Abstract

We consider two models of computation: local centralized algorithms and local distributed algorithms. Algorithms in one model are adapted to the other model to obtain improved algorithms.

Distributed vertex coloring is employed to design improved local centralized algorithms for: maximal independent set, maximal matching, and an approximation scheme for maximum matching over bounded degree graphs. The improvement is threefold: the algorithms are deterministic, stateless, and the number of probes is poly(log* n).

The recursive local centralized improvement technique by Nguyen and Onak [NO08] is employed to obtain an improved distributed approximation scheme for maximum matching. The number of rounds of the distributed algorithm is O(log* n) for bounded degree graphs.

Keywords. Centralized Local Algorithms, Distributed Local Algorithms, Sublinear Approximation Algorithms, Lower bounds, Graph Algorithms.

1 Introduction

Local Computation Algorithms, as defined by Rubinfeld et al. [RTVX11], are algorithms that answer queries regarding (global) solutions to computational problems by performing local (sublinear time) computations on the input. If there is more than one possible solution, then the answers to different queries must be consistent with the same solution. To make this notion concrete, consider the Maximal Independent Set problem, which we denote by MIS. Given a graph $G = (V, E)$ as input, the local algorithm $\text{ALG}$ gives the illusion that it “holds” a specific maximal independent set $V'$. Namely, given any vertex $v$ as a query, $\text{ALG}$ answers whether $v$ belongs to $V'$ even though $\text{ALG}$ cannot read all of $G$ or store $V'$. In order to answer such queries, $\text{ALG}$ can probe the graph $G$ by asking about the neighbors of a vertex of its choice.

A local algorithm may be randomized, so that the solution according to which it answers queries may depend on its internal coin flips. However, the solution should not depend on the
sequence of queries it is asked. The performance of a local computation algorithm is measured according to the maximum number of probes it makes to the input per query, its success probability over any sequence of queries, and the maximum space it uses between queries. It is desired that both the probe complexity and the space complexity of the algorithm be sublinear in the size of the graph (e.g., \(\text{polylog}(|V|)\)), and that the success probability be \(1 - 1/\text{poly}(|V|)\).

It is usually assumed that the maximum degree of the graph is upper-bounded by a constant, but we may allow non-constant upper bounds. For a formal definition of local algorithms in the context of graph problems, which is the focus of this work, see Subsection 2.2.

Local computation algorithms capture difficulties with very large inputs: reading the whole input or computing a solution is too costly, one may need only a very small part of a complete solution, and different uncoordinated servers answer queries and yet must agree on a specific solution. Local computation algorithms have been designed for various graph (and hypergraph) problems, including the abovementioned MIS [RTVX11, ARVX12], hypergraph coloring [RTVX11, ARVX12], maximal matching [MRVX12] and (approximate) maximum matching [MV13]. Local computation algorithms also appear implicitly in works on sublinear approximation algorithms for various graph parameters, such as the size of a minimum vertex cover [PR07, NO08, YYI12, ORRR12]. Some of these implicit results are very efficient in terms of their probe complexity (in particular, it depends on the maximum degree and not on \(|V|\)) but do not give the desired \(1 - 1/\text{poly}(|V|)\) success probability. We compare our results to both the explicit and implicit relevant known results.

As can be gleaned from the definition in [RTVX11], local computation algorithms are closely related to Local Distributed Algorithms. We discuss the similarities and differences in more detail in Subsection 1.1. In this work, we exploit this relation in two ways. First, we use techniques from the study of local distributed algorithms to obtain better local computation algorithms. Second, we apply techniques from the study of local computation algorithms (more precisely, local computation algorithms that are implicit within sublinear approximation algorithms) to obtain a new result in distributed computing.

In what follows we denote the aforementioned local computation model by \(\text{CENTLOCAL}\) (where the “CENT” stands for “centralized”) and the distributed (local) model \(\text{DISTLOCAL}\) (for a formal definition of the latter, see Subsection 2.3). We denote the number of vertices in the input graph by \(n\) and the maximum degree by \(\Delta\).

1.1 On the relation between CENTLOCAL and DISTLOCAL

The CENTLOCAL model is centralized in the sense that there is a single central algorithm that is provided access to the whole graph. This is as opposed to the DISTLOCAL model in which each processor resides in a graph vertex \(v\) and can obtain information only about the neighborhood of \(v\). Another important difference is in the main complexity measure. In the CENTLOCAL model, one counts the number of probes that the algorithm performs per query, while in the DISTLOCAL model, the number of rounds of communication is counted. This implies that a DISTLOCAL algorithm always obtains information about a ball centered at a vertex, where the radius of the ball is the number of rounds of communication. On the other hand, in the case of a CENTLOCAL algorithm, it might choose to obtain information about different types of

\[1\text{Strictly speaking, a distributed algorithm is considered local if it performs a number of rounds that does not depend on } |V|. \text{ We give an algorithm that is almost local, i.e., it has log}\^2 \text{ dependence on } |V|, \text{ but for simplicity refer to it as local. For the sake of comparison to previous work, we also consider algorithms in which the number of rounds is (poly)logarithmic in } |V|.\]
neighborhoods so as to save in the number of probes. Indeed (similarly to what was observed in the context of sublinear approximation algorithms [PR07]), given a DISTLOCAL algorithm for a particular problem with round complexity \( r \), we directly obtain a CENTLOCAL algorithm whose probe complexity is \( O(\Delta^r) \) where \( \Delta \) is the maximum degree in the graph. However, we might be able to obtain lower probe complexity if we don’t apply such a black-box reduction. In the other direction, CENTLOCAL algorithms with certain properties, can be transformed into DISTLOCAL algorithms.

1.2 Our results

The starting point for our results in the CENTLOCAL model is the ranking technique. To exemplify this, consider, once again, the MIS problem. A very simple (global “greedy”) algorithm for this problem works by selecting an arbitrary ranking of the vertices and initializing \( V' \) to be empty. The algorithm then considers the vertices one after the other according to their ranks and adds a vertex to \( V' \) if and only if it does not neighbor any vertex already in \( V' \). Such an algorithm can be “localized” as follows. For a fixed ranking of the vertices (say, according to their IDs), given a query on a vertex \( v \), the local algorithm performs a restricted DFS starting from \( v \). The restriction is that the search continues only on paths with monotonically decreasing ranks. The local algorithm then simulates the global one on the subgraph induced by this restricted DFS.

The main problem with the above local algorithm is that the number of probes it performs when running the DFS may be very large. Indeed, for some rankings (and queried vertices), the number of probes is linear in \( n \). In order to circumvent this problem, random rankings were studied. This brings up two questions, which were studied in previous works, both for the MIS algorithm described above [NO08, YYI12] and for other ranking-based algorithms [ARVX12, MRVX12, MV13]. The first is to bound the number of probes needed to answer a query with high probability. The second is how to efficiently store a random ranking between queries.

Orientations with bounded reachability. Our first conceptual contribution is a simple but very useful observation. Rather than considering vertex rankings, we suggest to consider acyclic orientations of the edges in the graph. Such orientations induce partial orders over the vertices, and partial orders suffice for our purposes. The probe complexity induced by a given orientation translates into a combinatorial measure, which we refer to as the reachability of the orientation. Reachability of an acyclic orientation is the maximum number of vertices that can be reached from any start vertex by directed paths (induced by the orientation). This leads us to the quest for a CENTLOCAL algorithm that computes an orientation with bounded reachability.

