Using Round Elimination to Understand Locality

Jukka Suomela[1]
Aalto University, Finland
https://jukkasuomela.fi/

Abstract

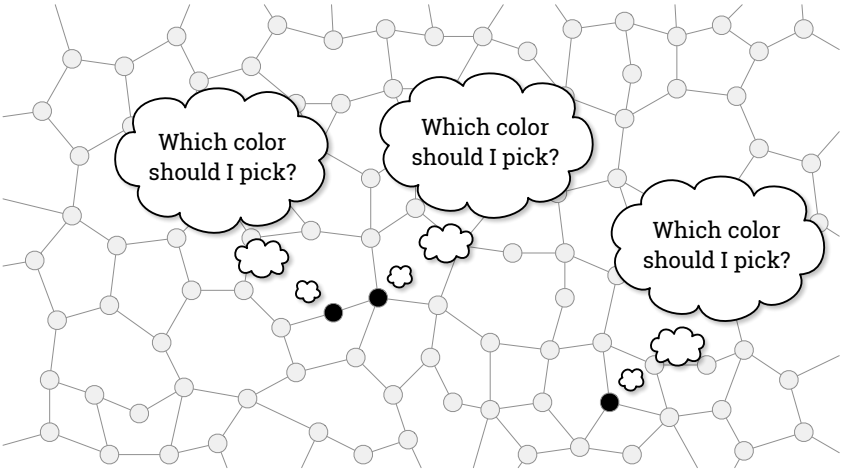
A key concept in the theory of distributed computing is *locality*: if I am a node in the middle of a large graph, *how far* do I need to see in order to pick my own part of the solution? A modern, effective technique for studying such questions is called *round elimination*.

With round elimination, one can turn any local graph problem Π into a new graph problem $\Pi' = \mathrm{re}(\Pi)$ that is strictly more localized: if problem Π can be solved so that each node needs information only within distance r, then problem Π' can be solved so that each node needs information only within distance $r - 1$. Round elimination is a mechanical procedure that can be fully automated: one can enter the description of Π to a computer program and out comes the description of $\Pi' = \mathrm{re}(\Pi)$.

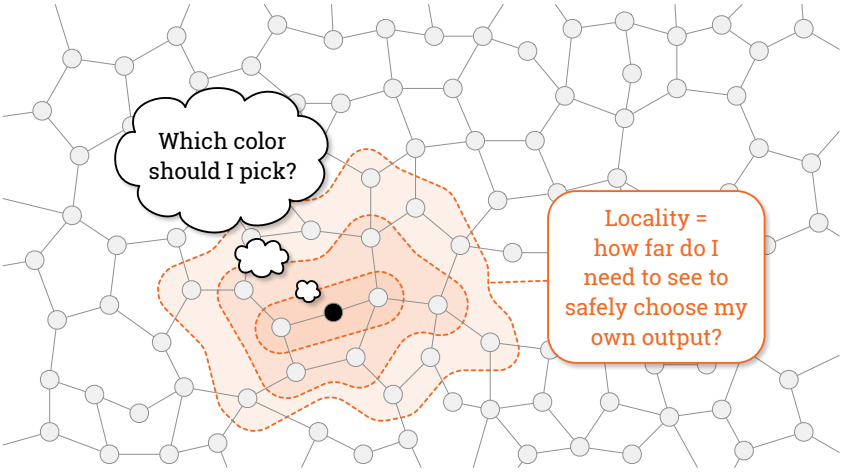
One particularly interesting phenomenon is that some problems Π are *fixed points* for re or re^2, that is, we have got $\Pi = \mathrm{re}(\Pi)$ or $\Pi = \mathrm{re}(\mathrm{re}(\Pi))$. Whenever this happens for a nontrivial problem, we immediately know that Π cannot be solved locally—this is a quick and easy way to prove lower bounds on the locality of many graph problems. In this short article I will give a brief introduction to the round elimination technique and its key applications.

1 Introduction

Locality of graph problems. Imagine you are a node in the middle of a large graph. Together with all other nodes, you will need to solve some graph problem, in a distributed manner: you will need to output your *own part of the solution*. For example, if we are interested in the vertex coloring problem, each node has to output its own color, and the local outputs have to form a proper coloring for the entire graph: adjacent nodes have to output different colors.



A key concept that the theory of distributed computing studies is *locality*: how far does each node need to see in the graph to pick its own part of the solution. If you are only aware of yourself and know nothing about the structure of the graph, there is not much you can do. And if you see the entire graph, then you have got enough information to solve any graph problem. But what happens between these extremes?



Which graph problems can be solved based on constant-radius neighborhoods? For example, what can you do if you see up to distance 3 in the graph, as shown in the above figure? What about slightly super-constant neighborhoods, e.g. radius $O(\log^* n)$ or $O(\log \log n)$? And which problems are inherently global and require you to see almost the entire graph in the worst case?

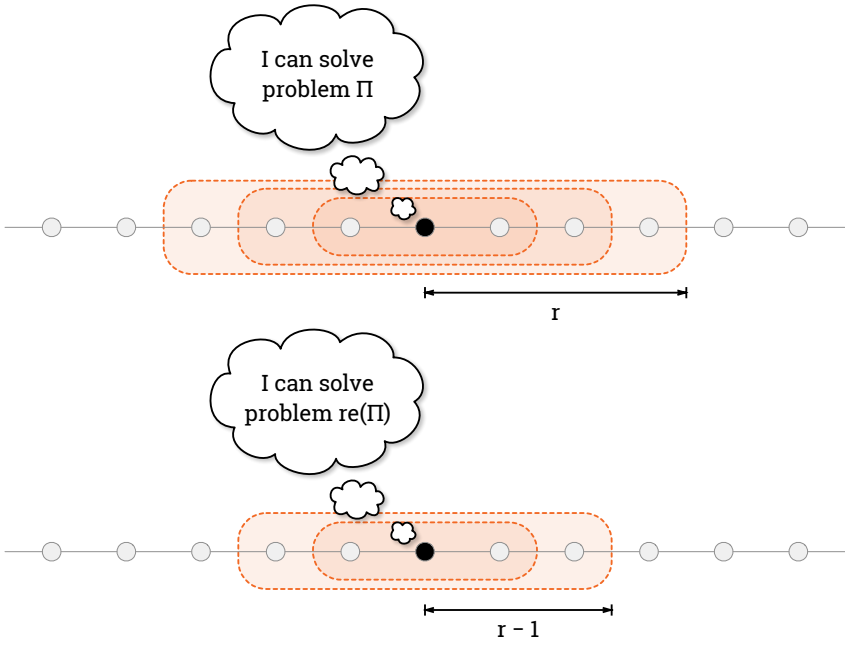


FIGURE 1: Round elimination turns a problem Π into a more localized problem re(Π).

