Towards Fully Automatic Distributed Lower Bounds

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

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

Fabian Kuhn
University of Freiburg, Germany

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

Joonatan Saarhelo
Unaffiliated, Helsinki, Finland

Abstract

In the past few years, a successful line of research has led to lower bounds for several fundamental local graph problems in the distributed setting. These results were obtained via a technique called *round elimination*. On a high level, the round elimination technique can be seen as a recursive application of a function that takes as input a problem Π and outputs a problem Π' that is one round easier than Π. Applying this function recursively to concrete problems of interest can be highly nontrivial, which is one of the reasons that has made the technique difficult to approach. The contribution of our paper is threefold.

Firstly, we develop a new and fully automatic method for finding so-called *fixed point relaxations* under round elimination. The detection of a non-0-round solvable fixed point relaxation of a problem Π immediately implies lower bounds of $\Omega(\log_\Delta n)$ and $\Omega(\log_\Delta \log n)$ rounds for deterministic and randomized algorithms for Π, respectively.

Secondly, we show that this automatic method is indeed useful, by obtaining lower bounds for defective coloring problems. More precisely, as an application of our procedure, we show that the problem of coloring the nodes of a graph with 3 colors and defect at most $(\Delta-3)/2$ requires $\Omega(\log_\Delta n)$ rounds for deterministic algorithms and $\Omega(\log_\Delta \log n)$ rounds for randomized ones. Additionally, we provide a simplified proof for an existing defective coloring lower bound. We note that lower bounds for coloring problems are notoriously challenging to obtain, both in general, and via the round elimination technique.

Both the first and (indirectly) the second contribution build on our third contribution: a new method to compute the one-round easier problem Π' in the round elimination framework. This method heavily simplifies the usage of the round elimination technique, and in fact it has been successfully exploited in a recent work in order to prove quantum advantage in the distributed setting [STOC '25].

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases round elimination, lower bounds, defective coloring

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.13

Related Version *Full Version*: https://arxiv.org/abs/2410.20224

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

1 Introduction

In the standard setting of distributed graph algorithms, known as the LOCAL model [29, 35], the nodes V of a graph $G = (V, E)$ communicate over the edges E of G in synchronous rounds. Initially, the nodes do not know anything about G (except for their own unique identifier and possibly some global parameters such as the number of nodes n or the maximum degree Δ) and at the end, each node must output its local part of the solution for the graph problem that needs to be solved. For example, if we intend to compute a vertex coloring of G, at the end, every node must output its own color in the final coloring. The time complexity of such a distributed algorithm is then measured as the number of rounds needed from the start until all nodes have terminated.

The study of the complexity of solving graph problems in the LOCAL model and in related distributed models has been a highly active area of research with a variety of substantial results over the last years. Apart from very significant and insightful new algorithmic results for distributed graph problems (e.g., [20, 19, 26, 36, 21, 25, 24]), the last ten years in particular also brought astonishing progress on proving lower bounds for distributed graph problems in the LOCAL model (e.g., [17, 19, 3, 5]). Essentially all of this recent progress on lower bounds has been obtained by a technique known as *round elimination*. The technique works for a class of problems known as *locally checkable* problems [33, 16]. Informally, a locally checkable problem is a problem for which there exists a constant-time algorithm satisfying the following: if a given solution is correct, the algorithm *accepts* on all nodes; otherwise, the algorithm *rejects* on at least one node. This class encompasses many of the most fundamental problems studied in the context of the LOCAL model, such as maximal matching, maximal independent set, and different variants of graph coloring.

Round Elimination. On a very high level, round elimination works as follows. Given a problem Π provided in the proper language, the round elimination framework provides a way to mechanically construct a problem $\Pi' = \hat{\mathcal{R}}(\Pi)$ that is exactly one round easier (under some mild assumptions). That is, if Π can be solved in R rounds, then Π' can be solved in $R - 1$ rounds (and vice versa).[1] For proving an R-round lower bound on problem Π, one then has to show that the problem $\hat{\mathcal{R}}^{(R-1)}(\Pi)$, or a relaxation of it, is not trivial, i.e., it cannot be solved in 0 rounds.

In its modern form, round elimination has first been used to show that the problems of computing a sinkless edge orientation or a Δ-vertex coloring of G require $\Omega(\log \log n)$ rounds with randomization and $\Omega(\log n)$ rounds deterministically [17, 19].[2] Subsequently Brandt [16] showed that round elimination can be applied to essentially every locally checkable problem and if a problem Π is specified in the right language, the problem $\hat{\mathcal{R}}(\Pi)$ can be computed in a fully automatic way. *Automatic round elimination* in the following led to a plethora of new distributed lower bounds. We next list some of the highlights. In [3], it was shown that even in regular trees, computing a maximal matching requires $\Omega(\min\{\Delta, \log_\Delta \log n\})$ rounds with randomized algorithms and $\Omega(\min\{\Delta, \log_\Delta n\})$ rounds with deterministic algorithms. Previously, the best known lower bound as a function of Δ for this problem was only

[1] Formally, round elimination has to be performed on a weaker version of the LOCAL model, which is known as the port numbering model. In the port numbering model, nodes do not have unique IDs, but they can distinguish their neighbors through different port numbers. Round elimination lower bounds in the port numbering model can then be lifted to lower bounds in the standard LOCAL model [5].

[2] We remark that although phrased differently, the classic proofs that 3-coloring a ring requires $\Omega(\log^* n)$ rounds [32, 29] can also be seen as round elimination proofs.

$\Omega(\log \Delta / \log\log \Delta)$ [28]. By a simple reduction, the same lower bound as for maximal matching also holds for computing a maximal independent set (MIS). In later work, the same lower bound was also proven directly for the MIS problem on trees and it was generalized in particular to the problems of computing ruling sets and of computing maximal matchings in hypergraphs, leading to tight (as a function of Δ) lower bounds for those problems [8, 4, 5, 7].

We emphasize that all the recent progress on developing new lower bounds for locally checkable problems in the LOCAL model has only been possible because the work of Brandt [16] describes an *automatic and generic* way to turn any locally checkable problem (given in the right formalism) into a locally checkable problem that is exactly one round easier. Moreover, for much of the progress, it was crucial that there exists efficient software as described by Olivetti in [34] that can be used to apply round elimination to concrete locally checkable problems.

Unfortunately, while round elimination has been extremely successful for proving many new lower bounds for computing locally checkable graph problems, the method has so far not been able to provide new lower bounds for many of the standard variants of *distributed graph coloring* and thus for some of the most important and most well-studied locally checkable problems. When applying round elimination to standard $(\Delta+1)$-coloring and related graph coloring problems, the descriptions of the problems in the sequence obtained by applying $\hat{\mathcal{R}}(\cdot)$ iteratively grow doubly exponential in *each* round elimination step (i.e., with each application of $\hat{\mathcal{R}}(\cdot)$) and thus even the one round easier problem $\hat{\mathcal{R}}(\Pi)$ often becomes too complex to understand.

We are convinced that in order to continue the present success story, further developing the existing automatic techniques will be indispensible and the main objective of this paper is to provide more efficient and more powerful methods for finding distributed lower bounds in an automatic fashion.

Distributed Coloring. As a concrete application, we aim to make progress towards obtaining lower bounds for distributed coloring problems. To achieve this, we consider the problem of computing a *d-defective c-coloring*. For two parameters c and d, a d-defective c-coloring of a graph $G=(V,E)$ is a partition of V into c color classes so that every node $v \in V$ has at most d neighbors that have the same color as v. Such colorings have become an important tool in many recent distributed coloring algorithms [15, 12, 13, 11, 14, 27, 6, 10, 23]. In [23], it is also argued that further progress on defective coloring algorithms might be key towards obtaining faster distributed $(\Delta+1)$-coloring algorithms and proving hardness results on distributed defective coloring algorithms might therefore also provide insights into understanding the hardness of the standard $(\Delta+1)$-coloring problem. To obtain proper colorings, defective colorings are commonly used as a subroutine in a recursive manner and to obtain efficient coloring algorithms using few colors, it would be particularly convenient to have algorithms that efficiently compute defective colorings with c colors and defect only $(1+o(1))\Delta/c$. Such defective colorings always exist [30] and efficient distributed algorithms for computing such colorings would immediately lead to faster $O(\Delta)$-coloring algorithms and potentially also to faster $(\Delta+1)$-coloring algorithms. In fact, a generalized variant of $(1+o(1))\Delta/2$-defective 2-colorings of line graphs have recently been used in a breakthrough result that obtains the first $\operatorname{poly}\log\Delta + O(\log^* n)$-round algorithm for computing a $(2\Delta-1)$-edge coloring of a graph [6].