Orientations and colorings. To obtain such an algorithm we actually design a CENTLOCAL coloring algorithm, since every vertex-coloring with \( k \) colors induces an orientation with reachability \( O(\Delta^k) \). The CENTLOCAL coloring algorithm applies techniques that were introduced for DISTLOCAL colorings algorithms [CV86, GPS88, Lin92, PS10]. Our CENTLOCAL algorithm is deterministic, does not use any space between queries, performs \( O(\Delta \cdot \log^* n + \Delta^2) \) probes per query, and computes a coloring with \( O(\Delta^2 \log \Delta) \) colors. (We refer to this problem as poly(\( \Delta \))-COLOR.) This implies that we can get an orientation whose reachability is \( \Delta^{O(\Delta^2 \log \Delta)} \) with the same probe complexity.
Applications of the local orientation algorithm. A CENTLOCAL algorithm for orientations with bounded reachability can be used to obtain several additional CENTLOCAL algorithms. This is done by applying a general transformation (similarly to what was shown in [ARVX12]) from global algorithms with certain properties to local algorithms. As a result we get deterministic algorithms for MIS and maximal matching (MLM), which significantly improve over previous work [RTVX11, ARVX12, MRVX12], and an algorithm for coloring with \((\Delta + 1)\) colors \(((\Delta + 1)\)-COLOR). Compared to previous work, for MIS and MLM the dependence on \(n\) in the probe complexity is reduced from \(\text{polylog}(n)\) to \(\log^* (n)\) and the space complexity is reduced from \(\text{polylog}(n)\) to 0.

The recursive local improvement technique. Our final result in the CENTLOCAL model is a \((1 - \epsilon)\)-approximation algorithm for maximum matching (MM). This algorithm applies an additional technique, introduced by Nguyen and Onak [NO08] in the design of their sublinear algorithm for approximating the size of a maximum matching. This technique, which we refer to as Local Improvement builds on augmenting paths. Roughly speaking, it consists of recursive applications of a CENTLOCAL algorithm for MIS to certain auxiliary graphs (defined by augmenting paths). Here we also reduce the dependence of the probe-complexity on \(n\) from \(\text{polylog}(n)\) [MV13] to \(\log^* n\) and the space is reduced from \(\text{polylog}(n)\) to 0.

A deterministic DISTLOCAL algorithm for approximate maximum matching. Let \(r\) denote an upper bound on the radius of the ball that contains all the probes of our CENTLOCAL algorithm for approximate maximum matching \(((1 - \epsilon)\)-MM). A deterministic DISTLOCAL\([r]\)-algorithm for \((1 - \epsilon)\)-MM follows immediately. Indeed, \(r\) is \(\Delta^{O(1/\epsilon)} + O \left( \frac{1}{\epsilon^3} \right) \cdot \log^* n\). In [LPSP08] a randomized DISTLOCAL\([O((\log n)/\epsilon^3)]\)-algorithm was presented for \((1 - \epsilon)\)-MM. The number of rounds in [LPSP08] does not depend on \(\Delta\), which is, therefore, unbounded. (See also [LPSR09, PS10] for constant approximation DISTLOCAL\([\text{polylog}(n)]\)-algorithms for maximum weighted matching.) Hence we get an improved result when \(\Delta^{O(1/\epsilon)} = o(\log n)\) (and, in particular, for constant \(\Delta\) and \(\epsilon\)). Note that an \(O(1)\)-approximation of a maximum matching in a \(n\)-node ring cannot be computed by a DISTLOCAL\([O(1)]\)-algorithm [CHW08, LW08].

1.3 Related work

Comparison to previous (explicit) CENTLOCAL algorithms. A comparison of our results with previous CENTLOCAL algorithms is summarized in Table 1. The results are presented for \(\Delta = O(1)\) and \(\epsilon = O(1)\). The dependency of our algorithms on \(\Delta\) and \(\epsilon\) can be found in Table 2 and in the corresponding theorems. The dependence on \(\Delta\) and \(\epsilon\) of previous algorithms is not explicit; our understating is that they depend exponentially on \(\Delta\) and \(1/\epsilon\).

Comparison to previous CENTLOCAL oracles in sublinear approximation algorithms. A sublinear approximation algorithm for a certain graph parameter (e.g., the size of a minimum vertex cover) is given probe access to the input graph and is required to output an approximation of the graph parameter with high (constant) success probability. Many such algorithms work by designing an oracle that answers queries (e.g., does a given vertex belong to a fixed small vertex cover). The sublinear approximation algorithm estimates the graph parameter by performing (a small number of) queries to the oracle. The oracles are essentially CENTLOCAL algorithms
Previous work | Here (Deterministic, 0-Space)  
---|---
**Problem** | **Space** | **# Probes** | **success prob.** | **# Probes** | **apx. ratio**  
MIS | \(O(\log^4 n)\) | \(O(\log^2 n)\) | \(1 - \frac{1}{\text{poly}(n)}\) | \([\text{ARVX12}]\) | \(O(\log^* n)\) | [Coro. 7]  
(1 - \(\varepsilon\))-MLM | \(O(\log^3 n)\) | \(O(\log^3 n)\) | \(1 - \frac{1}{\text{poly}(n)}\) | \([\text{MRVX12}]\) | \(O(\log^* n)\) | [Coro. 7]  
MM | \(O(\log^4 n)\) | \(O(\log^2 n)\) | \(1 - \frac{1}{\text{poly}(n)}\) | \([\text{MV13}]\) | \(\text{poly}(\log^* n)\) | [Thm. 10]  
poly(\(\Delta\))-COLOR | none | none | none | none | \(O(\log^* n)\) | [Coro 5]  
\((\Delta + 1)\)-COLOR | none | none | none | none | \(O(\log^* n)\) | [Coro 4]  

Table 1: A comparison between CentLocal algorithms. It is assumed that \(\Delta = O(1)\) and \(\varepsilon = O(1)\). Our algorithms are deterministic and stateless (i.e., have space zero). MLM denotes a maximal matching, MM denotes maximum matching.

but they tend to have constant error probability, and it is not clear how to reduce this error probability without significantly increasing their probe complexity. Furthermore, the question of bounded space was not an issue in the design of these oracles, since only few queries are performed by the sublinear approximation algorithm. Hence, they are not usually considered to be “bona fide” CentLocal algorithms. A comparison of our results and these oracles appears in Table 2.

| Problem | # Probes | success prob. | apx. ratio | # Probes | apx. ratio  
---|---|---|---|---|---
MIS | \(O(\Delta^4)\cdot \text{poly}\left(\frac{1}{\delta}\right)\) | \(1 - \delta\) | \(1 - \varepsilon\) | \([\text{YYI12}]\) | \(O(\Delta^4 \log \Delta) \cdot \log^* n\) | \(1\)  
MLM | \(O(\Delta^4)\cdot \text{poly}\left(\frac{1}{\delta}\right)\) | \(1 - \delta\) | \(1 - \varepsilon\) | \([\text{YYI12}]\) | \(O(\Delta^4 \log \Delta) \cdot \log^* n\) | \(1\)  
MM | \(\Delta\log(1/\delta) \cdot \text{poly}\left(\frac{1}{\delta}\right)\) | \(1 - \delta\) | \(1 - \varepsilon\) | \([\text{YYI12}]\) | \((\log^* n)^{1/\varepsilon} \cdot 2^{O(\Delta^{1/\varepsilon})}\) | \(1 - \varepsilon\)  
poly(\(\Delta\))-COLOR | none | none | none | none | \(O(\Delta \cdot \log^* n + \Delta^2)\) | \(-\)  
\((\Delta + 1)\)-COLOR | none | none | none | none | \(O(\Delta^2 \log \Delta) \cdot \log^* n\) | \(-\)  

Table 2: A comparison between CentLocal oracles in sub-linear approximation algorithms and our CentLocal (deterministic) algorithms. The former algorithms are designed to work with constant success probability and a bound is given on their expected probe complexity. When presenting them as CentLocal algorithms we introduce a failure probability parameter, \(\delta\), and bound their probe complexity in terms of \(\delta\). Furthermore, the approximation ratios of the sublinear approximation algorithms are stated in additive terms, and we translated the results so as to get a multiplicative approximation.

