# Local Computation Algorithms for Graphs of Non-Constant Degrees



by

Anak Yodpinyanee

B.S., Harvey Mudd College (2012)

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2014



Signature redacted

Author .....
Department of Electrical Engineering and Computer Science
August 29, 2014

Signature redacted

Certified by .....
Ronitt Rubinfeld
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, EECS Committee on Graduate Students

# Local Computation Algorithms for Graphs of Non-Constant Degrees

by

Anak Yodpinyanee

Submitted to the Department of Electrical Engineering and Computer Science
on August 29, 2014, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

## Abstract

In the model of *local computation algorithms* (LCAs), we aim to compute the queried part of the output by examining only a small (sublinear) portion of the input. Many recently developed LCAs on graph problems achieve time and space complexities with very low dependence on $n$, the number of vertices. Nonetheless, these complexities are generally at least exponential in $d$, the upper bound on the degree of the input graph. Instead, we consider the case where parameter $d$ can be moderately dependent on $n$, and aim for complexities with quasi-polynomial dependence on $d$, while maintaining polylogarithmic dependence on $n$. In this thesis, we give randomized LCAs for computing maximal independent sets, maximal matchings, and approximate maximum matchings. Both time and space complexities of our LCAs on these problems are $2^{O(\log^3 d)}\text{polylog}(n)$, $2^{O(\log^2 d)}\text{polylog}(n)$ and $2^{O(\log^3 d)}\text{polylog}(n)$, respectively.

Thesis Supervisor: Ronitt Rubinfeld
Title: Professor of Electrical Engineering and Computer Science

# Acknowledgments

This thesis is completed under the guidance of Professor Ronitt Rubinfeld, with financial support from NSF grants CCF-1217423 and CCF-1065125 and the DPST scholarship, Royal Thai Government.

# Contents

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

In the face of massive data sets, classical algorithmic models, where the algorithm reads the entire input, performs a full computation, then reports the entire output, are rendered infeasible. To handle these data sets, the model of *local computation algorithms* (LCAs) has been proposed. As defined in [35], these algorithms compute the queried part of the output by examining only a small (sublinear) portion of the input. Let us consider the problem of finding a maximal independent set (MIS) as an example. The algorithm $\mathcal{A}$ is given access to the input graph $G$, then it is asked a question: "is vertex $v$ in the MIS?" The algorithm then explores only a small portion of $G$, and answers "yes" or "no." The set of vertices $\{v : \mathcal{A}$ answers "yes" on $v\}$ must indeed form a valid MIS of $G$. LCAs have been constructed for many problems, including MIS, maximal matching, approximate maximum matching, vertex coloring, and hypergraph coloring ([35, 4, 25, 26, 14, 34]). In this thesis, we choose to study MIS, maximal matching, and approximate maximum matching because these are fundamental graph problems well-studied in many frameworks. Tools and results for these problems have proven to be useful as building blocks for more sophisticated and specialized problems in the field.

The LCA framework is useful in the circumstances where we never need the whole output at any time. This key characteristic of LCAs generalizes many other models from various contexts. For instance, LCAs may take the form of local filters and

reconstructors [11, 1, 9, 20, 36, 19, 10, 22]. Important applications include locally decodable codes (e.g., [38]), local decompression [28, 13], and locally computable decisions for online algorithms and mechanism design [25, 16]. There are a number of works on related models of local computation (e.g., [5, 8, 37, 31]) as well as lower bounds for the LCA framework and other models (e.g., [15]).

Many recently developed LCAs on graph problems achieve time and space complexities with very low dependence on $n$, the number of vertices. Nonetheless, these complexities are at least exponential in $d$, the upper bound on the degree of the input graph. While these papers often consider $d$ to be a constant, the large dependence on $d$ may forbid practical uses of these algorithms. In this work we consider the case where the parameter $d$ can be moderately dependent on $n$, and provide LCAs for complexities that have quasi-polynomial dependence on $d$, while maintaining polylogarithmic dependence on $n$. As noted in [25], whether there exist LCAs with polynomial dependence on $d$ for these problems is an interesting open question. This thesis aims at making progress and providing techniques useful towards resolving this problem.

## 1.1 Related Work

Many useful techniques for designing LCAs originate from algorithms for approximating the solution size in sublinear time. For example, if we wish to approximate the size of an MIS, we may sample a number of vertices and check whether each of them is in our MIS. The main difference, however, is that LCAs must be able to compute the answer to every query, while an approximation algorithm may ignore some queries, treating them as a small error in the approximation factor.

In this work, we primarily rely on the Parnas-Ron reduction, proposed in their paper on approximating the size of a minimum vertex cover (VC) [32]. This reduction turns a $k$-round distributed algorithm into an LCA by examining all vertices at distance up to $k$ from the queried vertex, then simulating the computation done by the distributed algorithm, invoking $d^{O(k)}$ queries to the input graph in total. Using

this technique, they obtain a $O(\log n)$-approximation for the size of a minimum VC (with $\epsilon n$ additive error) with query complexity $\tilde{d}^{O(\log \tilde{d})}$, where $\tilde{d}$ is the average degree. Marko and Ron later improve this result to a $(2+\delta)$-approximation with query complexity $d^{O(\log d)}$ [27]. However, these approaches for the minimum VC problem cannot be directly applied to packing problems such as MIS: while we can always add more vertices to the VC when unsure, such an approach may violate the independence constraint for MIS. Distributed algorithms for the MIS problem require more rounds, and consequently, a similar reduction only yields an LCA with query and time complexities $d^{O(d \log d)} \log n$ in [35].

Another powerful technique is the query tree method from the Nguyen-Onak algorithm [29]. This method simulates greedy algorithms on the vertices in a random order. The answer for each vertex only depends on vertices preceding it in this order, and thus the number of vertices to be investigated can be probabilistically bounded. This method is used in [4, 25, 26, 16, 34], giving query complexities with polylogarithmic dependence on $n$ for various problems. Unfortunately, the expected query tree size is exponential in $d$, which is considered constant in these papers. For certain problems, a slight modification of the Nguyen-Onak algorithm reduces the expected query tree size to $O(\tilde{d})$ [39, 30]. This gives algorithms with $\text{poly}(\tilde{d})$ query complexity for approximating the maximum matching and minimum VC sizes with multiplicative and additive errors. However, while the expected query tree size suffices for approximation algorithms, we require a bound on its maximum size to construct LCAs.

