ keskittyy paikallisiin ongelmiin, eli ongelmiin, joiden ratkaisu voidaan löytää käyttäen vain vähäinen määrä kommunikaatiota. Tutkitussa asetelmassa kommunikaatioverkko on myös ongelman syöte. Hajautetun järjestelmän täytyy löytää ratkaisu, joka liittyy tämän verkon rakenteeseen ja jokaisen verkon solmun täytyy tuottaa oma osansa tulosteesta.

Työssä tutkitaan hajautetun laskennan standardimalleihin kuuluvaa LOCAL-mallia. Tämä malli ei huomioi erilaisia hajautetun laskennan haasteita, kuten vikoja, ruuhkautumista, tai laskenta- ja muistivaatimuksia. Tätä mallia tutkitaan paikallisuuden ymmärtämiseksi: kuinka kauas informaatiota täytyy propagoida, jotta ongelma voidaan ratkaista hajautetusti? Kuinka monta kommunikaatiokierrosta ongelman ratkaiseminen vaatii?

Monet kiinnostavat ongelmat, kuten kommunikaatioverkon virittävän puun löytäminen, ovat globaaleja ongelmia. Tässä työssä tarkastelemme toista ääripäätä: ongelmia, jotka voidaan ratkaista ajassa, joka on kokonaan riippumaton tai korkeintaan heikosti riippuvainen kommunikaatioverkon koosta. Tälläisiä ongelmia ovat esimerkiksi klassiset symmetrianrikkomisongelmat kuten väritys ja riippumattomat joukot. Tyypillisesti näiden ongelmien aikavaativuus riippuu kahdesta tekijästä, verkon koosta ja sen maksimiasteluvusta. Yhteys verkon kokoon on melko hyvin ymmärretty, mutta riippuvuus maksimiasteluvusta ei. Tunnettujen ylä- ja alarajojen erotus on eksponentiaalinen. Kehittämämme alarajatekniikka antaa ensimmäisen maksimiasteluvun suhteen lineaarisen alarajan luonnolliselle verkko-ongelmalle.

Lisäksi olemme tutkineet solmuille annettavien uniikkien nimien vaikutusta. Vaikka tyypillisesti uniikit tunnisteet eivät auta vakioaikaisia algoritmeja, näytämme, että tietyissä tilanteissa niitä voidaan hyödyntää. Paikallisten päätösongelmien tapauksessa, jossa solmujen täytyy tarkastaa annettu ratkaisu uuden rakentamisen sijaan, karakterisoimme sen informaation määrän, joka riittää ja vaaditaan uniikkien tunnisteiden korvaamiseen.

Viimeiseksi tutkimme epädeterministisyyden vaikutusta paikallisiin päätösongelmiin. Kehitämme paikallisen päätöshierarkian, joka on analoginen keskitetyn laskennan polynomisen hierarkian kanssa, ja tutkimme tämän hierarkian rakennetta.

Väitöskirjatyö keskittyy rakenteellisiin tuloksiin, erityisesti mahdottomuustuloksiin. Verkon asteluvusta riippuvat alarajatulokset edustavat täysin uudenlaista alarajatekniikkaa, joka hyödyntää verkossa olevaa symmetriaa.

Avainsanat hajautettu laskenta, paikalliset algoritmit, alarajat

ISBN (painettu) 978-952-60-7138-1	**ISBN (pdf)** 978-952-60-7137-4	
ISSN-L 1799-4934	**ISSN (painettu)** 1799-4934	**ISSN (pdf)** 1799-4942
Julkaisupaikka Helsinki	**Painopaikka** Helsinki	**Vuosi** 2016
Sivumäärä 152	**urn** http://urn.fi/URN:ISBN:978-952-60-7137-4	

Preface

A great many people have helped me become the researcher that I am. First, I want to thank my supervisor Jukka Suomela for his guidance, and for always setting an example as a researcher. I also want to especially thank Patrik Floréen for pushing me towards becoming a researcher.

Thanks are also due to all of my fellow PhD students, past and present: Laurent Feuilloley, Mika Göös, Janne Korhonen, Tuomo Lempiäinen, and Joel Rybicki. Sorry for all those times I barged into your offices and interrupted your work! Many thanks to all of my colleagues at the University of Helsinki and at Aalto University, in particular Petteri Kaski, Juhana Laurinharju, Christopher Purcell, and Przemek Uznański.

Many thanks to Pierre Fraigniaud for the opportunity to visit Paris, and to all the people of the LIAFA laboratory there. I also want to thank my co-authors: Sebastian Brandt, Orr Fischer, Henning Hasemann, Barbara Keller, Reut Levi, Moti Medina, Stefan Schmid, and Jara Uitto.

Thanks to Fabian Kuhn and Danupon Nanongai for being the pre-examiners of my thesis. Many thanks to Michael Elkin for agreeing to be the opponent of my thesis defense.

Thanks to my fellow metaphysicists Eero, Eetu, Erno, Jukka, Miika, and Teemu: our discussions on everything have made me a much wiser scientist. I am grateful for the support of my parents, Eeva and Pekka, and my brother Olli. Finally, my most heartfelt thanks to Aino Kalmbach.

This work was supported in part by the Academy of Finland, Grants 132380 and 285721, the Foundation for Aalto University Science and Technology, and the Nokia Foundation.

Helsinki, 26th October 2016,

Juho Hirvonen

Contents

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 Juho Hirvonen and Jukka Suomela. Distributed maximal matching: greedy is optimal. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing (PODC 2012)*, Funchal, Portugal, pages 165–174, July 2012.

II Pierre Fraigniaud, Juho Hirvonen, and Jukka Suomela. Node labels in local decision. *Proceedings of the 22nd International Colloquim on Structural Information and Communication Complexity (SIROCCO 2015)*, Montserrat, Spain, pages 31–45, July 2015.

III Laurent Feuilloley, Pierre Fraigniaud, and Juho Hirvonen. A hierarchy of local decision. Accepted for publication in *Proceedings of the 43rd International Colloquim on Automata, Languages, and Programming (ICALP 2016)*, July 2016.

IV Henning Hasemann, Juho Hirvonen, Joel Rybicki, and Jukka Suomela. Deterministic local algorithms, unique identifiers, and fractional graph colouring. *Theoretical Computer Science*, Volume 610, part B, 11, pages 204–217, January 2016.

V Mika Göös, Juho Hirvonen, and Jukka Suomela. Linear-in-Δ lower bounds in the LOCAL model. Accepted for publication in *Distributed Computing*, 2016.

Author's Contribution

Publication I: "Distributed maximal matching: greedy is optimal"

The result and the write-up are joint work, with J. Suomela responsible for the idea of the lower bound proof.

Publication II: "Node labels in local decision"

The author had a major role in the research work and had a leading role in writing the paper.

Publication III: "A hierarchy of local decision"

The author had a major role in the research work and had a leading role in writing the paper.

Publication IV: "Deterministic local algorithms, unique identifiers, and fractional graph colouring"

All results are joint work. The author was mainly responsible for the lower bound results (Theorem 2).

Publication V: "Linear-in-Δ lower bounds in the LOCAL model"

The author was mostly involved in the proofs of the lower bound for the EC and the simulation for the PO model. The idea and the write-up are due to M. Göös.

1. Introduction

This thesis studies distributed computing in large-scale communication networks. We are interested in the basic question of how far information has to be propagated in those networks in order to solve problems related to their structure. We are especially interested in tasks that can be solved *locally*, that is, the solution in one part of the communication network does not depend on the solution in other far-away parts of the network. The technical focus is on structural results, especially lower bound results. This introduction is intended, in part, as a toolbox of lower bound techniques in distributed computing.

1.1 The setting

This work studies synchronous distributed message passing models. In this setting, the communication network is also the input graph – the computers are the vertices and the communication links the edges of the graph. The computers run the same deterministic algorithm, communicating via the communication links in synchronous rounds, with the aim of solving some problem related to the structure of the communication network itself. We are interested in locality. Which problems are such that each computer only requires information local to its own neighbourhood in the network to decide its own output. To this end, we use a model, known as the LOCAL model [45, 54], where all other challenges of distributed computing are abstracted away: there are no faults, no bounds on message size, no bounds on the memory nor the computation performed by the computers, and no Byzantine adversaries. Due to its strength, the LOCAL model is a good model for proving impossibility results: any results applicable to it are also applicable to a wide variety of models in distributed computing.

Since the question is one of information propagation and not computation,

we measure the distributed time complexity as the number of communication rounds the system has to take until all vertices of the network have stopped and produced their own local part of the output. Since there are no bounds on communication, the running time of each vertex corresponds exactly to the distance from which it has to receive information in order decide its own output, making time and distance essentially interchangable. In this sense, many interesting distributed problems are clearly *global* instead of local. Such problems include computing spanning trees, efficient routing, or finding the optimum solution of various optimization problems. We are interested in the opposite type of problems: local problems, problems for which the dependence of the running time on the size of the graph can be made very small.

From the perspective of locality, one of the most important classes of problems are so-called *locally checkable labellings* [49]. Informally, a graph problem is locally checkable, if given a solution, each vertex can verify locally that the solution is correct in its local neighbourhood. As an example of such a problem, consider k-colouring, the problem of labelling the vertices with k colours in such a way that no two vertices connected by an edge have the same colour. This problem is locally checkable since each vertex can ask its neighbours for their colours and check that they differ from its own.

The best we could hope for is that a problem could be solved in constant time, that is, the time complexity of a problem would be completely independent of the size of the input graph. Such an algorithm is also known as a *local algorithm*. There are many problems for which local algorithms do not exist: In his seminal work, Linial showed that 3-colouring an n-vertex ring network requires $\Omega(\log^* n)$ time [45]. This is also the true complexity of graph colouring [17]. By reduction, the same lower bound also applies to *maximal matching* and *maximal independent set*. These problems are classic examples of *symmetry breaking problems*. Broadly speaking these are problems that require that the vertices produce different outputs. It turns out that symmetry breaking is in general impossible for constant-time algorithms, even if the vertices are labelled with globally unique names [49].

On the positive side, there are non-trivial problems that can be solved in constant time. The key observation is that solving some problems does not require symmetry breaking in symmetric graphs. Typically the bad inputs are cycles or other regular graphs of even degree so that the local

structure of the graph does not give any symmetry breaking information. For many problems a trivial, completely symmetric solution is feasible in such graphs. Examples of such problems include 2-approximation of vertex cover [6], many linear programming problems [21, 41, 42], and the linear relaxation of maximal matching [6]. See the survey of Suomela for a general overview of locally solvable tasks [58].

Somewhat orthogonally to the question of how fast a problem can be solved as a function of the graph size n, it often happens that the best known algorithms, as a function of the size of the graph, have a linear dependency on the maximum degree Δ of the input graph [13, 50]. Unlike the dependency on n, this dependency is fairly poorly understood and generally no matching lower bounds exist. The work presented in this thesis contains the first linear-in-Δ lower bounds. We argue that this linear dependency, presents a second challenge of *local coordination* to distributed computing, one that is orthogonal to symmetry breaking [59]. While a lower bound that simultaneously captures aspects of symmetry breaking and local coordination is still beyond the reach of current techniques, in Publication V we show that there are problems that cannot be solved in time $o(\Delta)$ independent of n.

Randomization is often of little help when trying to solve locally checkable labelling problems in constant or almost constant time. Naor showed that randomization cannot reduce the complexity of colouring below $\Omega(\log^* n)$ rounds [48]. Naor and Stockmeyer showed that constant-time algorithms do not gain benefit from randomization when solving any locally checkable labelling problems. In this work we will concentrate on *deterministic* algorithms, unless otherwise specified.

In this thesis I study several questions in this extremely local corner of the distributed complexity landscape.

- In Publication IV we study the power of unique identifiers in constant-time distributed computation. We show that there are locally checkable problems in which unique names do help. This result also shows that the technical limitations of the result of Naor and Stockmeyer are also necessary [49].

- In Publication I and in Publication V we study the distributed complexity as a function of the maximum degree of the input graph. We give the first lower bounds that are linear in Δ, which is an exponential improvement over the previous best lower bounds [42].

- In Publication II we study the power of unique identifiers in the context of local decision. We establish the minimal amount of information that captures the power of unique names. It turns out that the useful property of unique identifiers is different from the case of corresponding construction tasks.
- In Publication III we study the power of non-determinism in local decision. We present a *local hierarchy* (analogous to the polynomial hierarchy) and give several structural results concerning it.

1.1.1 Structure of this work

We will continue by giving necessary definitions about graphs and distributed graph algorithms in Section 1.2. After that, we will give brief introductions to the topics of this thesis in Sections 1.3, 1.4, and 1.5. In Section 1.6 I give a general non-exhaustive overview of lower bound techniques for distributed graph algorithms.

The latter part is divided in three chapters. In Chapter 2 I present two linear-in-Δ lower bounds, one for maximal matching in an anonymous model and a second one for fractional maximal matching in the standard LOCAL model. In Chapter 3 we look at the results in Publication IV, where we study the power of unique identifiers in constant-time computing. Finally, in Chapter 4 we study the power of unique identifiers and the power of nondeterminism in the context of local decision.

1.2 Definitions

In this section I present definitions that will be used throughout this work. Some definitions, that are only used in one of the chapters, will be presented as required.

Sets. The set of integers $\{1, 2, \ldots, n\}$ is denoted in short by $[n]$.

1.2.1 Graph theory

In this section I give the necessary graph theoretic definitions. Most definitions are given as characterizations via subgraphs [20].

Graphs. A *graph* G is denoted by a pair (V, E), where $V = V(G)$ is a set of *vertices* and $E = E(G)$ a set of *edges*. The *size* of a graph G is defined to

be the size of its vertex set $|V(G)|$, denoted by n. By default the edges are undirected pairs of vertices $\{u, v\}$. We denote directed edges by $e = (u, v)$, where u is the *tail* of e and v is the *head* of v. For each edge $e = \{u, v\}$, we say that u and v are *adjacent* and that both u and v are *incident* to edge e. A vertex u adjacent to another vertex v is a *neighbour* of v, and the set of neighbours of v is denoted by $N(v)$.

The *degree* $\deg_G(v)$ of vertex v in graph G is the number of edges incident to v. The *maximum degree* $\Delta = \Delta(G)$ of graph G is the maximum degree among its vertices. A family $\mathcal{F}$ of graphs has *bounded degree* if there is a constant c such that for all $G \in \mathcal{F}$, we have that $\Delta(G) \leq c$. A graph is d*-regular*, if each of its vertices has degree d.

The *distance* $\mathrm{dist}_G(v, u)$ between two vertices v and u is the length of the shortest path between u and v, that is, the number of edges on that path. The *girth* $g(G)$ of graph G is equal to the length of its shortest cycle. The *radius* $r(G)$ of G is defined as $\min_{v \in V(G)} \max_{u \in V(G)} \mathrm{dist}_G(v, u)$.

Graph problems. Typically we will deal with graph problems where the solution is either a subset of vertices of a subset of edges.

- An *independent set* is a subset of vertices $I \subseteq V(G)$ such that there are no edges $\{v, u\}$ for any $u, v \in I$. An independent set is *maximal*, if it is not a proper subset of any other independent set.
- A *matching* is a subset of edges $M \subseteq E(G)$ such that no two edges $e, e' \in M$ share an endpoint. A matching is *maximal*, if it is not a proper subset of any other matching.
- A k*-vertex colouring* is a labelling $c \colon V(G) \to [c]$ such that for every pair of adjacent vertices u and v we have that $c(v) \neq c(u)$.

1.2.2 Models of distributed computing

The underlying model of distributed computing is throughout this work the model of Linial [45], known as the LOCAL model, with synchronous rounds of communication. The specific input labels, such as unique identifiers, change depending on the context. We will usually assume that the input graphs have bounded maximum degree, except in the context of local decision.

The LOCAL model. In the LOCAL model, the input graph G also represents the communication network. Each vertex is running the same, deterministic algorithm. The vertices operate in synchronous communication rounds, during each of which each vertex may send a message to each

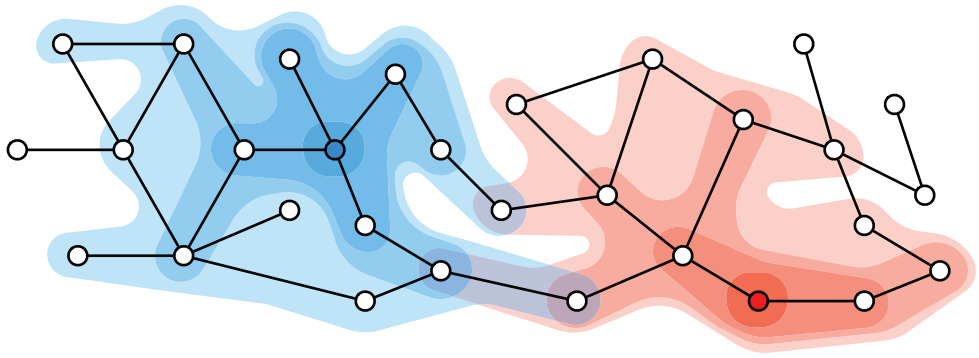

Figure 1.1. The information gathering radii $t = 0, 1, 2, 3$ for two vertices in the synchronous message passing model. Note that edges which have both of their endpoints at distance t of a vertex are not visible to vertex v after t rounds of communication.

of its neighbours and receive a message from each of its neighbours. The size of these messages is unbounded and there are no vertex failures.