## 2 Preliminaries

### 2.1 Notations

Let \(G = (V, E)\) denote an undirected graph. Let \(n\) denote the number of vertices and \(m\) denote the number of edges. We denote the degree of \(v\) by \(\text{deg}(v)\). Let \(\Delta\) denote the maximum degree, i.e., \(\Delta \triangleq \max_{v \in V} \{\text{deg}(v)\}\). Let \(\Gamma(v)\) denote the set of neighbors of \(v \in V\). The length of a path equals the number of edges along the path. We denote the length of a path \(p\) by \(|p|\). For \(u, v \in V\) let \(\text{dist}(u, v)\) denote the length of the shortest path between \(u\) and \(v\). The ball of radius \(r\) centered at \(v\) is defined as follows

\[
B_r(v) \triangleq \{u \in V \mid \text{dist}(v, u) \leq r\}.
\]

For \(k \in \mathbb{N}^+\) and \(n > 0\), let \(\log^k n\) denote the \(k\)th iterated logarithm of \(n\). Note that \(\log^0 n \triangleq n\) and if \(\log^i n = 0\), we define \(\log^{i+1} n = 0\), for every \(i > 0\). For \(n \geq 1\), define \(\log^* n \triangleq \min\{i : \log^i n \leq 1\}\).
2.2 The CENTLOCAL Model

The model of centralized local computations was defined in [RTVX11]. In this section we describe this model for problems over labeled graphs.

Labeled graphs. An undirected graph \( G = (V, E) \) is labeled if: (1) The vertices have unique names. For simplicity, assume that the vertex names are in \( \{1, \ldots, n\} \). We denote the vertex whose name is \( i \) by \( v_i \). (2) Each vertex \( v \) holds a list of \( \text{deg}(v) \) pointers, called ports, that point to the neighbors of \( v \). The assignment of ports to neighbors is arbitrary and fixed.

Problems over labeled graphs. Let \( \Pi \) denote a computational problem over labeled graphs (e.g., maximum matching, maximal independent set, vertex coloring). A solution for problem \( \Pi \) over a labeled graph \( G \) is a function, the domain and range of which depend on \( \Pi \) and \( G \). For example: (1) In the Maximal Matching problem, a solution is an indicator function \( M : E \to \{0, 1\} \) of a maximal matching in \( G \). (2) In the problem of coloring the vertices of a graph by \( (\Delta + 1) \) colors, a solution is a coloring \( c : V \to \{1, \ldots, \Delta + 1\} \). Let \( \text{sol}(G, \Pi) \) denote the set of solutions of problem \( \Pi \) over a labeled graph \( G \).

Probes. In the CENTLOCAL model, access to the labeled graph is limited to probes. A probe is a pair \((v, i)\) that asks "who is the \( i \)th neighbor of \( v \)". The answer to a probe \((v, i)\) is as follows. (1) If \( \text{deg}(v) < i \), then the answer is "null". (2) If \( \text{deg}(v) \geq i \), then the answer is the vertex \( u \) that is pointed to by the \( i \)th port of \( v \). For simplicity, we assume that the answer also contains the port number \( j \) such that \( v \) is the \( j \)th neighbor of \( u \). (This assumption reduces the number of probes by at most a factor of \( \Delta \).)

Online algorithms in the CENTLOCAL model. An online deterministic algorithm \( \text{ALG} \) for a problem \( \Pi \) over labeled graphs in the CENTLOCAL model is defined as follows. The input for the algorithm consists of three parts: (1) access to a labeled graph \( G \) via probes, (2) the number of vertices \( n \) and the maximum degree \( \Delta \) of the graph \( G \), and (3) a sequence \( \{q_i\}_{i=1}^N \) of queries. Each query \( q_i \) is a request for an evaluation of \( f(q_i) \) where \( f \in \text{sol}(G, \Pi) \). The algorithm is online because it must output an evaluation of \( f(q_i) \) without any knowledge of subsequent queries.

We say that \( \text{ALG} \) is consistent with \((G, \Pi)\) if

\[
\exists f \in \text{sol}(G, \Pi) \text{ s.t. } \forall N \in \mathbb{N} \forall \{q_i\}_{i=1}^N \forall i : y_i = f(q_i).
\] (1)

Consider, for example, the problem of computing a \((\Delta + 1)\) vertex coloring. Consistency in this example means the following. The online algorithm is input a sequence of queries, each of which is a vertex. The algorithm must output the color of each queried vertex. If a vertex is queried twice, then the algorithm must return the same color. Moreover, queried vertices that are neighbors must be colored by different colors. Thus, if all vertices are queried, then the answers constitute a legal vertex coloring that uses \((\Delta + 1)\) colors. We now describe two measures of performance that are used in the CENTLOCAL model.

Performance measures. In the CENTLOCAL model, two computational resources are considered: space and number of probes. The state of algorithm \( \text{ALG} \) is the information that \( \text{ALG} \) saves between queries. The space of algorithm \( \text{ALG} \) is the number of bits required to encode
the state of ALG. The state is used to ensure consistency. We note that the running time used to answer a query is not counted.

**Definition 1.** An online algorithm is a CENTLOCAL \([q, s] \) algorithm for \( \Pi \) if (1) it is consistent with \((G, \Pi)\), (2) it performs at most \( q \) probes, and (3) it needs at most space \( s \).

The goal in designing algorithms in the CENTLOCAL model is to minimize the number of probes and the space (in particular \( q, s = o(n) \)). A CENTLOCAL \([q, s] \) algorithm with \( s = 0 \) is called stateless or zero-space. In this case, we refer to the algorithm as a stateless CENTLOCAL \([q] \)-algorithm.

**Randomized local algorithms.** If ALG is a randomized algorithm, the consistency requirement is parameterized by the failure probability \( \delta \). We say that ALG is a CENTLOCAL \([q, s, \delta]\) algorithm for \( \Pi \) with probability at least \( 1 - \delta \) if it is consistent with \((G, \Pi)\), performs at most \( q \) probes, and needs at most space \( s \). The standard requirement is that \( \delta = 1/poly(n) \).

**Parallelizability and query order obliviousness.** In \[ARVX12,MRVX12,MV13\] two requirements are introduced: parallelizability and query order obliviousness. These requirements are fully captured by the definition of a consistent, online, deterministic algorithm with zero space. That is, every online algorithm that is consistent, zero-space, and deterministic is both parallelizable and query order oblivious.

### 2.3 The DISTLOCAL Model

The model of local distributed computation is a classical model (see \[Lin92,Pel00,Suo13\]).