Recently, a new method for deterministically constructing a good ordering of vertices is given in [14] based on graph coloring, which improves upon LCAs for several graph problems. They construct an LCA to $O(d^2 \log d)$-color the graph while making only $\text{poly}(d)$ queries. Using this LCA as a subroutine, they reduce the query complexity of their algorithm for the MIS problem to $d^{O(d^2 \log d)} \log^* n$. This gives the lowest dependence on $n$ currently known, as well as a new direction for developing deterministic LCAs. Because random bits are not needed, their LCAs also require no extra local computation memory.

While all of these algorithms have complexities with exponential dependence on $d$ for the problems studied in this thesis, there has been no significant lower bound. To the best of our knowledge, the only explicit lower bound of $\Omega(\tilde{d})$ is given by Parnas and Ron [32]. Even though we are able to reduce the dependence on $d$ to quasi-polynomial, closing this gap between the upper and lower bounds remains an important open question for this framework.

## 1.2 Our Contribution and Approach

This thesis addresses the MIS problem in detail, then gives results and proof sketches for the maximal matching problem and the $(1-\epsilon)$-approximate maximum matching problem for constant $\epsilon$. The comparison between our results and other approaches is given in table 1.1. This thesis provides the first LCAs whose complexities are both quasi-polynomial in $d$ and polylogarithmic in $n$ for these problems. More concretely, when $d$ is non-constant, previously known LCAs have complexities with polylogarithmic dependence on $n$ only when $d = O(\log\log n)$. Our LCAs maintain this dependence even when $d = \exp(\Theta((\log\log n)^{1/3}))$ for the MIS and approximate maximum matching problems, and $d = \exp(\Theta((\log\log n)^{1/2}))$ for the maximal matching problem.

In this thesis, we construct two-phase LCAs similar to that of [35], which are based on Beck's algorithmic approach to Lovász local lemma [7]. In the first phase, we find a large partial solution that breaks the original graph into small connected components. The LCAs for the first phase are obtained by applying the Parnas-Ron reduction on distributed algorithms. The distributed algorithm for the first phase of the MIS problem in [35] requires $O(d\log d)$ rounds to make such guarantee. Ours are designed based on recent ideas from [6] so that $O(\text{polylog}(d))$ rounds suffice. As a result, after applying the Parnas-Ron reduction, the query complexity on the first phase is still subexponential in $d$. Then, in the second phase, we explore each component and solve our problems deterministically; the complexities of this phase

| Problem | Citation | Type | Time | Space |
|---|---|---|---|---|
| MIS | [35] | Rand. | $2^{O(d\log^2 d)}\log n$ | $O(n)$ |
| | [4] | Rand. | $2^{O(d\log^2 d)}\log^3 n$ | $2^{O(d\log^2 d)}\log^2 n$ |
| | [14] | Det. | $2^{O(d^2\log^2 d)}\log^* n^\dagger$ | none |
| | [34] | Rand. | $2^{O(d)}\log^2 n$ | $2^{O(d)}\log n\log\log n$ |
| | | | $2^{O(d)}\log n\log\log n$ | $2^{O(d)}\log^2 n$ |
| | reduction [6] | Rand. | $2^{O(\log^2 d\cdot\sqrt{\log n})}$ | $O(n\log^2 d)$ |
| | this thesis | Rand. | $2^{O(\log^3 d)}\log^3 n$ | $2^{O(\log^3 d)}\log^2 n$ |
| Maximal Matching | [25] | Rand. | $O(\log^3 n)^\ddagger$ | $O(\log^3 n)^\ddagger$ |
| | [14] | Det. | $2^{O(d^2\log^2 d)}\log^* n^\dagger$ | none |
| | [34] | Rand. | $2^{O(d)}\log^2 n$ | $2^{O(d)}\log n\log\log n$ |
| | | | $2^{O(d)}\log n\log\log n$ | $2^{O(d)}\log^2 n$ |
| | reduction [6] | Rand. | $2^{O(\log^2 d+\log d\log^4\log n)}$ | $O(n\log d)$ |
| | this thesis | Rand. | $2^{O(\log^2 d)}\log^3 n$ | $2^{O(\log^2 d)}\log^2 n$ |
| Approximate Maximum Matching | [26] | Rand. | $O(\log^4 n)^\ddagger$ | $O(\log^3 n)^\ddagger$ |
| | [14] | Det. | $2^{\mathrm{poly}(d)}\mathrm{poly}(\log^* n)^\dagger$ | none |
| | this thesis | Rand. | $2^{O(\log^3 d)}\mathrm{polylog}(n)$ | $2^{O(\log^3 d)}\mathrm{polylog}(n)$ |

Table 1.1: The summary of complexities of various LCAs. For the approximate maximum matching problem, $\epsilon$ is assumed to be constant. "Rand." and "Det." stand for randomized and deterministic, respectively. Reduction refers to an LCA obtained by directly applying the Parnas-Ron reduction on the distributed algorithm from the cited paper. † indicates query complexity, when time complexity is not explicitly given in the paper. ‡ indicates hidden dependence on $d$, which is at least $2^{O(d)}$ but not explicitly known.

are bounded by the component sizes.

In addition, we employ the approach from [4] to reduce the amount of space required by our LCAs. This is accomplished through the observation that as we limit the size of the subgraph explored, we also limit the required dependence between random bits. Therefore, we may instead use a construction of $k$-wise independent

random variables. This reduces the amount of truly random bits from the tape to that of the seed length, which has roughly the same asymptotic bound as the time and query complexities.

# Chapter 2

# Preliminaries

## 2.1 Graphs

The input graph $G = (V, E)$ is a simple undirected graph with $|V| = n$ vertices and a bound on the degree $d$, which is allowed to be dependent on $n$. Both parameters $n$ and $d$ are known to the algorithm. Each vertex $v \in V$ is represented as a unique positive ID from $[n] = \{1, \ldots, n\}$. For $v \in V$, let $\deg_G(v)$ denote the degree of $v$, $\Gamma_G(v)$ denote the set of neighbors of $v$, and $\Gamma^+_G(v) = \Gamma_G(V) \cup \{v\}$. For $U \subseteq V$, define $\Gamma^+_G(U) = \cup_{u \in U} \Gamma^+_G(u)$. The subscript $G$ may be omitted when it is clear from the context.