In contrast, the best known algorithms for computing an $O(\Delta)$ or $(\Delta+1)$-vertex coloring require time polynomial in Δ [11, 22, 14, 31]. For vertex coloring, it is already known that computing $(1+o(1))\Delta/2$-defective 2-colorings requires $\Omega(\log n)$ rounds even in bounded-

degree graphs [9]. This raises the important question whether an increased number of $c > 2$ colors can admit the desired efficient $(1 + o(1))\Delta/c$-defective c-colorings. Already the case of $c = 3$ was wide open previous to our work and an important open problem in its own right: obtaining the desired efficient defective coloring algorithm for $c = 3$ would have fundamental consequences by improving the complexity of $O(\Delta)$-coloring (and of $(\Delta + 1)$-coloring if extendable to list defective colorings [23]), while proving a substantial lower bound for any such algorithm might pave the way for proving similar lower bounds for larger c in the future. As one of the main technical results of this paper, we show that computing $(1 + o(1))\Delta/3$-defective colorings (and in fact $(1 - o(1))\Delta/2$-defective colorings) with 3 colors requires $\Omega(\log n)$ rounds. We conjecture that a similar result should also hold for more than 3 colors and we hope that such a result can be proven by extending the techniques that we introduce in this paper.

1.1 Our Contributions and High-Level Ideas of Our Techniques

In the present paper, we take the task of automating round elimination and thus automating the search for distributed lower bounds one step further. In the following, we provide a high-level discussion of the contributions of the paper.

1.1.1 An Automatic Way of Generating Round Elimination Fixed Points

Chang, Kopelowitz, and Pettie [19] showed that in the LOCAL model, every locally checkable problem Π can either be solved deterministically in $f(\Delta) \cdot O(\log^* n)$ rounds (for some function $f(\cdot)$) or Π has a deterministic $\Omega(\log_\Delta n)$ and a randomized $\Omega(\log_\Delta \log n)$ lower bounds. In the following, we call problems of the first type *easy* problems and problems of the second type *hard* problems.

Fixed Points Imply Hardness Results. A particularly elegant way to prove that a problem is of the second type is through round elimination fixed points. A locally checkable problem Π is called a round elimination fixed point if $\hat{\mathcal{R}}(\Pi) = \Pi$, i.e., if the problem that is "one round easier" than Π is Π itself. We say that a problem Π is a *non-trivial fixed point* if Π is a round elimination fixed point that cannot be solved in 0 rounds. If a problem Π is a non-trivial fixed point, existing standard techniques directly imply that Π is a hard problem, i.e., that any deterministic LOCAL algorithm to solve Π requires at least $\Omega(\log_\Delta n)$ rounds and every randomized such algorithm requires at least $\Omega(\log_\Delta \log n)$ rounds (see, e.g., [5]). Moreover, we obtain the same lower bounds for Π if Π is not a fixed point itself but can be relaxed to a non-trivial fixed point $\tilde{\Pi}$. In fact, while interesting problems exist that are non-trivial fixed points themselves (see, e.g., [17]), finding a non-trivial *fixed point relaxation* $\tilde{\Pi}$ for Π (which we may simply call a *fixed point for* Π) is a more common way to prove lower bounds for a given problem Π (see, e.g., [2, 5, 7]). Furthermore, as shown in [5, 7], surprisingly, fixed points can also be used to prove lower bounds on the Δ-dependency of *easy* problems, i.e., problems that can be solved in time $f(\Delta) \cdot O(\log^* n)$.

Fixed Points Can Be Large. In order to understand the distributed complexity of locally checkable problems, we therefore need methods to find non-trivial fixed points for such problems in case such fixed points exist. We argue that, similarly to performing and analyzing round elimination, also finding new fixed points will in many cases require some automated support for searching for fixed points. Note that in general, even for a relatively simple problem Π with a small description, the smallest fixed point relaxation $\tilde{\Pi}$ of Π might

be much more complex and have a much larger description than the original problem Π. Consider for example the Δ-coloring problem in Δ-regular graphs. While the problem itself can be described[3] with Δ different labels and Δ different node configurations (one for each possible color), the round elimination fixed point for Δ-coloring that has been described in [5] consists of 2^Δ different labels and $2^\Delta - 1$ different node configurations (and no smaller fixed point for Δ-coloring is known or suspected to exist). Finding fixed points for problems that are not as symmetric and not as well-behaved as Δ-coloring might quickly become infeasible when it has to be done by hand, even when using the support of existing software for performing single round elimination steps.

Our Contribution: a Procedure for Finding Fixed Points Automatically. While the round elimination procedure is fully automatic in the sense that it gives a mechanical way to compute $\hat{\mathcal{R}}(\Pi)$ as a function of Π, finding fixed point relaxations is a *manual* process, where it is required to *guess* what is the right relaxation $\tilde{\Pi}$ of Π and then test if $\tilde{\Pi}$ is indeed a non-trivial fixed point.

One of the major open questions in the field is understanding whether one can automatically *decide* whether a given problem is easy or hard. Essentially, the only known way to prove that a problem is hard is producing a non-trivial fixed point relaxation of it, and it is not even known if all hard problems admit a non-trivial fixed point relaxation.

As our first main contribution, we make substantial progress on this question, by providing a method to *automatically* generate relaxations $\tilde{\Pi}$ of a given locally checkable problem Π that are *guaranteed to be* fixed points under the round elimination framework. While there is no formal guarantee that the resulting problem is non-trivial (and hence that the procedure succeeded in giving a lower bound), this procedure is able to automatically derive *all* fixed point relaxations that have been manually obtained in the past. Moreover, we additionally show that the procedure provides novel results.

Our Contribution: a First Simple Application of Our Procedure. As a first direct application we get a simpler proof of a result of [5]: by applying our method to the Δ-coloring problem, we directly obtain the Δ-coloring fixed point that was presented in [5].

1.1.2 Lower Bounds for Defective Coloring Problems

Understanding the complexity of $(\Delta+1)$-coloring is one of the major open questions in the field, and one of the very few techniques known to be able to prove lower bounds for local problems is round elimination. Unfortunately, $(\Delta+1)$-coloring, and many other variants of coloring, behave very badly in round elimination, in the sense that the problem Π' constructed via round elimination is typically doubly exponentially larger than the original problem Π. Understanding *defective* colorings seems to be one of the main obstacles that we need to overcome in order to make progress on $(\Delta+1)$-coloring, for different reasons:

- While we still cannot *understand* the resulting problem Π', we can *observe* that it is some variant of coloring that includes, as a subproblem, variants of *defective* coloring. Hence, in order to prove *lower bounds* for $(\Delta+1)$-coloring, we need to understand defective colorings first;
- In [23] it has been argued that improving defective coloring algorithms might be the key for improving *upper bounds* for $(\Delta+1)$-coloring.

[3] For an introduction to the description of problems, see Section 1.1.3 or Section 3.2.

For defective 2-coloring, hardness results are known [9]. However, such results do not say much about defective c-coloring for $c \geq 3$, because defective 2-coloring is special, in the sense that this problem seems to be much harder than the case $c \geq 3$. In fact, even on graphs where each node has some minimum degree equal to some large constant, even the seemingly simpler task of requiring each node of the graph to have at least *two* neighbors of a different color is hard. While this is not our main contribution regarding defective coloring, we show that our fixed point procedure is able to automatically obtain a non-trivial fixed point relaxation for defective 2-coloring.