The distributed computation takes place in an undirected labeled graph \( G = (V, E) \). Each vertex models a processor, and communication is possible between neighboring processors. All processors execute the same algorithm. Initially, every \( v \in V \) is input a local input. The computation is done in \( r \in \mathbb{N} \) synchronous rounds as follows. In every round: (1) every processor receives a message from each neighbor, (2) every processor performs a computation based on its local input and the messages received from its neighbors, (3) every processor sends a message to each neighbor. We assume that a message sent in the end of round \( i \) is received in the beginning of round \( i + 1 \). After the \( r \)th round, every processor computes a local output.

The following assumptions are made in the DISTLOCAL model: (1) The local input to each vertex \( v \) includes the ID of \( v \), the degree of the vertex \( v \), the maximum degree \( \Delta \), the number of vertices \( n \), and the ports of \( v \) to its neighbors. (2) The IDs are distinct. For simplicity, we assume that the IDs are in the set \( \{1, \ldots, n\} \). (3) The length of the messages sent in each round is not bounded.

We say that a distributed algorithm is a DISTLOCAL \([r]\)-algorithm if the number of communication rounds is \( r \). Strictly speaking, a distributed algorithm is considered local if \( r \) is bounded by a constant. We say that a DISTLOCAL \([r]\)-algorithm is almost local if \( r = O(\log^* n) \). When it is obvious from the context we refer to an almost DISTLOCAL algorithm simply by a DISTLOCAL algorithm.

We remark that in the DISTLOCAL model, efficiency of the algorithm executed locally by the processors is not important. Namely, one does not bound the running time required to complete each round.
When counting the number of messages, one makes a distinction between empty messages and non-empty messages. The number of messages that a DISTLOCAL algorithms sent equals the number of non-empty messages.

2.4 Mutual Simulations Between DISTLOCAL and CENTLOCAL

One can simulate algorithms in one model by algorithms in the other model as follows (without any restriction on $\Delta$).

Simulation of DISTLOCAL by CENTLOCAL: Every deterministic DISTLOCAL[$r$]-algorithm $D$, can be simulated by a deterministic, stateless CENTLOCAL[$O(\Delta^r)$]-algorithm $C$. The simulation proceeds simply by probing all vertices in the ball of radius $r$ centered at the query.

Simulation of CENTLOCAL by DISTLOCAL: Every deterministic, stateless CENTLOCAL[$q$]-algorithm $C$, the probes of which are restricted to a ball of radius $r$ centered at the query, can be simulated by deterministic DISTLOCAL[$r$]-algorithm $D$. Moreover, the number of messages sent by $D$ is $q$.

If $\Delta = 2$, then balls are simple paths (or cycles) and hence simulation of a DISTLOCAL[$r$]-algorithm is possible by a CENTLOCAL[$2r$]-algorithm.

3 Acyclic Orientation with Bounded Reachability (OBR)

In this section we introduce the problem of Acyclic Orientation with Bounded Reachability (OBR). We then design a CENTLOCAL algorithm for OBR.

Notations. Let $H = (V, A)$ denote a directed graph, where $V$ is the set of vertices and $H \subseteq V \times V$. The reachability set of $v \in V$ is the set of vertices $R$ such that there is a path from $v$ to every vertex in $R$. We denote the reachability set of $v \in V$ in digraph $H$ by $R_H(v)$. Let $r_H(v) \triangleq |R_H(v)|$ and $r_{\max}^H \triangleq \max_{v \in V} r_H(v)$. We simply write $R(v), r(v), r_{\max}$ when the digraph $H$ is obvious from the context. We say that a digraph $H = (V, A)$ is an orientation of an undirected graph $G = (V, E)$ if $G$ is an underlying graph of $H$.

In the problem of acyclic orientation with bounded reachability (OBR), the input is an undirected graph. The output is an orientation $H$ of $G$ that is acyclic. The goal is to minimize $r_{\max}^H$.

Previous works obtain an acyclic orientation by random vertex ranking [NO08, YY12, ARVX12, MRVX12, MV13]. We propose to obtain an acyclic orientation by vertex coloring.

Proposition 1 (Orientation via coloring). Every coloring by $c$ colors induces an acyclic orientation with $r_{\max}^c \leq 2\Delta \cdot (\Delta - 1)^{c-2}$.

Proof. Direct each edge from a high color to a low color. By monotonicity the orientation is acyclic. Every directed path has at most $c$ vertices, and hence the reachability is bounded as required. \qed
3.1 A CENTLOCAL Algorithm for OBR

In the proof of Theorem 4, we present a deterministic, stateless CENTLOCAL [$O(\Delta \cdot \log^* n + \Delta^2)$]-algorithm that computes a vertex coloring that uses $c = O(\Delta^2 \log \Delta)$ colors. Orientation by this coloring yields an acyclic orientation with $r_{\text{max}}^n \leq \Delta^c$.

Acyclic orientation can be also obtained by simulating DISTLOCAL vertex coloring algorithms. Consider, for example, the $(\Delta + 1)$ coloring using $r_1 = O(\Delta) + \frac{1}{\Delta} \cdot \log^* n$ rounds of [BE09] or the $O(\Delta^2)$ coloring using $r_2 = O(\log^* n)$ rounds of [Lin92]. CENTLOCAL simulations of these algorithms require $O(\Delta^c)$ probes. Thus, in our algorithm, the number of probes grows (slightly) slower as a function of $n$ and is polynomial in $\Delta$.

Our algorithm relies on techniques from two previous DISTLOCAL coloring algorithms.

**Theorem 2** ([Lin92] Corollary 4.1). A $5\Delta^2 \log c$ coloring can be computed from a $c$ coloring by a DISTLOCAL [1]-algorithm.

**Theorem 3** ([PR01] Section 4). A $(\Delta + 1)$ coloring can be computed by a DISTLOCAL [$O(\Delta^2 + \log^* n)$]-algorithm.

**Theorem 4.** An $O(\Delta^2 \log \Delta)$ coloring can be computed by a deterministic, stateless CENTLOCAL [$O(\Delta \cdot \log^* n + \Delta^2)$]-algorithm.

**Proof.** We begin by describing a two phased DISTLOCAL [$O(\log^* n)$]-algorithm $D$ that uses $O(\Delta^2 \cdot \log \Delta)$ colors. Algorithm $D$ is especially designed so that it admits an “efficient” simulation by a CENTLOCAL-algorithm.

Consider a graph $G = (V, E)$ with a maximum degree $\Delta$. In the first phase, the edges partitioned into $\Delta^2$ parts, so that the maximum degree in each part is at most 2. Let $p_i(u)$ denote the neighbor of vertex $u$ point to by the $i$th port of $u$. Let $E_{i,j} \subseteq E$ be defined by

$$E_{i,j} \triangleq \{(u, v) \mid p_i(u) = v, p_j(v) = u, \text{ID}(u) < \text{ID}(v)\}.$$

Each edge belongs to exactly one part $E_{i,j}$. For each part $E_{i,j}$ and vertex $u$, at most two edges in $E_{i,j}$ are incident to $u$. Hence, the maximum degree in each part is at most 2 (i.e., each part is a disjoint union of simple paths and cycles). Each vertex can determine in a single round how the edges incident to it are partitioned among the parts. Let $G_{i,j} \triangleq (V, E_{i,j})$.