How to study locality? The study of locality was initiated by Linial [12, 13] in the late 1980s, but what was lacking for a long time was *systematic* techniques that can be applied to a broad range of graph problems. Until recently, the main technique that the researchers used to study the locality of graph problems was, in essence, thinking hard.

There is a small number of specific graph problems whose locality has been well-understood already for a long time, see, e.g., Linial [12, 13], Naor [14], and Kuhn et al. [9, 10, 11]. However, what do you do when you encounter a new graph problem nobody else has studied before?

We do not yet have a complete answer that could be applied to any given graph problem. However, many graph problems of interest are *locally verifiable*: if the solution is not feasible, at least one node will notice it immediately by looking at the solution in its own local neighborhood. For example, vertex coloring is by definition locally verifiable: if a coloring is not proper, then there is a node that is adjacent to another node with the same color. And whenever we have got a locally verifiable problem, there is now a mechanical technique that we can try to use: *round elimination*.

Round elimination. Round elimination is based on an old idea that was already used by Linial [12, 13] in his lower bound for the locality of vertex coloring, but for a long time it seemed to be just an ad-hoc trick that is applicable to one specific problem. However, in the past four years it was discovered that the same idea can be generalized and applied to study *any* locally verifiable problem [4, 6, 15].

Informally, round elimination is a function "re" that maps a locally verifiable graph problem Π to another locally verifiable graph problem $\Pi' = \mathrm{re}(\Pi)$. Round elimination will make problems *strictly more local* in the following sense (see Figure 1):

If Π has a (sufficiently small) locality r, then re(Π) has locality $r - 1$.

Here, "locality r" means that Π can be solved (for the worst-case inputs) if and only if all nodes see up to distance r in the graph.

Round elimination is a mechanical process—there is even a freely available computer program [15] that is happy to calculate re(Π) for any given problem Π, and we will also soon see how to use it.

Significance of round elimination. Now even if we can *apply* round elimination to a given graph problem Π, it does not mean that we will be able to immediately *prove* tight upper and lower bounds for the locality of Π. The whole procedure may seem a bit pointless at first: if we do not understand the locality of Π, why would it be helpful to construct another problem re(Π) that we do not understand either?

However, as we will see soon, *sometimes* we are lucky and a mechanical application of round elimination immediately gives a tight lower bound. For example, some problems Π are *fixed points* or *period-2 points* in round elimination, that is, we have

$$\Pi = \text{re}(\Pi) \text{ or } \Pi = \text{re}(\text{re}(\Pi)),$$

and whenever this happens we immediately get a lower bound result—more about this soon!

We are currently seeing more and more examples of results in which round elimination has played a key role, see, e.g., [1–3, 5, 6]. We have recently systematically explored large families of graph problems, and round elimination has been an essential tool that has made it possible to make rapid progress.

In this short article I will give a brief overview of the round elimination technique and some of its key applications; I will use the sinkless orientation problem as a running example to illustrate the key ideas.

2 Example: sinkless orientation

Problem definition. *Sinkless orientation* is the following graph problem; see Figure 2:

Given an undirected graph $G = (V, E)$, orient all edges such that each node with degree at least 3 has outdegree at least 1.

That is, high-degree nodes cannot be sink nodes. It is easy to see that such an orientation always exists, and there is a very simple centralized algorithm that finds such an orientation (w.l.o.g. assume that G is connected); see Figure 2c:

1. If G contains a cycle C, first orient all edges of C in a consistent direction (this way all nodes in C will have outdegree at least 1 and hence they are happy). Then orient all other edges towards C, breaking ties arbitrarily (this way all other nodes will also have outdegree at least 1, as there is at least one incident edge that points towards C).

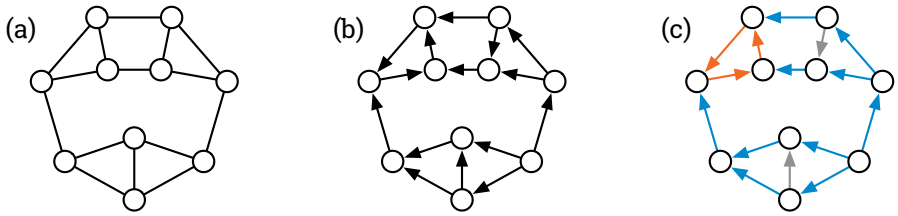


FIGURE 2: (a) Graph G. (b) A sinkless orientation for graph G. (c) A simple centralized algorithm for finding a sinkless orientation: find a cycle and orient it consistently (orange), orient all other edges towards the cycle (blue), break ties arbitrarily (gray).

2. Otherwise G is a tree and there is a node v of degree at most 1; then orient all edges towards v (this way all nodes except v will have outdegree 1).

Locality of sinkless orientation. The above algorithm is inherently global. Furthermore, it is *unnecessarily* global: the above algorithm will also ensure that nodes of degree 2 have outdegree at least 1, while this is not required in the problem definition. Maybe we could find a much more local solution to this problem?

This is indeed the case; it turns out that the locality of this problem is only $O(\log \log n)$ for randomized algorithms and $O(\log n)$ for deterministic algorithms [8]. The general case and the randomized algorithm are more complicated, but it is easy to see that at least in the case of *trees* the locality of the problem is $O(\log n)$. We describe the distributed algorithm from the perspective of an individual node v with $\deg(v) \geq 3$:

1. Node v finds the nearest node x such that $\deg(x) \leq 2$ (breaking ties arbitrarily). Note that such a node has to exist within distance $O(\log n)$ in any tree with n nodes.
2. Let $e = \{v, u\}$ be the unique edge incident to v that points towards x.
3. Node v will announce that e is oriented from v to u. Note that there are never conflicts: node u will not announce that e is oriented from u to v.

Finally, if there are some unoriented edges, orient them arbitrarily. This will ensure that all edges are oriented and, if a node has degree at least 3, it successfully orients at least one of its incident edges away from itself. Furthermore, each node only needs to see up to distance $O(\log n)$.

Models of distributed computing. The above idea can be implemented as a concrete distributed algorithm in many models of distributed computing, but for our purposes it is sufficient to note that it works in each of the following models (see Figure 3):

- *Port-numbering model:* For each node, there is a numbering of its incident edges, and for each edge, there is a numbering of its endpoints. We can refer to the 1st neighbor, 2nd neighbor, etc. of each node, and we can refer to the 1st endpoint and the 2nd endpoint of each edge. The port numbering comes from an adversary, and it is completely arbitrary: for example, if u is the 1st neighbor of v, it does not mean that v has to be the 1st neighbor of u.