We assume that the input graph $G$ is given through an adjacency list oracle $\mathcal{O}^G$ which answers neighbor queries: given a vertex $v \in V$ and an index $i \in [d]$, the $i^{\text{th}}$ neighbor of $v$ is returned if $i \leq \deg(v)$; otherwise, $\perp$ is returned. For simplicity, we will also allow a degree query which returns $\deg(v)$ when $v$ is given; this can be simulated via a binary-search on $O(\log d)$ neighbor queries.

An *independent set* $I$ is a set of vertices such that no two vertices in $I$ are adjacent. An independent set $I$ is a *maximal independent set* if no other vertex can be added to $I$ without violating this condition.

A *matching* $M$ is a set of edges such that no two distinct edges in $M$ share a common endpoint. A matching is a *maximal matching* if no other edge can be added

to $M$ out violating this condition. Let $V(M)$ denote the set of matched vertices, and $|M|$ denote the size of the matching, defined to be the number of edges in $M$. A *maximum matching* is a matching of maximum size.

## 2.2 Local Computation Algorithms

We adopt the definition of local computation algorithms from [35]. While they also consider the parameter $d$ in their analysis, we modify their definition to make this dependence explicit as follows.

**Definition 1** *A* $(t(n,d), s(n,d), \delta(n,d))$-local computation algorithm $\mathcal{A}$ *for a computation problem is a (randomized) algorithm with the following properties.* $\mathcal{A}$ *is given access to the adjacency list oracle* $\mathcal{O}^G$ *for the input graph* $G$*, a tape of random bits, and local read-write computation memory. The total size of the random tape and the local computation memory must not exceed* $s(n,d)$*. When given an input* $x$*,* $\mathcal{A}$ *must compute an answer for this input using computation time at most* $t(n,d)$*. This answer must only depend on* $x$*,* $G$*, and the random bits. The answers given by* $\mathcal{A}$ *to all possible queries must be consistent; namely, all answers must constitute some valid solution to the computation problem. The probability that* $\mathcal{A}$ *successfully answers all queries must be at least* $1 - \delta(n,d)$*.*

## 2.3 Parnas-Ron Reduction

Our algorithms apply the reduction from distributed algorithms to LCAs proposed by Parnas and Ron [32]. This reduction was originally created as a subroutine for approximation algorithms. Suppose that when a distributed algorithm $\mathcal{A}$ is executed on a graph $G$ with degree bounded by $d$, each processor (vertex) $v$ is able to compute some function $f$ within $k$ communication rounds under the $\mathcal{LOCAL}$ model.[1] Then

[1] In the $\mathcal{LOCAL}$ model, we optimize the number of communication rounds without limiting message size; in each round, each vertex may send an arbitrarily large message to each of its neighbors.

this function $f$ must only depend on the subgraph of $G$ induced by vertices at distance at most $k$ from $v$. We can then create an LCA $\mathcal{A}'$ that computes $f$ by simulating $\mathcal{A}$. Namely, $\mathcal{A}'$ first queries the oracle to learn the structure of this subgraph, then makes the same decision on this subgraph as $\mathcal{A}$ would have done. In total, this reduction requires $d^{O(k)}$ queries to $\mathcal{O}^G$.

# Chapter 3

# Maximal Independent Set

## 3.1 Overview

Our algorithm consists of two phases. The first phase of our algorithm computes a large MIS, using a variation of Luby's randomized distributed algorithm [24]. We begin with an initially empty independent set $I$, then repeatedly add more vertices to $I$. In each round, each vertex $v$ tries to put itself into $I$ with some probability. It succeeds if none of its neighbors also tries to do the same in that round; in this case, $v$ is added to $I$, and $\Gamma^+(v)$ is removed from the graph.

By repeating this process with carefully chosen probabilities, we show that once the first phase terminates, the remaining graph contains no connected component of size larger than $d^4 \log n$ with high probability. This phase is converted into an LCA via the Parnas-Ron reduction. Lastly, in the second phase, we locally compute an MIS of the remaining graph by simply exploring the component containing the queried vertex.

## 3.2 Distributed Algorithm - Phase 1

The goal of the first phase is to find an independent set $I$ such that removing $\Gamma^+(I)$ breaks $G$ into components of small sizes. We design our variation of Luby's algorithm

based on Beck's algorithmic approach to Lovász local lemma [7]; this approach has been widely applied in many contexts (e.g., [2, 21, 35, 6]). We design our algorithm based on the degree reduction idea from [6].[1] Our algorithm turns out to be very similar to the Weak-MIS algorithm from a recent paper by Chung et al. [12]. We state their version, given in Algorithm 1, so that we may cite some of their results.

Let us say that a vertex $v$ is *active* if $v \notin \Gamma^+(I)$; otherwise $v$ is *inactive*. As similarly observed in [33], applying a round of Luby's algorithm with selection probability $1/(d+1)$ on a graph with maximum degree at most $d$ makes each vertex of degree at least $d/2$ inactive with constant probability. To apply this observation, in each iteration, we first construct a graph $G'$ of active vertices. Next, we apply Luby's algorithm so that each vertex of degree at least $d/2$ becomes inactive with constant probability. We then remove the remaining high-degree vertices from $G'$ (even if they may still be active). As the maximum degree of $G$ is halved, we repeat this similar process for $\lceil \log d \rceil$ stages until $G'$ becomes edgeless, where every vertex can be added to $I$. Since each vertex becomes high-degree with respect to the maximum degree of $G'$ at some stage, each iteration gives every vertex a constant probability to become inactive.

Chung et al. use Weak-MIS to construct an independent set such that the probability that each vertex remains active is only $1/\text{poly}(d)$ [12]. We cite the following useful lemma that captures the key idea explained earlier.

**Lemma 1** *([12]) In Algorithm 1, if $v \in V_j$, then $v$ remains active after stage $j$ with probability at most $p$ for some constant $p < 1$.*

**Proof:** For each $u \in \Gamma_{G'}(v)$, let $E_u$ denote the event where $u$ is the only vertex in $\Gamma^+_{G'}(\{u,v\})$ that selects itself. Since the maximum degree in $G'$ is at most $d/2^{j-1}$, then

$$\Pr[E_u] = p_j(1-p_j)^{|\Gamma^+_{G'}(\{u,v\})|} \geq p_j(1-p_j)^{2d/2^{j-1}} \geq \frac{p_j}{e^2}.$$

[1] Applying a similar reduction on the unmodified version gives an LCA with complexities $2^{O(\log^3 d + \log d \log\log n)}$.