By Theorem 3 we 3-color each graph $G_{i,j}$ in $O(\log^* n)$ rounds. This induces a vector of $\Delta^2$ colors per vertex, hence a $3(\Delta^2)$ vertex coloring of $G$.

In the second phase, Algorithm $D$ applies Theorem 2 twice to reduce the number of colors to $O(\Delta^2 \log \Delta)$.

We now present an efficient simulation of algorithm $D$ by a CENTLOCAL-algorithm $C$. Given a query for the color of vertex $v$, Algorithm $C$ simulates the first phase of $D$ in which a 3-coloring algorithm is executed in each $E_{i,j}$. Since the maximum degree of each $G_{i,j}$ is two, a ball of radius $r$ in $G_{i,j}$ contains at most $2r$ edges. In fact, this ball can be recovered by at most $2r$ probes. It follows that a CENTLOCAL simulation of the 3-coloring of $G_{i,j}$ requires only $O(\log^* n)$ probes. Observe that if vertex $v$ is isolated in $G_{i,j}$, then it may be colored arbitrarily (say, by the first color). A vertex $v$ is not isolated in at most $\Delta$ parts. It follows that the simulation of the first phase requires $O(\Delta \cdot \log^* n)$ probes.

The second phase of algorithm $D$ requires an additional $\Delta^2$ probes, and the theorem follows. □

**Corollary 5.** There is a deterministic, stateless CENTLOCAL [$O(\Delta \cdot \log^* n + \Delta^2)$]-algorithm for OBR that achieves $r_{\text{max}}^n \leq \Delta^{O(\Delta^2 \log \Delta)}$. 

9
\textbf{Proof.} A \textsc{CentLocal}[$\Delta \cdot \log^* n + \Delta^2$]-algorithm for \textsc{Obr} is given a query $(v, i)$. The algorithm answers whether the edge $(v, p_i(v))$ is an incoming edge or an outgoing edge. The algorithm proceeds by querying the colors of $v$ and $p_i(v)$. The orientation of the edge $(v, p_i(v))$ is determined by comparing the colors of $v$ and $p_i(v)$. \hfill $\square$

4 Localization of Sequential Algorithms and Applications

A common theme in online algorithms and “greedy” algorithms is that the elements are scanned in query order or in an arbitrary order, and a decision is made for each element based on the decisions of the previous elements. Classical examples of such algorithms include the greedy algorithms for maximal matchings, $(\Delta + 1)$ vertex coloring, and maximal independent set. We present a compact and axiomatic \textsc{CentLocal} deterministic simulation of this family of algorithms. A randomized simulation appeared in [MRVX12]. Our simulation is based on an acyclic orientation that induces a partial order.

For simplicity, consider a graph problem $\Pi$, the solution of which is a function $g(v)$ defined over the vertices of the input graph. For example, $g(v)$ can be the color of $v$ or a bit indicating if $v$ belongs to a maximal independent set. (One can easily extend the definition to problems in which the solution is a function over the edges, e.g., maximal matching.)

We refer to an algorithm as a \textit{sequential algorithm} if it fits the scheme listed as Algorithm 1. The algorithm $\textsc{Alg}(G, \sigma)$ is input a graph $G = (V, E)$ and a permutation $\sigma : \{1, \ldots, n\} \rightarrow V$ of the vertices. The algorithm scans the vertices in the order induced by $\sigma$. It determines the value of $g(\sigma(i))$ based on the values of its neighbors whose value has already been determined. This decision is captured by the function $f$ in Line 2. For example, in vertex coloring, $f$ returns the smallest color that does not appear in a given a subset of colors.

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{Input:} $G = (V, E), \sigma$.
\For{$i = 1$ to $n$}
\State $g(\sigma(i)) \leftarrow f(\{g(v) : v \in \Gamma(\sigma(i)) \& \sigma^{-1}(v) < i\})$ \Comment{(Decide based on neighbors)}
\EndFor
\State \textbf{Output:} $g$.
\end{algorithmic}
\end{algorithm}

\textbf{Observation 1.} Let $G = (V, E)$ be a graph, let $H = (V, A)$ be an acyclic orientation of $G$ and let $P_\succ \subseteq V \times V$ denote the partial order defined by the transitive closure of $H$. Namely, $(u, v) \in P_\succ$ if and only if there exists a directed path from $u$ to $v$ in $H$. Let $\textsc{Alg}$ denote a sequential algorithm. For every permutation $\sigma : \{1, \ldots, n\} \rightarrow V$ that is a linear extension of $P_\succ$ (i.e., for every $(u, v) \in P_\succ$ we have that $\sigma^{-1}(u) > \sigma^{-1}(v)$), the output of $\textsc{Alg}(G, \sigma)$ is the same.

\textbf{Proof.} Consider two linear extensions $\sigma$ and $\pi$ of $P_\succ$. Let $g_\sigma$ denote the output of $\textsc{Alg}(G, \sigma)$ and define $g_\pi$ analogously. We prove, by induction on $i$, that $g_\sigma(\sigma(i)) = g_\pi(\sigma(i))$. The induction basis, for $i = 1$, holds because $\sigma(1)$ is a minimal element according to $P_\succ$ (a sink in $H$). Hence, $g_\sigma(\sigma(1)) = f(\emptyset) = g_\pi(\sigma(1))$. Turning to the induction step, we prove that the claim holds for $i > 1$, assuming it holds for every $1 \leq i' < i$. For each $v \in \Gamma(\sigma(i))$, if $(\sigma(i), v) \in A$ then $(\sigma(i), v) \in P_\succ$, so that $\sigma^{-1}(v) < i$ and $\tau^{-1}(v) < i$. On the other hand, if $(v, \sigma(i)) \in A$ then $(v, \sigma(i)) \in P_\succ$, so that $\sigma^{-1}(v) > i$ and $\tau^{-1}(v) > i$. Therefore, $g_\sigma(\sigma(i)) = g_\pi(\sigma(i))$.
then \((v, \sigma(i)) \in P_\succ\), so that \(\sigma^{-1}(v) > i\) and \(\tau^{-1}(v) > i\). By the induction hypothesis we have that the sets \(\{g_\alpha(v) : v \in \Gamma(\sigma(i)) \& \; \sigma^{-1}(v) < i\}\) and \(\{g_\tau(v) : v \in \Gamma(\sigma(i)) \& \; \tau^{-1}(v) < i\}\) are identical. Hence, \(g_\alpha(\sigma(i)) = g_\tau(\sigma(i))\).

**Theorem 6.** For every sequential algorithm \(\text{ALG}\), there exists a deterministic, stateless \(\text{CENTLOCAL}[\Delta \cdot \log^* n + \Delta^2, \log^* n]\)-algorithm \(\text{ALG}_c\) for which the following holds. For every graph \(G\), there exists a permutation \(\sigma\), such that \(\text{ALG}_c(G)\) simulates \(\text{ALG}(G, \sigma)\). That is, for every vertex \(v\) in \(G\), the answer of \(\text{ALG}_c(G)\) on query \(v\) is \(g_\alpha(v)\), where \(g_\alpha\) denotes the output of \(\text{ALG}(G, \sigma)\).

