Local Coordination and Symmetry Breaking

Jukka Suomela
Helsinki Institute for Information Technology HIIT,
Department of Information and Computer Science,
Aalto University, Finland · jukka.suomela@aalto.fi

Abstract

This article gives a short survey of recent *lower bounds* for *distributed graph algorithms*. There are many classical graph problems (e.g., maximal matching) that can be solved in $O(\Delta + \log^* n)$ or $O(\Delta)$ communication rounds, where n is the number of nodes and Δ is the maximum degree of the graph. In these algorithms, the key bottleneck seems to be a form of *local coordination*, which gives rise to the linear-in-Δ term in the running time. Previously it has not been known if this linear dependence is necessary, but now we can prove that there are graph problems that can be solved in time $O(\Delta)$ independently of n, and cannot be solved in time $o(\Delta)$ independently of n. We will give an informal overview of the techniques that can be used to prove such lower bounds, and we will also propose a roadmap for future research, with the aim of resolving some of the major open questions of the field.

1 Introduction

The research area of distributed computing studies computation in large computer networks. The fundamental research question is *what can be computed efficiently in a distributed system*. In distributed systems, communication is several orders of magnitude slower than computation: while a modern computer can perform arithmetic operations in a matter of *nanoseconds*, it can easily take dozens of *milliseconds* to exchange messages between two computers over the public Internet. To understand which computational tasks can be solved efficiently in a computer network, it is necessary to understand what can be solved with *very few communication steps*.

1.1 Model of Computing

Perhaps the most successful theoretical model for studying such questions is the LOCAL model [25,32]: We have an unknown graph G that represents a computer network. Every node of G is a computer and every edge of G is a communication link. Initially, each computer is only aware of its immediate surroundings. Computation proceeds in synchronous rounds; in each round, all nodes exchange messages with their neighbours. Eventually, all nodes have to stop and announce their local outputs—that is, their own part of the solution. For example, if we are studying graph colouring, each node has to output its own colour.

The *distributed time complexity* of a graph problem is the smallest t such that the problem can be solved with a distributed algorithm in t communication rounds. In the LOCAL model, parameter t plays a dual role—it represents both *time* and *distance*: in t rounds, all nodes can learn everything about graph G in their radius-t neighbourhood, and nothing else. In essence, a distributed algorithm with a running time of t is a mapping from radius-t neighbourhoods to local outputs—see Figure 1 for an illustration.

Therefore the defining property of fast distributed algorithms is *locality*: in a fast distributed algorithm each node only needs information from its local neighbourhood. In many cases, if we are given a graph problem, it is fairly easy to qualitatively classify it as a “local” or “global” problem; the key challenge is to understand precisely *how* local a given problem is.

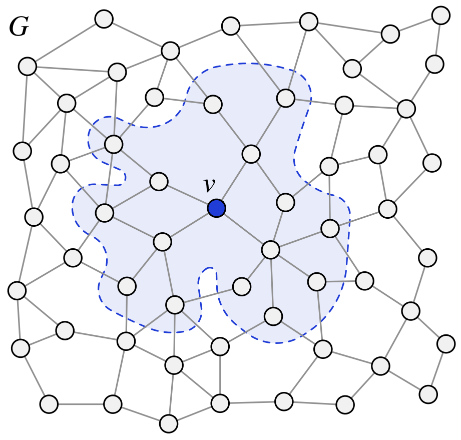


Figure 1: If we have a distributed algorithm with a running time of $t = 2$ rounds, then whatever node v outputs is a function of the information available in its radius-t neighbourhood (highlighted).

1.2 Symmetry Breaking vs. Local Coordination

Distributed time complexity and locality are commonly studied as a function of two parameters:

- n, the number of nodes in the network,
- Δ, the maximum degree of a node (the maximum number of neighbours).

As we will see, these parameters are often related to two different challenges in the area of distributed algorithms:

1. complexity as a function of n is related to *symmetry breaking*,
2. complexity as a function of Δ is related to *local coordination*.

The first aspect has been relatively well understood since Linial's seminal work in the 1990s [25]. However, our understanding of the second aspect has been poor—until very recently.

In this article, we will study the challenge of local coordination in more depth. We will survey recent work that has enabled us to better understand the distributed time complexity of certain graph problems not only as a function of n, but also as a function of Δ. Hopefully these preliminary results will provide a starting point for a research program that aims at resolving the long-standing open questions related to the distributed time complexity of classical graph problems such as maximal matchings, maximal independent sets, and graph colouring.

2 The LOCAL Model of Distributed Computing

To get started, let us first define the LOCAL model of distributed computing a bit more carefully. A reader familiar with the model can safely skip this section; further information can be found in many textbooks [27, 32, 34].

We will study distributed algorithms in the context of graph problems. We are given an unknown graph G, and the algorithm has to produce a feasible solution of the graph problem. Each node of graph G is a computational entity, and all nodes run the same distributed algorithm A. Eventually, each node v has to stop and produce its own part of the solution—the *local output* $A(G, v)$.

For example, if our goal is to find a proper vertex colouring of G, then the local output $A(G, v)$ will be the colour of node v in the solution. If our goal is to find a maximal matching, then the local output $A(G, v)$ will indicate whether v is matched and with which neighbour.

2.1 Synchronous Message Passing

In the LOCAL model, each node has a unique identifier, and initially each node knows only its own unique identifier and its degree in graph G. Computation proceeds in *synchronous communication rounds*. In every round, each node v that is still running performs the following steps, synchronously with all other nodes:

1. v sends a message to each of its neighbours,
2. v receives a message from each of its neighbours,
3. v updates its local state, and
4. possibly v stops and announces its local output $A(G, v)$.

The outgoing messages are a function of the old local state, and the new local state is a function of the old state and the messages that the node received.

The *running time* of an algorithm is defined to be the number of communication rounds until all nodes have stopped and announced their local outputs. This is the key difference between centralised and distributed computing: in the context of centralised algorithms we are interested in the number of *elementary computational steps*, while in the context of distributed algorithms the key resource is the number of *communication steps*. For our purposes, the cost of local computation is negligible.

2.2 Time and Distances Are Equivalent

In the LOCAL model, information can propagate at a maximum speed of 1 edge per round. If a node v stops after t rounds, then whatever v outputs can only depend on the information that was available within distance t from v in the input graph. In essence, a time-t algorithm in the LOCAL model is just a *mapping from radius-t neighbourhoods to local outputs*; recall Figure 1.

A bit more formally, let us write $x_t(v)$ for the local state of a node v after round t. Initially, $x_0(v)$ consists of just the unique identifier of node v and its degree. The key observation is that $x_{t+1}(v)$ is a function of $x_t(v)$ and the messages that v received from its neighbours during round $t + 1$. For each neighbour u of v, the message sent by u to v during round $t + 1$ is a function of $x_t(u)$. Hence the new state $x_{t+1}(v)$ is simply a function of the old states $x_t(v)$ and $x_t(u)$, where u ranges over the neighbours of v.