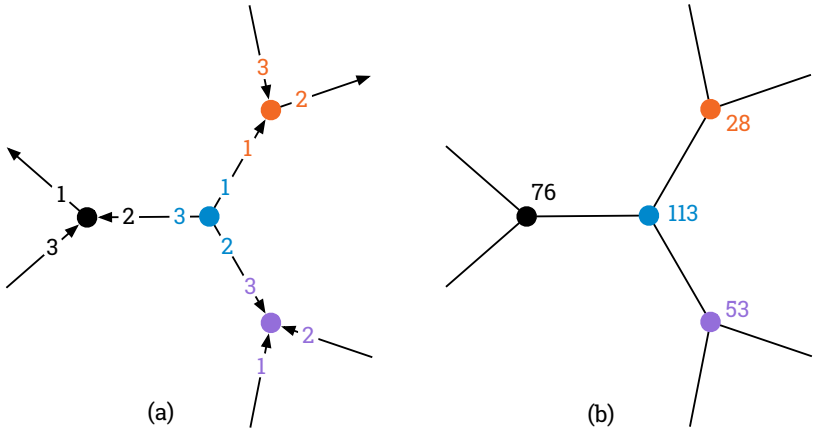


FIGURE 3: (a) Port-numbering model; here the numbering of the endpoints of each edge is represented with arrows and the numbering of the incident edges of each node is represented with numbers 1–3. (b) Deterministic LOCAL model; the numerical labels of the nodes are their unique identifiers.

- *Deterministic LOCAL model:* The nodes are labeled with unique identifiers. If there are n nodes, the unique identifiers are natural numbers between 1 and $\mathrm{poly}(n)$. The unique identifiers come from an adversary.
- *Randomized LOCAL model:* Each node has a source of independent random bits, and the solution is correct w.h.p.

We have been discussing here *locality* but we can equally well study the *round complexity of message-passing algorithms*: if we imagine that each node is a computer and each edge is a communication link, in r synchronous communication rounds each node can gather full information about its radius-r neighborhood and nothing more. Hence we can use the terms "locality" and "round complexity" (or distributed time complexity) interchangeably.

Lower bounds on the locality? Good, so far we know that the locality of the sinkless orientation problem is at most $O(\log n)$. But is this the best that we can do? Now we are getting closer to our topic.

We will use round elimination to show that, even if we are looking at trees, the sinkless orientation problem cannot be solved in the port-numbering model with locality $o(\log n)$, or, equivalently, in $o(\log n)$ rounds. This may sound like a weak statement, as it only applies to the port-numbering model. However, it turns out that proving a lower bound in the port-numbering model is the key step. As we will discuss in more detail in Section 6, tight bounds in the other models follow:

- In the randomized LOCAL model the locality is at least $\Omega(\log \log n)$ [6].
- In the deterministic LOCAL model the locality is at least $\Omega(\log n)$ [7].

3 How to represent problems?

Bipartite edge-labeling formalism. One thing we have learned over the recent years is that it is important to have the *right formalism* in which we can represent graph problems. The formalism that we will use here is defined so that it *makes round elimination as simple as possible*, but there is a downside: it is not a particularly natural formalism for a human being to think about graph problems. However, the formalism is very expressive, and in essence all locally verifiable graph problems can be turned into this formalism with some effort (with local reductions that only influence locality by a small additive or multiplicative constant). The right formalism looks like this:

- The input graph is an *infinite* 2*-colored tree*.
  - The first color class is called *active*, and each active node has degree d.
  - The second color class is called *passive*, and each passive node has degree δ.
- The task is to *label each edge* with a label from some finite set Σ.
- The graph problem is a pair $\Pi = (\mathsf{A}, \mathsf{P})$:
  - A is the constraint for active nodes.
  - P is the constraint for passive nodes.
- The solution is encoded in the local outputs of the active nodes. For each active node v, its local output contains the labels of each edge incident to v. The passive nodes do not produce any output.
  - The labels of the edges incident to an active node satisfy constraint A.
  - The labels of the edges incident to a passive node satisfy constraint P.

The constraints A and P are *sets of multisets of edge labels*; we will soon see an example that makes their role clear. Before that, one remark is in order: we study here infinite trees, but this is merely for convenience so that we do not need to worry about boundaries. Nevertheless, we can still prove lower bounds that apply to finite inputs: if we show that, for example, in an infinite tree a node has to see at least up to distance r, then the same will hold also in the middle of a finite tree in which all leaf nodes are at distance more than r; after all, the local neighborhoods are indistinguishable.

Representing sinkless orientation in the formalism. To prove a lower bound, it is sufficient to construct some family of instances that is hard, and as is often the case, it turns out that the sinkless orientation problem is hard to solve if you are *in the middle of a regular tree*. And it is easy to see that 3*-regular trees* have to be the most difficult case; after all the problem becomes trivial if all nodes have degree 2, and there is more slack if there are nodes of a degree higher than 3.

As discussed above, we can also first consider *infinite* trees and once we have a lower bound construction for infinite trees, we can zoom into a finite fragment of an infinite tree. So the specific problem that we will consider is "sinkless orientation in infinite 3-regular

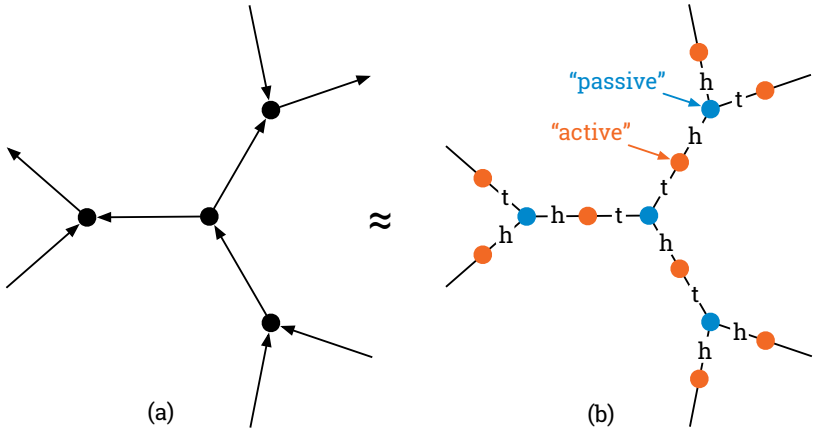


FIGURE 4: (a) Sinkless orientation in a 3-regular tree. (b) Equivalent problem encoded in the bipartite edge-labeling formalism; active nodes (orange) represent edges, passive nodes (blue) represent nodes, and the edge labels h and t represent "head" and "tail". In a feasible solution, the multiset of labels around an active node is always $\{h,t\}$, while the multiset of labels around a passive node is $\{h,h,t\}$, $\{h,t,t\}$, or $\{t,t,t\}$. The edge labels are encoded in the local outputs of the active nodes.

trees". But how do we map this to the formalism above? In our formalism we are supposed to have a 2-colored tree (which we do not have here), and we are supposed to be *labeling* edges (while here our task is to *orient* edges).