The problem of computing a d-defective 3-coloring is much more interesting, since, differently from defective 2-coloring, for a large range of values of d, it is known to be *easy*. In our work, we make substantial progress on defective coloring, by proving that, for $d \leq \frac{\Delta-3}{2}$, the d-defective 3-coloring problem is *hard*. Such a result is obtained by applying our fixed point procedure on defective 3-coloring. However, the obtained fixed point is so large that we needed to introduce an additional technique to let a computer verify that the resulting problem is indeed non-trivial. We believe that the three new techniques that we introduce to obtain our result on defective 3-coloring (namely, a new way of applying round elimination, a way to automatically compute fixed points, and a way to let computers verify that an entire family of problems is non-trivial) will pave the way to proving lower bounds for d-defective c-coloring for $c > 3$. In the following, we give more details on our contributions on defective coloring.

Our Contribution: Defective 2-Coloring. Not many bounds on the complexity of defective colorings are known (we discuss known bounds in the full version of this paper). An exception is the case of defective colorings with 2 colors, which is understood. By computing an MIS (which can be done in $O(\Delta + \log^* n)$ rounds [15]) and assigning the MIS nodes one of the colors and the remaining nodes the other color, one obtains a $(\Delta - 1)$-defective 2-coloring of the graph. Interestingly, the problem becomes hard if we try to just go one step further: in [9], it was shown that computing a $(\Delta - 2)$-defective 2-coloring is a hard problem. This result has been shown via a reduction from the hardness of sinkless orientation. However, this reduction is based on the construction of virtual graphs on which the defective coloring algorithm is executed in order to obtain a sinkless orientation on the original graph, and in particular the lower bounds are not proved by providing a non-trivial fixed point. As a second application of our fixed point generation method, we show the following.

There exists a non-trivial fixed point relaxation for $(\Delta - 2)$-defective 2-coloring.

This result is significant in light of the fundamental open question stated in [18, 5] asking whether, for every locally checkable problem Π that has a deterministic $\Omega(\log_\Delta n)$ and a randomized $\Omega(\log_\Delta \log n)$ lower bound, such a lower bound can be proven via a round elimination fixed point. The $(\Delta - 2)$-defective 2-coloring problem was one of an only very small number of such problems for which previously no fixed point lower bound proof was known.

Our Contribution: Defective 3-Coloring. As a main application of our automatic fixed point procedure, we study the defective coloring problem with 3 colors. From the arbdefective coloring lower bound of [5], it is known that d-defective 3-coloring is hard if $3(d+1) \leq \Delta$ and thus if $d \leq \frac{\Delta}{3} - 1$. In [9], it was further shown that if $d \geq \frac{2\Delta-4}{3}$, d-defective 3-coloring can be solved in $O(\Delta + \log^* n)$ rounds. By using our fixed point method, we manage to partially close this gap by proving the following statement.

For $d \leq \frac{\Delta-3}{2}$, the d-defective 3-coloring problem is a hard problem, i.e., it requires $\Omega(\log_\Delta n)$ rounds deterministically and $\Omega(\log_\Delta \log n)$ rounds with randomization.

This in particular implies that there is no $(1+o(1))\Delta/3$-defective 3-coloring algorithm that violates those time lower bounds, thereby ruling out the possibility of using defective 3-coloring as an approach for attacking $O(\Delta)$ and $(\Delta+1)$-coloring in the manner outlined before Section 1.1.

We note that the fixed point that we automatically generate for this problem is highly non-trivial, and that manually proving that the fixed point that we provide is indeed a fixed point would require to perform a case analysis over hundreds of cases. For this reason, we do not manually prove that the fixed point that we provide is indeed a fixed point. Instead, we provide a way to automate this process, by reducing the problem of determining whether a problem is a fixed point to the problem of proving that certain systems of inequalities have no solution. The remaining task of showing that said systems have no solution can be performed automatically via computer tools. This automatization process provides a partial answer to Open Question 9 in [5].

1.1.3 A More Efficient Method for Performing Round Elimination

Another major contribution of our work is providing a new way of applying round elimination. In particular, we provide a new procedure for computing the problem $\hat{\mathcal{R}}(\Pi)$. We would like to point out that after our work appeared online, our procedure has already proved to be extremely helpful for writing proofs based on round elimination. In fact, our procedure has been used to prove quantum advantage in the distributed setting [1]. We now provide more details on this procedure, and how we modified it to obtain the fixed point procedure mentioned in Section 1.1.1.

As a first step, in our work, we first provide a novel way for computing a locally checkable problem Π' that is exactly one round easier than Π. Then, we show that, by applying such a procedure in a slightly modified way, instead of obtaining the problem Π', we obtain some problem $\tilde{\Pi}$ which is guaranteed to be a fixed point relaxation of Π. While in some cases the obtained problem $\tilde{\Pi}$ may be solvable in 0 rounds (i.e., this *must* be the case when applying the procedure on an *easy* problem), the results presented in Section 1.1.2 are obtained by proving that the fixed points that we get by applying the procedure on defective colorings are non-trivial.

While our new procedure for applying the round elimination technique has applications for finding fixed points, this procedure is interesting on its own. In order to better explain the reason, we first highlight the main issue of the standard way of applying round elimination. While for a given locally checkable problem Π, the framework of Brandt [16] gives a fully automatic way for computing a locally checkable problem Π' that is exactly one round easier than Π, this computation is in general not computationally efficient. To illustrate why, we somewhat informally sketch how round elimination works (for a formal description we refer to Section 3.3).

How Round Elimination Works. For the automatic round elimination framework, a locally checkable problem on a Δ-regular graph $G=(V,E)$ is formalized on the bipartite graph H between the nodes V and the edges E of G.[4] That is, H is obtained by adding an additional

[4] More generally, round elimination can be defined on biregular bipartite graphs or hypergraphs (see Section 3).

node in the middle of the edges in E. Each edge of G is thus split into 2 halfedges. A solution to a locally checkable problem is given by an assignment of labels from a finite alphabet Σ to all edges of H (i.e., to each halfedge of G). The validity of a solution is given by a set of allowed node and edge configurations, where a node configuration is a multiset of labels of size Δ and an edge configuration is a multiset of labels of size 2. One step of round elimination on G is done by performing two steps of round elimination on H (note that one round on G corresponds to two rounds on H). When starting from a node-centric problem Π (i.e., a problem where the nodes in H corresponding to *nodes* in G assign the labels to their incident half-edges), the first step transforms Π into an edge-centric problem Π' that is exactly one round easier on H and the second step transforms the problem into a node-centric problem Π'' that is one round easier than Π' on H and thus one round easier than Π on G. The label set Σ' of Π' is the power set 2^Σ of Σ and the label set Σ'' of Π'' is the power set of Σ'. The allowed edge configurations of Π' are, roughly speaking, the multisets $\{\mathsf{L}_1, \mathsf{L}_2\}$ of labels $\mathsf{L}_1, \mathsf{L}_2 \in \Sigma' = 2^\Sigma$ such that *for all* $\ell_1 \in \mathsf{L}_1$ and $\ell_2 \in \mathsf{L}_2$, $\{\ell_1, \ell_2\}$ is an allowed edge configuration of Π.[5] The allowed node configurations of Π' are all the multisets $\{\mathsf{L}_1, \dots, \mathsf{L}_\Delta\}$ of labels $\mathsf{L}_i \in \Sigma'$ (that appear in some allowed edge configuration of Π') such that *there exists* an allowed node configuration $\{\ell_1, \dots, \ell_\Delta\}$ with $\ell_i \in \mathsf{L}_i$ in problem Π. In the second step, Π'' is obtained in the same way from Π', but by exchanging the roles of nodes and edges. That is, in the second step, the "for all" quantifier is applied to the allowed node configurations and the "exists" quantifier is applied to the allowed edge configurations (of Π').

The Computationally Expensive Part. Note that from a computational point of view, it is mainly the application of the "for all" quantifier on the edge side when going from Π to Π' and even more importantly on the node side when going from Π' to Π'' that is challenging. When implemented naively, one has to iterate over all possible size-2 multisets of Σ' in the first step and over all possible size-Δ multisets of Σ'' in the second step. While in general, the problem Π'' that is one round easier than the original problem Π on G can be doubly exponentially larger than Π, for interesting problems this is often not the case. For such more well-behaved problems, the "for all" case can potentially be computed in a much more efficient way.