In essence, in each communication round, each node just updates its local state based on the local states in its radius-1 neighbourhood. By a simple induction, the local state $x_t(v)$ is a function of the initial states $x_0(u)$, where u ranges over all nodes that are within distance t from v. Hence we can see that if a node stops after round t, its local output can only depend on the information that was available within distance t from v.

Conversely, it is easy to design an algorithm in which all nodes can gather their radius-t neighbourhoods in t communication rounds. As a corollary, if graph G is connected and has a diameter of D, then in time $t = D + O(1)$ all nodes can learn everything about the structure of the input graph G. Therefore in time $t = D + O(1)$ we can also solve any computable graph problem: each node can simply gather full information on the input, and then locally solve the problem and output the relevant part of the solution. The diameter is at most $O(n)$, and therefore linear-time algorithms are entirely trivial in the LOCAL model; the research on the LOCAL model focuses on algorithms that run in sublinear time.

3 Example: Maximal Matchings

As a concrete example, let us consider the problem of finding a *maximal matching* with a distributed algorithm in the LOCAL model. Recall that a *matching* of a simple undirected graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that each node is incident to at most one edge of M, and a matching is *maximal* if it is not a proper subset of another matching.

3.1 Simple Distributed Algorithms

It is easy to come up with a distributed algorithm that finds a maximal matching. Here is a simple example:

- Initialise $M \leftarrow \emptyset$.
- Use the unique node identifiers to assign a unique label for each edge.
- Repeat until G is empty:
  - Find the set $X \subseteq E$ that consists of the edges that are *local minima* with respect to the edge labels, i.e., their labels are smaller than the labels of any of their neighbours.
  - Add X to M, and remove all edges adjacent to X from G.

Unfortunately, the running time of such an algorithm can be linear in n, which makes it uninteresting from the perspective of the LOCAL model.

In the above algorithm, a key drawback is that set X can be very small in the worst case. We can do better with a simple *randomised* distributed algorithm [1, 20, 26]. Instead of picking local minima, we can pick a random matching X. More precisely, each edge e attempts to join X with a certain (carefully chosen) probability, and if none of its neighbours attempts to join simultaneously, we will have $e \in X$. The basic idea is that we can eliminate a constant fraction of the edges in each step, and it can be shown that the running time will be $O(\log n)$ with high probability.

3.2 State of the Art

The fastest distributed algorithms can find maximal matchings much faster than in logarithmic time—at least in sparse graphs. To better characterise the running time, we will use two parameters: n, the number of nodes, and Δ, the maximum degree of the graph. As a function of n and Δ, currently the fastest algorithms are these:

- Barenboim et al. [7]: a randomised algorithm, time $O(\log \Delta + \log^4 \log n)$.
- Hańćkowiak et al. [18]: a deterministic algorithm, time $O(\log^4 n)$.
- Panconesi and Rizzi [31]: a deterministic algorithm, time $O(\Delta + \log^* n)$.

Here $\log^* n$ is the iterated logarithm of n, a very slowly growing function.

3.3 Focus on Low-Degree Graphs

In general, charting the landscape of distributed time complexity throughout the (n, Δ)-plane is an arduous task. In this article, we will zoom into one corner of the plane: we will focus on the case of $n \gg \Delta$.

In this region, the Panconesi–Rizzi algorithm, which runs in time $O(\Delta + \log^* n)$, is the fastest known algorithm for finding maximal matchings. While the expression of the running time may not look particularly natural, it turns out there are many other problems for which the fastest algorithms in sparse graphs have a running time of precisely $O(\Delta + \log^* n)$—the key examples are maximal independent sets, vertex colouring, and edge colouring [5, 6, 21, 31].

While many different algorithms with precisely the same running time exist, we do not know if any of these are optimal. In particular, it is not known if any of these problems could be solved in time $o(\Delta) + O(\log^* n)$.

In what follows, we will dissect the running time of the Panconesi–Rizzi algorithm in two terms, $O(\log^* n)$ and $O(\Delta)$, and give an intuitive explanation for each of them. To do this, we will consider two corner cases: one in which we can find maximal matchings in time $O(\log^* n)$, and one in which the problem can be solved in time $O(\Delta)$. As we will see, there is a very good justification for the term $O(\log^* n)$, but there is no lower bound that would justify the existence of the term $O(\Delta)$.

3.4 Maximal Matchings as a Symmetry-Breaking Problem

First, let us have a look at the term $O(\log^* n)$. Maximal matchings are fundamentally a *symmetry-breaking problem.* To see this more clearly, we will define a restricted version in which we are left with a *pure* symmetry-breaking problem; in essence, we will eliminate the term Δ from the running time.

This turns out to be easy: we can simply focus on cycle graphs, in which case we have a constant maximum degree of $\Delta = 2$. We are now left with the seemingly trivial problem of finding a maximal matching in a cycle.

This special case highlights the need for symmetry breaking. While the topology of the input graph is symmetric, the output cannot be symmetric: any maximal matching has to contain some of the edges, but not all of them. We will have to resort to some means of symmetry breaking.

Problems of this type can be solved efficiently with the help of the Cole–Vishkin algorithm [8]. This is a deterministic distributed algorithm that can be used to e.g. colour a cycle with $O(1)$ colours in time $O(\log^* n)$. The algorithm relies heavily on the existence of unique node identifiers; it extracts symmetry-breaking information from the unique identifiers.

In essence, the Cole–Vishkin algorithm is a *colour reduction algorithm.* The unique identifiers provide a colouring with a large number of colours. Typically it is assumed that the range of unique identifiers is polynomial in n, and hence our starting point is a poly(n)-colouring of the cycle. The Cole–Vishkin algorithm then reduces the number of colours from any number ℓ to $O(\log \ell)$ in one step; it compares the *binary representations* of the old colours of adjacent nodes, and uses the index of the bit that differs, together with its value, to construct a new colour. Overall, we need only $O(\log^* n)$ iterations to reduce the number of colours to $O(1)$.

Once we have a colouring of the vertices with $O(1)$ colours, it is easy to find a maximal matching. For example, we can use the vertex colouring to derive an edge colouring. Then each colour class is a matching, and we can construct a maximal matching by considering the colour classes one by one. Hence the pure symmetry-breaking version of maximal matchings can be solved in time $O(\log^* n)$.

Remarkably, this is also known to be optimal. Linial's seminal lower bound [25] shows that symmetry breaking in a cycle requires $\Omega(\log^* n)$ rounds with deterministic algorithms, and Naor [28] shows that this holds even if we consider randomised algorithms.