**Algorithm 1** Chung et al.'s Weak-MIS algorithm

1: **procedure** WEAK-MIS($G, d$)
2: $I \leftarrow \emptyset$ ▷ begin with an empty independent set $I$
3: **for** iteration $i = 1, \ldots, c_1 \log d$ **do** ▷ $c_1$ is a sufficiently large constant
4: $G' \leftarrow G[V \setminus \Gamma^+(I)]$ ▷ subgraph of $G$ induced by active vertices
5: **for** stage $j = 1, \ldots, \lceil \log d \rceil$ **do**
6: $V_j \leftarrow \{v \in V(G') : \deg_{G'}(v) \geq d/2^j\}$ ▷ degree $\geq$ half of current max.
7: each $v \in V$ selects itself with probability $p_j = 1/\left(\frac{d}{2^{j-1}} + 1\right)$
8: **if** $v$ is the only vertex in $\Gamma^+_{G'}(v)$ that selects itself **then**
9: add $v$ to $I$ and remove $\Gamma^+_{G'}(v)$ from $G'$
10: remove $V_j$ from $G'$ ▷ remove high-degree vertices
11: add $V(G')$ to $I$ ▷ add isolated vertices in $G'$ to $I$
12: **return** $I$

Notice that $E_u$ is disjoint for each $u \in \Gamma_{G'}(v)$. Since $v \in V_j$, then $\deg_{G'}(v) \geq d/2^j$. Thus $v$ becomes inactive with probability at least

$$\sum_{u \in \Gamma_{G'}(v)} \Pr[E_u] \geq \deg_{G'}(v) \cdot \left(\frac{p_j}{e^2}\right) \geq \frac{1}{4e^2}.$$

That is, the theorem holds with parameter $p = 1 - 1/4e^2$. ■

Observe that for $v$ to remain active until the end of an iteration, it must be removed in step 10 due to its high degree. (If $v$ were removed in line 9 or line 11, then $v$ would have become inactive since either $v$ or one of its neighbors is added to $I$.) Therefore, $v$ must belong to one of the sets $V_j$. So, each vertex may remain active throughout the iteration with probability at most $p$. After $\Omega(\log d)$ iterations, the probability that each vertex remains active is only $1/\text{poly}(d)$, as desired.

Now we follow the analysis inspired by that of [6] to prove the guarantee on the maximum size of the remaining active components. Consider a set $S \subseteq V$ such that $\text{dist}_G(u, v) \geq 5$ for every distinct $u, v \in S$. We say that $S$ is active if every $v \in S$ is

active. As a generalization of the claim above, we show the following lemma.

**Lemma 2** *Let $S \subseteq V$ be such that $dist_G(u,v) \geq 5$ for every distinct $u, v \in S$, then $S$ remains active until Weak-MIS terminates with probability at most $d^{-\Omega(|S|)}$.*

**Proof:** First let us consider an individual stage. Suppose that $S$ is active at the beginning of this stage. By Lemma 1, each vertex $v \in S \cap V_j$ remains active after this stage with probability at most $p$. Notice that for each round of Luby's algorithm, whether $v$ remains active or not only depends on the random choices of vertices within distance 2 from $v$. Since the vertices in $S$ are at distance at least 5 away from one another, the events for all vertices in $S$ are independent. Thus, the probability that $S$ remains active after this stage is at most $p^{|S \cap V_j|}$.

Now we consider an individual iteration. Suppose that $S$ is active at the beginning of this iteration. By applying an inductive argument on each stage, the probability that $S$ remains active at the end of this iteration is at most $p^{\sum_{j=1}^{\lceil \log d \rceil} |S \cap V_j|}$. Recall that for $S$ to remain active after this iteration, every $v \in S$ must belong to some set $V_j$. So, $\sum_{j=1}^{\lceil \log d \rceil} |S \cap V_j| = |S|$. Thus, $S$ remains active with probability $\exp(-\Omega(|S|))$.

Lastly, we apply the inductive argument on each iteration to obtain the desired probability bound. ■

Now we are ready to apply Beck's approach to prove the upper bound on the maximum size of the remaining active components ([7], see also [35]).

**Theorem 1** WEAK-MIS$(G, d)$ *computes an independent set $I$ of an input graph $G$ within $O(\log^2 d)$ communication rounds, such that the subgraph of $G$ induced by active vertices contains no connected component of size larger than $d^4 \log n$ with probability at least $1 - 1/\text{poly}(n)$.*

**Proof:** We provide a proof sketch of this approach. There are at most $n(4d^5)^s$ distinct sets $S \subseteq V$ of size $|S| = s$ such that $\text{dist}_G(u,v) \geq 5$ for every distinct $u, v \in S$; this can be shown by counting the number of non-isomorphic trees with $s$ vertices embedded on the distance-5 graph $(V, \{(u,v) : \text{dist}_G(u,v) = 5\})$. By Lemma 2, the probability

that $S$ remains active is $d^{-\Omega(s)}$, so the probability that such active set exists is at most $n(4d^5)^s \cdot d^{-\Omega(s)}$. For $s = \log n$, this probability is at most $1/\text{poly}(n)$.

Observe that any connected subgraph of $G$ of size at least $d^4 \log n$ must contain some set $S$ satisfying the aforementioned condition. Thus, the probability that any such large set remains active after Weak-MIS terminates is also bounded above by $1/\text{poly}(n)$. ■

## 3.3 Constructing the LCA - Phase 1

We now provide the Parnas-Ron reduction of Weak-MIS in Algorithm 1 into an LCA. Let us start with a single stage, given as procedure LC-MIS-STAGE in Algorithm 2. The given parameters are the graph $G'$ (via oracle access), the degree bound $d$, the queried vertex $v$, the iteration number $i$ and the stage number $j$. Given a vertex $v$, this procedure returns one of the three states of $v$ at the end of iteration $i$, stage $j$:

- YES if $v \in I$ (so it is removed from $G'$)
- NO if $v \notin I$ and it is removed from $G'$
- $\perp$ otherwise, indicating that $v$ is still in $G'$

For simplicity, we assume that the local algorithm has access to random bits in the form of a publicly accessible function $B : V \times [c_1 \log d] \times [\lceil \log d \rceil] \to \{0, 1\}$ such that $B(v, i, j)$ returns 1 with the selection probability $p_j$ (from Algorithm 1, line 7) and returns 0 otherwise. This function can be replaced by a constant number of memory accesses to the random tape, and we will explore how to reduce the required amount of random bits in Section 3.5. Note also that we are explicitly giving the oracle $\mathcal{O}^{G'}$ as a parameter rather than the actual graph $G'$. This is because we will eventually simulate oracles for other graphs, but we never concretely create such graphs during the entire computation. Observe that each call to LC-MIS-STAGE on stage $j$ invokes up to $O(d^2)$ calls on stage $j - 1$. Simulating all stages by invoking this function with $j = \lceil \log d \rceil$ translates to $d^{O(\log d)}$ queries to the the base level $j = 0$.