Our Contribution. As our final contribution, we give a new elegant way to perform the application of the "for all" quantifier in round elimination. The method makes use of the fact that often the node and edge configurations of a problem can be represented by a relatively small number of *condensed configurations*. A condensed node or edge configuration is a multiset $\{S_1, \dots, S_k\}$ (where $k = \Delta$ for nodes and $k = 2$ for edges) of sets $S_1, \dots, S_k \subseteq \Sigma$ of labels, representing the set of all configurations $\{\ell_1, \dots, \ell_k\}$ for which $\ell_i \in S_i$ for all $i \in \{1, \dots, k\}$. We prove that the "for all" part of round elimination can be performed by a simple process that consists of steps of the following kind. In each step, we take two condensed configurations of the current problem and we combine those condensed configurations in some way to generate new condensed configurations. We then remove redundant configurations and continue until such a step cannot generate any new condensed configurations. In the end, each condensed configuration $\{S_1, \dots, S_k\}$ is interpreted as a multiset of labels of the new problem. We formally define the process and prove its correctness in Section 4.

[5] In the formally precise definition of the set of edge configurations of Π' provided in Section 3.3, we'll refine this definition slightly.

Informally, we prove that each configuration of the resulting problem can be described as a binary tree, where leaf nodes are condensed configurations of the original problem, and each internal node of the tree is the configuration obtained by combining its two children.

Since our new procedure is mainly used as a tool for obtaining fixed points, we do not formally state the benefits of this new procedure. However, we informally highlight the following:

- The new procedure avoids the cost of enumerating all possible size-Δ multisets of Σ'' in the second step, and its running time only depends on the number of input configurations, output configurations, and the height of the aforementioned trees. Such trees have height at most $\Delta \cdot |\Sigma'|$, and we observe that, for many natural problems, the height is much smaller. We thus obtain that, for many problems of interest, the running time of the new procedure is output-sensitive.
- Thanks to the new procedure, we obtain that, in order to check whether a problem is a fixed point, it is sufficient to check whether the combination of pairs of condensed configurations does not create new configurations. While this drastically reduces the time complexity of checking whether a problem is a fixed point, this also makes it much easier to *prove* that a problem is a fixed point. In fact, in the latter case, it is sufficient to consider two configurations at a time, instead of going through an exponential number of cases. We point out that, even if some friendly oracle gave us the fixed point for defective 3-coloring (which we present in the full version of this paper), we believe that, without exploiting this new procedure, proving that such a problem is indeed a fixed point would not have been possible.

2 Road Map

Further related work. In the full version of this paper, we provide further related work. More in detail, we discuss existing fixed points, the surprising fact that fixed points can be used to prove lower bounds as a function of Δ, and defective coloring.

Preliminaries. In Section 3, we provide some preliminaries. We first define the model of computation and the language that we use to formally describe problems. Then, we describe the round elimination framework.

A new way of applying round elimination. On a high level, round elimination allows us to start from a problem Π and to compute a problem Π' that, under some assumptions, is exactly one round easier (in the distributed setting) than Π. As it will become clear in Section 3, computing Π' as a function of Π can be a tricky process. In Section 4 we provide a novel and simplified way to compute Π' as a function of Π, and in the full version of this paper we prove its correctness.

Fixed point generation. A problem Π' is a non-trivial fixed point relaxation of Π if it satisfies the following: Π' can be solved in 0 rounds if we are given a solution for Π; Π' cannot be solved in 0 rounds in the so-called port numbering model (see Section 3 for the definition of this model); By applying round elimination on Π', we obtain Π' itself.

It is known by prior work (see Theorem 1) that, if there exists a non-trivial fixed point relaxation for a problem Π, then Π requires $\Omega(\log_\Delta n)$ rounds for deterministic algorithms and $\Omega(\log_\Delta \log n)$ rounds for randomized ones. In the full version of this paper, we provide an automatic way to obtain non-trivial fixed point relaxations. More in detail, we provide a

procedure FixedPoint that takes as input a problem Π and an object D (called diagram), and it produces a problem Π' that is always guaranteed to be a fixed point. Whether such a fixed point is non-trivial depends on Π and on the choice of D.

Selecting the right diagram. As mentioned, the choice of the diagram D may affect the triviality of the obtained fixed point. In the full version of this paper, we first provide a generic way to construct a diagram as a function of Π, that we call *default diagram*. Then, we show possible ways to modify the default diagram in the case in which the fixed-point obtained with the default diagram is a trivial one.

An alternative proof for the hardness of Δ-coloring. In the full version of this paper, we show a first application of our fixed point procedure, by providing a non-trivial fixed point relaxation for the Δ-coloring problem. Such a fixed point was already shown in [5], but here we show a much easier proof. While this section is not the main contribution of our work, its main purpose is to warm-up the reader for what comes later.

An alternative proof for the hardness of defective 2-coloring. In the full version of this paper, we show another application of our fixed point procedure, by providing a non-trivial fixed point relaxation for the $(\Delta - 2)$-defective 2-coloring problem. This is one of the few problems for which an $\Omega(\log_\Delta n)$ lower bound is known by prior work [9], but a non-trivial fixed point relaxation for this problem was unknown. Whether a non-trivial fixed point relaxation exists for all problems that require $\Omega(\log_\Delta n)$ deterministic rounds is one of the major open questions about round elimination, and hence we make progress in understanding it. Again, this result is not the main contribution of our work, and its main purpose is to prepare the reader for what comes next.

Defective 3-coloring. In the full version of this paper, we use our fixed point procedure to show a lower bound for defective 3-coloring. While the proofs for Δ-coloring and defective 2-coloring require a relatively short case analysis, the proof for defective 3-coloring requires to analyze hundreds of cases. For this reason, we prove that such a case analysis can be performed automatically by using computer tools. In particular, we reduce the task of checking whether a given problem Π is the result of applying our fixed point procedure, to proving that all systems of inequalities belonging to a certain finite set have no solution, which can be checked automatically via computer tools.

Open questions. We present some open questions in the full version of this paper.

3 Preliminaries

3.1 The LOCAL Model

The computational model that we consider is the standard LOCAL model of distributed computing [29, 35], where the nodes V of a graph $G = (V, E)$ communicate over the edges E. More precisely, time is divided into synchronous rounds, and in each round each node can send an arbitrarily large message to each neighbor. Moreover, between sending messages, nodes can perform any internal computation on the information they gathered so far. In the beginning of the computation, each node v is aware of its own degree $\deg(v)$, and has an internal ordering of its incident edges represented by the *ports* $1, \ldots, \deg(v)$ being assigned

bijectively to v's incident edges. We also assume that each node is aware of the number n of nodes and the maximum degree Δ of the input graph. As we will prove lower bounds in this work, this assumption makes our results only stronger. Moreover, each node is equipped with some *symmetry-breaking information* to avoid trivial impossibilities: in the case of deterministic algorithms, each node is assigned some globally unique ID of length $O(\log n)$ bits; in the case of randomized algorithms, each node instead has access to an unlimited amount of private random bits. Each node executes the same algorithm that governs which messages a node sends (depending on the accumulated knowledge of the node) and what the node outputs at the end of the computation. Each node has to terminate at some point and then provide a local output; all local outputs together form the global solution to the problem. The (*round* or *time*) *complexity* of a distributed algorithm is the number of rounds until the last node terminates. In the randomized setting, as usual, the algorithms are required to be Monte-Carlo algorithms that produce a correct solution with high probability, i.e., with probability at least $1 - 1/n$.

While the lower bounds we prove hold in the LOCAL model, for technical reasons we will also make use of the *port numbering* model along the way. The (deterministic) port numbering model is the same as the deterministic LOCAL model apart from two differences:

1. No symmetry-breaking information is provided, i.e., nodes are not equipped with IDs.
2. For each hyperedge e, a total order on the set of incident nodes is provided (which can be formalized via a bijection between this node set and the set $\{1, \dots, k\}$, where k denotes the number of nodes contained in e).