In summary, pure symmetry breaking problems are very well understood. As a simple corollary, it is not possible to find a maximal matching in time $O(\Delta) + o(\log^* n)$, or in time $f(\Delta) + o(\log^* n)$ for any function f.

3.5 Maximal Matchings as a Coordination Problem

Let us now turn our attention to the term $O(\Delta)$. We will argue that maximal matchings are also a *coordination problem*. To see this more clearly, we will again define a restricted version in which we are left with a *pure* coordination problem, which can be solved in time $O(\Delta)$ independently of n.

Of course we cannot simply set $n = O(1)$ to get rid of the term $O(\log^* n)$. However, we can consider a situation in which we have already solved the symmetry-breaking problem. We will *assume* that we are given a *bipartite graph*, in which one part is coloured black, another part is white. Our goal is to find a maximal matching in such a graph.

In essence, each black node should try to pick one of its white neighbours as its partner, and conversely each white node should try to pick one of its black neighbours as its partner. Here we have the coordination challenge: without any coordination, several black nodes may try to pick the same white node simultaneously.

In bipartite graphs, this coordination problem can be solved in time $O(\Delta)$ with a simple proposal algorithm [17]. The algorithm repeatedly performs the following steps:

1. Black nodes send "proposals" to their white neighbours, one by one (e.g. in the order of their unique identifiers).

2. White nodes "accept" the first proposal that they get (breaking ties with e.g. the unique identifiers of the senders), and "reject" all other proposals.

Each proposal–acceptance pair forms an edge in the matching. A black node will stop as soon as it becomes matched, or it runs out of white neighbours to whom to send proposals (in which case all of the neighbours are already matched). A white node will stop as soon as it becomes matched, or all of its black neighbours have stopped (in which case all of the neighbours are already matched). Clearly the output is a maximal matching.

While the simple proposal algorithm runs in time $O(\Delta)$ independently of n, it is not known if it is optimal. More specifically, it is not known if we can find a maximal matching in bipartite 2-coloured graphs in time $o(\Delta)$ independently of n. However, this is the best algorithm that we currently have for this problem; we have not been able to break the linear-in-Δ boundary (without introducing some dependence on n in the running time). Many other distributed algorithms have a similar issue at their core: in one way or another, there is a need for local coordination, and this is resolved in a fairly naive manner by considering neighbours one by one.

While symmetry breaking is well understood, local coordination is poorly understood. There are hardly any lower bounds. For bipartite maximal matchings we can apply the lower bound by Kuhn et al. [22–24], which shows that bipartite maximal matching requires $\Omega(\log \Delta)$ rounds in the worst case. However, this still leaves an *exponential* gap between the upper bound and the lower bound.

A particularly intriguing special case is that of *regular graphs*, i.e., the case that all nodes have the same degree of Δ. In this case, the simple proposal algorithm is still the fastest algorithm that we have, and hence the best known upper bound is still $O(\Delta)$. Somewhat surprisingly, there are no lower bounds at all for this case; the above-mentioned lower bound by Kuhn et al. cannot be applied here any more. Hence we have very simple coordination problems whose distributed time complexity is not understood at all.

4 Towards Coordination Lower Bounds

We have seen that in low-degree graphs, the fastest algorithm for maximal matchings has a running time of $O(\Delta + \log^* n)$. Here the term $O(\log^* n)$ is related to the well-known challenge of symmetry breaking, while the term $O(\Delta)$ appears to be related to a poorly-understood challenge of local coordination. The problem cannot be solved in time $O(\Delta) + o(\log^* n)$, but it is a major open question if it can be solved in time $o(\Delta) + O(\log^* n)$. Let us now have a look at possible ways towards answering this question.

4.1 Completable but Tight Problems

As we have discussed, maximal matchings are not an isolated example. There are numerous other graph problems for which the time complexity as a function of n is well-understood (at least for a small Δ), but the time complexity as a function of Δ is not really understood at all.

Perhaps the most prominent example is proper vertex colouring with $\Delta + 1$ colours [5, 6, 21]. This is yet another problem that can be solved in $O(\Delta + \log^* n)$ time, and it is known to require $\Omega(\log^* n)$ time, but it is not known if the problem can be solved in $o(\Delta) + O(\log^* n)$ time.

Other examples of such problems include edge colouring with $2\Delta - 1$ colours, and the problem of finding a maximal independent set. Incidentally, all of these problems can be characterised informally as follows:

1. Any partial solution can be *completed*. In particular, a greedy algorithm that considers nodes or edges in an arbitrary order can solve these problems.

2. However, there are situations in which a partial solution is *tight* in the sense that there is only one way to complete it. For example, if we have already

coloured all Δ neighbours of a node, and we have a colour palette of size $\Delta + 1$, we may have only 1 possible colour left.

While some counterexamples exist, it seems that many problems of this form have distributed algorithms with a running time of, e.g., $O(\Delta + \log^* n)$ or simply $O(\Delta)$, and we do not know if these algorithms are optimal. However, as soon as we step outside the realm of "completable but tight" problems, we see plenty of examples of running times that are either sublinear or superlinear in Δ:

1. If we drop the first restriction, running times typically increase to *superlinear* in Δ. Examples of such problems include matchings without short augmenting paths [2], which can be solved in time $\Delta^{O(1)}$, and locally optimal semi-matchings [9], which can be solved in time $O(\Delta^5)$.

2. If we drop the second restriction, there are many problems that can be solved in time *sublinear* in Δ. A good example is vertex colouring with $O(\Delta^2)$ colours. With such a large colour palette, local coordination becomes much easier: Linial's algorithm [25] solves this problem in time $O(\log^* n)$, independently of Δ.

Hence to resolve the long-standing open questions related to the distributed time complexity, it seems that we need to better understand the issue of local coordination in the context of "completable but tight" problems.

4.2 Major Hurdles and Recent Advances

Looking back at the previous attempts to prove e.g. tight lower bounds for the distributed time complexity of maximal matchings, it now seems that one of the major hurdles can be summarised as follows:

1. We do not understand the issue of local coordination even in isolation.

2. To prove tight lower bounds, we would need to understand not just local coordination in isolation, but also the complicated interplay of local coordination and symmetry breaking.

While the second issue seems to be still far beyond the reach of current research, we are now finally making progress with the first issue. The key idea is to identify *pure coordination problems*. These are "completable but tight" problems that can be solved in time $O(\Delta)$ independently of n, but cannot be solved in time $o(\Delta)$ independently of n. Such problems do not need any symmetry breaking, and hence we can focus solely on local coordination.