In t synchronous communication rounds, a vertex can gather *all* information about its radius-t local neighbourhood in the graph, including all input labellings such as unique identifiers, and possible additional inputs. See Figure 1.1 for illustration.

Local neighbourhoods. We will denote this ball of radius t around vertex v in graph G by $B_G(v, t)$. Note that edges are contained in $B_G(v, t)$ if at least one their endpoints is within distance $t - 1$ of v. If the neighbourhoods of two vertices u and v are isomorphic, we denote this by $B_G(v, t) \simeq B_G(u, t)$.

Constant-time algorithms. If the running time of an algorithm A is a constant t independent of n, then A can be defined as a mapping from the set of possible t-neighbourhoods to the set of possible outputs. In a bounded-degree graph we may assume that the vertices know this bound, as all of our lower bound constructions will still apply.

Locally checkable labellings. A locally checkable (vertex) labelling L (LCL for short) consists of an integer r, the locality parameter or *radius* of L, a finite set of input labels Σ, a finite set of output labels Γ, and a finite set of locally consistent labellings $\mathcal{C}$. Each element of $\mathcal{C}$ is a graph H of radius at most r with a fixed center vertex v. Each vertex of H is labelled with a pair from $\Sigma \times \Gamma$. A labelling $x \colon V(G) \to \Sigma \times \Gamma$ is consistent with L if for every $v \in V(G)$ there is an isomorphism that maps $B_G(v, r)$ to an element H of $\mathcal{C}$, such that v is the center of H and the mapping respects the input and output labellings. None of the LCLs considered in this thesis have input labels. Note that the finiteness of $\mathcal{C}$ implies that by definition,

only bounded-degree graphs admit locally checkable labellings, and the finiteness of Γ implies that all LCLs must have a bounded number of possible outputs.

The various models of distributed computing used throughout this work are presented next and illustrated in Figure 1.2.

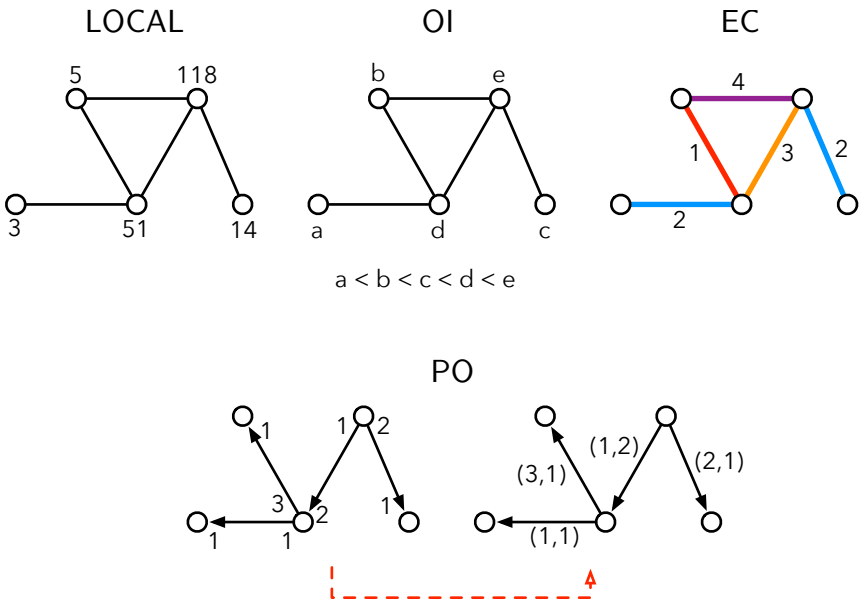

Figure 1.2. The models of distributed computing used in this work. In the PO model it is possible to represent a port-numbering as a digraph edge colouring, an vice versa.

Unique identifiers. In the standard LOCAL model, vertices are assumed to have globally unique names which are at most polynomial in the size of the graph. We will denote this identifier labelling by id. Unique identifiers can be used to construct other labellings, such as port-numberings and orientations. A model where the vertices have no labelling is known an *anonymous model*.

The fact that unique identifiers are assumed to be of polynomial size leaks information about the size of the graph to the vertices. How much this information helps is one of the main topics of this thesis.

Anonymous distributed algorithms. If the vertices do not have unique names or other vertex labels that break symmetry (such as vertex colouring), we say that the model is *anonymous*. Many basic tasks become impossible in anonymous models, such as detecting cycles or even producing different outputs [5]. The crucial observation is that an anonymous algorithm produces the same output in a graph and in its covering graph [5, 60]. This is due to the fact that its *view*, the information it can

extract from the network, is the same in both cases. We will use this property in our proofs in Chapter 2.

Port-numbering. Usually the vertices are assumed to have an ordering on their incident edges. This ordering is defined as a labelling p that assigns distinct labels from $[\deg_G(v)]$ to the edges incident to each v. We denote the port-number of $\{v, u\}$ at v by $p(v, u)$.

Model PO. We may consider a model where the edges have an orientation on them. Note that this orientation simply provides symmetry breaking information and does not affect communication. We will denote the anonymous model with a port numbering and an orientation on the edges by PO. This model is especially useful for proving lower bounds [33].

We will find it useful to consider a labelling that can be derived from a port-numbering and orientation. A *digraph k^2-edge colouring* is a labelling $c\colon E(G) \to [k] \times [k]$ such that for any pair of incident incoming or outgoing edges e, e', we have $c(e) \neq c(e')$. We represent the colours as pairs for technical reasons. First, a port-numbering and orientation can be used to derive a digraph Δ^2-colouring: each edge $e = (u, v)$ is labelled with $c(e) = (p(u, v), p(v, u))$. Second, a digraph Δ^2-colouring can be used to derive a port-numbering: each vertex first considers the outgoing edges in the order of their labels, and assigns them port numbers, and repeats this for outgoing edges.

We will use the two labellings as equivalent definitions of the PO model. Note that the described conversion do not necessarily produce the same port-numbering, but this will be fine from the perspective of worst-case analysis.

Model OI. Finally, it is often useful to consider order-invariant algorithms as stepping stone to the standard LOCAL model [33, 49]. We say that an algorithm A is *order-invariant* (with respect to the assignment of unique identifiers), if the output of the algorithm only depends on the relative order of the the identifiers, not their values. Put otherwise, if one changes the identifiers in the local neighbourhood of a vertex in a way that respects the same total order, then the output of the algorithm cannot change. We denote the model where algorithms are forced to be order-invariant by OI. This model can also be defined by a total order that is given to the vertices as a labelling.

Model EC. In the *k-edge colouring model*, denoted by EC, the edges are coloured with k colours such that incident edges have different colours.

The colouring is a function $c\colon E(G) \to [k]$.

The vertices are anonymous, so breaking symmetry between vertices is impossible in EC model. On the other hand it is a relatively strong model. For example a maximal matching can be found in constant time. We will use this model primarily as a lower bound model.

Randomization. We will model randomization as another part of the input. In a randomized model, each vertex has access to a private source of randomness and receives an infinite string of independent and uniform random bits.

1.3 Coordination and symmetry breaking

Typically the complexity of an LCL problem on bounded-degree graphs falls into one of two categories: either it is a relatively local symmetry breaking problem, like $(\Delta+1)$-colouring, and the complexity is $\Theta(\log^* n)$, or the problem is inherently global, like 2-colouring, and the complexity is $\Theta(n)$. There are problems, like Δ-colouring, for which the best known algorithm is polylogarithmic in n. No matching lower bounds have existed, however. While Kuhn et al. gave a lower bound of

$$\min\left\{\log\Delta/\log\log\Delta, \sqrt{\log n/\log\log n}\right\}$$

for maximal matching [42], their lower bound is unconditional only as a function of Δ. Recently, we gave the first examples of problems that are of neither type mentioned previously: for example Δ-colouring requires time $\Omega(\log\log n)$ [15] and there is an algorithm that runs in polylogarithmic time [51]. This randomized lower bound was used by Chang et al. to show a deterministic lower bound of $\Omega(\log n)$ for Δ-colouring [16]. They also show that there are no LCL problems of deterministic complexity $\omega(\log^* n)$ and $o(\log n)$.

While we can say that symmetry breaking is fairly well understood, sometimes even up to constants [57], the most efficient algorithms for symmetry breaking problems problems like $(\Delta+1)$-colouring, maximal matching, or maximal independent set also feature in their running time a component dependent on Δ. This dependency on Δ is much less well understood. As an example, there is no lower bound telling that the $O(\Delta+\log^* n)$-time maximal matching algorithm of Panconesi and Rizzi [50] could not be improved to something of the form $o(\Delta)+O(\log^* n)$. The state of the art lower bound has an exponential gap: Recall that Kuhn et al.

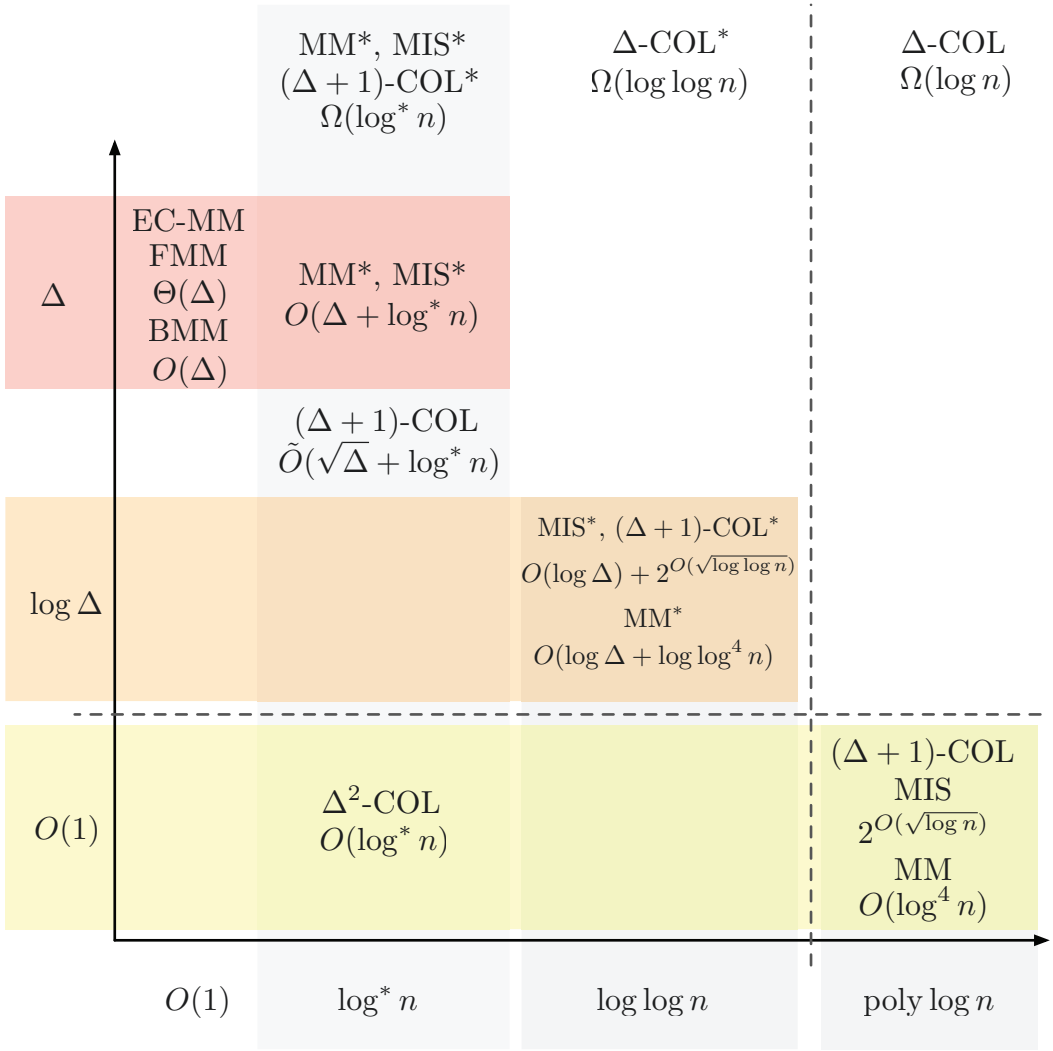

Figure 1.3. The complexity landscape of symmetry breaking and local coordination in distributed computing, specifically for LCL problems. For each problem I list the running time of the algorithm with the best known running time as a function of n, and algorithms that improve on the running time as a function of Δ. The problem names are abbreviated as follows. MIS = maximal independent set, MM = maximal matching, k-COL = k-vertex colouring, FMM = fractional maximal matching, and BMM = bipartite maximal matching. An asterisk after the problem's name indicates that the result applies to randomized algorithms. The dashed lines indicate the randomized lower bound of $\min\{\Omega(\log \Delta / \log\log \Delta)$ (horizontal), $\Omega(\sqrt{\log n / \log\log n})$ (vertical)$\}$ [42]. Other lower bounds: MM, MIS, $(\Delta+1)$-COL needs time $\Omega(\log^* n)$ [45], randomized Δ-COL needs time $\Omega(\log\log n)$ [15], and Δ-COL needs time $\Omega(\log n)$ [16]. MIS: in time $O(\Delta + \log^* n)$ [13], in time $2^{O(\sqrt{\log n})}$ [52], and in randomized time $O(\log \Delta) + 2^{O(\sqrt{\log\log n})}$ [31]. MM in time $O(\Delta + \log^* n)$ [50], in time $O(\log^4 n)$ [34], in randomized time $O(\log \Delta + \log\log^4 n)$ [12]. $(\Delta+1)$-colouring: in time $\tilde{O}(\sqrt{\Delta} + \log^* n)$ [11, 28], in time $2^{O(\sqrt{\log n})}$ [52]. Δ^2-COL in time $O(\log^* n)$ [45]. FMM in time $O(\Delta)$ [6]. BMM in time $O(\Delta)$ [18].

showed that many problems, among them maximal matching, require time $\Omega(\log \Delta / \log \log \Delta)$ [42].

The lower bound is provably tight for some problems, such as constant-approximation of minimum vertex cover [10], or provably almost tight for some others, such as constant-approximation of many linear packing and covering problems [41, 42]. The lower bound is specifically designed for approximation problems: a maximal matching lower bound is achieved through the fact that a maximal matching is a 2-approximation of maximum matching. One of the main themes of this work is to argue that there exists a class of problems, loosely speaking those problems for which a greedy approach works, such that the true local complexity is linear in the maximum degree of the input graph, Δ. Such problems include natural and fundamental problems in distributed computing, such as maximal matching, maximal independent set, and bipartite maximal matching. See Figure 1.3 for an illustration of the complexity landscape of LCL problems.

From a slightly different perspective, I will argue that in addition to symmetry breaking, there is another fundamental distributed computing challenge that we will call *local coordination* [59]. If, broadly speaking, symmetry breaking is about ensuring that two vertices are able to produce different outputs, then coordination is about ensuring that the solution is a local equilibrium; that there are no vertices that could still contribute more to the solution when the other vertices' outputs remain constant. From an algorithmic perspective, this is exactly what a greedy algorithm does.

There are two lines of work that give evidence to such a hypothesis. First, there are plenty of problems, for which the algorithm with the best running time as a function of n runs in time $O(\Delta)$. Some examples include maximal independent set in time $O(\Delta + \log^* n)$ [13], maximal matching in time $O(\Delta + \log^* n)$ [50], bipartite maximal matching in time $O(\Delta)$ [34], and fractional maximal matching in time $O(\Delta)$ [6]. Second, there are the lower bound results included in this work. In Publication I we gave the first linear-in-Δ lower bound for a natural graph problem, maximal matching. The catch is that this result only applies in anonymous, edge-coloured networks. We showed that in k-edge coloured networks, maximal matching requires exactly $k - 1$ rounds, which is achieved by the greedy algorithm. Later, in Publication V, we showed that fractional maximal matching requires time $\Omega(\Delta)$ in the standard LOCAL model, if the running time is independent of n. Combined with the algorithm of Åstrand and Suomela [6], this result is tight.