**Algorithm 2** LCA of Phase 1 for a single stage of Weak-MIS

**procedure** LC-MIS-STAGE($\mathcal{O}^{G'}, d, v, i, j$)
  **if** $j = 0$ **then return** $\perp$ ▷ every vertex is initially in $G'$
  **if** LC-MIS-STAGE($\mathcal{O}^{G'}, d, v, i, j-1$) $\neq \perp$ **then**
    **return** LC-MIS-STAGE($\mathcal{O}^{G'}, d, v, i, j-1$) ▷ $v$ is already removed
  **for** each $u$ within distance 2 from $v$ on $G'$ **do** ▷ check vertices near $v$
    **if** LC-MIS-STAGE($\mathcal{O}^{G'}, d, u, i, j-1$) $= \perp$ **then**
      status($u$) $\leftarrow$ removed
    **else**
      **if** $B(u, i, j) = 1$ **then** status($u$) $\leftarrow$ selected ▷ $u$ chooses itself
      **else** status($u$) $\leftarrow$ not selected
  **if** status($v$) = selected AND $\forall u \in \Gamma_{G'}(v)$: status($u$) $\neq$ selected **then**
    **return** YES ▷ $v$ is added to $I$
  **if** status($v$) = not selected **then**
    **if** $\exists u \in \Gamma_{G'}(v)$: status($u$) = selected AND
    $\forall w \in \Gamma_{G'}(u)$: status($w$) $\neq$ selected **then**
      **return** NO ▷ a neighbor $u$ of $v$ is added to $I$, so $v$ is removed
  **if** $|\{u \in \Gamma_{G'}(v) : \text{status}(u) \neq \text{removed}\}| \geq d/2^j$ **then**
    **return** NO ▷ $v$ is removed due to its high degree
  **return** $\perp$ ▷ $v$ remains in $G'$

Next we give the LCA LC-MIS-ITERATION for computing a single iteration in Algorithm 3. The parameters are similar to that of LC-MIS-STAGE, but the return values are slightly different:

- YES if $v \in I$ (so it is inactive)
- NO if $v \in \Gamma_G^+(I)$ (so it is inactive)
- $\perp$ otherwise, indicating that $v$ is still active

In case $v$ is still active by the end of iteration $i-1$, we must simulate iteration $i$ using LC-MIS-STAGE. We must return YES if $v \in I$ (line 7 of Algorithm 3); which may occur in two cases. It can be added to $I$ in some stage, making LC-MIS-STAGE returns YES (corresponding to line 9 of Algorithm 1). It may also remain in $G'$ through all iterations, making LC-MIS-STAGE return $\perp$; $v$ is then added to $I$ because it is isolated in $G'$ (line 11 of Algorithm 1).

**Algorithm 3** LCA of Phase 1 for a single iteration of Weak-MIS

1: **procedure** LC-MIS-ITERATION($\mathcal{O}^G, d, v, i$)
2: **if** $i = 0$ **then return** $\perp$ ▷ $I$ is initially empty
3: **if** LC-MIS-ITERATION($\mathcal{O}^G, d, v, i-1$) $\neq \perp$ **then**
4: **return** LC-MIS-ITERATION($\mathcal{O}^G, d, v, i-1$) ▷ $v$ is already inactive
5: $\mathcal{O}^{G'} \leftarrow$ oracle of $G$ induced by $\{u :$ LC-MIS-ITERATION($\mathcal{O}^G, d, u, i-1$) $= \perp\}$
6: **if** LC-MIS-STAGE($\mathcal{O}^{G'}, d, v, i, \lceil \log d \rceil$) $\neq$ NO **then**
7: **return** YES ▷ $v$ is added to $I$ in some stage (YES) or at the end ($\perp$)
8: **else if** $\exists u \in \Gamma_G(v)$: LC-MIS-STAGE($\mathcal{O}^{G'}, d, u, i, \lceil \log d \rceil$) $\neq$ NO **then**
9: **return** NO ▷ a neighbor of $u$ of $v$ is added to $I$
10: **return** $\perp$ ▷ $v$ is still active

To implement this LCA, we must simulate an adjacency list oracle for $G'$ using the given oracle for $G$, which can be done as follows. For a query on vertex $v$, we call LC-MIS-ITERATION($\mathcal{O}^G, d, u, i-1$) on all vertices $u$ at distance at most 2 away from $v$. This allows us to determine whether each neighbor of $v$ is still active at the beginning of iteration $i$, as well as providing degree queries. We then modify the

ordering of the remaining neighbors (by preserving the original ordering, for example) to consistently answer neighbor queries. That is, $\mathcal{O}^{G'}$ can be simulated through at most $O(d^2)$ function calls of the form LC-MIS-ITERATION$(\mathcal{O}^G, d, u, i-1)$.

Using this subroutine, the LCA LC-MIS-PHASE1 for Weak-MIS can be written compactly as given in Algorithm 4. Via a similar inductive argument, we can show that each call to LC-MIS-PHASE1 translates to $d^{O(\log^2 d)} = 2^{O(\log^3 d)}$ calls to the original oracle $\mathcal{O}^G$. The running time for the LCA is clearly given by the same bound. The memory usage is given by the amount of random bits of $B$, which is $O(n \log^2 d)$. We summarize the behavior of this LCA through the following lemma.

**Lemma 3** LC-MIS-PHASE1 *is a* $(2^{O(\log^3 d)}, O(n \log^2 d), 0)$*-local computation algorithm that computes the distributed Weak-MIS algorithm.*

**Algorithm 4** LCA of Phase 1 (Weak-MIS)

1: **procedure** LC-MIS-PHASE1$(\mathcal{O}^G, d, v)$
2: **return** LC-MIS-ITERATION$(\mathcal{O}^G, d, v, c_1 \log d)$

## 3.4 Constructing the LCA - Phase 2 and the Full LCA

Let $G''$ be graph $G$ induced by active vertices after Weak-MIS terminates. By Theorem 1, with high probability, $G''$ contains no component of size exceeding $d^4 \log n$. Therefore, to determine whether an active vertex $v$ is in the MIS, we first apply a breadth-first search until all vertices in $C(v)$, the component containing $v$, are reached. Then we compute an MIS deterministically and consistently (by choosing the lexicographically first MIS, for instance). This procedure is summarized as LC-PHASE2 in Algorithm 5.