**Proof.** Consider the acyclic orientation \(H\) of \(G\) induced by the \(\text{CENTLOCAL}[\Delta \cdot \log^* n + \Delta^2]\)-algorithm for OBR presented in Corollary 5. Let \(P_\succ\) denote the partial order that is induced by \(H\), and let \(\sigma\) be any linear extension of \(P_\succ\) (as defined in Observation 1). On query \(v \in V\) the value \(g_\alpha(v)\) can be computed by performing a (directed) DFS on \(H\) that traverses the subgraph of \(H\) induced by \(R_H(v)\) and computes \(g_\alpha(v)\) when backtracking from \(v\). The DFS uses the \(\text{CENTLOCAL}\) algorithm for OBR to determine the orientation of each incident edge and continues only along outward-directed edges. Since \(r_H^{\text{max}} = \Delta^2 \log \Delta\), it follows that \(\Delta^2 \log \Delta \cdot \log^* n\) probes suffice.

**Corollary 7.** There are deterministic, stateless \(\text{CENTLOCAL}[\Delta^2 \log \Delta, \log^* n]\) algorithms for \((\Delta + 1)\)-vertex coloring, maximal independent set, and maximal matching.

Note that the probes of the \(\text{CENTLOCAL}\)-algorithms in this corollary are confined to a ball of radius \(\max\{O(\log^* n), O(\Delta^2 \log \Delta)\}\). The number of probes of the \((\Delta + 1)\)-coloring in this corollary is exponential in \(\Delta\); compare this with the \(O(\Delta^2 \log \Delta)\)-coloring in Theorem 4 that has \(O(\Delta \cdot \log^* n + \Delta^2)\) probes.

### 5 A CENTLOCAL Algorithm for \((1 - \varepsilon)\) Maximum Matching (MM)

In this section we present a \(\text{CENTLOCAL}\) algorithm that computes a \((1 - \varepsilon)\)-approximation of a maximum matching. The algorithm is based on a \(\text{CENTLOCAL}\)-algorithm for maximal independent set (see Coro. 7) and on the local improvement technique of Nguyen and Onak [NO08].

Let \(M^*\) denote a maximum matching of \(G\). We say that a matching \(M\) is a \((1 - \varepsilon)\)-Maximum Matching for some \(0 < \varepsilon < 1\) if

\[
|M| \geq (1 - \varepsilon) \cdot |M^*|.
\]

Observe that if \(\Delta < 2\), then \(M^* = E\), so that the maximum matching problem can be solved trivially (in particular by a local algorithm). Hence, we assume from this point on that \(\Delta \geq 2\).

**Notation.** Let \(M\) be a matching in \(G = (V, E)\). A vertex \(v \in V\) is \(M\)-free is \(v\) is not an endpoint of an edge in \(M\). A path is \(M\)-alternating if it consists of edges drawn alternatingly from \(M\) and from \(E \setminus M\). A path is \(M\)-augmenting if it is \(M\)-alternating and if both of the path’s endpoints are \(M\)-free vertices. Note that the length of an augmenting path must be odd. The set of edges in a path \(p\) is denoted by \(E(p)\), and the set of edges in a collection \(P\) of paths is denoted by \(E(P)\). Let \(A \oplus B\) denote the symmetric difference of the sets \(A\) and \(B\).

---

2Given that the CENTLOCAL algorithm for OBR works by running a CENTLOCAL coloring algorithm, we can actually use the latter algorithm directly.
Global Algorithm Description. Similarly to [NO08] our local algorithm simulates the following global algorithm, which builds on the next two lemmas of Hopcroft and Karp [HK73], and Nguyen and Onak [NO08], respectively.

Lemma 8 ([HK73]). Let $M$ be a matching in a graph $G$. Let $k$ denote the length of a shortest $M$-augmenting path. Let $P^*$ be a maximal set of vertex disjoint $M$-augmenting paths of length $k$. Then, $(M \oplus E(P^*))$ is a matching and the length of every $(M \oplus E(P^*))$-augmenting path is at least $k + 2$.

Lemma 9 ([NO08, Lemma 6]). Let $M^*$ be a maximum matching. Let $M$ be a matching in a graph $G$. Let $2k + 1$ denote the length of a shortest $M$-augmenting path. Then

$$|M| \geq \frac{k}{k + 1}|M^*|.$$

The (global) algorithm is given as input a graph $G$ and an approximation parameter $\varepsilon \in (0, 1)$. The algorithm works in $k$ iterations, where $k \geq \frac{1}{\varepsilon} - 1$. Initially, $M_0 = \emptyset$. The invariant of the algorithm is that $M_i$ is a matching, every augmenting path of which has length at least $2i + 1$. Given $M_{i-1}$, a new matching $M_i$ is computed as follows. Let $P_i$ denote the set of shortest $M_{i-1}$-augmenting paths. Let $P_i^* \subseteq P_i$ denote a maximal subset of vertex disjoint paths. Define $M_i \triangleq M_{i-1} \oplus E(P_i^*)$. By Lemmas 8 and 9, $M_k$ is a $(1 - \varepsilon)$-approximation of a maximum matching.

The intersection graph $H_i$. The description of the local simulation of the $i$th iteration of the global algorithm uses an intersection graph $H_i = (P_i, C_i)$ defined as follows. The set of nodes $P_i$ is the set of shortest $M_{i-1}$-augmenting paths. Note that the length of each path in $P_i$ is $2i - 1$. We connect two paths $p, q \in P_i$ by an edge $(p, q) \in C_i$ if $p$ and $q$ intersect (i.e., share a vertex in $V$). Note that $H_1$ is the line graph of $G$ and that $M_1$ is simply a maximal matching in $G$. Observe that $P_i^*$ as defined above is a maximal independent set in $H_i$. Thus, iteration $i$ the global algorithm proceeds by: “constructing” $H_i$, “finding” a maximal independent set $P_i^*$ in $H_i$, and “augmenting” the matching by $M_i \triangleq M_{i-1} \oplus (E(P_i^*))$.

CENTLOCAL Algorithm Description. We begin with a review of the recursive local improvement technique in [NO08 Sec. 3.3]. A local simulation of the global algorithm is based on a sequence of oracles $O_1, O_2, \ldots, O_k$. The input to oracle $O_i$ is an edge $e \in E$, and the output is a bit that indicates whether $e \in M_i$. Oracle $O_i$ proceeds by computing two bits $\tau$ and $\rho$. The bit $\tau$ indicates whether $e \in M_{i-1}$, and is computed by invoking oracle $O_{i-1}$. The bit $\rho$ indicates whether $e \in E(P_i^*)$ (where $P_i^*$ is an MIS in $H_{i-1}$). Oracle $O_i$ returns $\tau \oplus \rho$ because $M_i = M_{i-1} \oplus E(P_i^*)$.

Determining whether $e \in E(P_i^*)$ is based on simulating probes to $H_i$ to which explicit probing is not possible. The simulation requires the following steps: (1) Enumeration: construct the set $P_i(e) \triangleq \{p \in P_i \mid e \in E(p)\}$. Note that $e \in E(P_i^*)$ if and only if $P_i(e) \cap P_i^* \neq \emptyset$. (2) MIS step: for each $p \in P_i(e)$, input the query $p$ to an MIS-algorithm for $H_i$ to test whether $p \in P_i^*$. We now elaborate on how these two steps are carried out by a CENTLOCAL-algorithm.

The enumeration of all the paths in $P_i(e)$ can be done if the balls of radius $2i - 2$ centered at the endpoints of $e$ are known. To determine whether a path is alternating, one needs to know whether $e' \in M_{i-1}$ for each edge $e'$ incident to vertices in the balls. This step can be completed by at most $\text{enum}(H_i) \triangleq 2\Delta^{2i}$ probes to the graph $G$ and $\text{enum}(H_i)$ queries to $O_{i-1}$. 