There are some caveats to the above hypothesis. First, if an algorithm is allowed a much larger time budget as a function of n than $O(\log^* n)$, the dependency on Δ can become sublinear. As an example, there exists a deterministic maximal matching algorithm with a time complexity of $O(\log^4 n)$ [34], that is, with no dependency on Δ. Similarly, on the randomized side, maximal independent set can be solved in time $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$ [31]. Where and how this transition occurs is a very interesting open question. Second, there is another recent line of research, where Barenboim showed that the greedy version of colouring, $(\Delta + 1)$-colouring, can be solved in time $O(\Delta^{3/4} + \log^* n)$ [11]. This was improved by Fraigniaud et al. [28], who showed that a certain class of colouring problems, including $(\Delta + 1)$-colouring, can be solved in time $\tilde{O}(\sqrt{\Delta} + \log^* n)$, where $\tilde{O}$ hides polylogarithmic factors of Δ. Barenboim also gives a separation between edge colouring and maximal matching: in the EC model $(2\Delta - 1)$-edge colouring can be solved in time $O(\Delta^{3/4} + \log^* k)$, where k is the number of edge colours [11].

It is still an open question if *local coordination* and *symmetry breaking* exist as separate basic problems in the LOCAL model. A lower bound of the form $\Omega(\Delta) + \Omega(\log^* n)$ for maximal matching or maximal independent set would be a definite answer to this question. Unfortunately, such a lower bound is beyond our current lower bound techniques. As a stepping stone to lower bounds for maximal matching, one should consider the problem of *bipartite* maximal matching: here is a problem that combines elements from the LOCAL model (2-colouring breaks symmetry) and from the anonymous models (simple, no unique identifiers with their strange effects). Proving a lower bound for this problem would allow a two-phase approach: worry about the symmetry breaking information first, and only then about the effect of uniqueness of the identifiers.

1.4 Unique identifiers and constant-time algorithms

Unique identifiers are potentially useful in several different ways. First and foremost, they locally break symmetry, allowing vertices to detect the true local structure of the input graph. The unique identifiers can be interpreted as colourings, which admit efficient bit manipulation algorithms [17, 49]. Unique identifiers, by definition, leak information about the size of the graph. This information can be used at least in the context of local decision [26]. The last aspect, uniqueness, does not seem to be as

useful, at least from the perspective of fast distributed algorithms. For example, recently Chang et al. showed that any algorithm for an LCL problem running in time $o(\log n)$ can be sped up to run in time $O(\log^* n)$. Their proof uses a colour reduction technique to produce locally distinct "identifiers" from a smaller set. A distributed algorithm cannot tell if the input graph is smaller or the identifiers have changed, and has to run in time $O(\log^* n)$.

One important feature of constant-time distributed algorithms is the apparent uselessness of unique identifiers. While one can do colour reduction from k colours to $O(\log^{(t)} k)$ colours in t communication rounds [17], this does not yield a constant-size labelling when $t = O(1)$ and one starts with unique identifiers that are polynomial in n. In their seminal work Naor and Stockmeyer showed that when one considers locally checkable labelling problems from the perspective of constant-time algorithms, then unique identifiers are only as powerful as the OI model [49]. That is, if there exists an algorithm that uses the actual values of the unique identifiers, then there also exists an algorithm that only uses their relative order. Their result also extends to randomization.

In the worst case, a total order provides very little symmetry breaking information. As an example, consider an n-cycle where the order is increasing clockwise. Since initially there is only one place on the cycle where the order is not increasing, colouring the cycle with *any* number of colours requires $\Theta(n)$ time in the OI model.

More recently, we showed that for a large class of natural optimization problems such as independent sets, matchings, vertex covers, and dominating sets, the worst-case approximation ratios of deterministic constant-time algorithms using unique identifiers are equal to what anonymous algorithms that only use a port-numbering and an orientation on the edges as input (the PO model) can achieve [33]. This is a completely anonymous model, and many symmetry breaking tasks cannot be solved *at all*. As with ordering, consider again an n-cycle. The edges can be oriented in a consistent way, and the port numbers arranged similarly. Every vertex has the same view up to any distance, and is therefore unable to break symmetry.

The two results about the uselessness of unique identifiers share a common feature: they require that the problem is such that the size of the local output is bounded by a constant. In principle this restriction could be a proof artefact from an argument using Ramsey's theorem [56], shared

by the proofs of both results. In Publication IV we show, however, that this restriction is actually necessary: there are natural optimization problems with unbounded outputs, such as *fractional graph colouring* and *fractional domatic partition*, where constant-time algorithms using unique identifiers can achieve non-trivial approximation ratios, but anonymous algorithms cannot. The result is abusing the unboundedness of the outputs: it produces a schedule of activity as an output, where lengths of the single intervals depend inversely on the size of the identifiers space (and thus the size of the input graph). A similar result is implied by the prior *multicoloring* algorithm of Kuhn [40].

There is nothing special about unique identifiers from the perspective of this result. Indeed, any colouring is sufficient. Even more, a good colouring produces a much better solution from the perspective of output schedule's granularity.

1.5 Local decision

Analogous to sequential computing, it is natural to study decision problems in the distributed setting. In distributed decision, as defined by Fraigniaud et al. [23], the task is to decide, in the standard LOCAL model, whether a given graph G, possibly with some input labelling x, satisfies a given graph property P. The twist is that, unlike in sequential computing, each vertex is only allowed to see its own local neighourhood in the graph and must produce its own local binary output, *yes* or *no*. The natural decision procedure now becomes asymmetric: we say that the system *accepts* the input if every vertex accepts, and *rejects* otherwise.

As an example, let P be the property of x being a 3-colouring of G. This can be decided with a 1-hop algorithm: every vertex looks at the inputs of its neighbours and checks that they differ from its own. If they do, the vertex accepts and if not, it rejects. This same procedure works, naturally, for all locally checkable labelling problems.

More generally we say that $\mathsf{LD}(t)$ consists of graph properties P, that is, subsets of labelled graphs (G, x), such that there exists a t-time distributed algorithm A for which

$$(G, x) \in P \iff \forall v \in V(G), A(v) \text{ accepts.} \tag{1.1}$$

We will be especially interested in the class of properties which can be decided in constant time. From now on, this class $\mathsf{LD}(O(1))$ will be denoted

simply by LD.

Remark 1. Note that there is the technical issue of quantifying unique identifiers in (1.1). We will assume that all properties are independent of the identifier assignment id on G. In addition, in the above definition, the algorithm must accept *yes*-instances on all identifier assignments.

Locally checkable labellings are one example of locally decidable properties [49]. Fraigniaud et al. defined the class LD [23, 25, 27], and also studied the effects of non-determinism and randomization on local decision.

Korman and Kutten studied the verification of minimum spanning trees and general non-deterministic certificates [37, 38]. Göös and Suomela studied a slightly stronger model of non-determinism [32]. Recently Förster et al. studied non-deterministic local decidability of s-t reachability, acyclicity, and other similar properties under a weaker model [22].

Work related to local decision has also been done in the context of self-stabilizing algorithms. The protocol of Afek et al. verifying a spanning tree [2] is especially important in our work. Further connections between self-stabilization and local checkability have been studied by, for example, Awerbuch et al. [8]. Finally, the distributed verification of global structures, such as minimum spanning trees has been studied [19], and it has connections to non-deterministic proof complexity [37, 39].

1.5.1 Power of unique identifiers

In this work we look at two aspects of local decision: the power of unique identifiers and the power of nondeterminism. On a high level, unique identifiers, like nondeterminism, can be treated as an oracle: by definition the identifiers leak information about the size of the graph and, as we will see, this allows the vertices to collectively guess the size of the input graph (or at least an upper bound for it).

In Publication II we study the power of unique identifiers in the context of local decision. In particular, we characterise the minimal information that is sufficient to replace unique identifiers.

This line of research began when Fraigniaud et al. showed that for a large class of problems, the actual values of unique identifiers do not help in local decision [24]; for these problems, an algorithm can be modified so that after (G, x) is fixed, the output of the algorithm at each vertex is independent of the assignment of unique identifiers. Such an algorithm is called *identifier oblivious*. We will denote the class of properties that

can be decided using constant-time identifier-oblivious algorithms by LDO. Fraignaud et al. also conjectured that LD $=$ LDO [24]. This conjecture turned out to be false: there exist properties in LD $\setminus$ LDO [26].

The separation of LD and LDO is fundamentally a question of computability. To gain intuition on this, consider the following promise-version P of a property separating the two classes [26]. Let (G, M) be an n-cycle where every vertex is labelled with the same Turing machine M. We promise that if M halts on an empty tape, it does so in $s \leq n$ steps. Now (G, M) is a *yes*-instance if and only if M does not halt. If algorithms are allowed to behave arbitrarily when the promise is violated, then it holds that $P \in$ LD: every vertex v simulates M for $\mathrm{id}(v)$ steps and outputs *no* if M halts, and *yes* if M does not halt. On the other hand we have that $P \notin$ LDO: an identifier-oblivious algorithm has to solve the halting problem since it has no information about the running time of M. In Section 4.1 we sketch how the promise in the above problem can be removed.

A separation argument based on computability is also necessary: if we assume that the local algorithm is allowed to be uncomputable, identifier-oblivious algorithms can easily simulate unique identifiers. Given a normal local algorithm A, such an identifier-oblivious algorithm consists of testing whether A accepts with *every* identifier assignment to the vertex's local neighbourhood, and accepting if and only if A does.

The separation is due to the fact that unique identifiers *leak information about the size of the graph*. The crucial property in the above example is that since unique identifiers are indeed unique, there is a vertex v with $\mathrm{id}(v) \geq n \geq s$, the halting time bound on M. Note that this is a different property from that which is useful in the context of symmetry breaking: there the running times are typically dependent on an *upper bound* on the size of the identifiers, and the uniqueness property does not play a role. Indeed, often a proper colouring with a small number of colours allows for even faster distributed algorithms, such as in the case of most symmetry breaking problems.

In Publication II we characterise the minimal labellings that give an identifier-oblivious algorithm the power of unique identifiers. These labellings must have the following property: each subset S of vertices receives at least one label with size $|S|$. This clearly allows an identifier-oblivious algorithm to decide the property P related to the halting of Turing machines: since there is one label of size at least n, every vertex simulates the Turing machine M for a number of steps equal to its label.

Now let us define things slightly more formally. We say that a *scalar oracle* is a function f that maps each positive integer n to a multiset of n integers $(f_1, f_2, \ldots, f_n)$. Now, given a graph G, an oracle f produces a set of labels and an *adversary* assigns the labels to the graph. Each vertex then sees, in addition to the input and the unique identifiers, the assignment of f in its own local neighbourhood. For a given oracle f, we say that LDO^f is the class of properties decidable using identifier-oblivious local algorithms. The class LD^f is defined similarly. Scalar oracles are a direct generalization of unique identifiers: indeed, unique identifiers can be defined a class of scalar oracles F such that if $f \in F$, then there exists a constant c such that $f(n)$ consists of n distinct elements from $[n^c]$ for all n.

Denote by $f_k^{(n)}$ the kth largest label produced by f on input n. It turns out that the following property is crucial. We say that an oracle is *large*, if

$$\forall c > 0, \exists k \geq 1, \forall n \geq k, f_k^{(n)} \geq c. \tag{1.2}$$

If f is not large, it is *small*. The above property says, roughly, that for any constant bound c, there is a bound k such that every k-subset of labels has a label of size at least c.

Our main result is to show the following theorem.

Theorem 2. $\mathsf{LDO}^f = \mathsf{LD}^f$ *if and only if* f *is large.*

In addition, we show that if f is small, there exists a property $P \in \mathsf{LD} \setminus \mathsf{LDO}^f$. Since by definition identifier-oblivious algorithms see the local structure of the graph, the two results combine to show that our definition of a large oracle gives the *minimal* sufficient property for a labelling to be as strong as unique identifiers. For *any* large oracle, we have the following relations between the four classes:

$$\mathsf{LDO} \subsetneq \mathsf{LD} \subseteq \mathsf{LDO}^f = \mathsf{LD}^f, \tag{1.3}$$

where the inclusion of LD in LDO^f may be strict, depending on f. The structure of these classes with regard to various oracles is covered in Section 4.1.

1.5.2 Non-determinism in local decision

It is natural to ask how non-determinism affects local decision. From a more practicable point of view this can be seen as the question of locally *verifying* a given answer, even if computing it requires global information. Several models of non-deterministic distributed decision have been proposed, including at least *proof labelling schemes* [36–38], *non-deterministic*

local decision (NLD for short) [23, 27], and *locally checkable proofs* [32]. In each of the models, the vertices are allowed to receive non-deterministic advice as input when making their decisions. On the other hand, there are technical differences regarding the use and assignment of unique identifiers, and the size of the labels.

As an example, the class $\mathsf{LCP}(t)$ is defined as the graph properties such that if and only if $(G, x) \in P$, there exists a t-bit *certificate* or *proof* $y\colon V(G) \to \{0,1\}^t$ and a local algorithm A such that each vertex accepts the proof with the input x. In symbols we say that

$$(G, x) \in P \iff \exists y : \forall v \in V(G), A(G, x, v, y) \text{ accepts.}$$

Locally checkable proofs are a strictly stronger version of the proof labelling schemes, as the running time of the local algorithm is not as restricted and the non-deterministic labels are allowed to refer to the identifiers of the vertices. In NLD the sizes of the certificates are not restricted, but the certificate has to work for *every* identifier assignment.

Compared to LD, many NP-complete decision problems, such as 3-colouring or the graph encoding of SAT are in $\mathsf{LCP}(O(1))$. For example, in the former case $y(v)$ simply encodes the colour of vertex v, and each vertex accepts if $y(v) \neq y(u)$ for each neighbour u of v. Crucially, a spanning tree can be certified using $O(\log n)$ bits per vertex [2, 32]. This can be used as a building block to distribute information globally: a spanning tree can be used to gather the number of nodes, to elect a leader, or to reverse decisions [32]. This last property is interesting because of the asymmetric nature of local decision.

Göös and Suomela also show that $\Omega(\log n)$ proof bits are necessary for certifying a spanning tree [32]. This makes LCP with $O(\log n)$-bit certificates, or $\mathsf{LCP}(\log n)$, a natural and robust class of properties.

1.5.3 A hierarchy of local decision

In Publication III we study the generalization of $\mathsf{LCP}(\log n)$ analogous to the polynomial hierarchy in the centralized computing. The first level of the hierarchy, $\mathsf{LCP}(\log n)$, is the class of problems such that the correctness of a solution is easy to verify, similar to the class NP. Now additional nondeterminism can be seen as a game between two players, similar to the polynomial hierarchy.

We will call the two players *prover* and *disprover*. The players take turns, assigning $O(\log n)$-bit labels $\ell_1, \ell_2, \ldots, \ell_k$ to the vertices of the input

graph. Prover wins, if it can make the local algorithm accept everywhere, and disprover wins otherwise. Roughly speaking property P is at the kth level of the hierarchy if the prover has a winning strategy if and only if $(G, x) \in P$.

More formally we can define classes Σ_k and Π_k via alternating quantification of the proof labels. Define Σ_k as the class of properties P such that there exists a local algorithm A for which we have that

$$(G, x) \in P \iff \exists \ell_1, \forall \ell_2, \ldots, \mathrm{Q}\, \ell_k : \forall v \in V(G), A(G, v, x, \ell_1, \ell_2, \ldots, \ell_k) = 1,$$

where Q denotes the existential quantifier, if k is odd and the universal quantifier otherwise. Similarly Π_k is defined as the class of properties P such that there exists a local algorithm A for which we have that

$$(G, x) \in P \iff \forall \ell_1, \exists \ell_2, \ldots, \mathrm{Q}\, \ell_k : \forall v \in V(G), A(G, v, x, \ell_1, \ell_2, \ldots, \ell_k) = 1,$$

where Q denotes the existential quantifier, if k is odd and the universal quantifier otherwise.

It turns out that the classes where the quantification ends in a universal quantifier collapse to the previous level. Therefore we can define a *local hierarchy* $\mathsf{LH} = \cup_k \Lambda_k$ as follows. When $k = 0$, we have $\Lambda_0 = \mathsf{LD} = \Sigma_0 = \Pi_0$. For $k > 0$, define

$$\Lambda_k = \begin{cases} \Sigma_k \text{ if } k \text{ is odd, and} \\ \Pi_k \text{ if } k \text{ is even.} \end{cases}$$

In Publication III we show structural results about this hierarchy. Trivially we have that $\Lambda_k \subseteq \Lambda_{k+1}$ for all k. The *complement* $\bar{P}$ of property P is the set $\{(G, x) : (G, x) \notin P\}$. Due to the asymmetric nature of the decision procedure, define the complement of Λ_k as

$$\mathrm{co}{-}\Lambda_k = \{P : \bar{P} \in \Lambda_k\}.$$

Since decision cannot be simply reversed by the local algorithm, we do not necessarily have that $\mathrm{co}{-}\Lambda_k \subseteq \Lambda_k$. We do show, however, that $\mathrm{co}{-}\Lambda_k \subseteq \Lambda_{k+1}$. In a symmetric fashion we also have that $\mathrm{co}{-}\Lambda_k \subseteq \mathrm{co}{-}\Lambda_{k+1}$ for all k, and that $\Lambda_k \subseteq \mathrm{co}{-}\Lambda_{k+1}$. Properties in the intersection $\Lambda_k \cap \mathrm{co}{-}\Lambda_k$ can be decided in a symmetric fashion: the prover can force all vertices to accept *yes*-instances and the disprover can force all vertices to reject *no*-instances.