It turns out that there is a simple trick we can apply: we split all edges in two halves, and label the half-edges using alphabet $\Sigma = \{h,t\}$, where h denotes "head" and t denotes "tail"; see Figure 4. And now the constraints are simply the following:

- For each original edge, exactly one of the half-edges has to be labeled with h and the other half-edge has to be labeled with t (so that each edge has a well-defined orientation).
- For each original node, at least one of the incident half-edges has to be labeled with t (so that there is at least one outgoing edge).

Let us formalize this. Let $G = (V,E)$ be the original (infinite) tree. We construct a new (infinite) tree $G' = (V',E')$, where the set of nodes is $V' = V \cup E$ and the set of edges is $E' = \big\{\{v,e\} : v \in e \in E\big\}$. Now V forms one color class and E forms another color class; nodes in V have degree 3 and nodes in E have degree 2. We will now (somewhat arbitrarily) decide to call E the set of active nodes and V the set of passive nodes, that is, active nodes have degree 2 and passive nodes have degree 3.

As the problem will be the starting point for round elimination, we will call it $\Pi_0 = (\mathsf{A}_0, \mathsf{P}_0)$, and we will use $\Sigma_0 = \{h,t\}$ to denote the set of labels. To complete the definition, we need to specify the two constraints A_0 and P_0. Here A_0 is the set of possible multisets of labels incident to active nodes. As the active nodes correspond to the original edges, we will only allow one possible multiset that contains exactly one h and exactly one t:

$$\mathsf{A}_0 = \big\{\{h,t\}\big\}. \tag{1}$$

However, for the passive nodes there are more possibilities: we can have one, two, or three incident t labels, and hence

$$\mathsf{P}_0 = \big\{\{h,h,t\},\{h,t,t\},\{t,t,t\}\big\}. \tag{2}$$

Equivalence of representations. Now we have completed the definition of $\Pi_0 = (\mathsf{A}_0, \mathsf{P}_0)$. The key observation is that problem Π_0 faithfully captures the locality of the sinkless orientation problem in 3-regular trees: if we have an algorithm that finds a sinkless orientation, we can simulate it as a black box to find a solution for Π_0, and vice versa, and the simulation changes locality only by a constant factor (distance r in graph G corresponds to distance $2r$ in graph G').

Hence, from now on we can forget about the sinkless orientation problem and study the locality of problem $\Pi_0 = (\mathsf{A}_0, \mathsf{P}_0)$ specified in (1)–(2).

Roadmap. We will now apply round elimination to Π_0 in two ways: first the hard way in Section 4, and then the easy way in Section 5 with the help of computers. In both cases we will see that two applications of round elimination will get us back to the same problem, i.e., $\mathrm{re}(\mathrm{re}(\Pi_0)) = \Pi_0$; what that implies is then discussed in more detail in Section 6.

4 Round elimination, done by hand

Basic idea. Imagine you have some algorithm $\mathcal{A}_0$ that solves Π_0 in r rounds in any infinite tree G. That is, each active node v only needs to see up to distance r in G to determine what are the labels of its incident edges—which of them to label with h and which of them to label with t.

We will construct now a new algorithm $\mathcal{A}_1$ that solves *another* problem Π_1 in $r-1$ rounds. The idea is very simple; algorithm $\mathcal{A}_1$ works as follows (see Figure 5):

- Let u be a passive node in Π_0, and let v_1 be one of its active neighbors.
- Node u gathers all information within radius $r-1$ and sees what it now knows about the output of v_1 in algorithm $\mathcal{A}_0$.
- Let $X_1 \subseteq \Sigma_0$ be the set of all possible labels that v_1 might produce for the edge $\{u, v_1\}$.
- Node u labels the edge $\{u, v_1\}$ with the set X_1.

Note that here we have *switched the roles of active and passive nodes*: what was previously passive in Π_0 will take an active role in Π_1 and vice versa; hence we will refer to e.g. Π_0-active and Π_1-active nodes. Also note that the set of labels changes: if our original set of labels was Σ_0, the new set of labels Σ_1 will consist of nonempty subsets of Σ_0.

Algorithm $\mathcal{A}_1$ by construction works based on information within distance $r-1$. However, what is not at all obvious is this: is $\mathcal{A}_1$ really solving a nontrivial problem Π_1? After all, could it not happen that $\mathcal{A}_1$ just labels all edges with the set Σ_0? As we will see through examples, this is not going to be the case!