Now we finally have the first natural example of a pure coordination problem for which we can prove tight lower bounds: *maximal fractional matching*. This

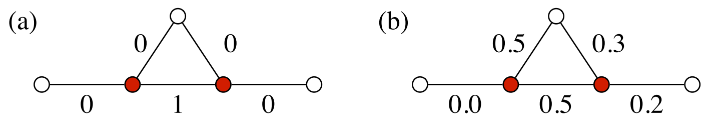


Figure 2: (a) A maximal matching. (b) A maximal fractional matching. The saturated nodes are highlighted.

was known to be solvable in time $O(\Delta)$ independently of n [3], and there is now a lower bound that shows that the problem cannot be solved in time $o(\Delta)$ independently of n [14]. In the next section, we will discuss this problem in more depth.

5 Recent Progress: Maximal Fractional Matchings

A *fractional matching* is a linear programming relaxation of a matching. While maximal matchings are a symmetry-breaking problem, it turns out that the problem of finding a *maximal fractional matching* is a pure coordination problem—it does not require any symmetry breaking.

5.1 Problem Formulation

If a matching can be interpreted as an integer-valued function $y\colon E \to \{0, 1\}$ that indicates which edges are in the matching, a fractional matching is a real-valued function $y\colon E \to [0, 1]$.

Formally, let $G = (V, E)$ be a simple undirected graph and let $y\colon E \to [0, 1]$ associate weights to the edges of G. Define for each $v \in V$ the sum of incident edges

$$y[v] := \sum_{e \in E : v \in e} y(e).$$

The function y is called a *fractional matching* if $y[v] \leq 1$ for each node v. A node v is *saturated* if $y[v] = 1$. A fractional matching is *maximal* if each edge has at least one saturated endpoint.

Informally, in a maximal fractional matching we cannot increase the weight of any edge without violating a constraint. See Figure 2 for examples.

In combinatorial optimisation, maximal fractional matchings and maximal matchings have similar applications. It is well known that we can use a maximal matching to construct a 2-approximation of a minimum vertex cover. It turns out that we can use maximal fractional matchings equally well: in any maximal fractional matching, the set of saturated nodes forms a 2-approximation of a minimum vertex cover [4].

5.2 No Need for Symmetry Breaking

From the perspective of centralised algorithms, the distinction between maximal matchings and maximal fractional matchings is not particularly interesting; either of the problems can be solved easily in linear time. However, the two problems are very different from the perspective of efficient distributed or parallel algorithms.

While the problem of finding a maximal matching requires symmetry breaking (recall Section 3.4), this is not the case with maximal fractional matchings. Consider, for example, a k-regular graph. In any such graph, there is a trivial maximal fractional matching that sets $y(e) = 1/k$ for all $e \in E$. In essence, highly symmetric graphs are trivial from the perspective of maximal fractional matchings; only non-symmetric inputs require some effort.

The general case is not that easy to solve efficiently, but it turns out that there is an algorithm [3] that finds a maximal fractional matching in $O(\Delta)$ time in the LOCAL model. The running time does not have any dependence on n, but it is still linear in Δ, as the algorithm resorts to a fairly naive one-by-one approach to overcome challenges related to local coordination.

In summary, we can characterise the state-of-the-art algorithms for these two problems as follows:

1. Maximal matchings:
   - symmetry breaking necessary
   - algorithms resort to local coordination
   - time $O(\Delta + \log^* n)$.
2. Maximal fractional matchings:
   - symmetry breaking not needed
   - algorithms resort to local coordination
   - time $O(\Delta)$.

5.3 A New Lower Bound

Maximal fractional matchings are a genuine example of a pure coordination problem. They are a "completable but tight" problem, in the sense discussed in Section 4.1. They do not need any symmetry breaking. Most importantly, now we can show that the $O(\Delta)$-time algorithm is optimal (at least for sparse graphs): the problem cannot be solved in time $o(\Delta)$, independently of n, with any distributed algorithm (deterministic or randomised).

Before we continue, it is important to emphasise that the result does not yet tell anything more than what is stated above. In particular, it has not yet been ruled out that there could exist a sublinear-in-Δ algorithm whose running time depends moderately on n. For example, algorithms of a running time $o(\Delta) + O(\log^* n)$ cannot be yet excluded. This is also the reason why this result does not tell anything interesting about maximal matchings: a simple corollary would be that maximal matchings cannot be found in time $o(\Delta)$ independently of n, but this we already know, as any algorithm for maximal matchings has a running time $\Omega(\log^* n)$ that certainly depends on n. Nevertheless, this result demonstrates that there are techniques with which we can prove tight lower bounds for pure coordination problems, and this hopefully paves the road for future research in which we can tackle also algorithms with running times that have a moderate dependence on n.

A key insight in the proof is that we will not try to prove the result directly for the LOCAL model, but we will first consider *weaker* models of distributed computing (Sections 5.4 and 5.7). In weaker models, lower bounds are of course easier to prove but less interesting. Only then we will amplify the result so that we have similar lower bounds also for the LOCAL model.

On a high level, the proof builds on the following techniques:

1. The *unfold-and-mix* technique for proving lower bounds in very weak models of distributed computing (Section 5.6).

2. A general technique for amplifying lower bounds from weak models to the usual LOCAL model (Section 5.8). The main ingredient here is the construction of so-called *homogeneous graphs* (Section 5.9).

Both of these techniques were originally presented in PODC 2012 [13, 19], but it was not until PODC 2014 [14] that we managed to put these techniques together in order to prove lower bounds for the maximal fractional matching problem in the usual LOCAL model.

5.4 Port-Numbering Model and Edge Colouring Model

We start by introducing two models of distributed computing: the *port-numbering model*, in short PN, and the *edge-colouring model*, in short EC. While at least the PN model is also interesting in its own right, for our purposes these models serve as a stepping stone towards more interesting results—lower bounds in the LOCAL model.

In many ways, the PN and EC models are similar to the usual LOCAL model: Each node is an autonomous entity that maintains its own local state. Computation proceeds in synchronous rounds. In each round, all nodes in parallel (1) send messages to their neighbours, (2) receive messages from their neighbours, (3) update their local state, and (4) possibly announce their local output and stop.