The second difference can be seen as an analog (on the hyperedge side) of the port numbers via which the nodes can distinguish between incident hyperedges.

3.2 Problems

The problems we study in this work fall into the class of *locally checkable problems*. Locally checkable problems are problems that can be defined via local constraints and encompass the vast majority of problems studied in the LOCAL model. A modern formalism to define these problems is given by the so-called black-white formalism that we will also use in this paper. In fact, as we will see, this formalism captures locally checkable problems not only on graphs, but more generally on hypergraphs (where we will denote the maximum number of nodes in a hyperedge by δ). In the full version of this paper we provide an example illustrating (some of) the definitions provided in this section.

The black-white formalism. In the black-white formalism, a locally checkable problem is given as a triple $\Pi = (\Sigma_\Pi, \mathcal{N}_\Pi, \mathcal{E}_\Pi)$. Here, Σ_Π is a finite set of elements, called *labels*, $\mathcal{N}_\Pi = (\mathcal{N}_1, \dots, \mathcal{N}_\Delta)$ and $\mathcal{E}_\Pi = (\mathcal{E}_1, \dots, \mathcal{E}_\delta)$, where each $\mathcal{N}_i$ and $\mathcal{E}_i$ is a collection of multisets of cardinality i with labels from Σ_Π. We call $\mathcal{N}_\Pi$ the *node constraint* of Π and $\mathcal{E}_\Pi$ the *edge constraint* of Π. On a hypergraph, a correct solution for Π is an assignment of labels from Σ_Π to the incident node-hyperedge pairs such that for each node v, the multiset of labels corresponding to v is contained in $\mathcal{N}_{\deg(v)}$, and analogously for hyperedges w.r.t. the respective $\mathcal{E}_i$. More formally, let $\mathcal{F}$ denote the set of pairs (v, e) where e is a hyperedge incident to v. A correct solution for Π on a hypergraph $G = (V, E)$ is a mapping $f \colon \mathcal{F} \to \Sigma_\Pi$ such that, for each $v \in V$, we have $\{f(v, e') \mid e' \ni v\} \in \mathcal{N}_{\deg(v)}$, and, for each $e \in E$, we have $\{f(v', e) \mid v' \in e\} \in \mathcal{E}_{\mathrm{rank}(e)}$. Here, the *rank* $\mathrm{rank}(e)$ of a hyperedge e is the number of nodes contained in e, and the displayed sets are to be understood as multisets.

When solving a locally checkable problem in the distributed setting, each node v has to output one label for each "incident" node-hyperedge pair in $\mathcal{F}$ such that the induced global solution is correct. While the improvements for the general round elimination technique

(discussed below) that we will obtain in this work apply to the general hypergraph setting, for the results about concrete problems that we provide we can restrict attention to the special case of graphs. In this special case, each hyperedge is of rank 2, and consequently we will replace the edge constraint $(\mathcal{E}_1, \ldots, \mathcal{E}_\delta)$ by $\mathcal{E}_2$. Moreover, to simplify notation, in this case, we will set $\mathcal{E} := \mathcal{E}_2$.

We remark that besides providing a formalism for graphs by considering them as a special case of hypergraphs, the black-white formalism provides a (different) way to encode and study problems on *bipartite* graphs, by identifying the "black" nodes in the bipartition with the *nodes* in the above formalism, and the "white" nodes with the *hyperedges*. This relation to bipartite graphs is also where the name "black-white formalism" comes from.

As can be observed, the definition of the problems in this formalism depends on Δ (and δ), which provides the power to also describe important problems like $(\Delta+1)$-coloring in this formalism. If we are to be very precise, in this formalism each problem is a collection of problems indexed by Δ (and, if considered on hypergraphs, δ). Throughout the paper, we implicitly assume that some (arbitrary) Δ (and, if required, some δ) is fixed. Note that this does not impact the generality of our results.

Finally, we remark that, for simplicity, we consider two locally checkable problems given in the black-white formalism as identical if one can be obtained from the other by renaming the labels used to describe the latter.

Configurations. We will use the term *configuration* to refer to a multiset of labels, and write it in either of the two equivalent forms $\{\ell_1, \ldots, \ell_i\}$ and $\ell_1 \ldots \ell_i$. Note that the order of the ℓ_j does not matter (also in the second form): all configurations that can be obtained from a configuration by reordering are considered to be the same configuration. When referring to the multiset of labels assigned to the pairs (v, e') incident to a fixed node v, we will use the term *node configuration*; when referring to the multiset of labels assigned to the pairs (v', e) corresponding to a fixed (hyper)edge e, we will use the term *edge configuration*. Moreover, for simplicity we may slightly abuse notation by writing $\{\ell_1, \ldots, \ell_i\} \in \mathsf{L}_1 \times \cdots \times \mathsf{L}_i$ if $\mathsf{L}_1, \ldots, \mathsf{L}_i$ are sets containing the labels $\ell_1, \ldots, \ell_i$, respectively.

It will be convenient to refer to certain collections of configurations in a condensed manner. A *condensed configuration* $\mathcal{C}$ is a configuration $\{\mathsf{L}_1, \ldots, \mathsf{L}_i\}$ of sets of labels. Configuration $\mathcal{C}$ is to be understood as the set of all configurations $\{\ell_1, \ldots, \ell_i\} \in \mathsf{L}_1 \times \cdots \times \mathsf{L}_i$ (though we will also consider the condensed configuration $\mathcal{C}$ as a configuration of sets when convenient). To indicate that a configuration of sets represents a condensed configuration, we will often write each set in the configuration in the form $[\ell_1 \ \ldots \ \ell_j]$ (unless the set only contains one element ℓ, in which case we will simply write the set as ℓ).

Diagrams. A useful way of capturing certain aspects of problems is via so-called diagrams. A *diagram* $D = (\Sigma_D, E_D)$ is nothing else than a directed acyclic graph with node set Σ_D and edge set E_D. The *edge diagram* of a problem $\Pi = (\Sigma_\Pi, \mathcal{N}_\Pi, \mathcal{E}_\Pi)$ is the diagram D obtained by setting $\Sigma_D := \Sigma_\Pi$ and defining E_D as the set of those directed edges (ℓ, ℓ') that satisfy that $\ell' \neq \ell$ and, for every configuration $\{\ell_1, \ldots, \ell_\delta\} \in \mathcal{E}_\Pi$ with $\ell_i = \ell$ for some $1 \leq i \leq \delta$, also $\{\ell_1, \ldots, \ell_{i-1}, \ell', \ell_{i+1}, \ldots, \ell_\delta\} \in \mathcal{E}_\Pi$. When displaying a diagram, we often omit arrows that can be obtained as the composition of displayed arrows. We call a subset $S \subseteq \Sigma_D$ *right-closed* (w.r.t. D) if, for any edge $(\ell, \ell') \in E_D$, $\ell \in S$ implies $\ell' \in S$.

3.3 The Round Elimination Technique

In this section, we give a formal introduction to round elimination. As some of the definitions provided in this section are fairly technical, the reader is encouraged to consult the illustrating example provided in the full version of this paper alongside reading the definitions.

For technical reasons, round elimination requires the considered input (hyper)graphs to be regular (and uniform). As such, we will assume throughout the paper that every node of the input (hyper)graph has the same degree Δ and every (hyper)edge has the same rank δ (which, in the case of graphs, is simply 2). This also simplifies the representation of locally checkable problems $\Pi = (\Sigma_\Pi, \mathcal{N}_\Pi, \mathcal{E}_\Pi)$: now we can assume that $\mathcal{N}_\Pi$ and $\mathcal{E}_\Pi$ are collections of multisets of cardinalities Δ and δ, respectively, instead of sequences of similar collections. Note that, as we will prove lower bounds in this work, the inherent restriction to regular graphs makes our results only stronger.