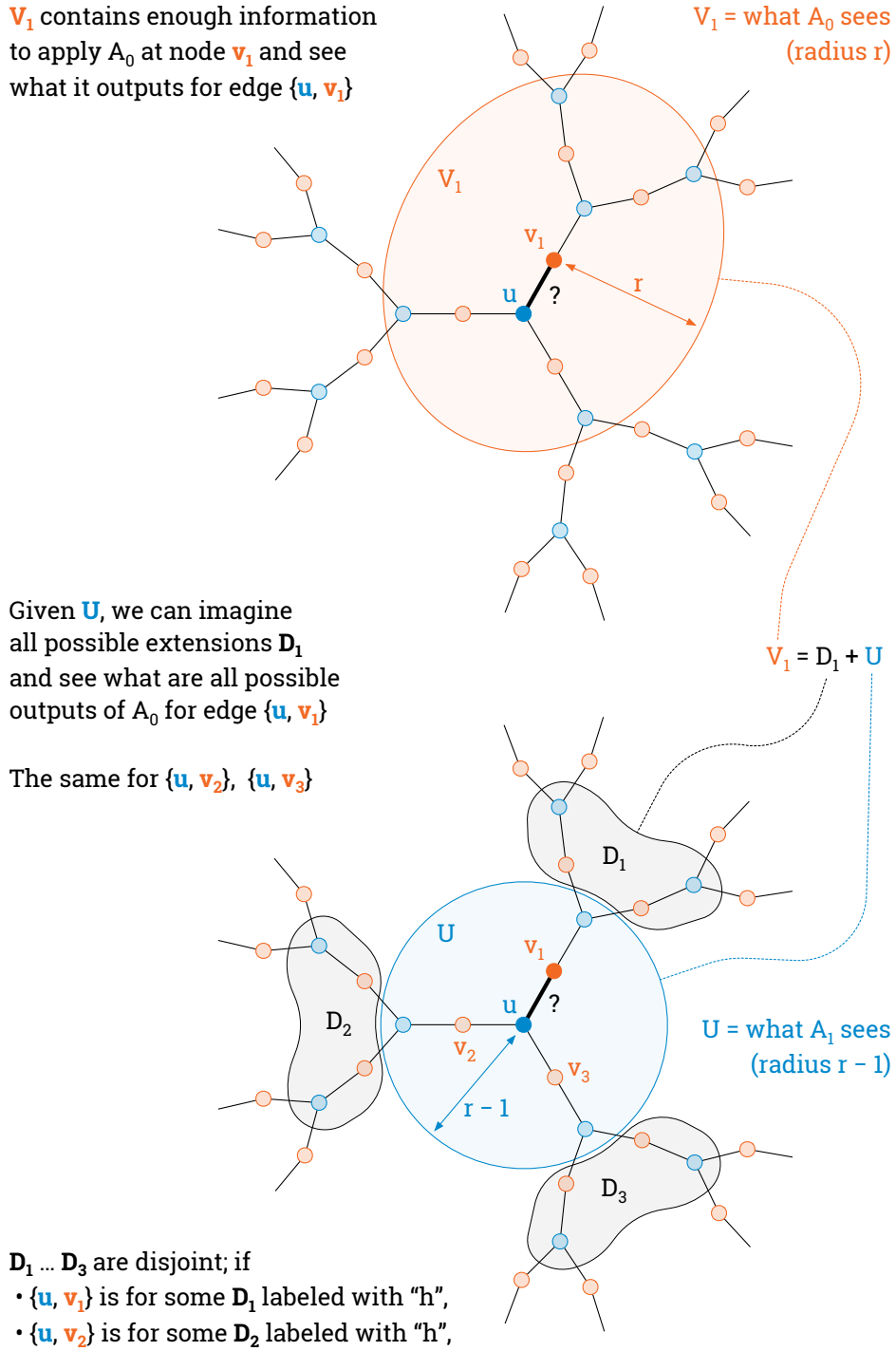


FIGURE 5: A schematic illustration of the round elimination step $\Pi_1 = \mathrm{re}(\Pi_0)$.

First step. Let us now apply this idea to the sinkless orientation problem, which we formalized as problem $\Pi_0 = (\mathsf{A}_0, \mathsf{P}_0)$ in (1)–(2). Assume that $\mathcal{A}_0$ solves Π_0 based on radius-r neighborhoods, for some value r. As we work in the port-numbering model, and the structure of the tree is regular (alternately nodes of degree 2 and degree 3), the only unknown information that $\mathcal{A}_0$ can use is the specific assignment of port numbers.

Now consider a Π_0-passive node u. Recall that passive nodes had degree 3, so there are three active neighbors, v_1, v_2, and v_3. Let U consist of all information within distance $r-1$ from u, and let V_i consist of all information within distance r from v_i; see Figure 5 for illustrations.

Now V_i is exactly enough information to determine what $\mathcal{A}_0$ would output for the edge $\{u, v_i\}$. However, U represents only a part of V_i, and hence U is not enough to simulate $\mathcal{A}_0$. The key observation is this: V_i can be decomposed in two parts, U and D_i, as shown in Figure 5. Furthermore, parts D_1, D_2, and D_3 are disjoint—this is going to be crucial as we will soon see!

When we run algorithm $\mathcal{A}_1$ at node u, it first gathers all information in U. Then for each i and each label $x \in \Sigma_0$, we will make the following thought experiment: is there some D_i such that given $V_i = U + D_i$ algorithm $\mathcal{A}_0$ would label the edge $\{u, v_i\}$ with x? Let X_i be the set of all such labels x; algorithm $\mathcal{A}_1$ will simply output the set X_i for the edge $\{u, v_i\}$. Now what are the properties that the output of $\mathcal{A}_1$ will satisfy?

- Let u be a Π_0-passive node. Assume that for each i there was some D_i such that $\mathcal{A}_0$ labels $\{u, v_i\}$ with h. But parts D_i are disjoint, and therefore we could construct an input in which $\mathcal{A}_0$ simultaneously labels all edges incident to u with h, which is not allowed in P_0 according to (2). Therefore there has to be at least one index i such that $\mathcal{A}_0$ never labels $\{u, v_i\}$ with h, and therefore $\mathcal{A}_1$ will label $\{u, v_i\}$ with the set $\{t\}$. The other edges might be labeled with $\{h\}$, $\{t\}$, or $\{h, t\}$.

- Let v be a Π_0-active node. If $\mathcal{A}_1$ labels an edge e with set $X \subseteq \Sigma_0$, then we know that $\mathcal{A}_0$ labels the same edge with some value $x \in X$ in this set; after all, the real input that $\mathcal{A}_0$ sees is among the possible worlds that $\mathcal{A}_1$ will consider in its thought experiment. By assumption, the output of $\mathcal{A}_0$ will satisfy constraint A_0 defined in (1), and therefore it will label exactly one edge e_1 incident to v with h and the other edge e_2 with t. Therefore $\mathcal{A}_1$ will label e_1 with either $\{h\}$ or $\{h, t\}$ and e_2 with either $\{t\}$ or $\{h, t\}$.

Nodes that were passive in $\mathcal{A}_0$ are now active in $\mathcal{A}_1$ and vice versa, so if we look at the new roles of the nodes, we can summarize that the output $\mathcal{A}_1$ satisfies the following properties:

- Π_1-active nodes (degree 3): at least one incident edge is labeled with $\{t\}$.
- Π_1-passive nodes (degree 2): one incident edge is labeled with $\{h\}$ or $\{h, t\}$ and the other with $\{t\}$ or $\{h, t\}$.