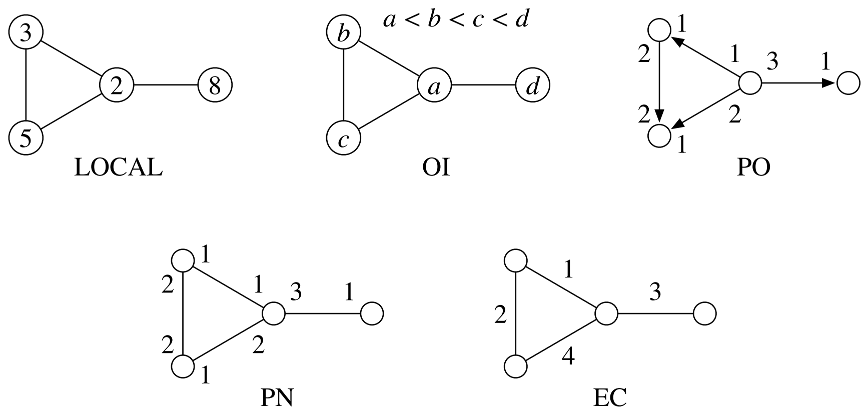
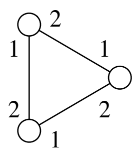


Figure 3: Models LOCAL, OI, PO, PN, and EC.

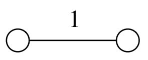
The key difference is related to the initial information that is available in the network. In PN and EC, the nodes are *anonymous*—they do not have any unique identifiers. However, each node can still distinguish between their neighbours; see Figure 3:

- In the PN model, the endpoints of the edges (“ports”) are numbered so that a node of degree d is incident to endpoints with numbers $1, 2, \ldots, d$. In essence, a node can refer to its neighbours by numbers $1, 2, \ldots, d$.
- In the EC model, the edges are properly coloured with $O(\Delta)$ colours. Each node knows the colours of the incident edges, and it can use the colours to refer to its neighbours.

Note that EC is strictly stronger than PN. Given an edge colouring, it is easy to derive a port numbering. The converse is not true: for example, in the EC model it is trivial to find an edge colouring, but in the PN model it is not possible in general with any deterministic algorithm. To see this, consider a cycle with a symmetric port numbering; no PN-algorithm can break the symmetry here:



However, note that EC is also a fairly weak model. For example, no deterministic algorithm can break the symmetry in a graph with just two nodes:



In particular, it is not possible to find a proper vertex colouring with either PN or EC algorithms here. While everything was solvable in linear time in the LOCAL model, there are numerous seemingly trivial problems that cannot be solved at all in PN or EC.

5.5 Maximal Matching in the EC Model

Maximal matching cannot be solved with deterministic algorithms in the PN model. However, in the EC model, there is a very simple greedy algorithm for finding a maximal matching M in time $O(\Delta)$. We consider each colour class $1, 2, \ldots, O(\Delta)$ one by one. In step i, we find all edges of colour i that are not yet adjacent to any edge of M, and add them to M. Clearly, the end result is a maximal matching.

A key observation at this point is that maximal matchings in the EC model do not need any symmetry breaking. Symmetry between adjacent edges is already broken by the edge colouring. We only need to deal with local coordination, in order to avoid adding two adjacent edges simultaneously to M.

The greedy algorithm solves the local coordination challenge by a very naive technique: it considers colour classes one by one, which results in a linear-in-Δ running time. The key question is now if this algorithm is optimal in the EC model. Perhaps e.g. a clever divide-and-conquer approach could solve the problem in only $O(\log \Delta)$ time?

It turns out the greedy algorithm is indeed optimal. We can show that there is no algorithm that finds a maximal matching in $o(\Delta)$ time in the EC model [19]. This was the first linear-in-Δ lower bound for a natural coordination problem.

5.6 Unfold and Mix

We will now give an overview of the techniques that can be used to prove lower bounds for the maximal matching problem in the EC model. Assume that we have a deterministic distributed algorithm A that finds a maximal matching in any given graph, for any given edge colouring. With this knowledge, we can construct "fragile" instances in which algorithm A is forced to produce *perfect matchings*. Informally, perfect matchings are more tightly constrained than maximal matchings; minor changes in the input may cause major changes in the output.

To force algorithm A to produce a perfect matching, we will to study graphs with self-loops. For our purposes, a graph G with self-loops is just a compact representation of a large (possibly infinite) graph G' that does not have any loops. To construct G', we just "unfold" all loops of G; see Figure 4.

Each self-loop represents *symmetry*. If e is a self-loop in graph G, and we unfold e to construct graph G', then G' will be symmetric with respect to e. More precisely, in the EC model deterministic algorithms cannot distinguish between

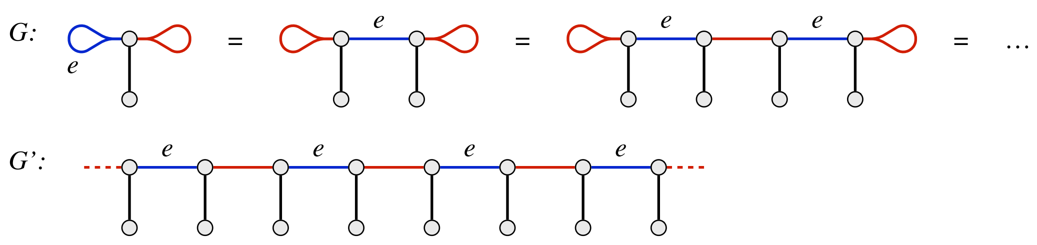


Figure 4: Graph G with self-loops is a compact representation of graph G'.

the two endpoints of e. If one of the endpoints produces the local output indicating that it is unmatched, the other endpoint will produce the same output—but this is a contradiction, as in a maximal matching we cannot have a pair of adjacent nodes such that both of them are unmatched. Therefore all nodes that have self-loops must be always matched. As long as we have at least one self-loop attached to each node, the output will be a perfect matching.

To prove the lower bound of $\Omega(\Delta)$, we will study *critical pairs*. We say that G and H form an r-critical pair of graphs if:

- G has a self-loop e and H has a self-loop f,
- loops e and f have the same colour,
- the radius-r neighbourhoods of e and f are isomorphic,
- algorithm A makes a different decision for e and f.

By a different decision we mean that in graph G algorithm A outputs a matching $A(G)$ with $e \in A(G)$, and in graph H algorithm A outputs a matching $A(H)$ with $f \notin A(H)$. Naturally, if such a pair exists, then the running time of A has to be at least r; otherwise A would not be able to distinguish between e and f.

For any algorithm A, it is fairly straightforward to find a 0-critical pair that consists of just a pair of nodes so that both of them have $\Theta(\Delta)$ self-loops. Once we have a 0-critical pair, we will apply a technique called *unfold-and-mix* repeatedly. In each iteration we will lose some self-loops but gain criticality. We can repeat the process for $\Theta(\Delta)$ times until we run out of self-loops. The end result will be a $\Theta(\Delta)$-critical pair of graphs of maximum degree Δ. The existence of such a pair is enough to demonstrate that the running time of algorithm A is $\Omega(\Delta)$.

The unfold-and-mix technique is illustrated in Figure 5. In each inductive step, our starting point is a k-critical pair of graphs, G and H, with the special loops e and f. Then we

1. "unfold" the loops e and f to obtain graphs GG and HH,

2. "mix" the graphs GG and HH together to obtain another graph GH that combines elements from G and H.