$\mathcal{R}(\cdot)$ and $\overline{\mathcal{R}}(\cdot)$. At the heart of the round elimination technique lie the round elimination operators $\mathcal{R}$ and $\overline{\mathcal{R}}$, which are functions that take a locally checkable problem in the black-white formalism as input and return such a problem. More precisely, for a locally checkable problem $\Pi = (\Sigma_\Pi, \mathcal{N}_\Pi, \mathcal{E}_\Pi)$, the locally checkable problem $\mathcal{R}(\Pi) = (\Sigma_{\mathcal{R}(\Pi)}, \mathcal{N}_{\mathcal{R}(\Pi)}, \mathcal{E}_{\mathcal{R}(\Pi)})$ is defined as follows.

The label set $\Sigma_{\mathcal{R}(\Pi)}$ of $\mathcal{R}(\Pi)$ is simply the set of non-empty subsets of Σ_Π, i.e., $\Sigma_{\mathcal{R}(\Pi)} := 2^{\Sigma_\Pi} \setminus \{\{\}\}$. For the definition of the edge constraint $\mathcal{E}_{\mathcal{R}(\Pi)}$ of $\mathcal{R}(\Pi)$, we need the notion of a *maximal configuration*. Let $\mathcal{Z}$ be a collection of configurations of sets of labels. Then, a configuration $\mathsf{L}_1 \dots \mathsf{L}_i \in \mathcal{Z}$ is maximal (in $\mathcal{Z}$) if there is no configuration $\mathsf{L}'_1 \dots \mathsf{L}'_i \in \mathcal{Z}$ (of the same length) such that there exists a bijection $\phi\colon \{1, \dots, i\} \to \{1, \dots, i\}$ satisfying $\mathsf{L}_j \subseteq \mathsf{L}'_{\phi(j)}$ for all $1 \le j \le i$ and $\mathsf{L}_j \subsetneq \mathsf{L}'_{\phi(j)}$ for at least one $1 \le j \le i$. In other words, a configuration of sets is maximal if no other configuration in the considered configuration space can be reached by enlarging (some of) the sets (and reordering the sets).

Now we can define $\mathcal{E}_{\mathcal{R}(\Pi)}$ as follows. Let $\mathcal{E}$ denote the collection of all configurations $\mathsf{L}_1 \dots \mathsf{L}_\delta$ such that $\mathsf{L}_1, \dots, \mathsf{L}_\delta \in \Sigma_{\mathcal{R}(\Pi)}$ and for all choices $(\ell_1, \dots, \ell_\delta) \in \mathsf{L}_1 \times \dots \times \mathsf{L}_\delta$ of labels we have $\{\ell_1, \dots, \ell_\delta\} \in \mathcal{E}_\Pi$. Then, $\mathcal{E}_{\mathcal{R}(\Pi)}$ is obtained from $\mathcal{E}$ by removing all configurations that are not *maximal* in $\mathcal{E}$. Finally, the node constraint $\mathcal{N}_{\mathcal{R}(\Pi)}$ of $\mathcal{R}(\Pi)$ is defined as the collection of all configurations $\mathsf{L}_1 \dots \mathsf{L}_\Delta$ such that each L_i appears in at least one configuration from $\mathcal{E}_{\mathcal{R}(\Pi)}$ and there exists a choice $(\ell_1, \dots, \ell_\Delta) \in \mathsf{L}_1 \times \dots \times \mathsf{L}_\Delta$ of labels satisfying $\{\ell_1, \dots, \ell_\Delta\} \in \mathcal{N}_\Pi$.

The problem $\overline{\mathcal{R}}(\Pi) = (\Sigma_{\overline{\mathcal{R}}(\Pi)}, \mathcal{N}_{\overline{\mathcal{R}}(\Pi)}, \mathcal{E}_{\overline{\mathcal{R}}(\Pi)})$ is defined dually to $\mathcal{R}(\Pi)$, where the role of nodes and hyperedges are reversed. More precisely, we have the following. As before, $\Sigma_{\overline{\mathcal{R}}(\Pi)} = \Sigma_{\mathcal{R}(\Pi)} = 2^{\Sigma_\Pi} \setminus \{\{\}\}$. The node constraint $\mathcal{N}_{\overline{\mathcal{R}}(\Pi)}$ of $\overline{\mathcal{R}}(\Pi)$ is the collection of *maximal* configurations $\mathsf{L}_1 \dots \mathsf{L}_\Delta$ such that $\mathsf{L}_1, \dots, \mathsf{L}_\Delta \in \Sigma_{\overline{\mathcal{R}}(\Pi)}$ and for all choices $(\ell_1, \dots, \ell_\Delta) \in \mathsf{L}_1 \times \dots \times \mathsf{L}_\Delta$ of labels we have $\{\ell_1, \dots, \ell_\Delta\} \in \mathcal{N}_\Pi$. The edge constraint $\mathcal{E}_{\overline{\mathcal{R}}(\Pi)}$ of $\overline{\mathcal{R}}(\Pi)$ is the collection of all configurations $\mathsf{L}_1 \dots \mathsf{L}_\delta$ such that each L_i appears in at least one configuration from $\mathcal{N}_{\overline{\mathcal{R}}(\Pi)}$ and there exists a choice $(\ell_1, \dots, \ell_\delta) \in \mathsf{L}_1 \times \dots \times \mathsf{L}_\delta$ of labels satisfying $\{\ell_1, \dots, \ell_\delta\} \in \mathcal{E}_\Pi$.

We will refer to the operation of deriving $\mathcal{E}_{\mathcal{R}(\Pi)}$ from $\mathcal{E}_\Pi$ (and $\mathcal{N}_{\overline{\mathcal{R}}(\Pi)}$ from $\mathcal{N}_\Pi$) as *applying the universal quantifier (to $\mathcal{E}_\Pi$ and $\mathcal{N}_\Pi$, respectively)* and say that a problem *satisfies the universal quantifier* if it is the result of such an operation.

The hard part in computing $\mathcal{R}(\Pi)$ and $\overline{\mathcal{R}}(\Pi)$ is applying the universal quantifier. In fact, consider the problem $\mathcal{R}(\Pi)$. There is an easy way to compute $\mathcal{N}_{\mathcal{R}(\Pi)}$, that is the following. Start from all the configurations in $\mathcal{N}_\Pi$, and for each configuration add to $\mathcal{N}_{\mathcal{R}(\Pi)}$ the condensed configuration obtained by replacing each label ℓ by the set that contains all label sets in $\Sigma_{\mathcal{R}(\Pi)}$ containing ℓ.

The round elimination sequence. In the round elimination framework, the two operators $\mathcal{R}$ and $\overline{\mathcal{R}}$ are used to define a sequence of problems that is essential for obtaining complexity lower bounds via round elimination. This sequence $\Pi_0, \Pi_1, \Pi_2, \ldots$ is defined via $\Pi_{i+1} := \overline{\mathcal{R}}(\mathcal{R}(\Pi_i))$ for all $i \geq 0$, where Π_0 is the given problem of interest. The following theorem provides a way to obtain lower bounds for the complexity of Π_0 via analyzing the 0-round-solvability of the problems in the sequence. It is a simplified version of Theorem 7.1 from [5].

▶ **Theorem 1.** *Let $\Pi_0, \Pi_1, \ldots, \Pi_t$ be a sequence of problems satisfying $\Pi_{i+1} = \overline{\mathcal{R}}(\mathcal{R}(\Pi_i))$ for all $0 \leq i \leq t-1$. Moreover, let B be an integer (that may depend on n and/or Δ) such that $|\Sigma_{\Pi_i}| \leq B$ for all $0 \leq i \leq t$, and $|\Sigma_{\mathcal{R}(\Pi_i)}| \leq B$ for all $0 \leq i \leq t-1$. Then, if Π_t is not 0-round-solvable in the port numbering model, Π_0 has lower bounds of $\Omega(\min\{t, \log_\Delta n - \log_\Delta \log B\})$ rounds in the deterministic* LOCAL *model and $\Omega(\min\{t, \log_\Delta \log n - \log_\Delta \log B\})$ rounds in the randomized* LOCAL *model.*

Fixed points. As implied by Theorem 1, it is crucial for proving lower bounds via round elimination to be able to determine the 0-round solvability of problems in the round elimination sequence produced by the studied problem Π_0. A class of problems that produces very simple sequences are so-called fixed points. A locally checkable problem Π is called a *fixed point* if $\overline{\mathcal{R}}(\mathcal{R}(\Pi)) = \Pi$. Moreover, for a fixed point Π, the problem $\Pi' := \mathcal{R}(\Pi)$ is called the *intermediate problem*. Note that such an intermediate problem Π' satisfies $\mathcal{R}(\overline{\mathcal{R}}(\Pi')) = \Pi'$. We get the following corollary from Theorem 1.