12
In the MIS step, a query asking whether $p \in P_i^*$ is answered by simulating an MIS algorithm over $H_i$. Let $L(H_i)$ denote an upper bound on the number of probes in $H_i$ required by this MIS algorithm. Each probe $p$ to $H_i$ is simulated by probing all the neighbors of $p$ in $H_i$. As in the enumeration step, the probe in $H_i$ can be obtained by (1) “revealing” the balls in $G$ of radius $2i \cdot 2$ centered at endpoints of edges in $E(p)$, and (2) finding out which edges within these balls are in $M_{i-1}$. This step can be completed by at most $\text{probe}(H_i) \triangleq 2i \cdot 2^{2i}$ probes to $G$ and $\text{probe}(H_i)$ queries to $O_{i-1}$. We conclude that a simulation of a single query to the MIS-algorithm over $H_i$ requires $\text{probe}(H_i) \cdot L(H_i)$ probes in $G$ as well as $\text{probe}(H_i) \cdot L(H_i)$ queries to $O_{i-1}$.

Let $Q_i$ denote the number of probes to $G$ performed by the oracle $O_i$. The following recurrence bounds $Q_i$.

$$Q_i \leq (1 + Q_{i-1}) \cdot \left(1 + \text{enum}(H_i) + \max_{e} |P_i(e)| \cdot L(H_i) \cdot \text{probe}(H_i)\right).$$

(2)

In the randomized algorithm for $O_i$ in [NO08], $Q_i$ is bounded by $2^{O(\Delta^i)}$ in expectation over the bit flips of the algorithm and a uniform choice of an edge. As indicated in Table 1, an exponential improvement was presented in [YYI12].

On the other hand, we obtain a deterministic CENTLOCAL-algorithm that computes a $(1 - \varepsilon)$-approximation for maximum matching by using the deterministic MIS CENTLOCAL-algorithm presented in Coro. 7.

**Theorem 10.** There is a deterministic, stateless, $(1 - \varepsilon)$-approximate CENTLOCAL $[\varphi]$-algorithm for maximum matching, where

$$\varphi = (\log^* n)^{1/\varepsilon} \cdot 2^{O(\Delta^{i/\varepsilon})}.$$

**Proof.** We bound the parameters in recurrence equation (2) as follows. Let $n_i$ denote the number of vertices in $H_i$. Clearly, $n_i \leq n^{2i}$. Note that $\log^* n_i = O(\log^* n)$. Each vertex $v$ appears in $O(i \Delta^{2i})$ paths in $P_i$. Hence $\max_e |P_i(e)| = O(i \Delta^{2i})$. Let $\Delta_i$ denote the maximum degree of $H_i$. Clearly, $\Delta_i \leq O(i^2 \cdot \Delta^{2i})$.

The number of probes $L(H_i)$ of the MIS algorithm presented in Coro. 7 is $\Delta_i^{O(\Delta_i \log \Delta_i)} \cdot \log^* n_i = 2^{O(\Delta^{2i} \log \Delta_i)} \cdot \log^* n$.

Plugging in these bounds to the recurrence equation yields

$$Q_i \leq (1 + Q_{i-1}) \cdot \left(2\Delta^{2i} + i\Delta^{2i} \cdot 2^{O(\Delta^{2i})} \cdot \log^* n \cdot 2i\Delta^{2i}\right)$$

(3)

$$\leq (1 + Q_{i-1}) \cdot 2^{O(\Delta^{2i})} \cdot \log^* n.$$

It follows that

$$Q_k \leq \prod_{i=1}^{k} \log^* n \cdot 2^{O(\Delta^{2i})}$$

$$= (\log^* n)^k \cdot 2^{O(\Delta^{2k})}.$$

Set $k \geq \frac{1}{\varepsilon} - 1$, and the theorem follows.

**Remark 1.** Note that for $\Delta = O(1)$ and $\varepsilon = O(1)$ we get that $\varphi = O(\text{poly}(\log^* n))$. 

13
Remark 2. The CENTLOCAL-algorithm for MIS from Coro. uses the vertex coloring algorithm from Theorem 2. This coloring algorithm requires IDs and port numbers to decompose the edge set. We now elaborate on how IDs and port numbers can be assigned in the simulation of probes to $H_i$. First, we assign each node $p$ in $H_i$ a unique ID, which is simply the sequence of the vertices of $G$ in the path $p$. Order the neighbors (of each $p$ in $H_i$) in lexicographic order of their IDs. The $j$th neighbor of $p$ (the neighbor connected to $p$ via port $j$ of $p$) is simply the $j$th neighbor according to this order. Note that the simulation determines all the neighbors of a node $p$ in $H_i$, hence the $j$th neighbor can be easily determined.

6 DISTLOCAL Results

In this section we present results for algorithms in the DISTLOCAL model over bounded degree graphs. First we list DISTLOCAL-algorithms for acyclic orientation that are obtained from vertex coloring algorithm and perform $O(\log^* n)$ rounds. We then prove that such orientations cannot be computed in $o(\log^* n)$ rounds. Thus, $\Theta(\log^* n)$ rounds are necessary and sufficient for computing an acyclic orientation with reachability that is bounded by any function of the maximum degree.

Finally, we present a $(1-\varepsilon)$-approximation algorithm for maximum matching, the number of rounds of which is linear in $\log^* n$ (the dependency on $\Delta$ and $\varepsilon$ is $\Delta^O(1/\varepsilon)$).

6.1 DISTLOCAL Algorithms for OBR

As observed in Proposition 1, every vertex coloring induces an acyclic orientation. This implies that a DISTLOCAL $c$-coloring algorithm can be used to compute an acyclic orientation with reachability $r_{H}^{\max} = O(\Delta c)$ by performing the same number of rounds. The distributed coloring algorithms [Lin92, Theorem 4.2] [BE09, Theorem 4.6] imply the following corollary.

Corollary 11. There are DISTLOCAL algorithms for OBR with the following parameters:

1. Reachability $O(\Delta^2 \Delta^2)$ in $O(\log^* n)$ rounds.

2. Reachability $O(\Delta^3)$ in $O(\Delta) + \frac{1}{2} \log^* n$ rounds.

6.2 Lower Bound for OBR in the DISTLOCAL Model

In this section we consider the problem of computing an acyclic orientation $H$ of a graph $G$ with reachability $r_{H}^{\max}$ that does not depend on the number of vertices $n$.

Definition 2. Let $g : \mathbb{N} \rightarrow \mathbb{N}$ denote a function. In the OBR($g$)-problem, the input is a graph $G$ with maximum degree $\Delta$. The goal is to compute an orientation $H$ of $G$ with reachability $r_{H}^{\max} \leq g(\Delta)$ (if such an orientation exists).

Note that every graph $G$ (with degree bounded by $\Delta$) admits an acyclic orientation with reachability $O(\Delta^{2+1})$. Our goal is to prove the following theorem.