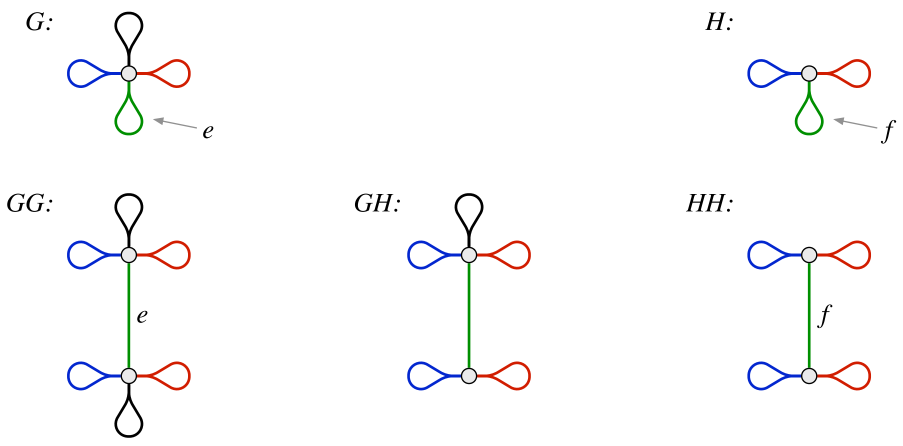


Figure 5: The unfold-and-mix technique.

Intuitively, the output of A in GG and the output of A in HH are not compatible with each other—by assumption, the outputs of e and f are different. Hence the algorithm is now in trouble:

- It has to do something different in GH in comparison with what it did in either GG or HH.
- It cannot sweep the problem under the rug easily, as the instances are "fragile": graph GH still has self-loops, and therefore the algorithm has to output a perfect matching.

With some effort, we can now show that among the three graphs that we have constructed (GG, GH, and HH), we can always find at least two that satisfy the conditions of a $(k + 1)$-critical pair.

In essence, each unfold-and-mix step makes the problem instance more difficult from the perspective of the algorithm that we study: the algorithm has to look further (i.e., spend more time) in order to distinguish the two graphs that form a critical pair. This way we can show that algorithm A cannot find a maximal matching in time $o(\Delta)$. Similar ideas can be used to prove an analogous lower bound also for maximal fractional matchings.

5.7 More Models

So far we have seen how to prove tight lower bounds for coordination problems in the EC model. However, we are interested in the usual LOCAL model, and the EC model and the LOCAL model are very different from each other.

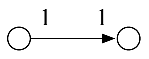
We will introduce two new models that serve as intermediate steps that bridge the gap between the EC model and the LOCAL model; see Figure 3:

- *Port numbering and orientation* (PO): The model is a stronger version of the PN model. In addition to the port numbering, we are also given an *orientation*, i.e., for each edge one of the endpoints is labelled as the head. The orientation does not restrict how we can send information in the network; it is just additional symmetry-breaking information that the algorithm can use.

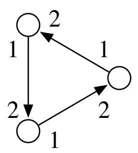
- *Order-invariant algorithms* (OI): This model is a weaker version of the LOCAL model. The nodes have unique identifiers, but algorithms can only use the relative order of the nodes and not the numerical values of the identifiers. Put otherwise, if we relabel the nodes but preserve their relative order, the output of the algorithm must not change.

For randomised algorithms these models are not that interesting, as a randomised algorithm can use random bits to e.g. generate labels that are unique with high probability (at least if we have some estimate of the size of the network). However, for deterministic algorithms the differences between the models PN, PO, OI, and LOCAL become interesting.

First, we can see that the PO model is strictly stronger than the PN model. For example, in a graph with just one edge, a PO algorithm can use the orientation to break the symmetry between the endpoints, while in PN this is not possible:



We can also see that OI is strictly stronger than PO. The OI model is clearly at least as strong as the PO model: the ordering of the nodes can be used to derive both a port numbering and an orientation. Moreover, in the OI model we can always break symmetry e.g. in a cycle (there is always a unique node that is smaller than any other node), while this is not necessarily the case in the PO model:



Finally, there are many problems that can be solved faster in the LOCAL model than in the OI model. Maximal matchings in a cycle are a good example: it takes $\Theta(n)$ time to find a maximal matching in a cycle in the OI model in the worst case, but as we have discussed in Section 3.4, this is possible in $\Theta(\log^* n)$ time in the LOCAL model. In summary, for deterministic algorithms the relative strengths of the models are PN $\subsetneq$ PO $\subsetneq$ OI $\subsetneq$ LOCAL and PN $\subsetneq$ EC.

5.8 Amplifying Lower Bounds

We have seen above that in general, the PO model can be much weaker than the LOCAL model. Indeed, there are many algorithms that exploit the properties of the LOCAL model in order to solve graph problems efficiently. For example, numerous algorithms that solve symmetry breaking in time $O(\log^* n)$ specifically rely on the unique identifiers and cannot be used in the PO model.

However, if we have a look at problems that can be solved in time $f(\Delta)$ independently of n—for example, pure coordination problems—the situation looks very different. As we can see from the survey [33], for numerous classical graph problems, the best $f(\Delta)$-time deterministic approximation algorithms in the LOCAL model do not make any use of unique identifiers. We could easily run the same algorithms in the PO model as well (and sometimes also in the PN model).

It turns out that this is not just a coincidence. We can now prove that the PO, OI, and LOCAL models are equally strong, at least in the following case [13]:

1. We use distributed algorithms with a running time of $f(\Delta)$ for some f, independently of n.

2. We are interested in so-called *simple* PO-*checkable graph optimisation problems*. This includes many classical packing and covering problems such as vertex covers, edge covers, matchings, independent sets, dominating sets, and edge dominating sets.

To prove that PO and LOCAL are equally strong for this family of problems, we proceed in two steps, using the OI model as an intermediate step:

- PO $\approx$ OI: We introduce so-called *homogeneous graphs*, in which nodes are ordered so that the ordering provides as little additional information as possible. There is only a small fraction of nodes for which OI algorithms may have an advantage over PO algorithms. See Section 5.9 for more details.

- OI $\approx$ LOCAL: We apply *Ramsey's theorem* [16] to assign unique identifiers in an unhelpful manner, so that algorithms in the LOCAL model do not have any advantage over OI algorithms.

The use of Ramsey's theorem in such a context is nowadays a standard technique [10, 29]. The key novelty is the introduction of homogeneous graphs, and in particular, a proof that shows that finite high-girth high-degree homogeneous graphs indeed exist.