▶ **Corollary 2.** *Let Π be a fixed point in the round elimination framework. Then, if Π is not 0-round-solvable in the port numbering model, Π has lower bounds of $\Omega(\log_\Delta n)$ rounds in the deterministic* LOCAL *model and $\Omega(\log_\Delta \log n)$ rounds in the randomized* LOCAL *model.*

0-round-solvability. Due to Theorem 1, we are interested in determining whether a problem can be solved in 0 rounds or not. For technical reasons, throughout the paper, whenever we consider the 0-round-solvability of a problem, we will consider it in the *port numbering model*. In the port numbering model, 0-round-solvability admits a simple characterization: a problem Π is 0-round-solvable if and only if there is a configuration $\ell_1 \ \ldots \ \ell_\Delta \in \mathcal{N}_\Pi$ such that, for any δ (not necessarily distinct) labels $\ell'_1, \ldots, \ell'_\delta \in \{\ell_1, \ldots, \ell_\Delta\}$, it holds that $\ell'_1 \ \ldots \ \ell'_\delta \in \mathcal{E}_\Pi$. We will use the terms *trivial* and *non-trivial* to refer to 0-round-solvable and non-0-round-solvable problems, respectively. In particular, we will be interested in trivial and non-trivial fixed points.

For the interested reader, in the full version of this paper we provide an example of the application of the round elimination procedure to a simple problem called sinkless orientation.

4 A New Way of Applying Round Elimination

In this section, we describe a novel and simple way for applying the round elimination technique. As already discussed in Section 3.3, the hard and error-prone part in applying the $\mathcal{R}(\cdot)$ and $\overline{\mathcal{R}}(\cdot)$ operators consists in applying the universal quantifier. Let $\Pi = (\Sigma_\Pi, \mathcal{N}_\Pi, \mathcal{E}_\Pi)$ be the problem of interest, where $\mathcal{N}_\Pi$ contains multisets of size Δ and $\mathcal{E}_\Pi$ contains multisets of size δ. Also, let $\Pi' = \mathcal{R}(\Pi) = (\Sigma_{\Pi'}, \mathcal{N}_{\Pi'}, \mathcal{E}_{\Pi'})$. Recall that applying the universal quantifier means computing $\mathcal{E}_{\Pi'}$ as follows. First, let $\mathfrak{C}$ be the maximal set such that for all $\mathsf{L}_1 \ \ldots \ \mathsf{L}_\delta \in \mathfrak{C}$ it holds that, for all i, $\mathsf{L}_i \in 2^{\Sigma_\Pi} \setminus \{\{\}\}$, and all multisets $\{\ell_1, \ldots, \ell_\delta\} \in \mathsf{L}_1 \times \ldots \times \mathsf{L}_\delta$ are in $\mathcal{E}_\Pi$. Then, $\mathcal{E}_{\Pi'}$ is obtained by removing all non-maximal configurations from $\mathfrak{C}$. This definition, if implemented in a naive manner, requires considering all possible configurations from labels in 2^{Σ_Π}, and then, for each of them, checking if all possible configurations obtained by selecting one label from each set in the configuration are contained in $\mathcal{E}_\Pi$.

4.1 A new way to compute $\mathcal{E}_{\Pi'}$

We show a drastically simplified way of applying the universal quantifier, that, at each point in time, requires to consider only two configurations and to perform elementary operations on those.

Input of the new procedure. While, formally, the given constraint $\mathcal{E}_\Pi$ is described as a set of multisets, in some cases the given constraint is described in a more compact form, that is, by providing *condensed configurations*. The procedure that we describe does not need to unpack condensed configurations into a set of non-condensed ones, and this feature allows to apply this new procedure more easily. For this reason, we assume that $\mathcal{E}_\Pi$ is described as a set Γ_Π of condensed configurations, that is, Γ_Π contains multisets, where each multiset $\mathcal{L} \in \Gamma_\Pi$ is of the form $\{\mathsf{L}_1, \ldots, \mathsf{L}_\delta\}$, and for all $1 \leq i \leq \delta$ it holds that $\mathsf{L}_i \subseteq \Sigma_\Pi$. Clearly, if we are given $\mathcal{E}_\Pi$ as a list of non-condensed configurations, we can convert it into this form by replacing each label with a singleton set. While we assume that the input is described as a set of condensed configurations, the output of the procedure is going to be a set of non-condensed configurations. We call the condensed configurations in Γ_Π *input configurations*.

Combining configurations. At the heart of our procedure lies an operation that *combines* two given configurations of sets. We now formally define what it means to combine two such configurations. Let $\mathcal{L} = \{\mathsf{L}_1, \ldots, \mathsf{L}_\delta\}$ and $\mathcal{L}' = \{\mathsf{L}'_1, \ldots, \mathsf{L}'_\delta\}$ be two configurations, where L_i and L'_i are sets. Let $\phi\colon \{1, \ldots, \delta\} \to \{1, \ldots, \delta\}$ be a bijection, i.e., a permutation of $\{1, \ldots, \delta\}$. Let $u \in \{1, \ldots, \delta\}$. Combining $\mathcal{L}$ and $\mathcal{L}'$ w.r.t. ϕ and u means constructing the configuration $\mathcal{C} = \{\mathsf{C}_1, \ldots, \mathsf{C}_\delta\}$ where $\mathsf{C}_i = \mathsf{L}_i \cup \mathsf{L}'_{\phi(i)}$ if $i = u$ and $\mathsf{C}_i = \mathsf{L}_i \cap \mathsf{L}'_{\phi(i)}$ otherwise. In other words, we consider an arbitrary perfect matching between the sets of the two configurations, and we take the union for one matched pair and the intersection for the remaining matched pairs. In Figure 1, we show an example of a combination of two configurations.

The New Procedure. In the following, we construct a sequence (Ψ_i) of sets of configurations until certain desirable properties are obtained. The first step of the procedure is setting $\Psi_0 = \Gamma_\Pi$. The next step is to apply a subroutine that creates Ψ_{i+1} as a function of Ψ_i, and this subroutine is repeatedly applied until we get that $\Psi_{i+1} = \Psi_i$. Let the final result be $\mathcal{E}^*_{\Pi'}$.

The subroutine computes all possible combinations of pairs of configurations (including a configuration with itself) that are in Ψ_i, for all possible permutations ϕ and for all possible choices of u. If a resulting configuration contains an empty set, the configuration is discarded. Let Ψ_{i+1} be the set of configurations obtained by starting from the configurations in Ψ_i, adding the newly computed configurations, and then removing the non-maximal ones. We call the defined procedure NewRE, which is described more formally in Algorithm 1.

In the full version of this paper, we will first provide an example of execution of our new procedure, and then we will prove that the constraint $\mathcal{E}^*_{\Pi'}$ returned by NewRE is equal to the constraint $\mathcal{E}_{\Pi'}$ as defined according to the definition of round elimination given in Section 3. Finally, we will also prove that the procedure always terminates.

$$\{\mathsf{I},\mathsf{O}\} \cup \{\mathsf{O}\} = \{\mathsf{I},\mathsf{O}\}, \qquad \{\mathsf{I},\mathsf{O}\} \cap \{\mathsf{I},\mathsf{O}\} = \{\mathsf{I},\mathsf{O}\}, \qquad \{\mathsf{O}\} \cap \{\mathsf{I},\mathsf{O}\} = \{\mathsf{O}\}$$

Figure 1 One possible way to combine $\{\mathsf{I},\mathsf{O}\}\,\{\mathsf{I},\mathsf{O}\}\,\{\mathsf{O}\}$ with itself. The resulting configuration is $\{\mathsf{I},\mathsf{O}\}\,\{\mathsf{I},\mathsf{O}\}\,\{\mathsf{O}\}$.