But if these are the only constraints we have, the label $\{h\}$ looks rather pointless. Whenever $\mathcal{A}_1$ would output $\{h\}$, it could equally well output $\{h, t\}$ without violating the above constraints. So there are only two meaningful outputs for $\mathcal{A}_1$; let us give them some names:

$$\Sigma_1 = \{T, X\}, \text{ where } T = \{t\} \text{ and } X = \{h, t\}.$$

Using this notation, we can restate the properties satisfied by the output of $\mathcal{A}_1$:

- Π_1-active nodes: at least one incident edge is labeled with T.
- Π_1-passive nodes: at least one incident edge is labeled with X.

We can now write this down in our formalism as the specification of a graph problem $\Pi_1 = (\mathsf{A}_1, \mathsf{P}_1)$ as follows:

$$\mathsf{A}_1 = \big\{\{T, X, X\}, \{T, T, X\}, \{T, T, T\}\big\},$$
$$\mathsf{P}_1 = \big\{\{T, X\}, \{X, X\}\big\}.$$

But note that there is one further simplification that we can do (just to keep the problem definition as short as possible): if we have a feasible solution in which an active node labels multiple incident edges with T, it can simply replace some of them with Xs without violating either of the above constraints. Hence w.l.o.g. we can constrain the problem as follows:

$$\mathsf{A}_1 = \big\{\{T, X, X\}\big\}, \tag{3}$$
$$\mathsf{P}_1 = \big\{\{T, X\}, \{X, X\}\big\}. \tag{4}$$

Now we have done *one* step of round elimination: we started with problem Π_0, assumed that it is solvable in r rounds with some algorithm $\mathcal{A}_0$, and we constructed a new problem $\Pi_1 = (\mathsf{A}_1, \mathsf{P}_1)$ that is by construction solvable in $r - 1$ rounds using an algorithm $\mathcal{A}_1$ that, in essence, just simulates $\mathcal{A}_0$ as a black box (and does some minor local post-processing so that we have got a more concise description for problem Π_1).

It turns out that this conversion from Π_0 to Π_1 is "if and only if": assuming there is some algorithm $\mathcal{A}_1$ that solves Π_1 in $r - 1$ rounds, we can also recover a solution for Π_0 in r rounds. It is not hard to see this is indeed the case: in essence we can map T to t and then X to either t or h so that each Π_0-active node is adjacent to exactly one t.

Second step. We repeat exactly the same reasoning to construct algorithm $\mathcal{A}_2$ that solves some problem Π_2 in $r - 2$ rounds. Consider a Π_1-passive node u. Passive nodes have degree 2, so there are two active neighbors, v_1 and v_2. Let U consist of all information within distance $r - 2$ from u, let V_i consist of all information within distance $r - 1$ from v_i, and let us decompose V_i in two parts, U and D_i.

When we run algorithm $\mathcal{A}_2$ at node u, it again gathers all information in U and labels $\{u, v_i\}$ with the set of all labels that $\mathcal{A}_1$ might output for some choice of D_i. What are the properties that the output of $\mathcal{A}_2$ will satisfy?

- Let u be a Π_1-passive node. Constraint P_1 in (4) implies that $\mathcal{A}_1$ cannot ever label all edges incident to u with T, and hence $\mathcal{A}_2$ will label at least one edge $\{u, v_i\}$ with the set $\{X\}$ that does not contain T.

- Let v be a Π_1-active node. Constraint A_1 in (3) implies that $\mathcal{A}_1$ will label one of the incident edges with T and the other two with X. Therefore $\mathcal{A}_2$ will label one of the incident edges with $\{T\}$ or $\{T, X\}$, which are the subsets that contain T, and the other two with $\{X\}$ or $\{T, X\}$, which are the subsets that contain X.

So to recap, after switching the roles of the nodes, we have got:

- Π_2-active nodes: at least one incident edge is labeled with $\{X\}$.
- Π_2-passive nodes: one incident edge is labeled with $\{T\}$ or $\{T, X\}$ and the two others with $\{X\}$ or $\{T, X\}$.

Again there is a label that is rather pointless: any $\{T\}$ can be replaced with $\{T, X\}$. Hence there are only two meaningful outputs for $\mathcal{A}_1$; let us again give some convenient names for them:

$$\Sigma_2 = \{\mathcal{T}, \mathcal{X}\}, \text{ where } \mathcal{T} = \{T, X\} \text{ and } \mathcal{X} = \{X\}.$$

Using this notation, we can now write down the properties satisfied by the output of $\mathcal{A}_2$ as a specification of a graph problem $\Pi_2 = (\mathsf{A}_2, \mathsf{P}_2)$ as follows:

$$\mathsf{A}_2 = \big\{\{\mathcal{X}, \mathcal{T}\}, \{\mathcal{X}, \mathcal{X}\}\big\},$$
$$\mathsf{P}_2 = \big\{\{\mathcal{X}, \mathcal{X}, \mathcal{T}\}, \{\mathcal{X}, \mathcal{T}, \mathcal{T}\}, \{\mathcal{T}, \mathcal{T}, \mathcal{T}\}\big\}.$$

And again we can simplify it further by observing that if $\mathcal{A}_2$ would label two incident edges with $\mathcal{X}$, it can safely replace one of them with $\mathcal{T}$. We arrive at the following problem:

$$\mathsf{A}_2 = \big\{\{\mathcal{X}, \mathcal{T}\}\big\}, \tag{5}$$
$$\mathsf{P}_2 = \big\{\{\mathcal{X}, \mathcal{X}, \mathcal{T}\}, \{\mathcal{X}, \mathcal{T}, \mathcal{T}\}, \{\mathcal{T}, \mathcal{T}, \mathcal{T}\}\big\}. \tag{6}$$

And the good news is that we can stop now!

We found a periodic point! Let us recap. We started with Π_0, sinkless orientation, and then applied round elimination twice to obtain $\Pi_1 = \mathrm{re}(\Pi_0)$ and $\Pi_2 = \mathrm{re}(\mathrm{re}(\Pi_0))$. But now if we compare Π_0 in (1)–(2) with Π_2 in (5)–(6), we notice that they are the same problem, up to the naming of the output labels!

We have found a periodic point in round elimination: $\Pi_0 = \mathrm{re}(\mathrm{re}(\Pi_0))$.

We will discuss how to interpret this in more detail in Section 6, but we will first see how to find such a periodic point with a significantly smaller amount of effort.

5 Round elimination, with the help of a computer

Round Eliminator tool. We have now spent several pages doing a somewhat technical and error-prone analysis to first construct problem $\mathrm{re}(\Pi_0)$ and then another problem $\mathrm{re}^2(\Pi_0)$, and along the way we did also some simplifications that may seem like ad-hoc steps. Is round elimination really that hard?