5.9 Key Ingredient: Homogeneous Graphs

We introduce the following technical definition that is helpful in the context of the OI model. We say that graph G is $(1 - \epsilon, r)$-homogeneous if the nodes can be ordered so that a fraction $1-\epsilon$ of all radius-r neighbourhoods are isomorphic, with respect to both topology and ordering. If we have such an ordering of the nodes, then any OI-algorithm with a running time at most r will be in trouble: almost all neighbourhoods look identical.

To show that models PO and OI are equally strong for any $f(\Delta)$-time algorithm, for a broad range of graph problems, it is desirable to have graphs with the following properties, for any g, r, k, and $\epsilon > 0$:

(1) the graph is $(1 - \epsilon, r)$-homogeneous,
(2) the graph is $2k$-regular,
(3) the graph has girth at least g, and
(4) the graph is finite.

It turns out that satisfying any three out of the four properties is easy—see Figure 6 for examples:

(a) Regular high-girth graphs satisfy all properties except (1).
(b) Cycles satisfy all properties except (2).
(c) Regular grids satisfy all properties except (3).
(d) Infinite trees satisfy all properties except (4).

However, satisfying all four properties simultaneously is more challenging. The construction that we use in our recent work [15] is based on the Cayley graphs of certain groups (variants of so-called iterated wreath products) that have several desirable properties: moderate growth, relatively large girth [12], and a convenient geometric embedding.

5.10 Putting Everything Together

With the help of homogeneous graphs we have been able to show that the models PO, OI, and LOCAL are equally strong from the perspective of $f(\Delta)$-time algorithms. We also know that maximal matchings take $\Omega(\Delta)$ time in the EC model. Ideally, we would now like to put the two results together and show that maximal matchings cannot be solved in $o(\Delta) + O(\log^* n)$ time in the LOCAL model, either.

Unfortunately, we do not know how to do this yet. Even if we put aside the issue of somehow bridging the gap between the EC model and the PO model, we have a much bigger obstacle in front of us: the term $O(\log^* n)$ in the running time is enough to separate the models PO and LOCAL, and kill the argument of Section 5.8. In essence, we still cannot deal with symmetry-breaking problems.

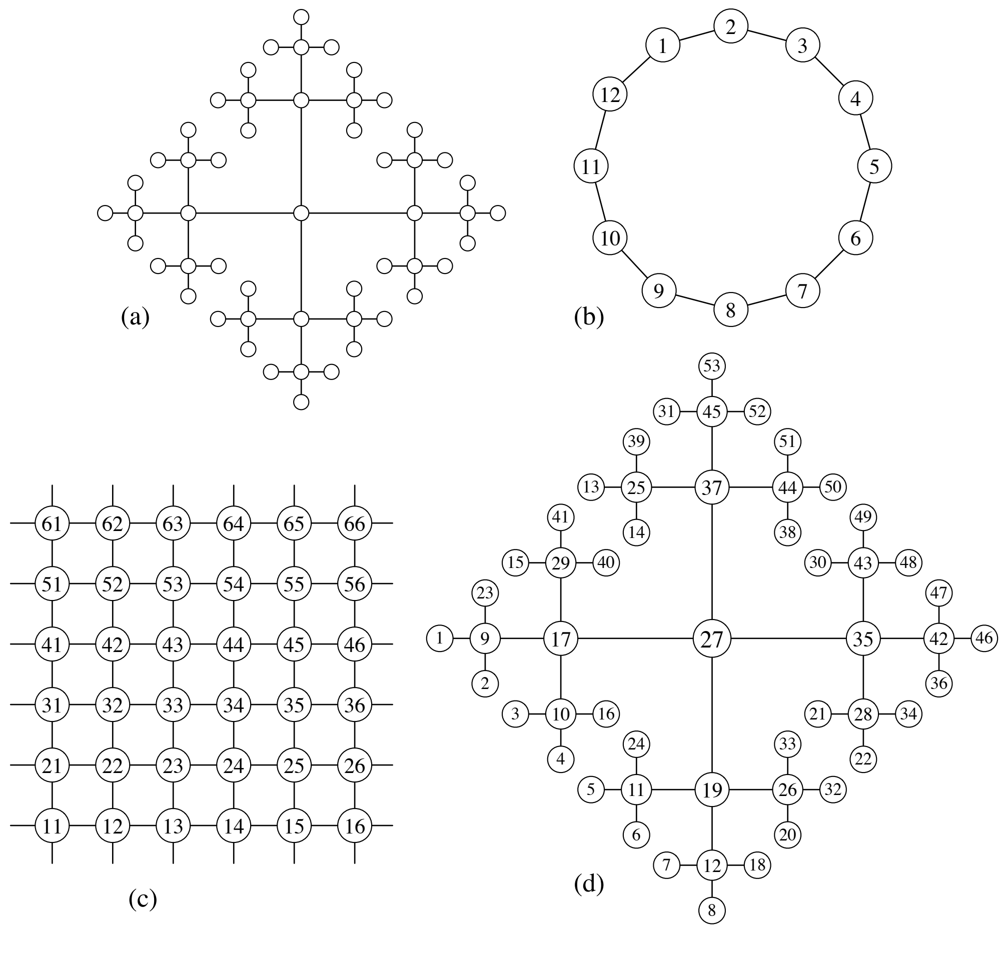


Figure 6: Examples of graphs that satisfy some of the desirable properties from Section 5.9.

However, what we can do is to put these ingredients together to prove a lower bound for maximal *fractional* matchings. We can show that maximal fractional matchings cannot be solved in time $o(\Delta)$ independently of n in the LOCAL model—recall that there is a matching upper bound of $O(\Delta)$. There are many technical details to be taken care of, but the key steps are as follows [14]:

- Prove a lower bound for the EC model, with the help of the unfold-and-mix technique (Section 5.6).
- EC $\approx$ PO: Use the properties of maximal fractional matchings to show that a lower bound for EC implies a lower bound for PO.
- PO $\approx$ OI: Use homogeneous graphs (Section 5.9) to show that a lower bound for PO implies a lower bound for OI.
- OI $\approx$ LOCAL: Use Ramsey's theorem to show that a lower bound for OI implies a lower bound for LOCAL.

Finally, we can prove that in the LOCAL model, randomised $f(\Delta)$-time algorithms cannot do any better than deterministic algorithms with this kind of graph problems, and we have the theorem that we have been looking for: even if we use randomised algorithms, and even if we can exploit the full power of the LOCAL model, there is no distributed algorithm that finds a maximal fractional matching in time $o(\Delta)$ independently of n.

6 Conclusions

So far we have identified the first pure coordination problem—maximal fractional matchings—with the following properties: the problem can be solved in time $O(\Delta)$ independently of n, and there is a matching lower bound showing that it cannot be solved in time $o(\Delta)$ independently of n.