We also give a conditional collapse result: if there are k and $k + 1$ such that $\Lambda_k = \Lambda_{k+1}$, then the whole hierarchy collapses to its kth level. This proof uses a straightforward simulation argument.

Finally, we show that there are languages outside the local hierarchy. This is done using a counting argument on the languages on 0/1-labelled paths, similar to many other existence results in various models of computational complexity.

On the positive side we show, for example, that many NP-hard optimization problems such as maximum independent set or minimum dominating set fall into $\mathrm{co}{-}\Lambda_1$ (and therefore the second level Λ_2). More generally, the graph version of satisfiability for quantified Boolean formulas with k alternations of quantifiers (a complete problem for the kth level of polynomial hierarchy) is in Λ_k.

The first two levels of the hierarchy have an arguably reasonable interpretation as distributed protocols. In addition, the two player protocol can have an exponentially smaller requirement on the size of the certificates: for example non-3-colourability requires an $\Omega(n^2/\log n)$ size proof in LCP, but is clearly in Λ_2.

In comparison, Balliu et al. consider the generalization of NLD into a corresponding local hierarchy [9]. This model differs in two crucial aspects: one labelling has to work for all identifier assignments, and the size of the labelling is unbounded. The hierarchy, defined this way, collapses to its second level.

1.6 Lower bound techniques

One aim of this work is to present new lower bounds and lower bound techniques for distributed computing. Existing lower bounds are often shown via one of two basic techniques: *indistinguishability* and *simulation*. In what follows I give a brief overview of the these two techniques in local distributed computing. This overview will not cover lower bound techniques such as communication complexity arguments for the CONGEST model, which are not applicable in the LOCAL model.

1.6.1 Indistinguishability

The indistinguishability principle says that if two vertices see the same information, they must produce the same output. This produces a lower bound if the vertices are located in two different problem instances where they need to produce different outputs. This approach has been very powerful in the context of message passing models [3, 42]. For an example

of a typical lower bound of this type in the LOCAL model, consider the problem of 2-colouring. Consider two paths P_n and P_{n+1} on n and $n+1$ vertices, respectively. In order to decide whether the endpoints should have different colours or not, these vertices need to see $\Omega(n)$ steps away.

Typically the indistinguishability argument is well suited to proving approximation lower bounds, as in the case of the lower bound construction of Kuhn et al. [42]. In its most simplified form, one constructs two d-regular graphs with girth $2g+1$. Now in both graphs, for each $t \leq g$, the radius-t neighbourhood of each vertex is a d-regular tree. Now if one of the graphs has a large optimum solution and the other a small one, no t-time algorithm can distinguish the two graphs and therefore must produce (with a suitable identifier assignment) a small solution in the graph with a large optimum [3, 35]. Graphs with suitable properties can be derived for example from the theory of random graphs [4, 14, 29] or sometimes via explicit constructions [46, 47].

Recently a form of indistinguishability argument has been used to show lower bounds for optimization in CSP problems. Gamarnik and Sudan gave an upper bound on the size of independent sets that constant-time (randomized) algorithms can find in d-regular graphs [30]. Their proof is based on the *shattering* phenomenon of random CSP instances [1], where the geometry of the solution space undergoes a rapid transition when the target solution size grows large enough. In particular they show that in large solutions, the outputs of vertices must be globally correlated, or in other words, any two solutions either have a large overlap or almost no overlap. Vertices in different parts of the graph are unable to coordinate their choices. Later, Rahman and Virág showed that constant-time algorithms are exactly half-optimal in this setting [55]. Interestingly this coincides with the best known polynomial-time algorithms for this problem [43].

This work. In Publication IV our lower bound uses a simple indistinguishability argument, essentially as described in the previous section, based on the proof of Bollobas that there are graphs of high girth with no large independent sets [14] (and therefore no short fractional graph colourings).

Our EC lower bound for maximal matching in Publication I can be seen as a kind of iterated indistinguishability argument: at each step t we find two graphs with a pair of vertices such that their neighbourhoods agree up to distance t, but the vertices must produce different outputs. Therefore

the vertices must see up to distance $t + 1$.

Our separation of LDO^f and LD for small oracles f in Publication II uses an indistinguishability argument: we show that the oracle labels can be assigned in such a way that vertices with potentially useful oracle labels cannot separate between the instances where they should accept and the instances where the should reject. The actual construction is fairly involved and described in some detail in Section 4.1. We also use indistinguishability arguments (combined with uncomputability arguments) to further study the relations of the classes LDO, LDO^f, LD, and LDO^f for various oracles f.

1.6.2 Simulation

Simulation is an extremely powerful lower bound technique in distributed computing. At its heart, the classic lower bound of Linial [45] is a simulation argument [44]. On a high level, the argument goes as follows: assume that there is a t-time c-colouring algorithm A for (oriented) cycles. Now we construct a $(t-1)$-time 2^c-colouring algorithm A': the output of A' is the set of possible outputs for A given the $(t-1)$-neighbourhood of the vertex. Now we begin with the assumption of having a t-time 3-colouring algorithm. In each iteration the simulation argument produces a faster colouring algorithm with a larger colour palette. Finally, any 0-round algorithm must use at least n colours to avoid any conflicts, which gives a lower bound of $t = \Omega(\log^* n)$.

More recently we used a simulation argument to show that sinkless orientation requires $\Omega(\log \log n)$ rounds in Δ-regular graphs and trees [15]. This is another iterated simulation argument, but instead of using a larger colour palette, the success probability decreases in each step. Any 0-round algorithm must have a large error probability.

This work. Our maximal fractional matching lower bound in Publication V consists of two parts: one is the new iterated indistinguishability technique presented in Publication I, and the other is a series of simulations that lift the result from the anonymous EC model to the standard LOCAL model. While the last step $\mathsf{OI} \rightsquigarrow \mathsf{LOCAL}$ is a fairly standard Ramsey argument, the other three steps are novel and can be of independent interest. In general the approach of proving lower bounds in a weak (anonymous) model, where such proofs are relatively easy, has proven to be powerful [33, 49].

On the side of local decision, our result showing that $\mathsf{LDO}^f = \mathsf{LD}^f$ for large oracles f is a straightforward simulation argument. An identifier-

oblivious algorithm is given enough information by the oracle label f so that each vertex can simulate identifier assignments in its local neighbourhood. The property that f is large will guarantee that there is a set of such simulations, one at each vertex, which form a consistent assignment of unique identifiers to the graph. Finally, many of the structural results in Publication III are based on a simulation argument.

2. Lower bounds for coordination problems

In this chapter I present the first linear-in-Δ lower bounds for two natural graph problems, maximal matching and its linear relaxation, fractional maximal matching. These lower bounds hint that in addition to symmetry breaking, there is another basic distributed challenge, local coordination.

I will present an overview of the proofs for the two results, highlighting how the maximal matching lower bound at the heart of both results forces *any* algorithm to have greedy-like properties. Our lower bound in Publication I represents a new type of lower bound argument: we tie the running time of an algorithm to the amount of symmetries in the neighbourhood of each vertex.

In Publication V we extend this lower bound to fractional maximal matching in the standard LOCAL model. This proof has the same symmetry-based argument at its heart, but uses simulation arguments, some of which are novel, to lift the original lower bound to the stronger model. This combined approach allows us to tackle different challenges to lower bounds separately.

2.1 New lower bounds for coordination problems

In Publication I we gave the first linear-in-Δ lower bound for a natural graph problem, maximal matching. This proof comes with a large caveat, though: it is not for the standard LOCAL model, but for the relatively strong anonymous model known as the *edge-colouring model*, or EC for short. In this model, vertices do not have any identifiers (and are thus unable to break symmetry), but the edges are coloured with $\Delta + 1$ colours. We show the following theorem.

Theorem 3. *Let $k \geq 3$ be an integer. Finding a maximal matching in anonymous k-edge coloured graphs requires $k - 1$ rounds.*

This result is also tight: the greedy algorithm actually runs in $k - 1$ rounds.

We will see the proof of Theorem 3 in Section 2.2. On a high level, it works by iteratively constructing, for any given algorithm, pairs of graphs such that

(1) the output of the algorithm differs on the two graphs, but
(2) to distinguish the two graphs the algorithm must look further and further away.

The difficult inputs will be ones that contain lots of symmetries. In particular, *every* vertex in these graphs will have some number neighbours with isomorphic infinite-radius neighbourhoods, that is, there is an automorphism of the graph that switches the place of the two nodes. These symmetries are a crucial resource in the proof: loosely speaking, in each step, the algorithm can force the number of symmetries to decrease by one at *some* vertex. The induction stops once some vertex runs out of all of its symmetries. As a byproduct we will see, as we discuss in Section 2.2.3, that *any* algorithm must behave in a somewhat greedy fashion.

There are two main weaknesses to the main result of Publication I. First, the EC model is a "toy" model and we would really like to prove linear-in-Δ lower bounds in the standard LOCAL model. Second, the model is incomparable with the LOCAL model, since symmetry breaking is both impossible and unnecessary for solving maximal matching. Therefore the result says nothing of what happens when the running time can be a combination of a function of Δ and a function of n.

We address the first weakness in Publication V, where we use a similar construction to prove a lower bound for fractional maximal matchings. This lower bound is then lifted through a series of simulation results all the way to the standard LOCAL model. Formally we have the following theorem.

Theorem 4. *Any algorithm for the LOCAL model that finds a fractional maximal matching in time independent of n must have running time $\Omega(\Delta)$.*

This is the first linear-in-Δ lower bound for any natural graph problem in the LOCAL model. This result, again, comes with a caveat: we do not rule out an algorithm for fractional maximal matching with a running time of $o(\Delta) + O(\log^* n)$. This is due to the same reason as to why our proof does not yield a lower bound for maximal matching: current lower bound techniques are unable to both handle unique identifiers and to prove

lower bounds of order $\omega(\log^* n)$. This barrier for showing lower bounds for maximal matching with both the coordination and the symmetry breaking elements is discussed in Section 2.4.

2.2 Linear-in-Δ lower bound for maximal matching

In this section we present a slightly more concise version of the proof of the main theorem in Publication I. We completely omit the group theoretic notation of the proof in Publication I and use notation similar to the proof in Publication V.

The general picture is that given an algorithm (that can be non-uniform over d, the graph degree), we inductively construct a series of graphs such that in the ith graph the algorithm has to look at least i hops away (at some vertex).

In the following sections I present the necessary notations specific to the proof of Theorem 3.

2.2.1 Edge-colouring model EC

In what follows, we will use the non-standard EC model. Recall that in this model, the vertices of the input graph $G = (V, E)$ do not have any labels, they are completely anonymous. We will focus on the special case of d-regular graphs with a $(d+1)$-edge colouring. Each vertex v has exactly one *missing colour*, that is, a colour with no edge of that colour incident to it, denoted by $f(v)$.

Multigraphs. We will consider, in addition to simple graphs, also multigraphs as representations of simple graphs with symmetries. In particular, we will allow graphs with loops, that is, edges from a vertex to itself. To this end, we will allow edges of the form $\{v, v\}$, and consider E to be a multiset. The graphs considered here will not have multiple edges of the form $\{v, u\}$ with $v \neq u$. For the rest of this chapter, we will call multigraphs simply graphs.

An edge is *simple*, if it is not a loop. The *simple degree* of a vertex v is the number of simple edges incident to v. In the following proof, the simple degree is a measure of asymmetries in the neighbourhood of a vertex.

Covering graphs and lifts. A graph G with an edge colouring c is the *lift* of another graph H with edge colouring c' if and only if there exists an onto graph homomorphism from G to H that preserves the edge colours.

That is, a mapping $\phi\colon V(G) \to V(H)$ is a graph homomorphism (with some abuse of notation due to loops), if and only if

(1) for every $\{v,u\} \in E(G)$ with $c(\{v,u\}) = \chi$ there exists $\{\phi(v),\phi(u)\} \in E(H)$ with $c'(\{\phi(v),\phi(u)\}) = \chi$, and
(2) ϕ preserves vertex degrees.

A crucial property of lifts is that for all vertices v, u such that $\phi(v) = \phi(u)$, we have that the neighbourhoods of v and u are isomorphic up to any distance. This property will be at the heart of our proof.

Universal covers. For a given graph G, the *universal cover* U_G of G is defined as follows. Fix an arbitrary vertex $v \in V(G)$ (each choice produces and isomorphic graph). The set $V(U_G)$ consists of all non-backtracking walks in G, and two vertices $u, w \in V(U_G)$ are connected by an edge of colour χ if u as a walk follows w to its endpoint and then takes one more step along an edge of colour χ. Note that U_G is always a tree and that it can be infinite. This is not a problem, as we can consider the execution of constant-time algorithms also on infinite graphs by cutting a finite piece of them.

Factor graphs. Conversely to the universal covers, the factor graph F_G of graph G is the smallest graph H such that G is a lift of H. The factor graph is a representation of the symmetries in the original graph: each vertex $v \in V(F_G)$ represents an equivalence class of vertices that look the same up to any distance and that must all have the same output in the original graph G. We say that vertices u and v in G are *identical*, if they are mapped to the same vertex on F_G.

Executing distributed algorithms on multigraphs. In the proof that follows, we will consider the virtual execution of distributed algorithms on multigraphs. In practice, we define this to be the following process.

(1) Given the input graph G, take the universal cover U_G of G.
(2) Run algorithm A on U_G.
(3) For each $v \in V(G)$, we define that $A(v, G) = A(\phi^{-1}(v), U_G)$, that is, the output of v is equal to the output of the vertices that map to it on its universal cover.

Loopy graphs. We say that a vertex v in graph G is *k-loopy*, if $\phi(v) \in F_G$ has at least k loops. Graph G is said to be k-loopy, if *every* vertex in F_G has k loops. Loopiness has the following consequence, which is crucial to our argument.

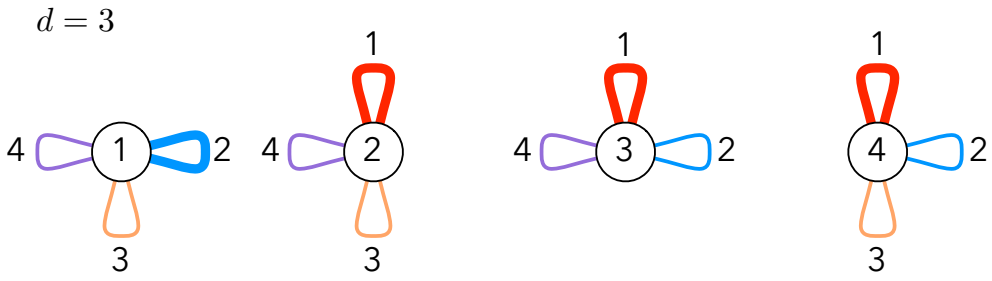

Figure 2.1. The multigraphs $G(c)$ for the base case of the induction when $d = 3$. Each graph is a single-vertex multigraph with the vertices labelled with their respective missing colours. A possible output, which corresponds to the output of the greedy algorithm, in bold.

Fact 5. Any anonymous distributed maximal matching algorithm must produce a perfect matching on loopy (edge-coloured) graphs.

To see this, consider a loopy vertex that is unmatched: by definition, it has at least one neighbour which is identical to it, and is therefore also unmatched, a contradiction with the maximality of the solution.

Now we are ready to proceed to the lower bound construction.

2.2.2 The lower bound construction

In this section we give a concise proof of Theorem 3.

Proof of Theorem 3. For any constant d, any distributed algorithm A_d, and any $t \in [d]$ we will construct a pair of d-regular graphs (G_t, H_t) such that the existence of G_t and H_t forces A_d to have running time at least $t + 1$.

More concretely, there are vertices $g_t \in V(G_t)$ and $h_t \in V(H_t)$ such that the t-neighbourhood of g_t is isomorphic to the t-neighbourhood of h_t, but their outputs differ. The proof is by induction and in order to sustain the induction argument, we will in addition require that the outputs differ on a loop edge, that G_t and H_t are $(d - t)$-loopy, and that the structure of both G_t and H_t, ignoring loop edges, is a tree. We call such a pair of graphs *t-critical* for A_d.