No, the good news is that we can reproduce exactly what we did in Section 4 with a couple of mouse clicks if we use the Round Eliminator tool by Olivetti [15]! The only thing we need to know is the language in which we specify problems.

Round Eliminator notation. Recall the original definition of problem Π_0 in (1)–(2):

$$\mathsf{A}_0 = \big\{\{h,t\}\big\}$$
$$\mathsf{P}_0 = \big\{\{h,h,t\},\{h,t,t\},\{t,t,t\}\big\}.$$

In the computer-readable version we simply write each possible multiset on its own line; the elements are space-separated and we can list them in any order, for example:

Active	Passive

`h t`	`h h t`
	`h t t`
	`t t t`

(7)

However, often the number of possible multisets gets very large and a shorthand notation is needed. The language that we use with Round Eliminator allows us to write the same problem also as follows:

Active	Passive
`h t`	`ht ht t`

(8)

The interpretation of the line `ht ht t` is that for the first element we can choose either `h` or `t`, and the same holds for the second element. So expanding this will result in four possibilities, `h h t`, `h t t`, `t h t`, and `t t t`, the second and the third one being the same multiset. So (8) is exactly the same problem as (7). We have got a very concise way of specifying the sinkless orientation problem.

We do not necessarily need to use the concise form when specifying problems, but we will nevertheless need to be familiar with it so that we can understand the output of the Round Eliminator tool.

First step. Let us open the Round Eliminator tool [15] and enter problem (8); we press "start" and then "speedup" to perform one step of round elimination. We get the following problem back:

Active	Passive
`B B A`	`AB B`

(9)

If we rename `A` to T and `B` to X, we can see that Round Eliminator gave us back exactly the same problem as what we had in (3)–(4) after one manual round elimination step. It is easier to see the equivalence if we expand the shorthand notation and reorder it a bit:

Active	Passive
`A B B`	`A B` `B B`

Second step. Let us continue and press the "speedup" button again. Now we get the following problem back:

Active	Passive
`B A`	`AB AB B`

(10)

If we rename `A` to $\mathcal{X}$ and `B` to $\mathcal{T}$, we see that this is exactly the same problem as what we had in (5)–(6) after two round elimination steps. And on the other hand we can see that this is also identical to the problem (8) that we started with (the only difference being the names of the labels). So we have with a couple of mouse clicks identified that $\Pi_0 = \mathrm{re}(\mathrm{re}(\Pi_0))$, and hence we are at the same point as where we were at the end of Section 4.

6 Consequences of a periodic point

Immediate lower bound in the port-numbering model. We have now seen (twice) that $\Pi_0 = \mathrm{re}^2(\Pi_0)$, that is, two steps of round elimination gives us back the same problem—or a bit more precisely, problem $\mathrm{re}^2(\Pi_0)$ is isomorphic to Π_0. There are at least two useful ways to interpret this, but both lead to the same conclusion:

1. If the locality of Π_0 in infinite trees is some finite value r, then the locality of the same problem $\Pi_0 = \Pi_2 = \mathrm{re}^2(\Pi_0)$ is $r - 2$, which is absurd. Hence the locality of Π_0 in infinite trees cannot be finite.

2. If we have an algorithm $\mathcal{A}_0$ that solves Π_0 in r rounds for some finite even r, we can use round elimination repeatedly to reduce r to 0. But it is easy to check that Π_0 cannot be solved in 0 rounds, so $\mathcal{A}_0$ cannot exist.

The conclusion in both cases is that sinkless orientation cannot be solved at all in the port-numbering model if you are in the middle of a regular tree and do not see any leaf nodes (or other irregularities).

And now it is enough to observe that we can construct a finite 3-regular tree with n nodes in which all leaf nodes are within distance $\Theta(\log n)$ from the midpoint, and hence we get a tight $\Omega(\log n)$ lower bound on the locality of sinkless orientation in the port-numbering model.

Consequences for other models of computing. Now let us look back at what we did: we performed round elimination in the *port-numbering model* (so e.g. the only "unknown" information in the distant parts D_i was the specific choice of *port numbering*).

We would like to do the same in the usual deterministic LOCAL model. Unfortunately, the unique identifiers are something that we cannot *directly* handle in the round elimination process. For example, if in some neighborhood U algorithm $\mathcal{A}_0$ outputs h for the edge $\{u, v_i\}$ if and only if it sees the unique identifier 1 in part D_i, then we cannot argue that $\mathcal{A}_0$ will also output h for all edges $\{u, v_i\}$ for some choice of unique identifiers. After all, identifier 1 cannot be simultaneously in part D_1 and part D_2.

However, we can take a *detour through randomized algorithms*. We simply assume that all nodes have random bits as inputs. We start with a randomized algorithm $\mathcal{A}_0$ that solves Π_0

so that the local failure probability is some small p_0. Then we can show that $\mathcal{A}_1$ will solve Π_1 so that the local failure probability is some p_1, which depends on p_0 and the structure of Π_0. Now if we apply this reasoning to the sinkless orientation problem, we can repeat the argument ℓ steps for an even ℓ and arrive at $\mathcal{A}_\ell$ that solves $\Pi_\ell = \Pi_0$ in $r - \ell$ rounds so that the local failure probability is p_ℓ. If we have $\ell = o(\log \log n)$, we can show that p_ℓ is not too high, and $\mathcal{A}_\ell$ has to do something nontrivial, and hence $r - \ell > 0$, which implies that the original algorithm $\mathcal{A}_0$ requires more than ℓ rounds. Therefore the locality of Π_0 has to be $\Omega(\log \log n)$ for randomized algorithms. For the details of this kind of an analysis, see, for example, [2, 6].

Now that we have a lower bound for the randomized LOCAL model, it immediately implies a lower bound in the deterministic LOCAL model: after all, any deterministic algorithm gives also an equally fast randomized algorithm (we can use randomness to construct unique identifiers w.h.p.). Hence we have got a lower bound of $\Omega(\log \log n)$ for the deterministic LOCAL model. But we can do better; the *gap result* by Chang et al. [7] shows that the existence of an $o(\log n)$-round deterministic algorithm for a locally verifiable problem implies the existence of an $O(\log^* n)$-round deterministic algorithm for the same problem, but we know that such an algorithm cannot exist. Hence the gap result implies that the locality of the sinkless orientation problem in the deterministic LOCAL model is $\Omega(\log n)$, which is tight.