Intuitively, the unfold-and-mix technique shows that there is need for local coordination with nodes that are at distance up to $\Theta(\Delta)$. All other parts of the proof—e.g. homogeneous graphs and Ramsey's theorem—are just technicalities that are needed to show that algorithms cannot "cheat" somehow by exploiting unique node identifiers and randomness.

6.1 Current Obstacles

Our hope is that we could prove that many other problems—for example, maximal matchings—also have a similar source of hardness that is related to local coordination with nodes at distance up to $\Theta(\Delta)$. We conjecture that, for example, maximal matching cannot be solved in time $o(\Delta) + O(\log^* n)$ with any algorithm. In essence, we believe that the Panconesi–Rizzi algorithm [31] with a running time of $O(\Delta + \log^* n)$ is optimal for $n \gg \Delta$.

Currently, there seem to be two obstacles that prevent us from proving such a theorem, both of which are related to the term $\Theta(\log^* n)$ in the lower bound that we are looking for.

1. The final step OI $\approx$ LOCAL (Section 5.8): The Ramsey-based argument that we use to show that OI and LOCAL are equally strong fails for $\Theta(\log^* n)$-time algorithms.

2. The starting point in the EC model (Section 5.6): In time $\Theta(\log^* n)$ an algorithm can find a vertex colouring that breaks the symmetry between adjacent nodes. Therefore it is no longer possible to use self-loops to construct instances that are "fragile".

However, once again we can try to deal with these two obstacles one by one. It turns out that there is a natural graph problem that would let us focus on the second obstacle first—the problem is bipartite maximal matching that we already discussed in Section 3.5.

6.2 Roadmap for the Future

Recall that maximal matching in bipartite 2-coloured graphs can be solved in time $O(\Delta)$ independently of n. We conjecture that bipartite maximal matchings are a pure coordination problem that cannot be solved in time $o(\Delta)$ independently of n. The current techniques are not yet sufficient to prove it, but it seems that we are now facing just one obstacle—how to use the unfold-and-mix technique to construct "fragile" instances even in the presence of an edge colouring that breaks symmetry.

This suggests the following roadmap for future research:

1. Extend the unfold-and-mix technique so that we can prove a linear-in-Δ bound for bipartite maximal matchings.

2. Then extend the Ramsey-based argument so that we can prove a similar bound for maximal matchings in general.

3. Then extend the techniques so that we can prove similar bounds for other coordination problems, for example, independent sets, vertex colourings, and edge colourings.

This seems to be a long road ahead, but it could lead to a resolution of major open questions related to distributed time complexity. Such results could find applications also in other areas of theoretical computer science. In prior work, tight lower bounds for distributed symmetry breaking have implied tight lower bounds for e.g. decision tree complexity and models of parallel computing [11, 30], and perhaps tight lower bounds for local coordination would find similar applications.

Acknowledgements

This article is based on the material that I presented in the ADGA 2014 workshop, `http://adga2014.hiit.fi/`. Many thanks to Christoph Lenzen for inviting me to give the talk, to the workshop participants for discussions, to Stefan Schmid for asking me to write this article and for his feedback on it, to Przemysław Uznański and Tuomo Lempiäinen for their helpful comments, and to my coauthors Mika Göös and Juho Hirvonen without whom the results described in this article would not even exist.

References

[1] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986. `doi:10.1016/0196-6774(86)90019-2`.

[2] Matti Åstrand, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, and Jara Uitto. Local algorithms in (weakly) coloured graphs, 2010. `arXiv:1002.0125`.

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

[4] Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981. `doi:10.1016/0196-6774(81)90020-1`.

[5] Leonid Barenboim and Michael Elkin. Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. In *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC 2009)*, pages 111–120. ACM Press, 2009. `doi:10.1145/1536414.1536432`.

[6] Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Morgan & Claypool, 2013. `doi:10.2200/S00520ED1V01Y201307DCT011`.

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

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

[9] Andrzej Czygrinow, Michał Hańćkowiak, Edyta Szymańska, and Wojciech Wawrzyniak. Distributed 2-approximation algorithm for the semi-matching problem. In *Proc. 26th International Symposium on Distributed Computing (DISC 2012)*, volume 7611 of *Lecture Notes in Computer Science*, pages 210–222. Springer, 2012. doi:10.1007/978-3-642-33651-5_15.

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

[11] Faith E. Fich and Vijaya Ramachandran. Lower bounds for parallel computation on linked structures. In *Proc. 2nd Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 1990)*, pages 109–116. ACM Press, 1990. doi:10.1145/97444.97676.

[12] Alex Gamburd, Shlomo Hoory, Mehrdad Shahshahani, Aner Shalev, and Balint Virág. On the girth of random Cayley graphs. *Random Structures & Algorithms*, 35(1):100–117, 2009. doi:10.1002/rsa.20266.

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

[14] Mika Göös, Juho Hirvonen, and Jukka Suomela. Linear-in-Δ lower bounds in the LOCAL model. In *Proc. 33rd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2014)*, pages 86–95. ACM Press, 2014. doi:10.1145/2611462.2611467. arXiv:1304.1007.

[15] Mika Göös and Jukka Suomela. No sublogarithmic-time approximation scheme for bipartite vertex cover. *Distributed Computing*, 27(6):435–443, 2014. doi:10.1007/s00446-013-0194-z. arXiv:1205.4605.

[16] Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer. *Ramsey Theory*. John Wiley & Sons, New York, 1980.

[17] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1998)*, pages 219–225. Society for Industrial and Applied Mathematics, 1998.

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

[19] Juho Hirvonen and Jukka Suomela. Distributed maximal matching: greedy is optimal. In *Proc. 31st Annual ACM Symposium on Principles of Distributed Computing (PODC 2012)*, pages 165–174. ACM Press, 2012. doi:10.1145/2332432.2332464. arXiv:1110.0367.

[20] Amos Israeli and Alon Itai. A fast and simple randomized parallel algorithm for maximal matching. *Information Processing Letters*, 22(2):77–80, 1986. doi:10.1016/0020-0190(86)90144-4.

[21] Fabian Kuhn. Weak graph colorings: distributed algorithms and applications. In *Proc. 21st Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2009)*, pages 138–144. ACM Press, 2009. doi:10.1145/1583991.1584032.

[22] 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. ACM Press, 2004. doi:10.1145/1011767.1011811.

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

[24] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: lower and upper bounds, 2010. arXiv:1011.5470.

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

[26] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986. doi:10.1137/0215074.

[27] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, San Francisco, 1996.

[28] 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.

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

[30] Noam Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991. doi:10.1137/0220062.

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

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

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

[34] Jukka Suomela. *Distributed Algorithms*. 2014. Online textbook. http://users.ics.aalto.fi/suomela/da/.