Base case $t = 0$. We begin by showing that for any $d \geq 3$, and any algorithm $A = A_d$, there exists a pair (G, H) of 0-critical graphs. We consider the $d + 1$ different single-vertex multigraphs, denoted by $G(c)$ for each choice of the missing colour c. See Figure 2.1 for illustration. Any algorithm must output a single colour not equal to c in each $G(c)$. It is easy to see that there must exist a colour χ and graphs $G(c)$ and $G(c')$, with $c, c' \neq \chi$, such that A outputs χ on $G(c)$, but does not output χ on $G(c')$.

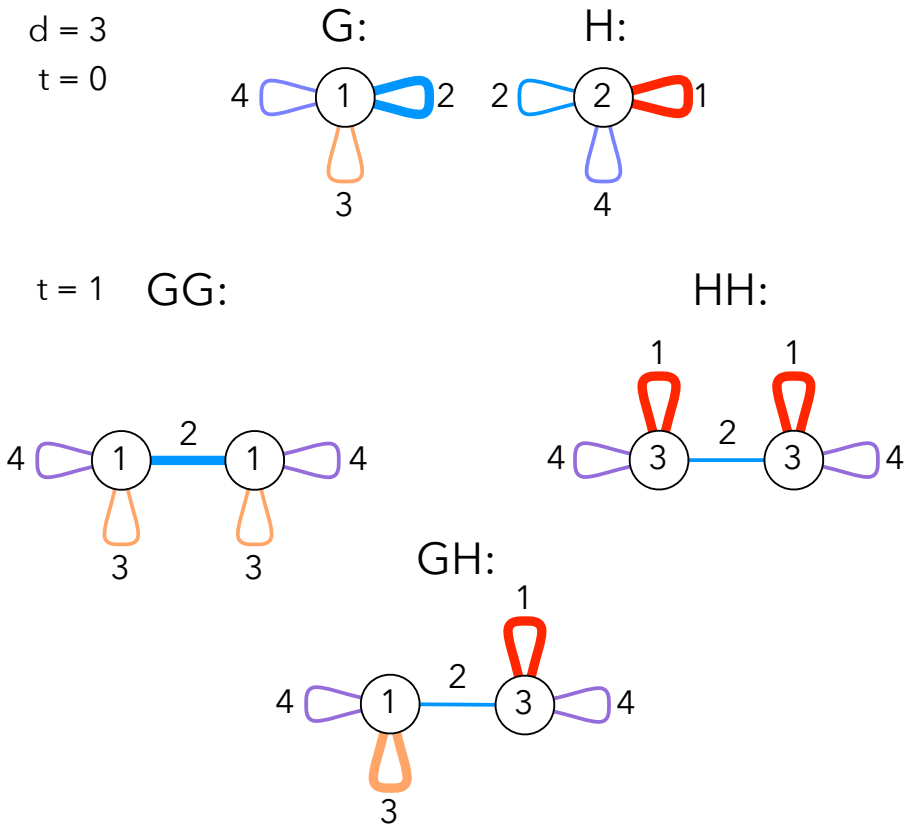

Figure 2.2. The mixing step for $d = 3$ and $t = 0$. We have two graphs, G and H that are a 0-critical pair, that is, their vertices have isomorphic radius-0 neighbourhoods (by definition) and different outputs. We construct the three graphs GG, HH, and GH. Depending on A, either GG and GH or GH and HH will become a 1-critical pair. In the example illustrated, the output of the left vertex differs between GG and GH, and these are the 1-critical pair.

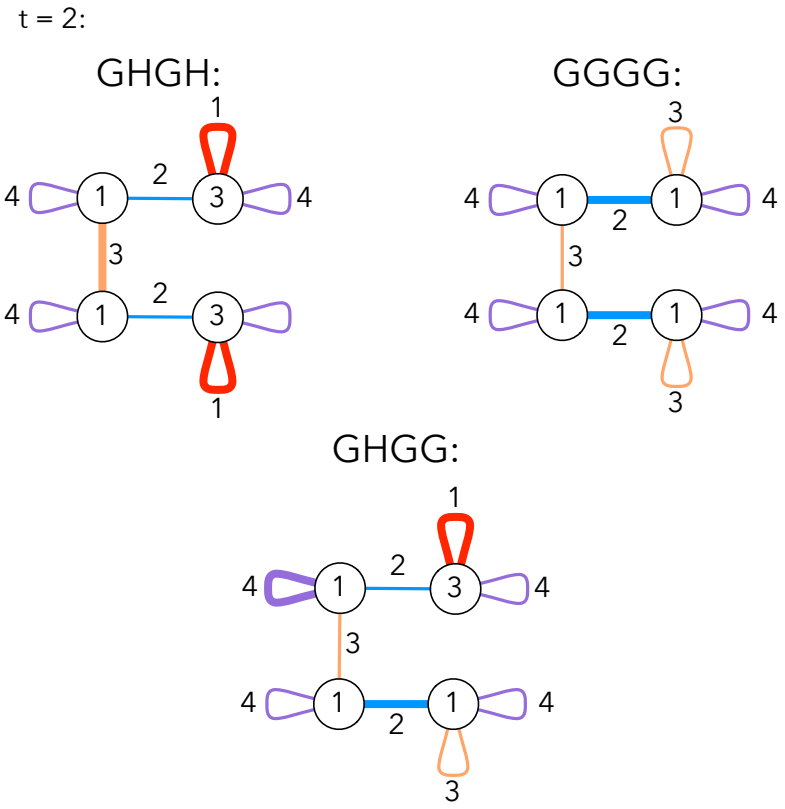

Figure 2.3. The mixing step for $d = 3$ and $t = 1$. The critical vertices are connected by a new edge. The greedy algorithm outputs a new loop colour.

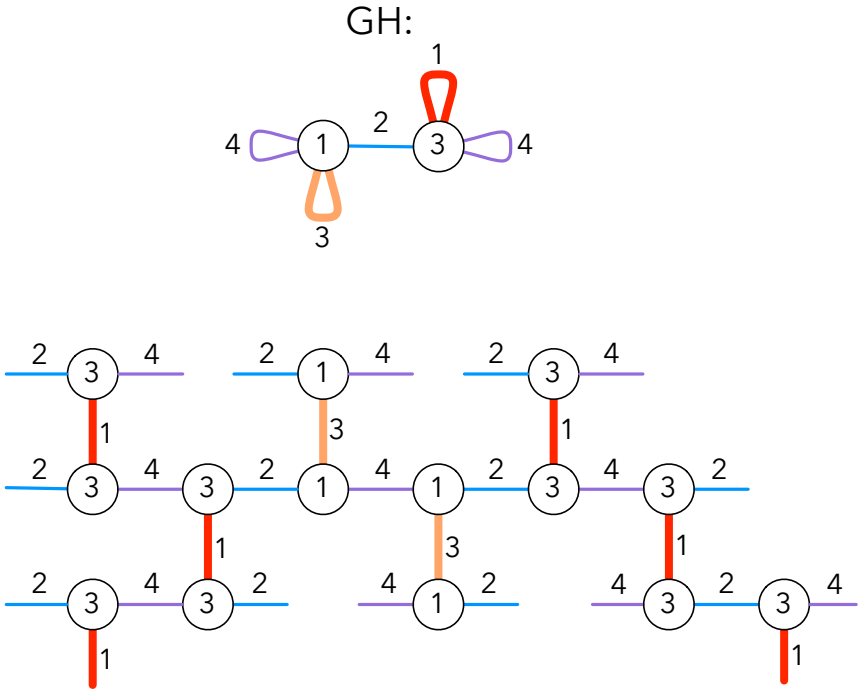

Figure 2.4. Graph GH from Figure 2.2 and a part of its unfolded universal cover U_G.

Now let G and H be a 0-critical pair. We construct a new graph GH by taking a copy of G and a copy H, with $V(G) = \{g\}$ and $V(H) = \{h\}$, remove the loop of colour χ from both G and H, and add a new edge $\{g, h\}$ of colour χ. Now the following is true.

1. We have for the neighbourhoods of g and h that $B_G(g, 0) \simeq B_{GH}(g, 0)$ and that $B_H(h, 0) \simeq B_{GH}(h, 0)$,
2. There exists a colour $\chi' \neq \chi$ such that the output of A on differs between two loop edges of colour χ' in at least one of the following cases.
   a. There is a loop of colour χ' incident to g in G and the output of g with respect to the loop of colour χ' differs between G and GH.
   b. There is a loop of colour χ' incident to h in H and the output of h with respect to the loop of colour χ' differs between H and GH.

See Figure 2.2 for illustration.

Mixing. Now assume that a t-critical pair of graphs $(G, H) = (G_t, H_t)$ exists for some $t < d$. In particular, there are vertices $g \in V(G)$ and $h \in V(H)$ such that $B_G(g, t-1) \simeq B_H(h, t-1)$, but $A(g, G) = c \neq A(h, H)$ for some loop edge c. We construct three new graphs GG, GH, and HH by gluing together copies of G and H. For example, to construct GH we

Step 1: take a copy of G and a copy of H and set $V(GH) = V(G) \cup V(H)$ and $E(GH) = E(G) \cup E(H)$,

Step 2: remove the loop edges of colour c incident to nodes $g \in V(G)$ and $h \in V(H)$, and finally

Step 3: add an edge $\{g, h\}$ of colour c to $E(GH)$.

See Figure 2.2 for illustration.

Propagation. Recall that we want to find a node such that it changes output on a loop edge. We claim that whatever A does, the change in the output of $\{g, h\}$ must propagate to some loop edge in GH. Now let us consider the output of A in GH. Since the outputs of $A(g, G)$ and $A(h, H)$ are incompatible in GH, at least one of them must change this output. The analysis splits into two cases.

Case 1: Vertices g and h output $A(g, GH) = A(h, GH) = c$. We have that $A(h, GH) \neq A(h, H) = A(h, HH) = \chi$, vertex h changes output.

Case 2: Vertices g and h output $A(g, GH), A(h, GH) \neq c$, that is, g changes output by $A(g, G) = A(g, GG) \neq A(g, GH) = \chi'$.

First observe that for all vertices $v \in V(G)$ and $u \in V(H)$ we have that $B_{GG}(v,t) \simeq B_{GH}(v,t+1)$ and $B_{HH}(u,t) \simeq B_{GH}(u,t+1)$. That is, if vertices change outputs between the two constructions, they must see *at least* $t+1$ steps away to differentiate between the two constructions. In particular, since the simple edges of GG, GH, and HH induce a tree, the distance to which a vertex v has to see to distinguish the two constructions is

$$t + \min\{\mathrm{dist}_{GH}(v,h), \mathrm{dist}_{GH}(v,g)\} + 1.$$

Finally, we will show that the change in outputs must propagate to some loop in GH. For simplicity, we will assume the first case where h changes output. We assume that the edge of colour χ incident to h is a simple edge $\{h, s_1\}$, as otherwise we have the claim. By the loopiness property s_1 must be matched, but to avoid changes in loops it must be matched via a simple edge to a vertex s_2 that is not matched via a loop in HH. Since the simple edges of GH form a tree, s_2 is at distance 2 from h. Again, since s_2 is matched via a simple edge to s_3 in HH, the vertex s_3 must be matched to some other vertex, as was the case for h. This propagation continues until there is some vertex s_ℓ such that s_ℓ changes the output of one of its loops. Now $s_\ell \in V(GH)$ and its corresponding copy $s'_\ell \in V(HH)$ have isomorphic local neighbourhoods up to distance $t+\ell+1$ and GH and HH form a $(t+\ell+1)$-critical pair. □

2.2.3 Coordination and greedy algorithms

The above proof shows that in $(\Delta+1)$-edge coloured graphs, the greedy algorithm is optimal. Naturally, for a larger number k of edge colours one can do colour reduction in time $O(\log^* k)$, but this does not allow circumventing the linear-in-Δ barrier.

Moreover, the proof shows that *any* optimal algorithm must have "greedy" properties. First, note that we can iterate the lower bound until at least one of the vertices in both constructions has simple degree d. In particular, only if the change propagates to a vertex with no loops, we do not get a new critical pair. This means that any optimal algorithm must always propagate the difference of the outputs of the two critical vertices into one of the loops of one of those vertices to force a vertex of simple degree d after d iterations of the proof. In addition, if an algorithm behaves optimally, as described above, then there exists a missing colour c such that any vertex with that missing colour could, a priori, output any of its incident edge colours. It should be noted that existing algorithms for maximal matching

tend to have similar, greedy features [18, 50]. Whether such an approach is also necessary, is an interesting open question.

2.3 Linear-in-Δ lower bound for maximal fractional matching

In this section we discuss how the proof technique from the previous setting can be used as a building block to show Theorem 4. The proof begins by showing an $\Omega(\Delta)$ lower bound for *fractional maximal matching* in the EC model using similar techniques as seen in lower bound proof for maximal matching. Then we lift this lower bound all the way to the LOCAL model via a series of simulation results

$$\mathsf{EC} \rightsquigarrow \mathsf{PO} \rightsquigarrow \mathsf{OI} \rightsquigarrow \mathsf{LOCAL},$$

The first simulation $\mathsf{EC} \rightsquigarrow \mathsf{PO}$ is a novel but fairly simple one, incurring an overhead of $O(\Delta)$ rounds. In the second simulation $\mathsf{PO} \rightsquigarrow \mathsf{OI}$ we show that port-numbering and orientation can be used to construct a locally consistent ordering on the vertices. This ordering can then be used to simulate any order-invariant algorithm. Finally, the simulation $\mathsf{OI} \rightsquigarrow \mathsf{LOCAL}$ uses fairly standard Ramsey techniques [18, 49] to control the unique identifiers.

Fractional maximal matching. A *fractional matching* $y\colon E \to [0,1]$ is an assignment of weights to the edges in such a way that the total weight of the edges incident to each vertex

$$y[v] = \sum_{e \ni v} y(e)$$

is at most one. A vertex v is *saturated*, if it holds that $y[v] = 1$. A fractional matching is *maximal*, if for every edge, at least one of its endpoints is saturated.

2.3.1 Lower bound for EC

The EC lower bound for fractional maximal matching is very close to the corresponding lower bound for maximal matching. One clear difference is that in d-regular graph solving fractional maximal matching is trivial: each edge outputs $1/d$. On the other hand, we still have the invariant that in a loopy graph every vertex has to be saturated.

The proof uses slightly different seed graphs for $t = 0$. The problem of the lower bound construction admitting a trivial solution can be fixed using

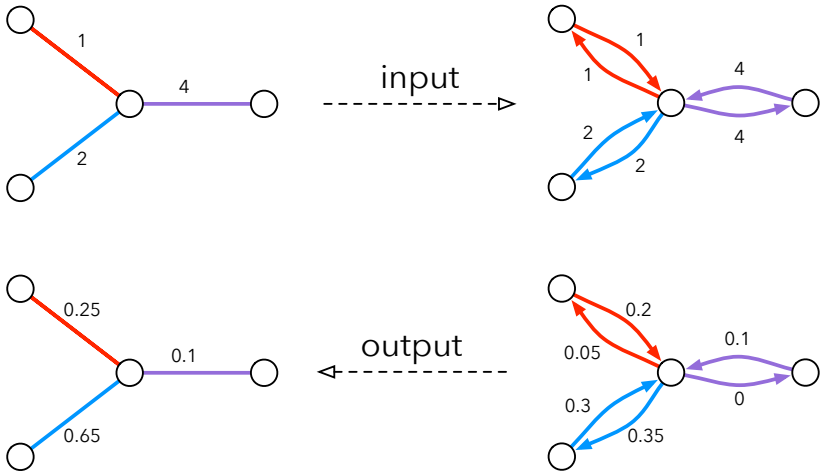

Figure 2.5. The simulation EC $\rightsquigarrow$ PO. Each vertex locally and virtually replaces each edge of colour c with two edges, both labelled with (c, c), denoted by c in the figure. Once the algorithm A has produced output in the virtual graph for each edge, these outputs are simply reduced back to their original edges.

seed graphs of degrees $d + 1$ and d. Consider the multigraph G with a single vertex g and loops of each colour in $[d+1]$: there must be at least one $c \in [d + 1]$ such that $A(g, c, G) \neq 0$. Now consider the H, the multigraph derived from G by removing the loop of colour c. Denote the vertex of H by h. Since the solution must be maximal, there is another colour χ such that $A(h, \chi, H) \neq A(g, \chi, G)$. Therefore G and H form a 0-critical pair.

After this, the proof proceeds as in the proof of Theorem 3: at each step we find a t-critical pair (G, H) such that for $g \in V(G)$ and $h \in V(H)$ we have that

(1) $B_G(g, t) \simeq B_H(h, t)$, and
(2) there is a loop colour c such that $A(g, G, c) \neq A(h, H, c)$.