Theorem 12. For every function $g$, there is no DISTLOCAL $[o(\log^* n)]$-algorithm that solves the OBR($g$)-problem.
Proof. The proof is based on a reduction from MIS to OBR\((g)\). Let \(D\) denote a DistLocal\([\cdot]\)-algorithm that computes an acyclic orientation \(H\) of \(G\) in \(r\) rounds. We claim that an MIS in \(G\) can be computed in \(r + r_{H}^{\text{max}}\) rounds. Indeed, given an orientation, we can execute the CentLocal-algorithm for MIS (see Coro. 7) with probes that are limited to the reachability set of the query. A distributed simulation of the MIS CentLocal-algorithm requires \(r'\) rounds where \(r'\) is the diameter of the reachability set, which is bounded by \(r_{H}^{\text{max}}\).

Let \(G_n\) denote an undirected ring with \(n\) vertices. Let \(g : \mathbb{N} \rightarrow \mathbb{N}\) be any function (e.g., Ackermann function). Assume, for the sake of contradiction, that there exists a DistLocal\([\cdot]\)-algorithm that computes an acyclic orientation \(H_n\) of \(G_n\) with reachability \(r_{H_n}^{\text{max}} \leq g(\Delta)\). Then, the aforementioned reduction implies a DistLocal\([r + g(\Delta)]\)-algorithm for MIS.

If \(r = o(\log^* n)\), then this contradicts the theorem of Linial [Lin92] that states that there is no DistLocal algorithm that computes an MIS in less than \(\frac{1}{2} \cdot \log^* n\) rounds.

Remark 3. Theorem 12 can be extended to reachability that is \(o(\log^* n)\).

Remark 4. Theorem 12 can be extended to randomized algorithms since the lower bound for MIS in [Lin92] holds also for randomized algorithms.

6.3 A DistLocal Algorithm for \((1 - \varepsilon)\) Maximum Matching (MM)

In this section we obtain a DistLocal-algorithm for MM by simulating the CentLocal algorithm presented in Theorem 10. The algorithm does not restrict the maximum degree (i.e., \(\Delta\) may be \(\omega(1)\)).

Theorem 13. There is a deterministic DistLocal\([\Delta^{O(1/\varepsilon)} + O\left(\frac{1}{\varepsilon^2}\right) \cdot \log^*(n)]\)-algorithm for MM.

The proof of the theorem is based on an upper bound on the radius of the ball that contains all the probes of the CentLocal-algorithm (see Sec. 2.4).

Lemma 14. All the probes of the CentLocal-algorithm in Theorem 10 are confined to a ball of radius

\[ r = \Delta^{O(1/\varepsilon)} + O\left(\frac{1}{\varepsilon^2}\right) \cdot \log^* n. \]

Proof. Let \(r(O_i)\) and \(r_{\text{MIS}}(H_i)\) denote the radius in which oracle \(O_i\) and the MIS-algorithm perform their probes. The radii satisfy the following equations.

\[
\begin{align*}
    r_{\text{MIS}}(H_i) &= O\left(\log^* n_i + \Delta_i^2 \log \Delta_i\right) \\
    r(O_i) &= \begin{cases} 
    r_{\text{MIS}}(H_1) & \text{if } i = 1, \\
    r(O_{i-1}) + 2i \cdot r_{\text{MIS}}(H_i) + 2i & \text{if } i \geq 2.
    \end{cases}
\end{align*}
\]

Since \(\Delta_i \leq i^2 \Delta^2_i\) and \(n_i \leq n^{2n}\) we get that

\[ 2i \cdot r_{\text{MIS}}(H_i) + 2i = O(i \cdot (\Delta_i^7 + \log^* n)). \]

It follows that

\[ r_k = \sum_{i=1}^{k} O\left(\Delta_i^{8i} + i \log^* n\right) = \Delta^{O(k)} + O(k^2) \log^* n. \]

Let \(k \geq \frac{1}{\varepsilon} - 1\), and the lemma follows. \(\square\)
Remark 5. A similar result can be obtained by using any DISTLOCAL \([O(\log^* n) + \text{poly}(\Delta)]\)-algorithm for MIS.

7 Discussion

In this work we design local centralized algorithms for several graph problems. Our algorithms are deterministic, do not use any space, and the number of probes (queries to the graph) is \(\text{poly}(\log^* n)\) where \(n\) is the number of graph vertices. Previously known algorithms for these problems make \(\text{polylog}(n)\) probes, use \(\text{polylog}(n)\) space, and have failure probability \(1/\text{poly}(n)\). While a basic tool in previous works was (random) vertex rankings, our basic (seemingly weaker) tool, is acyclic graph orientations with bounded reachability. That is, our algorithms use as a subroutine a local procedure that orients the edges of the graph while ensuring an upper bound on the number of vertices reachable from any vertex. To obtain such orientations we employ a local coloring algorithm which uses techniques from local distributed algorithms for coloring. On the other hand, by using a technique that was introduced for local computation in the context of sublinear approximation algorithms we get a new result in local distributed computing: A deterministic algorithm for approximating a maximum matching to within \((1 - \varepsilon)\) that performs \(\Delta O^{(1/\varepsilon)} + O\left(\frac{1}{\varepsilon^2}\right) \cdot \log^* n\) rounds where \(\Delta\) is the maximum degree in the graph. This is the best known algorithm for this problem for constant \(\Delta\).

References

[ARVX12] Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *SODA*, pages 1132–1139, 2012.

[BE09] Leonid Barenboim and Michael Elkin. Distributed \((\Delta + 1)\)-coloring in linear (in \(\Delta\)) time. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 111–120. ACM, 2009.

[CHW08] Andrzej Czygrinow, Michal Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *Distributed Computing*, pages 78–92. Springer, 2008.

[CV86] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.

[GPS88] Andrew V Goldberg, Serge A Plotkin, and Gregory E Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM Journal on Discrete Mathematics*, 1(4):434–446, 1988.

[HK73] John E Hopcroft and Richard M Karp. An \(n^{5/2}\) algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[Lin92] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[LPSP08] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. In *SPAA*, pages 129–136, 2008.
[LPSR09] Zvi Lotker, Boaz Patt-Shamir, and Adi Rosén. Distributed approximate matching. SIAM J. Comput., 39(2):445–460, 2009.

[LW08] Christoph Lenzen and Roger Wattenhofer. Leveraging Linial’s locality limit. In Distributed Computing, pages 394–407. Springer, 2008.

[MRVX12] Yishay Mansour, Aviad Rubinstein, Shai Vardi, and Ning Xie. Converting online algorithms to local computation algorithms. In Automata, Languages, and Programming, pages 653–664. Springer, 2012.

[MV13] Yishay Mansour and Shai Vardi. A local computation approximation scheme to maximum matching. In APPROX-RANDOM, pages 260–273, 2013.

[NO08] Huy N Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In Foundations of Computer Science, 2008. FOCS’08. IEEE 49th Annual IEEE Symposium on, pages 327–336. IEEE, 2008.

[ORRR12] Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In SODA, pages 1123–1131, 2012.

[Peleg00] David Peleg. Distributed computing: a locality-sensitive approach, volume 5. SIAM, 2000.

[PR01] Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. Distributed computing, 14(2):97–100, 2001.

[PR07] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. Theoretical Computer Science, 381(1):183–196, 2007.

[PS10] Alessandro Panconesi and Mauro Sozio. Fast primal-dual distributed algorithms for scheduling and matching problems. Distributed Computing, 22(4):269–283, 2010.

[RTVX11] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In ICS, pages 223–238, 2011.

[Suo13] Jukka Suomela. Survey of local algorithms. ACM Comput. Surv., 45(2):24:1–24:40, March 2013.

[YYI12] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. SIAM J. Comput., 41(4):1074–1093, 2012.