Algorithm 1 The new procedure.

```
▷ Applies the procedure to the input configurations Γ ◁
procedure NewRe(Γ, δ)
    Ψ_0 ← Γ
    for i ← 0, 1, 2, ... do
        Ψ ← Ψ_i
        for all 𝓛 ∈ Ψ_i do
            for all 𝓛' ∈ Ψ_i do
                for all permutations φ over the integers {1, ..., δ} do
                    for all 1 ≤ u ≤ δ do
                        𝓒 ← Combine(𝓛, 𝓛', δ, φ, u)
                        if {} ∉ 𝓒 then
                            Ψ ← Ψ ∪ {𝓒}
        Ψ_{i+1} ← DiscardNonMaximal(Ψ)
        if Ψ_{i+1} = Ψ_i then
            break
    return Ψ_i
▷ Combines two configurations w.r.t. a given permutation φ and position u ◁
procedure Combine(𝓛 = {L_1, ..., L_δ}, 𝓛' = {L'_1, ..., L'_δ}, δ, φ, u)
    for i ← 1, ..., δ do
        if i = u then
            C_i = L_i ∪ L'_{φ(i)}
        else
            C_i = L_i ∩ L'_{φ(i)}
    𝓒 ← {C_1, ..., C_δ}
    return 𝓒
▷ Returns the set of maximal configurations of Ψ ◁
procedure DiscardNonMaximal(Ψ)
    S ← {}
    for all 𝓛 ∈ Ψ do
        if ¬(∃𝓛' ∈ Ψ s.t. 𝓛' ≠ 𝓛 and Dominates(𝓛', 𝓛)) then
            S ← S ∪ {𝓛}
    return S
▷ Determines whether 𝓛' dominates 𝓛 ◁
procedure Dominates(𝓛' = {L'_1, ..., L'_δ}, 𝓛 = {L_1, ..., L_δ})
    return ∃ permutation φ such that, for all 1 ≤ i ≤ δ, L_i ⊆ L'_{φ(i)}
```

References

1 Alkida Balliu, Sebastian Brandt, Xavier Coiteux-Roy, Francesco D'Amore, Massimo Equi, Francois Le Gall, Henrik Lievonen, Augusto Modanese, Dennis Olivetti, Marc-Olivier Renou, Jukka Suomela, Lucas Tendick, and Isadora Veeren. Distributed quantum advantage for local problems. In *Proceedings of the 57th Annual ACM SIGACT Symposium on Theory of Computing, (STOC 2025)*, 2025.

2 Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. In *Proc. 34th Symp. on Distributed Computing (DISC)*, 2020.

3 Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. In *Proc. 60th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 481–497, 2019. doi:10.1109/FOCS.2019.00037.

4 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Improved distributed lower bounds for MIS and bounded (out-)degree dominating sets in trees. In *Proc. 40th ACM Symposium on Principles of Distributed Computing (PODC)*, 2021. doi:10.1145/3465084.3467901.

5 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed Δ-coloring plays hide-and-seek. In *54th ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 464–477, 2022.

6 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed edge coloring in time polylogarithmic in Δ. In *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, pages 15–25. ACM, 2022. doi:10.1145/3519270.3538440.

7 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed maximal matching and maximal independent set on hypergraphs. In *Proc. 34th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 2632–2676, 2023. doi:10.1137/1.9781611977554.CH100.

8 Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed lower bounds for ruling sets. In *Proc. 61st IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 365–376, 2020. doi:10.1109/FOCS46700.2020.00042.

9 Alkida Balliu, Juho Hirvonen, Christoph Lenzen, Dennis Olivetti, and Jukka Suomela. Locality of not-so-weak coloring. In *Structural Information and Communication Complexity - 26th International Colloquium, SIROCCO 2019, L'Aquila, Italy, July 1-4, 2019, Proceedings*, volume 11639 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2019. doi:10.1007/978-3-030-24922-9_3.

10 Alkida Balliu, Fabian Kuhn, and Dennis Olivetti. Distributed edge coloring in time quasi-polylogarithmic in delta. In *Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 289–298, 2020. doi:10.1145/3382734.3405710.

11 Leonid Barenboim. Deterministic (Δ+1)-Coloring in Sublinear (in Δ) Time in Static, Dynamic, and Faulty Networks. *Journal of ACM*, 63(5):1–22, 2016. doi:10.1145/2979675.

12 Leonid Barenboim and Michael Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Comput.*, 22:363–379, 2010. doi:10.1007/s00446-009-0088-2.

13 Leonid Barenboim and Michael Elkin. Deterministic distributed vertex coloring in polylogarithmic time. *Journal of ACM*, 58:23:1–23:25, 2011. doi:10.1145/2027216.2027221.

14 Leonid Barenboim, Michael Elkin, and Uri Goldenberg. Locally-iterative distributed $(\Delta + 1)$-coloring below Szegedy-Vishwanathan barrier, and applications to self-stabilization and to restricted-bandwidth models. In *Proc. 37th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 437–446, 2018. doi:10.1145/3212734.3212769.

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

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

17 Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed Lovász local lemma. In *Proceedings of the 48th ACM Symposium on Theory of Computing (STOC 2016)*, pages 479–488. ACM Press, 2016. doi:10.1145/2897518.2897570.

18 Sebastian Brandt and Dennis Olivetti. Truly tight-in-Δ bounds for bipartite maximal matching and variants. In *Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 69–78, 2020. doi:10.1145/3382734.3405745.

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

20 Yi-Jun Chang, Wenzheng Li, and Seth Pettie. An optimal distributed (Δ+1)-coloring algorithm? In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, (STOC 2018)*, pages 445–456, 2018. doi:10.1145/3188745.3188964.

21 Salwa Faour, Mohsen Ghaffari, Christoph Grunau, Fabian Kuhn, and Václav Rozhon. Local distributed rounding: Generalized to mis, matching, set cover, and beyond. In *Proc. 34th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 4409–4447, 2023. doi:10.1137/1.9781611977554.CH168.

22 Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local conflict coloring. In *Proc. 57th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 625–634, 2016. doi:10.1109/FOCS.2016.73.

23 Marc Fuchs and Fabian Kuhn. List defective colorings: Distributed algorithms and applications. In Rotem Oshman, editor, *Proc. 37th Int. Symp. on Distributed Computing (DISC)*, volume 281 of *LIPIcs*, pages 22:1–22:23, 2023. doi:10.4230/LIPICS.DISC.2023.22.

24 Mohsen Ghaffari and Christoph Grunau. Near-optimal deterministic network decomposition and ruling set, and improved MIS. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2148–2179. IEEE, 2024. doi:10.1109/FOCS61266.2024.00007.

25 Mohsen Ghaffari, Christoph Grunau, Bernhard Haeupler, Saeed Ilchi, and Václav Rozhon. Improved distributed network decomposition, hitting sets, and spanners, via derandomization. In *Proc. 34th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 2532–2566, 2023. doi:10.1137/1.9781611977554.CH97.

26 Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In *Proc. 59th Symp. on Foundations of Computer Science (FOCS)*, pages 662–673, 2018. doi:10.1109/FOCS.2018.00069.

27 Fabian Kuhn. Faster deterministic distributed coloring through recursive list coloring. In *Proc. 32st ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1244–1259, 2020. doi:10.1137/1.9781611975994.76.

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

29 Nathan Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.

30 L. Lovász. On decompositions of graphs. *Studia Sci. Math. Hungar.*, 1:237–238, 1966.

31 Yannic Maus and Tigran Tonoyan. Local conflict coloring revisited: Linial for lists. In *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 16:1–16:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.16.

32 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.

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

34 Dennis Olivetti. Round Eliminator: a tool for automatic speedup simulation, 2019. URL: https://github.com/olidennis/round-eliminator.

35 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. doi:10.1137/1.9780898719772.

36 Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proceedings of 52nd Annual ACM Symposium on Theory of Computing (STOC 2020)*, 2020. doi:10.1145/3357713.3384298.