Then we mix these graphs along the edge colour c, creating the graphs GG, GH, and HH. The difference of the outputs of G and H must propagate somewhere, and by the fact that all vertices must be saturated, it will propagate to a loop.

2.3.2 Simulation: EC $\rightsquigarrow$ PO

The first simulation EC $\rightsquigarrow$ PO is fairly straightforward. The vertices will locally construct a PO-graph based on the EC-graph and simulate any PO-algorithm with a small overhead. We will use the d^2-digraph edge colouring formulation of port numbering and orientation in the simulation.

Let A be a t-time PO-algorithm. The trick is that given an edge-coloured d-regular multigraph G, we will simulate A in a $2d$-regular PO-multigraph

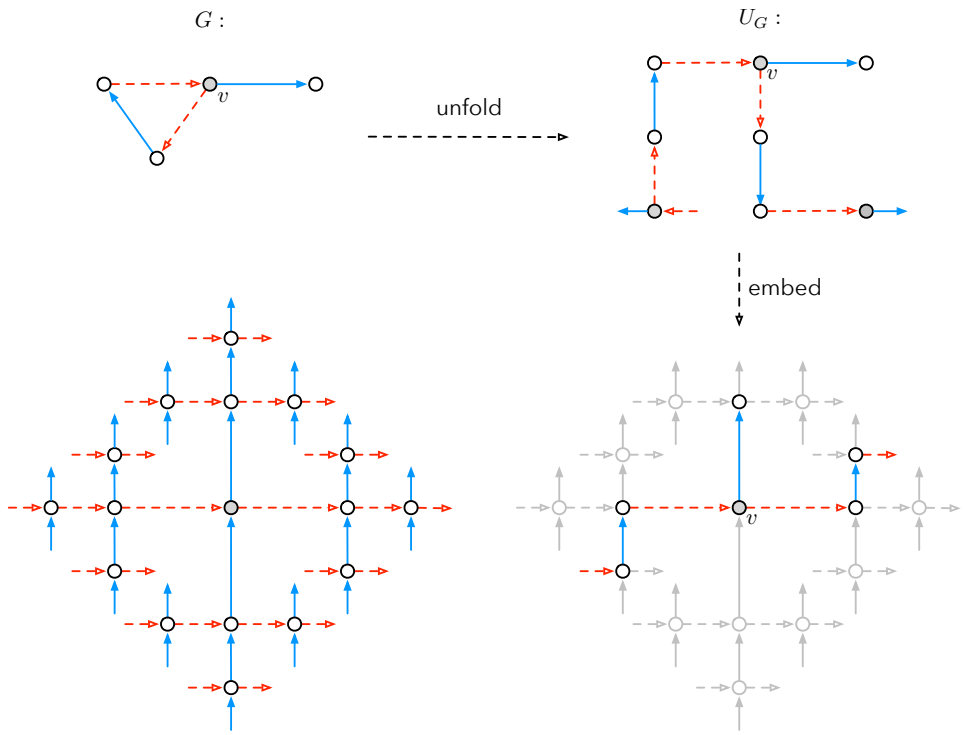

Figure 2.6. Simulation PO $\rightsquigarrow$ OI. Vertex v in G wants so simulate an order invariant algorithm A. First, v virtually unfolds its neighbourhood into its own local neighbourhood in U_G (this is essentially all it can do anyways, as anonymous algorithms are unable to detect cycles). Next v derives a locally consistent order on the vertices in its local neighbourhood from the 4-regular infinite tree T with a homogeneous total order.

constructed by replacing each edge $\{u, v\}$ of colour c with two directed (multi)edges, (u, v) and (v, u), both labelled with (c, c) or simply c. Now each vertex can virtually "unfold" these edges and simulate A in the simple PO graph H produced this way.

Finally, since A produces a maximal fractional matching on the virtual graph H, we can map the output of A back to G and define the output of the simulation A' as $A'(g, G, \{g, g'\}) = A(g, H, (g, g')) + A(g, H, (g', g))$. See Figure 2.5 for illustration.

Now if $t(\Delta) = o(\Delta)$, we have a contradiction: the simulation doubles the maximum degree of the graph, but this still yields a sublinear-in-Δ algorithm for the EC model.

2.3.3 Simulation: PO $\rightsquigarrow$ OI

In this section we discuss how a port-numbering and an orientation can be used to construct a locally consistent ordering on the vertices. This ordering will then allow the vertices to simulate an order-invariant algorithm in the PO model. This technique was first used to simulate OI algorithms in the

context of constant-time approximation [33].

Simulation is based on the fact that certain infinite trees have a useful total order on their vertices. For a fixed Δ, let r denote the number of different possible edge labels (a, b) in the PO model. The $2r$-regular infinite tree T has a total order on its vertices that is homogeneous, that is, the total order looks the same up to any distance at any two vertices. This also implies that the order of any two vertices depends only on the labels on the path that connects the vertices. We will use this order to construct a locally consistent ordering on any graph in the PO model.

Given such a total order, an algorithm A_{OI} with running time t can be simulated in the PO model roughly as follows.

Step 1: Each vertex v constructs its local, unfolded radius-t view, that is, its own local t-neighbourhood in U_G.

Step 2: Each vertex v then virtually embeds $B(v, U_G, t)$ in T and derives a total order on $B(v, U_G, t)$.

Step 3: Now each vertex can simulate A_{OI} on U_G. The solution is feasible, as the local orders derived from T form consistent total order on U_G. Finally, this solution is feasible on G as well, as anonymous algorithms are invariant over lifts.

See Figure 2.6 for illustration.

2.3.4 Simulation: OI $\rightsquigarrow$ LOCAL

The final simulation OI $\rightsquigarrow$ LOCAL uses a fairly standard Ramsey techniques to control the symmetry breaking effect of the unique identifiers [18, 49].

The trick is that Ramsey's theorem cannot be applied directly to a fractional maximal matching algorithm A, as discussed in Section 1.4, because the number of possible outputs is not bounded for linear programs. To compensate for this, we consider algorithm A^*, which works as a saturation indicator [7]: define that $A^*(v) = 1$ if v is saturated under A, and $A^*(v) = 0$ otherwise.

We show that in "loopy" OI-neighbourhoods, any algorithm A that uses identifiers can be forced to behave like an order-invariant algorithm, when identifiers are selected from a suitable set. Note that this last simulation, as the only simulation among the three, works only in some neighbourhoods. This will be enough for the purposes of the lower bound, however.

2.4 The barrier for proving maximal matching lower bounds in the LOCAL model

Our lower bound for fractional maximal matching does not imply a meaningful lower bound for maximal matching – as presented, it only applies when the running time of the algorithm does not depend on n and by Linial's lower bound *every* maximal matching algorithm runs in time $\Omega(\log^* n)$. Unfortunately the last simulation $\mathsf{OI} \rightsquigarrow \mathsf{LOCAL}$ presents a proof barrier that the current Ramsey technique cannot breach. If I denotes the size of the identifier space, then these techniques can only be applied when the running time of an algorithm is $o(\log^* n)$. The usual assumption of identifiers bounded polynomially in n means that any small construction must be padded so that its size allows a large enough identifiers space for controlling the effect of unique identifiers on the original construction.

The challenge of unique identifiers is twofold. First, they break the symmetry in the input graph, breaking the EC lower bound construction that relies on these symmetries. Second, controlling unique identifiers requires large constructions. Suomela suggests a two part approach to tackling this problem [59]: first, prove a lower bound for 2-coloured maximal matching, taking care of the symmetry breaking side. Then, given that a 2-colouring is unhelpful, show that the uniqueness property is also unhelpful.

3. Unique identifiers in distributed computing

In this chapter we see how unique identifiers can help constant-time algorithms, despite the negative results of Naor and Stockmeyer [49], and Göös et al. [33]. Both results require that the problem in question has a bounded number of different outputs. This technical condition is not true for the scheduling problems seen in this chapter. The algorithms for these scheduling should be considered more as structural results, showing that the bounding of outputs is necessary, and not just an artefact of the Ramsey-type proof technique. The algorithms also establish an interesting connection between randomized and deterministic constant-time algorithms.

We will also see a new lower bound for fractional domatic partition. This showcases the typical use of the indistinguishability argument.

3.1 Fractional graph colouring and scheduling

The scheduling problem considered in Publication IV is fractional graph colouring. This can be seen a scheduling problem, where at each time step the set of active vertices must be an independent set, that is, a set of vertices such that no two vertices are adjacent.

More formally, denote by $\mathcal{I}$ the set of independent sets in graph G. A fractional graph colouring x associates a value $x(I) \geq 0$ to each independent set I in G. Moreover, we must have for all $v \in V(G)$ that

$$\sum_{I \in \mathcal{I} : v \in I} x(I) \geq 1,$$

that is, every node is active for at least one time unit. The quality of a fractional graph colouring is measured by its length, defined as

$$\ell(x) = \sum_{I \in \mathcal{I}} x(I),$$

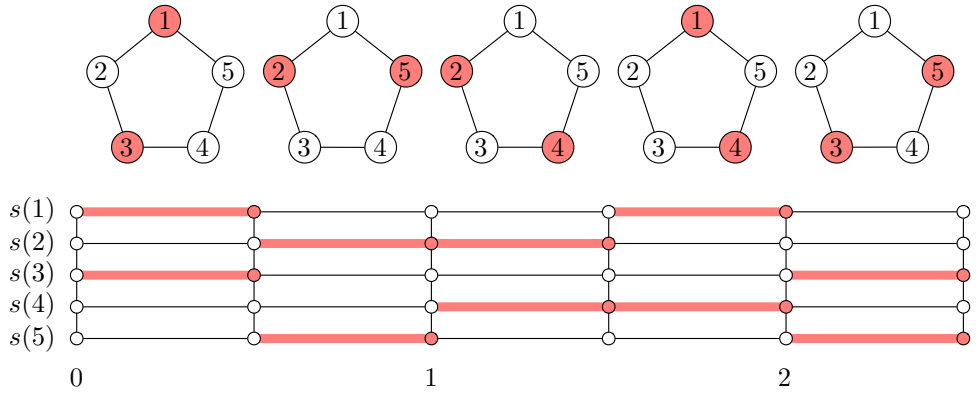

Figure 3.1. An example of the correspondence between a fractional graph colouring as a labelling $x\colon \mathcal{I} \to [0,1]$ of independent sets and the same labelling interpreted as a local schedule with each vertex outputting the intervals when it is active.

which the optimum fractional graph colouring minimizes.

This problem can be transformed into a scheduling problem in a straightforward manner. We order the independent sets $\mathcal{I} = (I_1, I_2, \ldots I_N)$ in an arbitrary manner. First, all vertices $v \in I_1$ are active for $x(I)$ time units, then, from time $x(I_1)$, all vertices in I_2 are active for $x(I_2)$ time units, and so on, for each of the N intervals.

Scheduling in a distributed setting. In the distributed setting, where the nodes are responsible for their own local outputs, we do not define the fractional graph colouring x explicitly. Instead, the output $s(v)$ of each vertex v consists of a disjoint set of intervals

$$s(v) = (a_1, b_1] \cup (a_2, b_2] \cup \cdots \cup (a_k, b_k],$$

where a_i, b_i are rational numbers. See Figure 3.1 for an illustration. The total length of the schedule is $\ell(v) = |s(v)| = \sum_{i=0}^{k}(b_i - a_i) \geq 1$. The set of active vertices at time t is denoted by $S(t) = \{v \in V : t \in s(v)\}$. For every t the set $S(t)$ must be an independent set, that is, for an edge $\{u, v\} \in E$, we have that $s(v) \cap s(u) = \emptyset$. This implicitly defines a fractional graph colouring: for each I we set $x(I)$ to be the sum of the lengths of the maximal intervals such that the set of active nodes equals I. This fractional graph colouring has length ℓ if $\max_{v \in V} \ell(v) = \ell$.

3.1.1 Abusing unique identifiers

In their seminal work, Naor and Stockmeyer study locally checkable labellings and prove the following theorem.

Theorem 6 ([49]). *If an LCL problem Π can be solved in a family of graphs $\mathcal{G}$ using a t-time local algorithm A, then it can be solved in $\mathcal{G}$ using a t-time*

identifier-oblivious algorithm A'.

This definition of an LCL problems requires two things. First, the outputs labels come from a finite set, and second, the input graph must be of bounded maximum degree.

We defined so-called *simple PO-checkable* optimization problems and showed that unique identifiers do not help constant-time algorithms to approximate these problems [33]. This class of problems is defined as follows.

Definition 7. A graph optimization problem Π is simple, if every feasible solution x of Π is a subset of vertices (or alternatively, every feasible x is a subset of edges). The problem Π is PO-checkable if the feasibility of each solution can be checked by a constant-time algorithm in the PO-model.

Similar to the class of LCL problems, this class includes many natural optimization problems, such as vertex covers, independent sets, dominating sets, matchings, and edge dominating sets.

Formally, we proved the following theorem.

Theorem 8 ([33], Theorem 1.3)**.** *Let* Π *be a simple PO-checkable graph problem. Assume that* $\mathcal{F}$ *is a family of bounded-degree graphs, and it is closed under lifts. If there is an ID-algorithm* A *with running time* $r = O(1)$ *that finds an* α*-approximation of* Π *in* $\mathcal{F}$*, then there is a PO-algorithm* B *with running time* r *that finds an* α*-approximation of* Π *in* $\mathcal{F}$*.*

The above theorem is a generalization of the fact that there are no constant-time approximation algorithms for the typical symmetry breaking problems, such as independent sets or matchings, even on cycles [18, 58]. In the PO-model it is impossible to break symmetry in any number of rounds: a cycle can be oriented in a consistent manner with outgoing ports numbered with 1s and incoming ports numbered with 2s. In such an input graph, after every round r, every node always sees the same r-neighbourhood. Thus the only feasible solution is the empty solution, and no approximation is possible. On the other hand, some problems, like 2-approximation of vertex cover do not require symmetry breaking, and our results show that local algorithms for such problems do not require the use of unique identifiers.

It is crucial that Theorem 8 applies only to *simple* PO-checkable problems. The impossibility of anonymous symmetry breaking means that algorithms cannot find fractional graph colourings in the PO-model. On the other hand,

since fractional graph colouring is not a simple graph problem, Theorem 8 does not apply to it. In Publication IV we give an algorithm that is able to abuse unique identifiers to give a non-trivial constant-time approximation algorithm for fractional graph colouring.

Theorem 9. *For any $\alpha > 1$ there exists a deterministic constant-time algorithm A with running time 1 such that A finds in any graph G a fractional graph colouring x of length $\alpha(\Delta(G) + 1)$.*

Note that this result can be seen as a proof that the requirement for constant-size outputs in the definition of LCL-problems is necessary as well. We can consider the problem of finding a fractional graph colouring such that each vertex v has a schedule of length at most $2(d(v) + 1)$. As it turns out, the proof of Theorem 9 shows that each individual vertex v has a schedule of length at most $\alpha(d(v) + 1)$, for any $\alpha > 1$.

3.1.2 Constant-time algorithm for fractional graph colouring

The algorithm given by Theorem 9 is not practical in bounded-degree graphs, and should be considered more as a structural result. It establishes an interesting connection between deterministic and randomized algorithms, as the fine structure of the output allows the deterministic algorithm for fractional graph colouring to simulate a randomized algorithm for independent sets.

A prior constant-time algorithm for fractional graph coloring due to Kuhn [40] uses a slightly different approach. Where we use a randomized algorithm as a black box and iterate over all random inputs, Kuhn's algorithm iterates over all permutations of the identifiers and selects the local maxima as active nodes.

In principle our simulation can use any randomized algorithm, but for simplicity's and concreteness' sake we fix the algorithm A to be simulated to be the algorithm where each vertex flips some constant number of random bits and joins the independent set if its value is the local maximum. The key idea in the simulation is that the schedule can be arbitrarily fine-grained. We abuse this by constructing a schedule at each point of time the set of active vertices corresponds to the independent set produced by A with some (deterministically chosen) random input.