The algorithm only reports ERROR when the component size exceeds the bound from Theorem 1. Clearly, a call to LC-PHASE2 makes at most $\text{poly}(d) \log n$ queries

**Algorithm 5** LCA of Phase 2

**procedure** LC-PHASE2($\mathcal{O}^{G''}, d, v$)
  breadth-first search for $d^4 \log n$ steps on $G''$ to find $C_v$
  **if** $|C_v| > d^4 \log n$ **then report** ERROR ▷ occurs with prob. $1/\text{poly}(n)$
  deterministically compute an MIS $I_{C_v}$ of $C_v$
  **if** $v \in I_{C_v}$ **then return** YES
  **else return** NO

to $\mathcal{O}^{G''}$ in total. The lexicographically first maximal independent set of $C_v$ can be computed via a simple greedy algorithm with $O(d \cdot |C_v|)$ time complexity. Overall, both time and query complexities are $\text{poly}(d) \log n$.

Combining both phases, we obtain the LCA LC-MIS for computing an MIS as given in Algorithm 6. We now prove the following theorem.

**Algorithm 6** LCA for computing a maximal independent set

**procedure** LC-MIS($\mathcal{O}^G, d, v$)
  **if** LC-MIS-PHASE1($\mathcal{O}^G, d, v$) $\neq \perp$ **then**
    **return** LC-MIS-PHASE1($\mathcal{O}^G, d, v$) ▷ $v$ is already inactive
  $\mathcal{O}^{G''} \leftarrow$ oracle of $G$ induced by $\{u : \text{LC-MIS-PHASE1}(\mathcal{O}^G, d, u) = \perp\}$
  **return** LC-MIS-PHASE2($\mathcal{O}^{G''}, d, v$)

**Theorem 2** *There exists a* $(2^{O(\log^3 d)} \log n, O(n \log^2 d), 1/\text{poly}(n))$*-local computation algorithm that computes a maximal independent set of* $G$.

**Proof:** To obtain the time complexity, recall from Lemma 3 that each call to LC-PHASE1 can be answered within $2^{O(\log^3 d)}$ time using $2^{O(\log^3 d)}$ queries to $\mathcal{O}^G$. Thus the adjacency list oracle $\mathcal{O}^{G''}$ can be simulated with the same complexities. The LCA for Phase 2 makes $\text{poly}(d) \log n$ queries to $\mathcal{O}^{G''}$, resulting in $2^{O(\log^3 d)} \log n$ total computation time and queries. The required amount of space is dominated by the random bits used in Phase 1. The algorithm only fails when it finds a component of size

larger than $d^4 \log n$, which may only occur with probability $1/\text{poly}(n)$ as guaranteed by Theorem 1. ■

## 3.5 Reducing Space Usage

In this section, we directly apply the approach from [4] to reduce the amount of random bits used by our LCA. The main insight of this proof requires the observation that our algorithm only explores a small number of vertices, so full independence between random bits generated by all vertices is not required. We simply need independence between the bits used by the algorithm on a single query, which can be generated using a small number of truly random bits. This approach yields the following theorem.

**Theorem 3** *There exists a* $(2^{O(\log^3 d)} \log^3 n, 2^{O(\log^3 d)} \log^2 n, 1/\text{poly}(n))$*-local computation algorithm that computes a maximal independent set of* $G$*.*

**Proof:** Observe that the MIS algorithm in Theorem 2 constructed throughout this chapter does not require fully independent random bits in function $B$. Our algorithm can answer a query for any vertex $v$ by exploring up to $q = 2^{O(\log^3 d)} \log n$ vertices in total. So, random bits used by vertices not explored by a query do not affect our answer. One bit is used by each vertex in each round of Phase 1, and thus we only create $b = O(\log^2 d)$ random bits for each vertex. Therefore, out of $O(n \log^2 d)$ bits from function $B$, only $q \cdot b$ bits are relevant for each query.

To generate random bits for function $B$, we will apply the following lemma from [3] as restated in [4].

**Lemma 4** *([3]) For* $1 \leq k \leq m$*, there exists* $k$*-wise independent random bits* $x_1, \ldots, x_m$ *with seed length* $O(k \log m)$*. Furthermore, for* $1 \leq i \leq m$*, each* $x_i$ *can be computed in space* $O(k \log m)$*.*

Note here that each random bit generated from this construction is either 0 or 1 with equal probability, but we may require bits of function $B$ to be 1 with probability

as low as $1/(d+1)$. Thus we will need up to $\lceil \log d \rceil$ bits from this construction to obtain one bit for function $B$. So in our case, we have $k = \lceil \log d \rceil \cdot q \cdot b = 2^{O(\log^3 d)} \log n$ and $m = \lceil \log d \rceil \cdot O(n \log^2 d) = O(n \log^3 d)$. Thus the amount of space can be reduced to $O(k \log m) = 2^{O(\log^3 d)} \log n \cdot \log(n \log^3 d) = 2^{O(\log^3 d)} \log^2 n$. Similarly, the required amount of time for computing each random bit becomes $2^{O(\log^3 d)} \log^2 n$. ■

# Chapter 4

# Other Results

In this chapter, we give a summary of our LCAs for the maximal matching problem and the approximate maximum matching problem.

## 4.1 Maximal Matching

We may apply our MIS algorithm to the line graph $L(G)$ in order to compute a maximal matching on $G$ within the same asymptotic complexities as Theorem 3. Nonetheless, in [18], Israeli and Itai proposed a randomized distributed algorithm which takes $O(\log n)$ rounds to compute a maximal matching. Similarly to the MIS problem, Barenboim et al. also create a variant of this algorithm that, within $O(\log d)$ rounds, finds a large matching that breaks the remaining graph into small connected components [6]. Specifically, by running Algorithm 7, the remaining graph satisfies the following lemma.