To recap, we took the following steps:

- Start with a locally verifiable problem Π_0.
- Show that Π_0 is a *periodic point* in round elimination: $\Pi_0 = \mathrm{re}^2(\Pi_0)$.
- It follows that the complexity of Π_0 in the port-numbering model is $\Omega(\log n)$.
- By taking into account failure probabilities, it follows that the complexity of Π_0 in the randomized LOCAL model is $\Omega(\log \log n)$.
- By amplifying this with the gap result, it follows that the complexity of Π_0 in the deterministic LOCAL model is $\Omega(\log n)$.

All of this generalizes. Now the key point is this: the above reasoning was in essence only using the fact that Π_0 is a locally verifiable problem with $\Pi_0 = \mathrm{re}^2(\Pi_0)$. Hence whenever we have a periodic point in round elimination, we get for free the same lower bound results: the locality is $\Omega(\log n)$ in the deterministic LOCAL model and $\Omega(\log \log n)$ in the randomized LOCAL model.

Of course we are not always lucky; however, whenever we encounter a locally verifiable problem Π of an unknown complexity, it makes sense to check if Π or e.g. some relaxation of Π happens to satisfy e.g. $\Pi = \mathrm{re}(\Pi)$ or $\Pi = \mathrm{re}^2(\Pi)$. Lots of examples of such problems are already known [16], and for all such problems we can immediately derive a lower bound with the help of the Round Eliminator tool.

Often we need to consider relaxations. There is one counterintuitive observation that is useful to keep in mind when applying round elimination: *often it is easier to prove lower bounds for easier problems*! For example, we have seen that sinkless orientation is a periodic point in round elimination. Now consider a more challenging version of the problem, *sinkless and sourceless orientation*, in which nodes of degree at least 3 must have both indegree and outdegree at least 1. Trivially, sinkless and sourceless orientation is at least as hard as sinkless orientation. However, sinkless and sourceless orientation is *not* a periodic point in round elimination. Therefore to prove a lower bound for sinkless and sourceless orientation, it is useful to note that it is a strict restriction of the sinkless orientation problem, which is one of the problems that is known to be a periodic point in round elimination, and this observation gives a lower bound also for sinkless and sourceless orientations.

7 There is a lot more we can do

Round elimination sequences. Round elimination is by no means merely about periodic points. More generally, we can construct e.g. a sequence of problems $\Pi_0, \Pi_1, \ldots, \Pi_k$ such that $\Pi_{i+1} = \mathrm{re}(\Pi_i)$ for each i. Such a sequence can be applied both to prove upper bounds and to prove lower bounds:

- Upper bounds: If Π_k is a *trivial* problem that can be solved in 0 rounds, then we know that the locality of Π_0 is at most k. We can even work backwards to extract a k-round algorithm for solving Π_0.
- Lower bounds: If Π_k is a *nontrivial* problem that cannot be solved in 0 rounds, then we know that the locality of Π_0 is more than k.

Restrictions and relaxations. One challenge here is that if the round elimination sequence does not converge to a periodic point, the size of the description of the problems can very rapidly increase, and even if Π_0 is a simple problem with a short description, Π_4 might already be too large to even write down explicitly in a computer, let alone to understand for a human being. Hence what we often do in practice is to consider restrictions and relaxations:

- An *upper bound sequence* is a sequence of problems $\Pi_0, \Pi_1, \ldots, \Pi_k$ such that Π_{i+1} is a *restriction* of $\mathrm{re}(\Pi_i)$ for each i. Now if Π_k is solvable in 0 rounds, the locality of Π_0 is at most k.
- A *lower bound sequence* is a sequence of problems $\Pi_0, \Pi_1, \ldots, \Pi_k$ such that Π_{i+1} is a *relaxation* of $\mathrm{re}(\Pi_i)$ for each i. Now if Π_k is not solvable in 0 rounds, the locality of Π_0 is more than k.

Lower bound sequences were used recently, for example, to show a lower bound for maximal matchings and maximal independent sets [1], and also Linial's [12, 13] lower bound can be interpreted as such a sequence of problems.

Further reading. For more information on the round elimination process, these are recommended reading:

- Brandt [4] gives the formal mathematical definition of the round elimination process.
- Olivetti [15] gives a detailed explanation of how to use the Round Eliminator tool.

Acknowledgments

Many thanks to Sebastian Brandt and Yannic Maus for their comments on this article, and also to everyone else who has taken part in the development of the round elimination technique!

References

[1] 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 Annual IEEE Symposium on Foundations of Computer Science (FOCS 2019)*, pages 481–497. IEEE, 2019. doi:10.1109/FOCS.2019.00037.

[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 International Symposium on Distributed Computing (DISC 2020)*, 2020. URL http://arxiv.org/abs/1911.13294.

[3] Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed Lower Bounds for Ruling Sets. In *Proc. 61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*, 2020. URL http://arxiv.org/abs/2004.08282.

[4] Sebastian Brandt. An Automatic Speedup Theorem for Distributed Problems. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pages 379–388. ACM Press, 2019. doi:10.1145/3293611.3331611.

[5] Sebastian Brandt and Dennis Olivetti. Truly Tight-in-Δ Bounds for Bipartite Maximal Matching and Variants. In *Proc. 39th ACM Symposium on Principles of Distributed Computing (PODC 2020)*, 2020. URL http://arxiv.org/abs/2002.08216.

[6] 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 ACM Symposium on Theory of Computing (STOC 2016)*, pages 479–488. ACM Press, 2016. doi:10.1145/2897518.2897570.

[7] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model. In *Proc. 57th IEEE Symposium on Foundations of Computer Science (FOCS 2016)*, pages 615–624. IEEE, 2016. doi:10.1109/FOCS.2016.72.

[8] Mohsen Ghaffari and Hsin-Hao Su. Distributed Degree Splitting, Edge Coloring, and Orientations. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 2505–2523. Society for Industrial and Applied Mathematics, 2017. doi:10.1137/1.9781611974782.166.

[9] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 300–309, New York, New York, USA, 2004. ACM Press. doi:10.1145/1011767.1011811.

[10] 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, New York, New York, USA, 2006. ACM Press. doi:10.1145/1109557.1109666.

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

[12] Nathan Linial. Distributive graph algorithms – Global solutions from local data. In *Proc. 28th Annual Symposium on Foundations of Computer Science (FOCS 1987)*, pages 331–335. IEEE, 1987. doi:10.1109/SFCS.1987.20.

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

[14] 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.

[15] Dennis Olivetti. Round Eliminator: a tool for automatic speedup simulation, 2020. URL https://github.com/olidennis/round-eliminator.

[16] Dennis Olivetti and Jukka Suomela. Periodic points in round elimination, 2020. URL https://github.com/suomela/fixed-points.