Let r denote the number of random bits used by the randomized independent set algorithm A. The constant r depends on how good solutions we want to achieve and affects the granularity of the output schedule. We

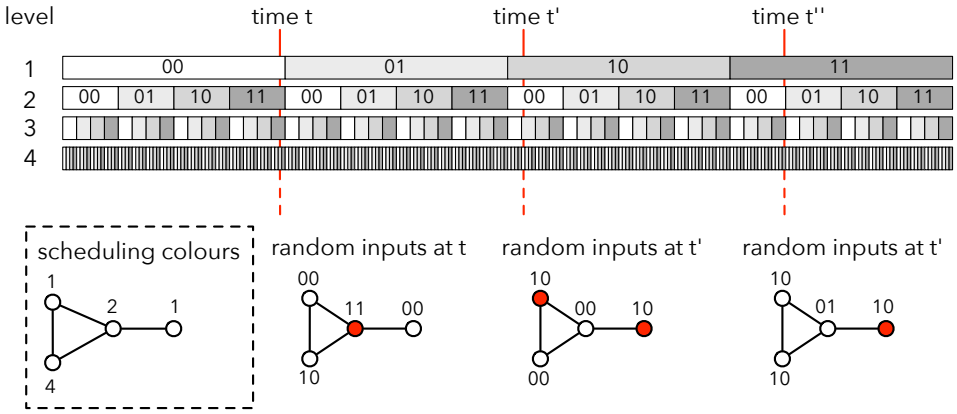

Figure 3.2. The scheduling scheme, illustrated using the first four levels, using $r = 2$. In the upper part, the static schedule goes over each possible assignment of two random bits in a recursive fashion. On the lower part, the vertices have computed their scheduling colours, which form a proper colouring of the input graph. At each time t, each vertex checks its own random input and its neighbours' random inputs from the scheduling scheme, from the levels given by their scheduling colours.

define a recursive simulation scheme that the vertices use to coordinate their simulations of A. On the first level, each of the 2^r possible random inputs is used for an interval of length $1/2^r$. On level i, for each maximal interval $[a, b)$ of level $i-1$ during which the random input remains constant, the scheme uses each of the possible random inputs for a subinterval of length $(b - a)/2^r$. See Figure 3.2 for illustration.

The vertices choose a *scheduling colour* based on their unique identifier and degree, and send this to their neighbours. The scheduling colours have some technical requirements, including that they must form a proper colouring. Each vertex uses at each time t the random bits given by the level in the scheduling scheme equal to its scheduling colour. For each time step t, vertices can check their own and their neighbour's random inputs and simulate A. This scheme ensures that each combination of random inputs occurs equally often and therefore the deterministic output of the algorithm corresponds to the expected output of the randomized algorithm.

Interestingly, as we saw in Section 2.3 that there is a problem, fractional maximal matching, which has unbounded outputs, but unique identifiers are unhelpful when solving it in constant time. This can be explained by the maximality condition of the solution: the output of any algorithm A can be reduced to a version A^*, which simply tells whether a vertex is fully saturated under A. This output is bounded, and can then be controlled via Ramsey techniques. A fractional graph colouring is not necessarily minimal in any analogous way.

3.1.3 Fractional domatic partitions

In Publication IV we also study the problem of fractional domatic partitions. In this section I will show how to give a tight lower bound, corresponding to the algorithm in Publication IV. To this end, we will next formally define the problem of fractional domatic partition. Given a graph G, let $\mathcal{D}$ denote the set of dominating sets in G, that is, sets $D \subseteq V(G)$ such that for all $v \in V(G)$, we have that $N(v) \cap D \neq \emptyset$. Similar to fractional graph colouring, a fractional domatic partition x assigns a weight $x(D)$ to each dominating set D of G. We require that for all $v \in V(G)$ it holds that

$$\sum_{D \in \mathcal{D} : D \ni v} x(D) \leq 1.$$

Fractional domatic partition is a maximization problem with objective function $\sum_{D \in \mathcal{D}} x(D)$.

Again, we can turn fractional domatic partition into a scheduling problem by defining an ordering $(D_1, D_2, \ldots, D_M)$ on the dominating sets and then scheduling each set one after the other. On an intuitive level, x can be seen as a schedule that covers the whole graph at all times, but each vertex has, for example, a battery which constrains how long it can be active.

Now in the context of distributed computing, we once again use the local schedule of each node to define a (fractional) domatic partition. For each v, let $s(v)$, or the schedule of v, consist of disjoint intervals

$$s(v) = (a_1, b_1] \cup (a_2, b_2] \cup \cdots \cup (a_k, b_k].$$

We say that s has length ℓ if for all $t \leq \ell$, we have that the set of active nodes $\{v : t \in s(v)\} \in \mathcal{D}$ is a dominating set.

3.1.4 Optimal lower bound for domatic partitions

In this section I show that the algorithm we present for fractional domatic partitions in Publication IV is asympototically optimal. This improves the trivial lower bound presented in Publication IV. I use the cycle cutting technique used at least by Alon [3]. The proof is due to the existence of graphs with minimum dominating sets of size $(1 + o(1))(\ln d/d)n$ [4].

In Publication IV we prove the following theorem.

Theorem 10 (Publication IV, Theorem 8)**.** *There exists a deterministic local algorithm that finds a domatic partition x for any graph G in one communication round. Moreover, the length of x is at least*

$$\frac{\delta + 1}{3 \ln(\delta + 1)},$$

where δ is the minumum degree of graph G.

As every domatic partition must have length at least $\delta + 1$, this result leaves a logarithmic gap between the lower and upper bounds. In this section I describe how this gap can be closed.

Similar to Theorem 2 in Publication IV, we have the following theorem.

Theorem 11. *Let $\mathcal{F}_d$ be the family of d-regular graphs, and let A_d be a deterministic constant-time algorithm that finds a domatic partition for any G_d in $\mathcal{F}_d$. For each d there exists G_d such that G_d has a domatic partition of length $d+1$, but A_d produces a domatic partition of length $O(d/\ln d)$.*

The proof is by a familiar indistinguishability argument: for every d and every constant g (in fact, if necessary, we could get $g = \Theta(\log_d n)$), there exist two d-regular graphs of girth at least g such that one has a domatic partition of length $d+1$, and the other has minimum dominating set of size $\Omega(n \ln d/d)$ and thus maximum (fractional) domatic partition of length $O(d/\ln d)$. We start with the two necessary lemmas.

Lemma 1. *For every pair of constants g and d, there exists a d-regular graph G of girth at least g such that G has a domatic partition of length $d+1$.*

Proof. The proof closely resembles the proof of Alon for the existence of graphs with high girth and dominating sets of size $1/(d+1)n$ [3, Lemma 2.1].

First, let $G = (V, E)$ consist of $2r$ disjoint stars of degree d, for r large enough, giving $n = 2r(d+1)$. In addition, we fix a $(d+1)$-colouring c on these stars: each middle vertex is coloured with the colour 1, and each of the leaf vertices gets a different colour from $\{2, \ldots, d+1\}$. Finally, to get a d-regular graph with a domatic partition of length $d+1$, we find an arbitrary perfect matching $E_{i,j}$, for every $i, j \in \{2, \ldots, d+1\}$ with $i < j$, in the complete bipartite graph on the vertices of colours i and j, and add these edges to E. We call the edges added in this fashion *non-star edges*. An edge with endpoints of colours i and j will be called an (i, j)-edge. In the graph G consisting of the original stars and each of the perfect matchings each colour class i is a dominating set, yielding a domatic partition of length $d+1$.

Next we will cut every cycle of length strictly less than g one by one. Consider the shortest such cycle C (of some length g') and an (i, j)-edge $e = \{u, v\}$, for any $i \neq j$, along that cycle. Since G is d-regular and

every node has an (i, j)-edge in its radius-2 neighbourhood, there must be an (i, j)-edge $e' = \{u', v'\}$ that is at distance $\Omega(\log_d n)$ from e. Now remove $\{u, v\}$ and $\{u', v'\}$ from E, and add $\{u, v'\}$ and $\{u', v\}$. Each colour class $c^{-1}(i)$ is still a dominating set, cycle C has been cut, and no new cycles of length at most g' were created. To see this, note that any cycle containing both $\{u, v'\}$ and $\{u', v\}$ must have length at least $2g'$, and any cycle containing at most one of those edges must have length at least $g + 1$. Repeatedly cutting short cycles we finally arrive at a graph with a domatic partition (given by the colouring) of size $d + 1$ and no cycles of length at most g. □

We will need a result of Alon and Wormald as the main ingredient of the lower bound.

Lemma 2 ([4])**.** *There exist d-regular graphs G of arbitrarily high constant girth g with domination number $\gamma(G) = \Omega(n \ln(d)/d)$.*

Now we are ready to prove Theorem 11.

Proof of Theorem 11. For any given d, assume that we have an algorithm A_d with constant running time $t(d)$. We construct two d-regular graphs G_1 and G_2 given by Lemmas 1 and 2 such that

- G_1 has girth $g_1 > 2t + 3$ and a domatic partition of length $d + 1$, and
- G_2 has girth $g_2 > 2t + 3$ and has a minimum dominating set of size $\Omega(n \ln(d)/d)$, and thus maximum (fractional) domatic partition of length $O(d/\ln d)$.

Now there must exist an identifier assignment to G_2 such that there is a node v such that v and all nodes in $N(v)$ are inactive after time $c(d/\ln d)$, for some constant c. Now we copy this identifier assignment in $B_{2t+3}(v, G_2)$ to the $(2t + 3)$-neighbourhood of an arbitrary node u in G_1. When A_d is run on G_1 with this identifier assignment, it produces a domatic partition of length at most $cd/\ln d$, while a perfect domatic partition of length $d + 1$ exists in G_1. □

4. Local decision

In this chapter we take a closer look at the techniques for proving the two main results of this work regarding local decision. In Publication II we characterise the power of unique identifiers in this context. In Publication III we present a hierarchy of local decision, analogous to the polynomial hierarchy, and study its structure.

4.1 Power of unique identifiers in local decision

Recall that a distributed oracle is a function that maps each $n \in \mathbb{N}$ to a set of n labels $(f_1, f_2, \ldots, f_n)$. We denote by $f_k^{(n)}$ the kth smallest label on input n. We defined that an oracle f is large, if

$$\forall c > 0, \exists k \geq 1, \forall n \geq k, f_k^{(n)} \geq c. \tag{4.1}$$

Otherwise f is small. As we will see in Section 4.1.2, a large f allows the vertices to recover a new labelling f', which has the crucial property that every subset S of vertices has a label of size at least $|S|$.

The main theorem of Publication II is a corollary of two lemmas. The first lemma gives a separation between LD and LDO with advice from any small oracle f.

Lemma 3. *For any small f, there is a property $P \in \mathsf{LD} \setminus \mathsf{LDO}^f$.*

Property P is a promise free version of the property we discussed in Section 1.5.1: similar to the separation of LDO and LD due to Fraigniaud et al. [26], we consider the class of graphs which encode the execution table of a Turing machine. The trick is to handle the oracle labels, which we do by padding the graph in suitable manner.

The second lemma says that this separation is as strong as possible, as large oracles are strong enough to simulate LD.

Lemma 4. *For any large f, we have that* $\mathsf{LDO}^f = \mathsf{LD}^f$.

This lemma is shown via a simulation argument: a large oracle will allow the vertices to simulate a non-identifier oblivious algorithm.

Combined, these two lemmas yield a total order on the four classes.

Theorem 12. $\mathsf{LDO} \subsetneq \mathsf{LD} \subseteq \mathsf{LDO}^f = \mathsf{LD}^f$ *for any large* f.

In Sections 4.1.1 and 4.1.2 we take a closer look at how these two results a shown.

We also study the relationships of the classes for small oracles f. We show that LD and LDO^f are incomparable. By Lemma 3 there is a property $P \in \mathsf{LD} \setminus \mathsf{LDO}^f$. We show that there is a small f and a property $P \in \mathsf{LDO}^f \setminus \mathsf{LD}$.

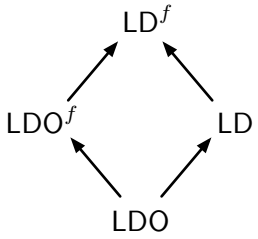

Figure 4.1. A partial order on the classes LDO, LD, LDO^f, and LD^f for small oracles f.

4.1.1 Proof of Lemma 3

In this section I present a simplified version of the proof of Lemma 3. We will ignore some technical aspects of the full proof, as they do not contribute to the intuition as to why the separation exists.

The overall structure of the proof is to construct a promise-free version of the uncomputability argument we saw in Section 1.5.1. This takes the form of a fairly standard graph encoding of the execution of a Turing machine, as in the proof of Fraigniaud et al. [26]. Then we modify the construction so that small oracles are useless as well.

Recall that the property P with promise was the following: given a path P_n on n vertices, each vertex gets a Turing machine M as input. We promise that if M halts on an empty tape, then it does so in $s \leq n$ steps. The instance (P_n, M) is a *yes*-instance, if M does not halt. This language is clearly in LD, if behaviour can be arbitrary when the promise doesn't hold: each vertex v simulates M for $\mathrm{id}(v)$ steps and at least one vertex detects that M halts if it does so. On a high level, the trick is to encode the halting

time s of M into the input without an explicit promise. That is, the very existence of a legal input should solve the halting problem.

To get a promise-free property, the structural properties of the input graphs G have to be locally checkable even using identifier oblivious algorithms. The promise-free property P_0 consists of labelled graphs $G(M, r)$, for each (halting) M and each constant running time r. Moreover, such a graph is not in P_0 if and only if M halts on an empty tape and outputs 1 (we can define P_1 by flipping the halting condition). The computationally hard task will be to separate the cases where M outputs 0 and where M outputs 1, as the input, together with local checkability, will give the fact that M halts for free. Now if there was a local algorithm A deciding P_0 in LDO^f, we could sequentially construct $G(M, r)$ and simulate A on it to separate languages P_0 and P_1, a contradiction to a known uncomputability result (see, for example, Papadimitiou [53]).

Construction $G(M, r)$. Let M be a Turing machine that halts in s steps on an empty tape (naturally if M does not halt, we cannot construct a finite input graph that encodes the infinite execution of M) and let r be the locality parameter. The construction consists of three parts.

Part I: An $(s+1) \times (s+1)$ sized 2-dimensional grid, the execution table of M, denoted by $T(M)$. Each vertex knows the identity of M. In addition, each vertex on row j holds the contents of the tape after step j, with the ith vertex on each row holding the contents of the ith place on the tape. In addition, on each row one vertex holds the machine head, including the state of M. Each vertex also holds its own coordinate (i, j) on the grid modulo 3, providing a consistent orientation of $T(M)$. It is essential that the vertices don't see their real coordinates, as this would leak too much information on the running time of M.

Part II: A *fragment collection* $\mathcal{F}$, consisting of every syntactically correct (according to the transition rules of M) $(2r+1) \times (2r+1)$ piece of execution table of M. Each fragment is connected to the the pivot along its edges.

Part III: A *tail* S_N, a sufficiently long path starting from the pivot, depending on the choice of f, the distributed oracle. The vertices of the path hold distance counters to the pivot, but not the identity of M.

Properties of $G(M, r)$. The three parts guarantee three crucial properties of the construction.

P I: The execution table ensures that the size of the graph $G(M, r)$ is connected to the number of steps s it takes for M to halt on an empty tape. Since $s \leq |T(m)|$, each vertex v on the table can simulate M for $\mathrm{id}(v)$ many steps and reject if M halts with output 1.

P II: The fragment collection ensures that a local algorithm that simply scans the execution table for a part where M halts with 1 will always reject. This is due to the fact that the fragment collection contains all legal pieces of the execution table, including those that witness M outputting 1 (from *some* starting state).

P III: The tail ensures that a small oracle will be unhelpful. A large tail makes sure that the adversary can label the whole execution table $T(M)$ with some predetermined constant label, due to the definition of a small oracle. All useful labels end up in the tail, which does not contain the identity of M, and cannot therefore use these labels.

There is still the technical issue that the structure of $G(M, r)$ has to be locally checkable for P_0 to separate LD and LDO^f. This can be dealt by attaching additional structure to the construction so that the vertices can detect if it is, for example, a wrapped around torus.

4.1.2 Proof of Lemma 4

The proof of Lemma 4 proceeds through a fairly straightforward simulation argument: the labels provided by a large oracle allow the vertices to recover (in a weak sense) an assignment of unique identifiers. With slight abuse of notation we will let $f(v) = f^{(n)}(v)$ denote the label of vertex v, assigned by the adversary, in some input graph G of size n. Now a large oracle f is necessary *and* sufficient to recover another oracle $\hat{f}$ with the following property: if $f^{(n)}(v)$ is the ith smallest label produced by f, then v can compute a new label $\hat{f}(v) \geq i$.

Let A_{ID} be a t-time algorithm that decides a property $P \in \mathsf{LD}$. Given an input (G, x), the identifier-oblivious simulation of A_{ID} proceeds as follows.

Step 1: Each vertex v computes $\hat{f}(v)$.

Step 2: Each vertex gathers $B_G(v, t)$ and the labelling $\hat{f}$ restricted to $B_G(v, t)$. Let $\hat{f}^*(v)$ denote the maximum of these values.

Step 3: Each vertex runs a local simulation of A_{ID} with each possible locally unique identifier assignment from $[\hat{f}^*(v)]$ to the vertices of $B_G(v, t)$.

If at least one of these simulations outputs *reject*, then reject, and otherwise accept.