**Lemma 5** *([6])* DISTRIBUTED-MM-PHASE1$(G, d)$ *computes a partial matching $M$ of an input graph $G$ within $O(\log d)$ communication rounds, such that the remaining graph contains no connected component of size larger than $O(d^4 \log n)$ with probability at least $1 - 1/\text{poly}(n)$.*

The proof of this lemma also makes use of Beck's analysis, but contains a more complicated argument which shows that with constant probability, each vertex loses

**Algorithm 7** Barenboim et al.'s variant of Israeli and Itai's algorithm for computing a partial matching (simplified)

1: **procedure** DISTRIBUTED-MM-PHASE1$(G, d)$
2: initialize matching $M = \emptyset$
3: **for** $i = 1, \ldots, c_2 \log d$ **do** ▷ $c_2$ is a sufficiently large constant
4: initialize directed graphs $F_1 = (V, \emptyset)$ and $F_2 = (V, \emptyset)$
5: each vertex $s$ chooses a neighbor $t$ (if any) uniformly at random, then add $(s, t)$ to $E(F_1)$
6: each vertex $t$ with positive in-degree in $F_1$ chooses a vertex $s \in \{s' : (s', t) \in E(F_1)\}$ with highest ID, then add $(s, t)$ to $E(F_2)$
7: each node $v$ with positive degree in $F_2$ chooses a bit $b(v)$ as follows:
if $v$ has an outgoing edge but no incoming edge, $b(v) = 0$
if $v$ has an ingoing edge but no outcoming edge, $b(v) = 1$
otherwise, chooses $b(v) \in \{0, 1\}$ uniformly at random
8: add every edge $(s, t) \in E(F_2)$ such that $b(s) = 0$ and $b(t) = 1$ to $M$, and remove matched vertices from $G$

some constant fraction of its neighbors in every round. Thus, applying such matching subroutine for $O(\log d)$ rounds suffices to remove or isolate each vertex with probability $1 - 1/\text{poly}(d)$. We apply the Parnas-Ron reduction on their algorithm as similarly done in Chapter 3 to obtain the following theorem, where the dependence on $d$ is reduced from $2^{O(\log^3 d)}$ to $2^{O(\log^2 d)}$.

**Theorem 4** *There exists a* $(2^{O(\log^2 d)} \log^3 n, 2^{O(\log^2 d)} \log^2 n, 1/\text{poly}(n))$*-local computation algorithm that computes a maximal matching of* $G$.

## 4.2 $(1 - \epsilon)$-approximate Maximum Matching

Our algorithm for computing a $(1 - \epsilon)$-approximate maximum matching follows the approach by Mansour and Vardi in [26], which was also used in the context of distributed computing and approximation (e.g., [23, 29, 39]). The main difference from their algorithm is that we employ our new LCA for the MIS problem. Additionally, we provide an explicit analysis of the dependence on $d$, which is considered to be a constant in their paper.

Our LCA is based on the following lemma by Hopcroft and Karp in [17], which allows us to compute an approximate maximal matching by repeatedly finding augmenting paths in order of increasing length.

**Lemma 6** *[17] Let* $M$ *and* $M^*$ *be a matching and a maximum matching in* $G$*. If the shortest augmenting path with respect to* $M$ *has length* $2k - 1$*, then* $|M| \leq (1 - 1/k)|M^*|$.

The algorithm can be summarized as follows. Let $M_0$ be an empty matching. For each $k = 1, \ldots, \lceil 1/\epsilon \rceil$, we compute a maximal set of vertex-disjoint augmenting paths of length $\ell = 2k - 1$ with respect to $M_{k-1}$, then augment these paths to $M_{k-1}$ to obtain $M_k$. As a result, $M_{\lceil 1/\epsilon \rceil}$ will be a $(1 - \epsilon)$-approximate maximum matching.

We construct a procedure LC-AMM-STAGE$(\mathcal{O}^G, k, e)$, which returns YES when $e \in M_k$, and returns NO otherwise. Notice that an edge $e$ belongs to the matching

$M_k$ when either $e \in M_{k-1}$, or $e$ is in one of the chosen augmenting paths of length $\ell$, but not both. Therefore, we need to check both of these conditions to determine whether $e \in M_k$ or not. The former can be determined with a single call to LC-AMM-STAGE($\mathcal{O}^G, k-1, e$), while the latter requires a more complicated subroutine, which also invokes the algorithm LC-MIS from Chapter 3. We give a high-level sketch of this function in Algorithm 8. A thorough version of the algorithm can be found in [26], but the discussion given below should be self-contained and covers all necessary modification to the analysis.

**Algorithm 8** Sketch of the LCA of a single stage for computing a $(1-\epsilon)$-approximate maximum matching

```
1: procedure LC-AMM-STAGE(O^G, k, e)
2:     if k = 0 then return NO                          ▷ M_0 is the empty matching
3:     InPreviousMatching ← LC-AMM-STAGE(O^G, k − 1, e)
4:     ℓ ← 2k − 1
5:     O^{C_k} ← oracle for the conflicting graph C_k
6:     InAugmentingPath ← NO
7:     for each augmenting path P of length ℓ containing e do
8:         if LC-MIS(O^{C_k}, ℓ^2 d^ℓ, v_P) = YES then InAugmentingPath ← YES
9:     return InPreviousMatching ⊕ InAugmentingPath
```

To check whether $e$ is in a chosen augmenting path, we compute a maximal set of vertex-disjoint augmenting paths of length $\ell$ by invoking the LCA for finding an MIS on the *conflict graph* $C_k$, defined as follows. Each vertex $v_P \in V(C_k)$ represents an augment path $P$ of length $\ell$ with respect to $M_{k-1}$. The edge set $E(C_k)$ contains $(v_P, v_{P'})$ if and only if $P$ and $P'$ are not vertex-disjoint. To simulate the oracle $\mathcal{O}^{C_k}$, notice that a vertex $v$ may occur in up to $O(\ell d^\ell)$ paths of length $\ell$. So, given a path $P$ consisting of $\ell$ edges, we check all $O(\ell d^\ell)$ paths $P'$ passing through each vertex in $P$ whether each of them is an augmenting path. Recall that a path $P'$ is an augmenting path with respect to $M_{k-1}$ if every odd-numbered edge is not in

$M_{k-1}$ but every even-numbered edge is in $M_{k-1}$. Thus checking whether each path $P'$ is an augmenting path requires $\ell$ calls of the form LC-AMM-STAGE$(\mathcal{O}^G, k-1, e')$. That is, each query to $\mathcal{O}^{C_k}$ can be simulated with $O(\ell^2 d^\ell) = O(k^2 d^{O(k)})$ calls to LC-AMM-STAGE$(\mathcal{O}^G, k-1, e')$.

Observe that $C_k$ is a graph of $nd^{O(k)}$ vertices with bounded degree $kd^{O(k)}$. By using our LCA for the MIS problem, checking whether a path $P$ is a chosen augmenting path of $C_k$ requires $2^{O(\log^3(kd^k))} \log(nd^{O(k)}) = 2^{O(k^3 \log^3(kd))} \log n$ queries to $\mathcal{O}^{C_k}$. We need to check on all paths containing an edge $e$, yielding $kd^{O(k)}$ calls to LC-MIS in total. Multiplying these terms together, we have that each call to LC-AMM-STAGE$(\mathcal{O}^G, k, e)$ requires up to $2^{O(k^3 \log^3(kd))} \log n$ calls to LC-AMM-STAGE$(\mathcal{O}^G, k-1, e')$.

By multiplying the query complexities of all $k = O(1/\epsilon)$ stages, the total query complexity becomes

$$2^{O(k^4 \log^3(kd))} \log^k n = 2^{O(\frac{1}{\epsilon^4} \log^3 \frac{d}{\epsilon})} \log^{\frac{1}{\epsilon}} n.$$

The running time is asymptotically within this same bound. The total number of random bits used in function $B$ is

$$k \cdot nd^{O(k)} \cdot \log^2(kd^{O(k)}) = \frac{nd^{O(\frac{1}{\epsilon})}}{\epsilon^3} \log^2\left(\frac{d}{\epsilon}\right).$$

Nonetheless, we may apply the same technique from [4] to reduce the amount of memory usage. When $\epsilon$ is constant, this approach results in the following theorem.

**Theorem 5** *For any constant $\epsilon > 0$, there exists a $(2^{O(\log^3 d)}\text{polylog}(n), 2^{O(\log^2 d)}\text{polylog}(n), 1/\text{poly}(n))$-local computation algorithm that computes a $(1-\epsilon)$-approximate maximum matching of $G$.*

# Bibliography


[1] Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Property-preserving data reconstruction. *Algorithmica*, 51(2):160–182, 2008.

[2] Noga Alon. A parallel algorithmic version of the local lemma. *Random Structures & Algorithms*, 2(4):367–378, 1991.

[3] 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.

[4] Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1132–1139. SIAM, 2012.

[5] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 475–486. IEEE, 2006.

[6] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 321–330. IEEE, 2012.

[7] József Beck. An algorithmic approach to the Lovász local lemma. I. *Random Structures & Algorithms*, 2(4):343–365, 1991.

[8] Christian Borgs, Michael Brautbar, Jennifer Chayes, Sanjeev Khanna, and Brendan Lucier. The power of local information in social networks. In *Internet and Network Economics*, pages 406–419. Springer, 2012.

[9] Zvika Brakerski. Local property restoring. *Unpublished manuscript*, 2008.

[10] Andrea Campagna, Alan Guo, and Ronitt Rubinfeld. Local reconstructors and tolerant testers for connectivity and diameter. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 411–424. Springer, 2013.

[11] Bernard Chazelle and C Seshadhri. Online geometric reconstruction. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 386–394. ACM, 2006.

[12] Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the lovász local lemma and graph coloring. *Submitted to PODC*, 2014.

[13] Akashnil Dutta, Reut Levi, Dana Ron, and Ronitt Rubinfeld. A simple online competitive adaptation of lempel-ziv compression with efficient random access support. In *Data Compression Conference (DCC), 2013*, pages 113–122. IEEE, 2013.

[14] Guy Even, Moti Medina, and Dana Ron. Best of two local models: Local centralized and local distributed algorithms. *arXiv preprint arXiv:1402.3796*, 2014.

[15] David Gamarnik and Madhu Sudan. Limits of local algorithms over sparse random graphs. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 369–376. ACM, 2014.

[16] Avinatan Hassidim, Yishay Mansour, and Shai Vardi. Local computation mechanism design. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 601–616. ACM, 2014.

[17] 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.

[18] Amos Israeli and Alon Itai. A fast and simple randomized parallel algorithm for maximal matching. *Information Processing Letters*, 22(2):77–80, 1986.

[19] Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of lipschitz functions with applications to data privacy. *SIAM Journal on Computing*, 42(2):700–731, 2013.

[20] Satyen Kale, Yuval Peres, and C Seshadhri. Noise tolerance of expanders and sublinear expander reconstruction. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 719–728. IEEE, 2008.

[21] Pierre Kelsen. Fast parallel matching in expander graphs. In *Proceedings of the fifth annual ACM symposium on Parallel algorithms and architectures*, pages 293–299. ACM, 1993.

[22] Reut Levi, Dana Ron, and Ronitt Rubinfeld. Local algorithms for sparse spanning graphs. *arXiv preprint arXiv:1402.3609*, 2014.

[23] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. In *Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pages 129–136. ACM, 2008.

[24] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.

[25] 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.

[26] Yishay Mansour and Shai Vardi. A local computation approximation scheme to maximum matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 260–273. Springer, 2013.

[27] Sharon Marko and Dana Ron. Approximating the distance to properties in bounded-degree and general sparse graphs. *ACM Transactions on Algorithms (TALG)*, 5(2):22, 2009.

[28] S Muthukrishnan, Martin Strauss, and Xian Zheng. Workload-optimal histograms on streams. In *Algorithms–ESA 2005*, pages 734–745. Springer, 2005.

[29] 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.

[30] Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1123–1131. SIAM, 2012.

[31] Lorenzo Orecchia and Zeyuan Allen Zhu. Flow-based algorithms for local graph clustering. In *SODA*, pages 1267–1286. SIAM, 2014.

[32] 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.

[33] D Peleg. Distributed computing: A locality-sensitive approach. 2000. *SIAM Philadelphia PA*.

[34] Omer Reingold and Shai Vardi. New techniques and tighter bounds for local computation algorithms. *arXiv preprint arXiv:1404.5398*, 2014.

[35] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. *arXiv preprint arXiv:1104.1377*, 2011.

[36] Michael Saks and C Seshadhri. Local monotonicity reconstruction. *SIAM Journal on Computing*, 39(7):2897–2926, 2010.

[37] Daniel A Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013.

[38] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the xor lemma. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 537–546. ACM, 1999.

[39] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 225–234. ACM, 2009.