Clearly the algorithm rejects correctly, as rejection by A_{ID} is independent of the identifier assignment by its correctness. To see that acceptance is correct, observe that there is a choice of simulations, one per vertex, such that the local identifier assignments form a globally consistent unique identifier assignment. In particular, the assignment id where $\mathrm{id}(v) = i$ when $f^{(n)}(v)$ is the ith smallest label (breaking ties in some consistent way) is used locally by each vertex in a simulation.

4.2 A hierarchy of local decision

In this section we look in more detail at the hierarchy of local decision presented in Publication III. First we will go through the structure of the local hiearchy. Then we will see the proof of one of these results in more detail. This is intended as an illustration of the typical simulation techniques that we are using.

To establish the structure of the hierarchy, we show that a final universal quantifier is unhelpful. Intuitively this happens because the algorithm cannot trust anything from the disprover without a corresponding answer from the prover.

Theorem 13. *For all even $k \geq 2$, we have that $\Sigma_k = \Sigma_{k-1}$, and for all odd $k \geq 3$, we have that $\Pi_k = \Pi_{k-1}$.*

This collapse establishes the structure of LH as the alternation of classes Π_{2k} and Σ_{2k+1}. The proof is by simple simulation argument: the prover can certify the the size of the graph to each vertex, and the vertices can then simulate all possible certificate assignments by the disprover, accepting if and only if they are all accepting.

4.2.1 Reversal of decisions

The following theorem states that decisions can be reversed using an additional level in the decision protocol.

Theorem 14. *If $P \in \mathrm{co}{-}\Lambda_k$, then $\bar{P} \in \Lambda_{k+1}$.*

The proof is similar to the proof of the fact that $\mathrm{co}\text{-}\mathsf{LCP}(0) \subseteq \mathsf{LCP}(\log n)$ due to Göös and Suomela [32].

We will look at the case for odd k. The even case is similar. Since $P \in \text{co}-\Lambda_k$, we have that $\bar{P} \in \Lambda_k$. Let A be an algorithm that decides $\bar{P}$. We have

$$(G,x) \notin P \iff \forall \ell_1, \exists \ell_2, \ldots, \exists \ell_k, \forall v \in V(G), A(v, G, x, \ell_1, \ldots, \ell_k) = 1.$$

We will simulate A on level $k+1$. If $(G,x) \in P$ and therefore $(G,x) \notin \bar{P}$, the prover can follow the winning strategy of the disprover for the first k rounds. This guarantees that there is a rejecting vertex v. Now for its last quantifier, the prover will construct a spanning tree to point to the rejecting vertex v. That vertex can simulate A and see that it is rejecting, and accept. Other vertices check that their spanning tree construction is valid, and accept. If $(G,x) \in \bar{P}$, the disprover has no winning strategy, and then at the $k+1$th level the disprover can always force all vertices to accept. Any spanning tree constructed by the prover will point to an accepting node, which will reject as required.

This proof is just one example of the importance of spanning tree certificates in broadcasting global information throughout the network.

4.2.2 Conditional collapse of the hierarchy

While we are unable to decide if the hierarchy is infinite, we can show the following conditional collapse result.

Theorem 15. *If for some k we have $\Lambda_k = \Lambda_{k+1}$, then $\mathsf{LH} = \Lambda_k$.*

Assume that there is an even k such that $\Lambda_k = \Lambda_{k+1}$ (again, the proof is similar for odd k). Starting from a contradiction, assume that there is a property $P \in \Lambda_{k+2} \setminus \Lambda_{k+1}$, decided by a local algorithm A. By definition we have that

$$(G,x) \in P \iff \forall \ell_1, \exists \ell_2, \ldots, \exists \ell_{k+2}, \forall v \in V(G), A(v, G, x, \ell_1, \ldots, \ell_{k+2}) = 1. \tag{4.2}$$

We can define a new property $\tilde{P} \in \Lambda_{k+1}$ based on P, where the first labelling of the disprover comes from the input and $(G,(x,\ell)) \in \tilde{P}$ if and only if the prover has a winning strategy under A. That is, we feed the labelling ℓ into A and define

$$(G,(x,\ell)) \in \tilde{P} \iff \exists \ell'_1, \forall \ell'_2, \ldots, \exists \ell'_{k+1}, \forall v \in V(G), A(v, G, x, \ell, \ell'_1, \ldots, \ell'_{k+1}) = 1.$$

Since this language is in Λ_{k+1}, by our assumption there is a local algorithm A' such that

$$(G,(x,\ell)) \in \tilde{P} \iff \forall \ell_1, \ldots, \exists \ell_k, \forall v \in V(G), A'(v, G, (x,\ell), \ell_1, \ldots, \ell_k) = 1.$$

Now we have a protocol for P on level k: given (G, x), the disprover begins by giving both ℓ and ℓ_1. If $(G, x) \in P$, the prover will always have a winning strategy, no matter how ℓ is chosen. On the other hand, if $(G, x) \notin P$, the disprover can find by (4.2) a labelling ℓ such that it has a winning strategy.

References

[1] Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 793–802. IEEE, October 2008. doi:10.1109/FOCS.2008.11.

[2] Yehuda Afek, Shay Kutten, and Moti Yung. The local detection paradigm and its applications to self-stabilization. *Theoretical Computer Science*, 186 (1–2):199–229, 1997. doi:10.1016/S0304-3975(96)00286-1.

[3] Noga Alon. On constant time approximation of parameters of bounded degree graphs. In Oded Goldreich, editor, *Lecture Notes in Computer Science*, volume 6390, pages 234–239. Springer, 2010. doi:10.1007/978-3-642-16367-8_14.

[4] Noga Alon and Nicholas Wormald. High degree graphs contain large-star factors. In *Fete of Combinatorics and Computer Science*, volume 20, pages 9–21. Springer, 2010. doi:10.1007/978-3-642-13580-4_1. arXiv:0810.2053.

[5] Dana Angluin. Local and global properties in networks of processors. In *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, pages 82–93. ACM Press, 1980. doi:10.1145/800141.804655.

[6] Matti Åstrand and Jukka Suomela. Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In *Proc. 22nd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2010)*, pages 294–302. ACM Press, 2010. doi:10.1145/1810479.1810533.

[7] Matti Åstrand, Patrik Floréen, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, and Jara Uitto. A local 2-approximation algorithm for the vertex cover problem. In *Proc. 23rd International Symposium on Distributed Computing (DISC 2009)*, volume 5805 of *Lecture Notes in Computer Science*, pages 191–205. Springer, 2009. doi:10.1007/978-3-642-04355-0_21.

[8] Baruch Awerbuch, Boaz Patt-Shamir, and George Varghese. Self-stabilization by local checking and correction. In *Proc. 32nd Annual Symposium on Foundations of Computer Science (FOCS 1991)*, pages 268–277. IEEE, 1991. doi:10.1109/SFCS.1991.185378.

[9] Alkida Balliu, Gianlorenzo D'Angelo, Pierre Fraigniaud, and Dennis Olivetti. Brief Announcement: Local Distributed Verification. In *Proc. 30th International Symposium on Distributed Computing (DISC 2016)*, volume 9888 of *Lecture Notes in Computer Science*, pages 461–464. Springer-Verlag Berlin Heidelberg, 2016. doi:10.1007/978-3-662-53426-7. arXiv:1605.03892.

[10] Reuven Bar-Yehuda, Keren Censor-Hillel, and Gregory Schwartzman. A Distributed $(2+\epsilon)$-Approximation for Vertex Cover in $O(\log \Delta/\epsilon \log\log \Delta)$ Rounds. In *Proc. 35th ACM Symposium on Principles of Distributed Computing (PODC 2016)*. ACM, 2016. arXiv:1602.03713.

[11] Leonid Barenboim. Deterministic $(\Delta+1)$-Coloring in Sublinear (in Δ) Time in Static, Dynamic and Faulty Networks. In *Proc. 34th ACM Symposium on the Principles of Distributed Computing (PODC 2016)*, pages 345–354. ACM, 2015. doi:10.1145/2767386.2767410.

[12] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. In *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 321–330. IEEE Computer Society Press, 2012. doi:10.1109/FOCS.2012.60.

[13] Leonid Barenboim, Michael Elkin, and Fabian Kuhn. Distributed $(\Delta+1)$-coloring in linear (in Δ) time. *SIAM Journal on Computing*, 43(1):72–95, 2014. doi:10.1137/12088848X.

[14] Béla Bollobás. The independence ratio of regular graphs. *Proceedings of the American Mathematical Society*, 83(2):433–436, 1981.

[15] 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 *Proc. 48th Annual Symposium on the Theory of Computing (STOC 2016)*, pages 479–488. ACM, 2016. doi:10.1145/2897518.2897570. arXiv:1511.00900v1.

[16] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An Exponential Separation Between Randomized and Deterministic Complexity in the LOCAL Model. In *Proc. 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2016)*, pages 615–624. IEEE, 2016. arXiv:1602.08166v2.

[17] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986. doi:10.1016/S0019-9958(86)80023-7.

[18] Andrzej Czygrinow, Michał Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *Proc. 22nd International Symposium on Distributed Computing (DISC 2008)*, volume 5218 of *Lecture Notes in Computer Science*, pages 78–92. Springer, 2008. doi:10.1007/978-3-540-87779-0_6.

[19] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM Journal on Computing*, 41(5):1235–1265, 2012. doi:10.1137/11085178X.

[20] Reinhard Diestel. *Graph Theory*. Springer, Berlin, 4th edition, 2010. http://diestel-graph-theory.com/.

[21] Patrik Floréen, Joel Kaasinen, Petteri Kaski, and Jukka Suomela. An optimal local approximation algorithm for max-min linear programs. In *Proc. 21st Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2009)*, pages 260–269. ACM Press, 2009. doi:10.1145/1583991.1584058. arXiv:0809.1489.

[22] Klaus-Tycho Förster, Thomas Luedi, Jochen Seidel, and Roger Wattenhofer. Local Checkability, No Strings Attached. In *Proc. 17th International Conference on Distributed Computing and Networking (ICDCN 2015)*, pages 21:1–21:10. ACM, 2015. doi:10.1145/2833312.2833315.

[23] Pierre Fraigniaud, Amos Korman, and David Peleg. Local distributed decision. In *Proc. 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011)*. IEEE Computer Society Press, 2011. doi:10.1109/FOCS.2011.17.

[24] Pierre Fraigniaud, Magnús M. Halldórsson, and Amos Korman. On the impact of identifiers on local decision. In *Proc. 16th International Conference on Principles of Distributed Systems (OPODIS 2012)*, volume 7702 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 2012. doi:10.1007/978-3-642-35476-2_16.

[25] Pierre Fraigniaud, Amos Korman, Merav Parter, and David Peleg. Randomized distributed decision. In *Proc. 26th International Symposium on Distributed Computing (DISC 2012)*, volume 7611 of *Lecture Notes in Computer Science*, pages 371–385. Springer, 2012. doi:10.1007/978-3-642-33651-5_26.

[26] Pierre Fraigniaud, Mika Göös, Amos Korman, and Jukka Suomela. What can be decided locally without identifiers? In *Proc. 32nd Annual ACM Symposium on Principles of Distributed Computing (PODC 2013)*, pages 157–165. ACM Press, 2013. doi:10.1145/2484239.2484264. arXiv:1302.2570.

[27] Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a Complexity Theory for Local Distributed Computing. *Journal of the ACM*, 60(5):1–26, 2013. doi:10.1145/2499228.

[28] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local Conflict Coloring. In *Proc. 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2016)*, pages 625–634. IEEE, 2016. doi:10.1109/FOCS.2016.73. arXiv:1511.01287.

[29] Joel Friedman. A proof of Alon's second eigenvalue conjecture. In *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC 2003)*, pages 720–724. ACM Press, 2003. doi:10.1145/780542.780646.

[30] David Gamarnik and Madhu Sudan. Limits of local algorithms over sparse random graphs. In *Proc. 5th Conference on Innovations in Theoretical Computer Science (ITCS 2014)*, pages 369–376. ACM Press, 2014. doi:10.1145/2554797.2554831. arXiv:1304.1831.

[31] Mohsen Ghaffari. An Improved Distributed Algorithm for Maximal Independent Set. In *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 270–277. Society for Industrial and Applied Mathematics, 2016. doi:10.1137/1.9781611974331.ch20. arXiv:1506.05093.

[32] Mika Göös and Jukka Suomela. Locally checkable proofs. *Theory of Computing*, 2014. To appear.

[33] Mika Göös, Juho Hirvonen, and Jukka Suomela. Lower bounds for local approximation. *Journal of the ACM*, 60(5):39:1–23, 2013. doi:10.1145/2528405. arXiv:1201.6675.

[34] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. *SIAM Journal on Discrete Mathematics*, 15(1):41–57, 2001. doi:10.1137/S0895480100373121.

[35] Juho Hirvonen, Joel Rybicki, Stefan Schmid, and Jukka Suomela. Large cuts with local algorithms on triangle-free graphs, February 2014. arXiv:1402.2543.

[36] Amos Korman and Shay Kutten. On distributed verification. In *Proc. 8th International Conference on Distributed Computing and Networking (ICDCN 2006)*, volume 4308 of *Lecture Notes in Computer Science*, pages 100–114. Springer, 2006. doi:10.1007/11947950_12.

[37] Amos Korman and Shay Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20(4):253–266, 2007. doi:10.1007/s00446-007-0025-1.

[38] Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010. doi:10.1007/s00446-010-0095-3.

[39] Amos Korman, Shay Kutten, and Toshimitsu Masuzawa. Fast and compact self stabilizing verification, computation, and fault detection of an MST. In *Proc. 30th Annual ACM Symposium on Principles of Distributed Computing (PODC 2011)*, pages 311–320. ACM Press, 2011. doi:10.1145/1993806.1993866.

[40] Fabian Kuhn. Local Multicoloring Algorithms: Computing a Nearly-Optimal TDMA Schedule in Constant Time. In *Proc. 26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009)*, volume 3 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 613–624. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009. doi:10.4230/LIPIcs.STACS.2009.1852. arXiv:arXiv:1310.1707v3.

[41] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, pages 980–989. ACM Press, 2006. doi:10.1145/1109557.1109666.

[42] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local Computation: Lower and Upper Bounds. *Journal of the ACM*, 63(2):17:1–17:44, 2016. doi:10.1145/2742012. arXiv:1011.5470.

[43] Joseph Lauer and Nicholas Wormald. Large independent sets in regular graphs of large girth. *Journal of Combinatorial Theory, Series B*, 97(6):999–1009, 2007. doi:10.1016/j.jctb.2007.02.006.

[44] Juhana Laurinharju and Jukka Suomela. Brief announcement: Linial's lower bound made easy. In *Proc. 33rd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2014)*, pages 377–378. ACM Press, 2014. doi:10.1145/2611462.2611505. arXiv:1402.2552.

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

[46] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[47] Moshe Morgenstern. Existence and explicit constructions of $q + 1$ regular Ramanujan graphs for every prime power q. *Journal of Combinatorial Theory, Series B*, 62(1):44–62, 1994. doi:10.1006/jctb.1994.1054.

[48] Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991. doi:10.1137/0404036.

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

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

[51] Alessandro Panconesi and Aravind Srinivasan. The local nature of Δ-colouring and its algorithmic applications. *Combinatorica*, 15(2):255–280, 1995. doi:10.1007/BF01200759.

[52] Alessandro Panconesi and Aravind Srinivasan. On the complexity of distributed network decomposition. *Journal of Algorithms*, 20(2):356–374, 1996. doi:10.1006/jagm.1996.0017.

[53] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.

[54] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 2000.

[55] Mustazee Rahman and Bálint Virág. Local algorithms for independent sets are half–optimal, 2014. arXiv:1402.0485v1.

[56] Frank P. Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, 30:264–286, 1930. doi:10.1112/plms/s2-30.1.264.

[57] Joel Rybicki and Jukka Suomela. Exact bounds for distributed graph colouring. In *Proc. 22nd International Symposium on Structural Information and Communication Complexity (SIROCCO 2015)*, volume 9439 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2015. doi:10.1007/978-3-319-25258-2. arXiv:1502.04963.

[58] Jukka Suomela. Survey of local algorithms. *ACM Computing Surveys*, 45 (2):24:1–40, 2013. doi:10.1145/2431211.2431223. http://www.cs.helsinki.fi/local-survey/.

[59] Jukka Suomela. Local coordination and symmetry breaking. *Bulletin of the EATCS*, 115:83–110, 2015. http://bulletin.eatcs.org/index.php/beatcs/issue/view/17.

[60] Masafumi Yamashita and Tsunehiko Kameda. Computing on anonymous networks: part I—characterizing the solvable cases. *IEEE Transactions on Parallel and Distributed Systems*, 7(1):69–89, 1996. doi:10.1109/71.481599.

Aalto-DD 239/2016

ISBN 978-952-60-7138-1 (printed)
ISBN 978-952-60-7137-4 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

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

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS