# A Unified Sparsification Approach for Matching Problems in Graphs of Bounded Neighborhood Independence

Lazar Milenković
School of Electrical Engineering, Tel Aviv University
Tel Aviv
milenkovic.lazar@gmail.com

Shay Solomon
School of Electrical Engineering, Tel Aviv University
Tel Aviv
solo.shay@gmail.com

## ABSTRACT

The neighborhood independence number of a graph $G$, denoted by $\beta = \beta(G)$, is the size of the largest independent set in the neighborhood of any vertex. Graphs with bounded neighborhood independence, already for constant $\beta$, constitute a wide family of possibly dense graphs, including line graphs, unit-disk graphs, claw-free graphs and graphs of bounded growth, which has been well-studied in the area of distributed computing. In ICALP'19, Assadi and Solomon [8] showed that, for any $n$-vertex graph $G$, a maximal matching can be computed in $O(n \log n \cdot \beta)$ time in the classic sequential setting. This result shows that, surprisingly, for almost the entire regime of parameter $\beta$, a maximal matching can be computed much faster than reading the entire input. The algorithm of [8], however, is inherently sequential and centralized. Moreover, a maximal matching provides a 2-approximate (maximum) matching, and the question of whether a better-than-2-approximate matching can be computed in sublinear time remained open.

In this work we propose a unified and surprisingly simple approach for producing $(1+\epsilon)$-approximate matchings, for arbitrarily small $\epsilon > 0$. Specifically, set $\Delta = O(\frac{\beta}{\epsilon} \log \frac{1}{\epsilon})$ and let $G_\Delta$ be a random subgraph of $G$ that includes, for each vertex $v \in G$, $\Delta$ random edges incident on it. We show that, with high probability, $G_\Delta$ is a $(1+\epsilon)$-*matching sparsifier* for $G$, i.e., the maximum matching size of $G_\Delta$ is within a factor of $1+\epsilon$ from that of $G$. One can then work on the sparsifier $G_\Delta$ rather than on the original graph $G$. Since $G_\Delta$ can be implemented efficiently in various settings, this approach is of broad applicability; some concrete implications are:

- A $(1+\epsilon)$-approximate matching can be computed in the classic sequential setting in $O(\frac{n \cdot \beta}{\epsilon^2} \cdot \log \frac{1}{\epsilon})$ time, shaving a $\log n$ factor from the runtime of [8] (for any constant $\epsilon$), and more importantly achieving an approximation factor of $1+\epsilon$ rather than 2. For constant $\epsilon$, our runtime is tight, matching a lower bound of $\Omega(n \cdot \beta)$ due to [5, 8].
- $G_\Delta$ can be computed in a single communication round in distributed networks. Consequently, a $(1+\epsilon)$-approximate matching can be computed in $\left(\frac{\beta}{\epsilon} \log \frac{1}{\epsilon}\right)^{O(1/\epsilon)} + O(\frac{1}{\epsilon^2}) \cdot \log^* n$ communications rounds, which reduces to $O(\log^* n)$ rounds when $\beta$ and $\epsilon$ are constants; the previous (deterministic) algorithm by Barenboim and Oren [16, 17] requires a similar number of rounds but its approximation factor is $2+\epsilon$. Our sparsifier also provides a rare example of an algorithm achieving a sublinear *message complexity*.
- A $(1+\epsilon)$-approximate matching can be dynamically maintained with update time $O(\frac{\beta}{\epsilon^3} \log \frac{1}{\epsilon})$; the previous (deterministic) algorithm by Barenboim and Maimon [14] achieves approximation factor 2 with a higher (by a factor of $\sqrt{\frac{n}{\beta}}$, for constant $\epsilon$) update time of $O(\sqrt{\beta n})$.


*SPAA '20, July 15–17, 2020, Virtual Event, USA*

ACM ISBN 978-1-4503-6935-0/20/07…$15.00
https://doi.org/10.1145/3350755.3400248

## CCS CONCEPTS

• **Theory of computation → Sparsification and spanners**; *Dynamic graph algorithms*; *Distributed algorithms.*

## KEYWORDS

distributed algorithm, graph matching, maximum matching, neighborhood independence, sublinear time, sparsification

**ACM Reference Format:**
Lazar Milenković and Shay Solomon. 2020. A Unified Sparsification Approach for Matching Problems in Graphs of Bounded Neighborhood Independence. In *ACM/IEEE Joint Conference on Digital Libraries in 2020 (SPAA '20), July 15–17, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3350755.3400248

## 1 INTRODUCTION

Graph matching is one of the most important and well-studied problems in combinatorial optimization. Perhaps the original motivations of the problem, dating back to the 40s, were to minimize transportation costs and optimize assignments of personnel to job positions [32, 50, 55, 82]. Since then, graph matching algorithms have found a rich plethora of applications, ranging from scheduling and object recognition to numerical analysis and computational chemistry. Moreover, matching algorithms were also used as key subroutines in other important optimization algorithms, such as the traveling salesman problem [29], planar Max-Cut [45, 73] and shortest paths [64]. In general, graph matching problems are intimately connected to various other important problems.

Maximum cardinality matching (MCM) is perhaps the most basic graph matching problem and its importance is hard to overstate. The first (sequential) algorithms for computing an MCM [33, 54], dating back half a century ago, are considered cornerstone results in computer science, having triggered numerous groundbreaking works on a variety of combinatorial optimization problems. In particular, the work of Micali and Vazirani [70, 83], building on that of Hopcroft and Karp for bipartite graphs [51], shows that

an *exact* MCM can be computed in time $O(m\sqrt{n})$. While the exact MCM problem is not fully resolved yet, the approximate version is (up to the $\epsilon$ dependence): the algorithm of [51, 70, 83] provides a $(1+\epsilon)$-approximate MCM within time $O(m/\epsilon)$, for any $\epsilon > 0$. In general graphs, the approximate $(1+\epsilon)$-MCM algorithm of [70, 83] is quite sophisticated; an alternative and much simpler approach is to settle for a *maximal matching*, which can be computed via a naïve greedy $O(m)$-time algorithm. Since a maximal matching provides a 2-approximate MCM, this easily solves the approximate MCM problem, albeit with an approximation factor of 2.

Due to its practical importance and wide applicability, graph matching has been studied in a variety of settings and computation models, including parallel algorithms (see, e.g., [36, 53, 58, 66]), distributed algorithms (see, e.g., [12, 49, 61, 67]), streaming algorithms (see, e.g., [1, 4, 43, 57]), online algorithms (e.g., [19, 26, 37, 59]), dynamic algorithms (see, e.g., [18, 23, 44, 80]), and massively parallel computation (MPC) (see, e.g., [4, 6, 31, 39]).

In a breakthrough paper from SODA'19, Assadi, Chen and Khanna [5] presented a randomized sequential algorithm for $(\Delta+1)$-(vertex) coloring that runs in time $\tilde{O}(n\sqrt{n})$, which in general is *sublinear in the input size*. The same paper also shows that such a result cannot be achieved for several related problems, including those of maximal matching and $(1+\epsilon)$-approximate MCM. In ICALP'19, Assadi and Solomon [8] gave a randomized algorithm for computing a maximal matching in $O(n \log n \cdot \beta)$ time, where $\beta = \beta(G)$ is the *neighborhood independence number* of the input graph $G$, which is the size of the largest independent set in the neighborhood of any vertex. Moreover, as shown in [8], the lower bound from [5] for general graphs can be extended to prove that $\Omega(n \cdot \beta)$ time is necessary for computing a maximal matching and $(1+\epsilon)$-approximate MCM on graphs with neighborhood independence $\beta$, giving a nearly tight (up to a $\log n$ factor) time bound for the maximal matching problem for any $\beta$, and also suggesting that $\beta$ is in some sense the "right" parameter for measuring the runtime of maximal matching algorithms. While a maximal matching provides a 2-approximate MCM, the following question remained open:

Question 1.1. *Can a better-than-2-approximate matching be computed in sublinear time for bounded $\beta$?*

Graphs with bounded neighborhood independence, already for constant $\beta$, constitute a wide family of graphs, including line graphs, unit-disk graphs, claw-free graphs and graphs of bounded growth. Such graphs are possibly dense already for small values of $\beta$ (e.g., the $n$-clique has $\Omega(n^2)$ edges and $\beta = 1$). (More details on this graph family appear in Section 1.1.) This family of graphs has been studied primarily in the area of distributed computing; see [10, 15–17, 38, 47, 65, 78, 79] and the references therein. A drawback of the algorithm of Assadi and Solomon [8] is that it is inherently sequential and centralized, and requires a global coordination; in particular, it is unclear if it can be parallelized or distributed efficiently.

In this work we present a unified approach for producing a $(1+\epsilon)$-approximate MCMs, for arbitrarily small $0 < \epsilon < 1$. Our approach is to construct a *matching sparsifier*, which is a sparse subgraph that approximately preserves the MCM size. As will be shown next, our sparsifier construction is as simple as it gets, it is inherently *local*, which makes it naturally suitable for several settings, such as the sequential, distributed, and dynamic computational models.

## 1.1 Our Contribution

A $\gamma$-matching sparsifier, for a graph $G = (V, E)$ and an approximation parameter $\gamma \geq 1$, is a subgraph $G'$ of $G$ that preserves the MCM size to within a factor of $\gamma$, that is, $|\mathsf{MCM}(G)| \leq \gamma \cdot |\mathsf{MCM}(G')|$; the parameter $\gamma$ will be referred to as the *approximation factor* of the sparsifier $G'$, and we shall focus on the regime of $\gamma = 1+\epsilon$, for an arbitrarily small $0 < \epsilon < 1$. Clearly, the sparsifier should use as few edges as possible (ideally, $O(n)$ or even $O(|\mathsf{MCM}(G)|)$) while achieving an approximation factor close to 1.

Our matching sparsifier $G_\Delta$ for a graph $G$ is constructed as follows. For a fixed $\Delta = O(\frac{\beta}{\epsilon} \log \frac{1}{\epsilon})$, each vertex marks $\Delta$ random edges incident on it in the graph $G$; the sparsifier $G_\Delta$ consists of all the marked edges.

The meta-theorem of our work is summarized in the following statement, which is deliberately vague in the sense that it does not state the runtime bounds of the construction; the exact runtime bounds are model-dependent, and are detailed for each model separately in Section 1.2.

Theorem 1.2. *One can "efficiently construct" a $(1+\epsilon)$-matching sparsifier with $O\left(n \cdot \frac{\beta}{\epsilon} \log \frac{1}{\epsilon}\right)$ edges, for any $0 < \epsilon < 1$ and $\beta = O\left(\frac{\epsilon n}{\log n}\right)$, where the bound on the approximation factor holds with high probability. The size bound is actually $O\left(|\mathsf{MCM}(G)| \cdot \frac{\beta}{\epsilon} \log \frac{1}{\epsilon}\right)$.*

**Bounded Neighborhood Independence Graphs.** Our meta-theorem concerns graphs of bounded neighborhood independence $\beta$. Already for constant $\beta$, this constitutes a rich family of graphs. A notable example is *line graphs*, which play a central role in the area of distributed computing, and have neighborhood independence number of at most 2. Another example is *bounded growth graphs*. A graph $G(V, E)$ is said to be of bounded growth iff there exists a function such that for every vertex $v \in V$ and integer $r \geq 1$, the size of the largest independent set in the $r$-neighborhood of $v$ is bounded by $f(r)$. Intersection graphs of geometrical objects such as proper interval graphs [48], unit-disk graphs [46], quasi-unit-disk graphs [62] and general disc graphs [47] are all bounded-growth graphs. The family of *bounded diversity graphs*, for which there has been a recent growing interest in efficient distributed algorithms [11, 13, 15], is yet another subfamily of bounded neighborhood independence graphs. The *diversity* of a vertex $v$ is the number of maximal cliques that $v$ belongs to in the graph, and the diversity of the graph is the maximum diversity of a vertex. Since each clique contains just one independent vertex, the neighborhood independence number of a graph with diversity $k$ is at most $k$.

We remark that graphs with neighborhood independence number $\beta$ are sometimes referred to as $(\beta+1)$-*claw-free graphs*, i.e., graphs that do not contain $K_{1,\beta+1}$ as an induced subgraph. These graphs were extensively studied in the context of structural graph theory; see the series of papers by Chudnovsky and Seymour, starting with [30], and also the survey by Faudree et al. [35].

We next consider several aspects of our meta-theorem.

**Uniform Sparsity.** We prove that the size of our sparsifier is upper bounded by $4|\mathsf{MCM}(G)| \cdot \Delta$, where $|\mathsf{MCM}(G)|$ is the MCM size of the input graph $G$ and $\Delta$ is the number of neighboring edges that every vertex marks. For super-constant values of $\beta$, this size bound

could be significantly smaller than the naïve $n \cdot \Delta$ upper bound. This more refined size upper bound of the sparsifier is proved in Section 2.2. Moreover, in the same section we show that our sparsifier is also *uniformly sparse*, which will be crucial for achieving a fast distributed algorithm (in Section 3.2). We use the *arboricity* (see Definition 2.11) as the measure of uniform sparsity.

**Randomization and Approximation.** Both randomization and approximation are required for our construction of sparsifiers, $G_\Delta$, where every vertex marks $\Delta$ (random) neighboring edges and the sparsifier consists of all the marked edges. First, it is easy to verify that if the edges marked for inclusion in the sparsifier are chosen deterministically, the resulting sparsifier could have an arbitrarily poor approximation ratio. Second, even allowing randomization, we demonstrate that our construction of sparsifiers cannot preserve the exact size of the MCM with reasonable probability, unless $\Delta$ is close to $n$. Further details are provided in Section 2.2.

## 1.2 Applications of Our Meta-Theorem

Our sparsification algorithm is inherently *local*, since every vertex chooses *independently* of other vertices which among its neighboring edges to mark for inclusion in the sparsifier. As a result, it can be distributed, parallelized and dynamized easily and efficiently, and is therefore of rather broad applicability.

We next highlight some concrete applications.

**Centralized Sequential Model.** In Section 3.1 we demonstrate that our sparsifier $G_\Delta$ can be computed within time linear in its size, namely, $O(n \cdot \frac{\beta}{\epsilon} \log \frac{1}{\epsilon})$. We can then run the standard $(1+\epsilon)$-approximate MCM algorithm of [52, 70, 83], which takes $O(m/\epsilon)$ time for any $m$-edge graph, leading to a total runtime of $O(n \cdot \frac{\beta}{\epsilon^2} \log \frac{1}{\epsilon})$, which is sublinear in the graph size for almost the entire regime of parameter $\beta$. (This result is given in Theorem 3.1.) Moreover, our algorithm achieves an approximation factor of $1+\epsilon$, while shaving a factor of $\log n$ from the runtime of [8] in the entire regime of parameter $\beta = O\left(\frac{n}{\log n}\right)$ (for any constant $\epsilon$). We stress that the complementary regime of $\beta = \omega\left(\frac{n}{\log n}\right)$ is irrelevant, since in that regime the runtime of [8] is inferior to the naïve $O(n^2)$ runtime of the greedy maximal matching algorithm. Finally, our runtime is *tight* in this entire regime of $\beta = O\left(\frac{n}{\log n}\right)$ (for constant $\epsilon$), as it matches the lower bound of $\Omega(n \cdot \beta)$ due to [5, 8]. In particular, this provides an optimal (up to the $\epsilon$-dependence) positive resolution to Question 1.1 for the regime $\beta = O\left(\frac{n}{\log n}\right)$.

In fact, by using the refined upper bound on the size of our construction, we obtain a runtime of $O(|\mathtt{MCM(G)}| \cdot \frac{\beta}{\epsilon^2} \log \frac{1}{\epsilon})$, which could be significantly smaller than the above bound of $O(n \cdot \frac{\beta}{\epsilon^2} \log \frac{1}{\epsilon})$.

**Distributed Computing.** In Section 3.2 we demonstrate the applicability of our sparsifier in the area of distributed computing. It is readily verified that our sparsifier can be computed in standard distributed networks within a single round of communication. Since the arboricity of our sparsifier is low, we can run on top of it a second sparsifier construction, due to Solomon from ITCS'18 [81], which has a low maximum degree. This will ultimately (see Theorem 3.2) give rise to a distributed algorithm for computing a $(1+\epsilon)$-approximate matching in $\left(\frac{\beta}{\epsilon}\right)^{O(1/\epsilon)} + O\left(\frac{1}{\epsilon^2}\right) \log^* n$ communication rounds; when $\beta$ and $\epsilon$ are constants, the number of rounds is reduced to $O(\log^* n)$, which provides an improvement over the (deterministic) algorithm of Barenboim and Oren [16, 17] that applies to graphs of constant neighborhood independence and requires the same number of rounds, but achieves a $(2+\epsilon)$-approximation.

We then apply our sparsifier construction to solve the problem of distributed $(1+\epsilon)$-approximate MCM while achieving a sublinear (depending, of course, on the neighborhood independence number $\beta$) *message complexity*. (Theorem 3.3 summarizes this result.) This provides a new example to the very small pool of distributed algorithms that achieve a sublinear message complexity.

**Dynamic Algorithm.** We employ our sparsifier construction $G_\Delta$ to efficiently maintain a $(1+\epsilon)$-approximate MCM in the standard fully dynamic setting. This dynamic setting is sequential and centralized, and it allows both insertions and deletions of edges, while the vertex set is fixed, where in each step a single edge is added to the graph or removed from it; such a step is called an *edge update* (or shortly, an *update*). A common assumption is that initially there are no edges in the graph, but this assumption does not lose generality if the sequence of updates is large enough. Various graph problems have been extensively studied in the fully dynamic setting since the 80s; refer to [18, 20–25, 28, 44, 71, 77, 80, 81] and the references therein for works on graph matching and related problems.

One may try to optimize the *amortized* (i.e., average) update time of the algorithm or its *worst-case* (i.e., maximum) update time, over a worst-case sequence of graphs; it is obviously more challenging to achieve a low worst-case update time. In Section 3.3 we demonstrate that a $(1+\epsilon)$-approximate MCM can be maintained in the fully dynamic setting with a worst-case update time of $O\left(\Delta/\epsilon^2\right) = O(\frac{\beta}{\epsilon^3} \log \frac{1}{\epsilon})$; the previous (deterministic) algorithm by Barenboim and Maimon [14] achieves approximation factor 2 with a higher (by a factor of $\sqrt{\frac{n}{\beta}}$, for constant $\epsilon$) update time of $O(\sqrt{\beta n})$.

Remarkably, our update time bound holds *deterministically* and the approximation factor of $(1+\epsilon)$ holds with high probability against an *adaptive adversary*. (Theorem 3.5 summarizes this result.) This is a rare example of a randomized algorithm that does not make the *oblivious adversary* assumption, in which the adversary (i.e., the entity inserting and deleting edges to and from the graph) cannot decide its updates adaptively based on the algorithm's output; all other randomized algorithms for dynamic graph matching problems make this assumption, with the sole exception of the very recent STOC'20 paper of Wajc [84].

## 1.3 Preliminaries

We shall sometimes abbreviate maximum (cardinality) matching as MCM, as was already done in Sections 1.1 and 1.2. Also, for a graph $G$, let $|\mathtt{MCM}(G)|$ denote the MCM size in $G$, where $\mathtt{MCM}(G)$ may denote an arbitrary MCM of $G$.

As usual, for a graph $G = (V, E)$, let $n = |V|$ denote the number of vertices in the graph.

The *degree* of a vertex $v$ in $G$, denoted by $\deg(v)$, is the number of edges incident on $v$ in $G$. The *maximum degree* of $G$ is the maximum degree of any vertex in $G$. Also, for a vertex set $V' \subseteq V$, let $\deg_{V'}(v)$ denote the degree of $v$ in the subgraph $G[V']$ of $G$ induced by $V'$; note that $\deg(v) = \deg_V(v)$.

We say that an event happens *with high probability* if it happens with probability at least $1-1/\text{poly}(n)$. In some situations it is further required that the success probability will be at least $1 - 1/n^c$, for an arbitrarily large constant $c > 1$.

## 2 THE SPARSIFIER

**The random sparsifier, $G_\Delta$.** Given a graph $G = (V, E)$, we "mark" $\Delta$ neighbors of each vertex uniformly at random (without replacement). If the degree of a vertex is at most $\Delta$, we mark all its neighbors. Denote by $E_\Delta$ the set of marked edges, and let $G_\Delta = (V, E_\Delta)$ be the subgraph of $G$ over the vertex set $V$ and the set $E_\Delta$ of marked edges. We will refer to $G_\Delta$ as a *random sparsifier* of $G$.

In Section 2.1 we bound the approximation factor of $G_\Delta$, which is the main technical contribution of this work. Some other useful properties of this sparsifier are provided in Section 2.2.

### 2.1 $G_\Delta$ is a $(1+\epsilon)$-sparsifier

This section is devoted to the proof of the following theorem. We stress that the proof is subtle. The naïve approach for proving this theorem overlooks strong probability dependencies and is therefore doomed to failure; more details are provided in the paragraph preceding Lemma 2.6 and in the proof of that lemma.

Theorem 2.1. *For any graph $G$ with neighborhood independence $\beta$, where $0 < \epsilon < 1$ and $\beta = O\left(\frac{\epsilon n}{\log n}\right)$, $G_\Delta$ is a $(1+\epsilon)$ matching sparsifier for $G$ with high probability, for $\Delta = \Theta\left(\frac{\beta}{\epsilon}\log\frac{1}{\epsilon}\right)$.*

**Remark.** The success probability is at least $1-1/\text{poly}(n')$, where $n'$ is the number of non-isolated vertices. Clearly, the success probability should depend on $n'$ rather than $n$, and we shall henceforth assume in what follows that there are no isolated vertices.

In what follows let $M$ be an arbitrary MCM in $G$, and denote the sets of matched and free vertices in $M$ by $V_M$ and $V_F$, respectively.

The following lemma, which will be used for proving Theorem 2.1, provides a lower bound on the MCM size in terms of the number of (non-isolated) vertices $n$ and $\beta$.

Lemma 2.2. $|M| \geq \frac{n}{\beta+2}$.

**Proof:** Suppose for contradiction that $|M| < \frac{n}{\beta+2}$, i.e., $|M|(\beta+2) < n$. Given that $|V_M| = 2|M|$, we get that $|V_F| > |M|\beta$. Since $M$ is an MCM, $V_F$ is an independent set, hence all the edges incident on $V_F$ lead to $V_M$. Racalling the assumption of having no isolated vertices, each vertex of $V_F$ is adjacent to at least one edge, and the number of edges from $V_F$ to $V_M$ is at least $|V_F|$. Then, by the pigeonhole principle there exists an edge $e$ of $M$ that is incident to more than $\beta$ vertices of $V_F$. If all the vertices are incident to one endpoint of $e$, this contradicts the fact that the neighborhood independence of $G$ is $\beta$. On the other hand, if each endpoint of $e$ leads to $V_F$, then since $e$ is incident to at least $\beta+1$ vertices of $V_F$, this yields a length-3 augmenting path, which contradicts the fact that $M$ is of maximum size. It follows that $|M| \geq \frac{n}{\beta+2}$, as required. ∎

To prove Theorem 2.1, we will show that the subgraph $G_\Delta[V_M]$ of the sparsifier $G_\Delta$ induced by $V_M$ contains a matching of size greater than $\left(1-\frac{\epsilon}{2}\right)|M|$ with high probability.

Definition 2.3. *A vertex is said to be of* high degree *(respectively,* low degree*) if its degree is bigger than (resp., at most) $\Delta$.*

We stress that this partition of the vertex set into high and low degree vertices depends only on the input graph and $\Delta$, and it does not depend on randomness or the choices made by the algorithm in any way; our proof makes critical use of this observation.

Let $M_\Delta$ be an arbitrary MCM in $G_\Delta[V_M]$ *that maximizes the number of low degree matched vertices.* Let $W_M$ and $W_F$ denote the sets of matched and free vertices in $M_\Delta$, respectively, and note that $W_M, W_F \subseteq V_M$. Our new goal is to show that $|W_M| > \left(1-\frac{\epsilon}{2}\right)|V_M|$ with high probability, from which Theorem 2.1 would follow.

Claim 2.4. *$W_F$ contains only high degree vertices.*

**Proof:** Suppose for contradiction that there is a low degree vertex $w$ in $W_F$. Since $w$ is also in $V_M$, it has a mate $mate(w)$ in $M$, and as $w$ is of low degree, all its neighborhood from the original graph is included in the sparsifier, hence edge $(w, mate(w))$ belongs to $G_\Delta$. It follows that $mate(w)$ is in $W_M$, as otherwise we could add $(w, mate(w))$ to $M_\Delta$, which is a contradiction to the fact that $M_\Delta$ is an MCM in $G_\Delta[V_M]$ . Let $x$ be a mate of $mate(w)$ in $M_\Delta$. We modify $M_\Delta$ by removing $(mate(w), x)$ from it and adding $(w, mate(w))$ to it in its place. Clearly, the cardinality of the resulting matching remains the same, hence we can apply the same argument for $x$ instead of $w$, in case $x$ is of low degree. Indeed, the mate of $x$ in $M$, $mate(x)$, must be in $W_M$, otherwise the resulting matching were not of maximum size. This process yields an alternating path, and it must terminate since the symmetric difference between the matching resulting at each step and M strictly decreases. Upon termination, the number of low degree vertices in the resulting matching has increased, which contradicts our choice of $M_\Delta$. ∎

Since $W_F$ is an independent set, Claim 2.4 yields the following corollary.

Corollary 2.5. *There exists an independent set in $G_\Delta$ of at least $|W_F|$ high degree vertices of $V_M$.*

The following lemma is central to the proof of Theorem 2.1. As shown below, we do not reason probabilistically on the set $W_F$ directly, since this set depends on the random sparsifier $G_\Delta$. In particular, the probability that a certain event involving some vertex $v$ occurs, when conditioned on $v$ belonging to $W_F$, could be very different than without this conditioning. This technical hurdle enforces us to employ a more subtle argument.

Lemma 2.6. *$|W_F| < \frac{\epsilon}{2}|V_M|$ with high probability, for $\beta = O\left(\frac{\epsilon n}{\log n}\right)$.*

**Proof:** To prove the lemma, we upper bound the probability that a large independent set $U$ of high degree vertices exists in $G_\Delta$. (By Corollary 2.5, if no such independent set exists, the size of $W_F$ cannot be too big, which will complete the proof of the lemma.)

We next make the following claim, which by the union bound over all such sets $U$ provides the required result.

Claim 2.7. *Fix an arbitrary subset $U$ of $V_M$ of size at least $\frac{\epsilon}{2}|V_M|$ such that all its vertices are of high degree. The probability that $U$ forms an independent set in $G_\Delta$, denoted by $\mathcal{E}^{(U)}$, is at most $(\epsilon/24)^{|U|}$.*

**Proof:** Throughout the proof we take $\Delta$ to be $20\frac{\beta}{\epsilon}\ln\frac{24}{\epsilon} = O\left(\frac{\beta}{\epsilon}\log\frac{1}{\epsilon}\right)$.

Notice that every vertex in $V_M$ has at most $\beta$ neighbors in $V_F$, since $V_F$ is an independent set and the neighborhood independence number of $G$ is $\beta$. For an arbitrary vertex $v \in U$, denote the event that all edges marked due to $v$ lie outside $U$ (recall that all such edges are taken to the sparsifier $G_\Delta$) by $\mathcal{E}_v^{(U)}$. Then we have

$$\begin{aligned}\mathbf{P}(\mathcal{E}_v^{(U)}) &= \prod_{i=1}^{\Delta}\left(\frac{\deg_{V\setminus U}(v)-i+1}{\deg(v)-i+1}\right)\\ &\le \left(\frac{\deg_{V\setminus U}(v)}{\deg(v)}\right)^{\Delta}\\ &\le \left(\frac{\deg_{V_M\setminus U}(v)+\beta}{\deg_{V_M}(v)+\beta}\right)^{\Delta}, \qquad (1)\end{aligned}$$

where $\deg(v)$ denotes the degree of $v$ in $G$, and for any $W \subseteq V$, $\deg_W(v)$ denotes the degree of $v$ in the subgraph of $G$ induced by $W$.

In what follows we restrict our attention to the subgraph $G[V_M]$ of $G$ induced by $V_M$ (ignoring vertices of $V_F$ and edges incident on them). We apply the following definition from [8] to $G[V_M]$.

Definition 2.8. *Fix $0 < \delta < 1$. We say that $v \in U$ is a $\delta$-good vertex if its degree in $V_M$ is at most $1/\delta$ or its degree in $U$ is at least a $\delta$ fraction of its degree in $V_M \setminus U$, namely $\deg_U(v) \ge \delta\deg_{V_M\setminus U}(v)$.*

Lemma 7 of [8] implies that, if $\delta = \frac{|U|}{4\kappa\beta}$, at least half of the vertices in $U$ are $\delta$-good, where $\kappa = |V_M|$. Denote the set of good vertices in $U$ by $U_G$. Observe that $U_G$ is determined by $U$ and by the partition of $V$ into $V_F$ and $V_M$, hence it is fixed prior to the construction of the random sparsifier $G_\Delta$. Also, the random choices made due to any vertex $v$ are independent of random choices made due to other vertices. We summarize this in the following observation.

Observation 2.9. *For any good vertex $v$, we have $\mathbf{P}(\mathcal{E}_v^{(U)}) \le \left(\frac{\deg_{V_M\setminus U}(v)+\beta}{\deg_{V_M}(v)+\beta}\right)^{\Delta}$, and this bound holds independently of random choices made due to other vertices.*

Since $\Delta \ge \beta + 8\beta/\epsilon$, every vertex in $U$ has a degree (in $V_M$) at least $8\beta/\epsilon$, which, in turn, exceeds $1/\delta$, and therefore every good vertex $v$ satisfies $\deg_U(v) \ge \delta\deg_{V_M\setminus U}(v)$. We employ this lower bound on $\deg_U(v)$ to strengthen the upper bound on $\mathbf{P}(\mathcal{E}_v^{(U)})$ provided by Observation 2.9 for an arbitrary good vertex $v$ as follows:

$$\begin{aligned}\mathbf{P}(\mathcal{E}_v^{(U)}) &\le \left(\frac{\deg_{V_M\setminus U}(v)+\beta}{\deg_{V_M}(v)+\beta}\right)^{\Delta} \le \left(\frac{\frac{1}{1+\delta}\deg_{V_M}(v)+\beta}{\deg_{V_M}(v)+\beta}\right)^{\Delta}\\ &\le \left(1-\frac{\frac{\epsilon}{9\beta}\deg_{V_M}(v)}{\deg_{V_M}(v)+\beta}\right)^{\Delta} \le \left(1-\frac{\epsilon}{10\beta}\right)^{\Delta}, \qquad (2)\end{aligned}$$

where the second inequality holds since $v$ is a good vertex and the third inequality holds by the choice of $\delta$. Write $U_G = \{v_1,\ldots,v_g\}$, where $g = |U_G| \ge |U|/2$ is the number of good vertices. Recall that $\mathcal{E}^{(U)}$ denotes the event that $U$ is an independent set in $G_\Delta$, Observation 2.9 and Equation (2) yield

$$\begin{aligned}\mathbf{P}(\mathcal{E}^{(U)}) &\le \mathbf{P}(\mathcal{E}_{v_1}^{(U)}\cap\mathcal{E}_{v_2}^{(U)}\cap\ldots\cap\mathcal{E}_{v_g}^{(U)})\\ &= \mathbf{P}(\mathcal{E}_{v_1}^{(U)} \mid \mathcal{E}_{v_2}^{(U)}\cap\ldots\cap\mathcal{E}_{v_g}^{(U)})\\ &\quad\cdot\mathbf{P}(\mathcal{E}_{v_2}^{(U)} \mid \mathcal{E}_{v_3}^{(U)}\cap\ldots\cap\mathcal{E}_{v_g}^{(U)})\\ &\quad\cdots\\ &\quad\cdot\mathbf{P}(\mathcal{E}_{v_g}^{(U)})\\ &= \mathbf{P}(\mathcal{E}_{v_1}^{(U)})\cdot\mathbf{P}(\mathcal{E}_{v_2}^{(U)})\cdot\ldots\cdot\mathbf{P}(\mathcal{E}_{v_g}^{(U)})\\ &\le \left(\left(1-\frac{\epsilon}{10\beta}\right)^{\Delta}\right)^{g} \le \left(1-\frac{\epsilon}{10\beta}\right)^{\frac{\Delta|U|}{2}},\end{aligned}$$

which is upper bounded by $(\epsilon/24)^{|U|}$ for any $\Delta \ge 20\frac{\beta}{\epsilon}\ln\frac{24}{\epsilon} = O\left(\frac{\beta}{\epsilon}\log\frac{1}{\epsilon}\right)$. Claim 2.7 follows. ∎

To complete the proof of Lemma 2.6, denote by $\mathcal{E}$ the event that $|W_F| \ge \frac{\epsilon}{2}|V_M|$ and let $\mathcal{U}$ be the collection of all subsets of $V_M$ that satisfy the conditions of Claim 2.7 (namely, of size at least $\frac{\epsilon}{2}|V_M|$ and such that all vertices are of high degree). By Corollary 2.5, it holds that $\mathbf{P}(\mathcal{E}) \le \mathbf{P}(\bigcup_{U\in\mathcal{U}}\mathcal{E}^{(U)})$. By the union bound,

$$\mathbf{P}(\mathcal{E}) \le \sum_{U\in\mathcal{U}}\mathbf{P}(\mathcal{E}^{(U)}) \le \sum_{i=\frac{\epsilon}{2}\kappa}^{\kappa}\binom{\kappa}{i}\cdot\left(\frac{\epsilon}{24}\right)^{i}. \qquad (3)$$

(Recall that $\kappa = |V_M|$.) The ratio of successive terms in the sum of the right-hand side of Equation (3) is upper bounded by

$$\frac{\binom{\kappa}{i+1}\left(\frac{\epsilon}{24}\right)^{i+1}}{\binom{\kappa}{i}\left(\frac{\epsilon}{24}\right)^{i}} = \left(\frac{\kappa-i}{i+1}\right)\left(\frac{\epsilon}{24}\right) \le \left(\frac{\kappa-\frac{\epsilon}{2}\kappa}{\frac{\epsilon}{2}\kappa+1}\right)\left(\frac{\epsilon}{24}\right) \le \frac{1}{12},$$

where the first inequality holds since, in the range $\frac{\epsilon}{2}\kappa \le i \le \kappa$, the term $\frac{\kappa-i}{i+1}$ is maximized when $i = \frac{\epsilon}{2}\kappa$. Hence the sum of the right-hand side of Equation (3) is dominated by a geometric sum with common ratio of $1/12$ and $q_0 = \binom{\kappa}{\frac{\epsilon}{2}\kappa}\left(\frac{\epsilon}{24}\right)^{\frac{\epsilon}{2}\kappa}$ being the first term, which can naively be upper bounded by $2q_0$. It follows that

$$\begin{aligned}\sum_{i=\frac{\epsilon}{2}\kappa}^{\kappa}\binom{\kappa}{i}\cdot\left(\frac{\epsilon}{24}\right)^{i} &\le 2\binom{\kappa}{\frac{\epsilon}{2}\kappa}\left(\frac{\epsilon}{24}\right)^{\frac{\epsilon}{2}\kappa}\\ &\le 2\left(\frac{1}{2}\right)^{\frac{\epsilon}{2}\frac{2n}{\beta+2}} \quad (\kappa = 2|M| \ge \frac{2n}{\beta+2} \text{ by Lemma 2.2})\\ &\le \frac{1}{n^c} \text{ (holds for } \beta = O\left(\frac{\epsilon n}{\log n}\right)) \qquad (4)\end{aligned}$$

The constant $c$ can be made arbitrarily large by decreasing the constant hiding under the $O$-notation in the definition of $\beta$.

Plugging Equation (4) into Equation (3), we obtain $\mathbf{P}(\mathcal{E}) \le \frac{1}{n^c}$, where $\mathcal{E}$ is the event that $|W_F| \ge \frac{\epsilon}{2}|V_M|$. We conclude that $|W_F| < \frac{\epsilon}{2}|V_M|$ with high probability, thus proving Lemma 2.6. ∎

**Completing the proof of Theorem 2.1.** Lemma 2.6 implies

$$|M_\Delta| = \frac{1}{2}\left(|V_M| - |W_F|\right) > \frac{1}{2}\left(|V_M| - \frac{\epsilon}{2}|V_M|\right) = \left(1-\frac{\epsilon}{2}\right)|M|,$$

which yields $M_\Delta(1+\epsilon) > |M|$. In other words, we have shown that $G_\Delta$ is a $(1+\epsilon)$-sparsifier for $G$, when $\Delta = \Theta\left(\frac{\beta}{\epsilon}\log\frac{1}{\epsilon}\right)$, for any $0 < \epsilon < 1$ and $\beta = O\left(\frac{\epsilon n}{\log n}\right)$, as required.

## 2.2 Additional properties of $G_\Delta$

In this section we present a few observations regarding the sparsifier $G_\Delta$, some of which will play a major role in the applications of Section 3.

*2.2.1 $G_\Delta$ is sparse.* Clearly, the sparsifier $G_\Delta$ contains at most $n \cdot \Delta$ edges. This size bound will be used for achieving a sublinear-time algorithm in the centralized sequential setting (Section 3.1) and a distributed algorithm that uses a sublinear number of messages (Section 3.2.1).

The following observation shows that, roughly speaking, one can replace $n$ in this naïve size bound by the MCM size. This sharper size bound will be crucial for the performance of our dynamic algorithm (Section 3.3).

Observation 2.10. *$G_\Delta$ contains at most $2|\mathtt{MCM}(G)| \cdot (\Delta+\beta)$ edges.*

**Proof:** Let $V_M$ and $V_F$ be the sets of matched and free vertices with respect to any MCM (or even a maximal matching) $M$. Since $V_F$ is an independent set, all edges touching $V_F$ must lead to $V_M$. Since the neighborhood independence number is $\beta$, each vertex in $V_M$ has at most $\beta$ neighbors in $V_F$, so the number of edges touching $V_F$ in the graph, and thus in the sparsifier, is at most $|V_M| \cdot \beta$. The remaining edges in the sparsifier have both endpoints in $V_M$, so their number is trivially bounded by $V_M \cdot \Delta$. Overall, the sparsifier contains at most $|V_M| \cdot \beta + |V_M| \cdot \Delta \leq 2|\mathtt{MCM}(G)| \cdot (\Delta+\beta)$ edges. ∎

**Remark.** Since $\Delta$ is larger than $\beta$ (see the statement of Theorem 2.1), the size upper bound provided by Observation 2.10 does not exceed $4|MCM(G)| \cdot \Delta$. For super-constant values of $\beta$, this size bound could be significantly smaller than the naïve bound of $n \cdot \Delta$.

*2.2.2 $G_\Delta$ is uniformly sparse.* Observation 2.10 shows that $G_\Delta$ is sparse. We next observe that it is also *uniformly sparse*, which will be crucial for achieving a fast distributed algorithm (Section 3.2).

Definition 2.11. *The* arboricity *of an undirected graph $G = (V, E)$ is defined as $\alpha(G) = max_{U \subseteq V}\left\lceil \frac{|E(U)|}{|U|-1} \right\rceil$, where $E(U)$ is the set of edges induced by $U$ (which we assume has size $|U| \geq 2$).*

Observation 2.12. *The arboricity of $G_\Delta$ is at most $2\Delta$.*

The proof is straightforward, so we will omit it.

*2.2.3 $G_\Delta$ is not deterministic or exact for a reason.*

We first argue that randomization is required for our specific construction of sparsifiers, $G_\Delta$, where every vertex marks $\Delta$ neighboring edges and the sparsifier consists of all the marked edges. The argument is a standard one and follows similar lines as those used in the proof of Theorem 11 in [8]; it is provided here for completeness.

Lemma 2.13. *Fix $n$ and $\Delta$ such that $\Delta < n/2$. Consider any deterministic algorithm that constructs a sparsifier $G_\Delta$ in every graph $n$-vertex graph $G$ with $\beta(G) = 2$, where the algorithm queries up to $\Delta$ entries of the adjacency array of each vertex and includes in $G_\Delta$ up to $\Delta$ adjacent edges for each vertex (possibly different than the queried ones). Then the approximation ratio of $G_\Delta$ (for some graphs $G$ with $\beta(G) = 2$) is no better than $\frac{n}{2\Delta}$.*

**Remark.** When $\Delta \geq n/2$, an approximation no better than $\frac{n}{2\Delta}$ vacuously holds, hence the restriction in the lemma statement.

**Proof:** Consider the family of graphs $\mathcal{G}_n$, for an even $n \in \mathbb{N}$, obtained by removing a single edge from a clique $K_n$ on $n$ vertices. For a graph $G \in \mathcal{G}_n$, we denote the single edge not in $G$ by $\bar{e}(G)$, and refer to it as the non-edge of $G$. Clearly $\beta(G) = 2$ for every $G \in \mathcal{G}_n$ and $G$ contains a (perfect) matching of size $n/2$.

Let $\mathcal{A}$ be a deterministic algorithm for constructing sparsifier $G_\Delta$ on every graph $G \in \mathcal{G}_n$. We consider a game between $\mathcal{A}$ and an adaptive adversary that answers the probes of $\mathcal{A}$ to the adjacency array of the input. Each time $\mathcal{A}$ probes a new entry of the adjacency array for some vertex $u \in V$, the adversary outputs a vertex $v \in V$ in the neighborhood of $u$ that has not been so far returned as an answer to a query on $u$.

The adversary picks an arbitrary set $D$ of $\Delta$ vertices at the outset. On the other hand, the non-edge of the graph will be chosen adaptively by the adversary. We may assume that the algorithm $\mathcal{A}$ knows the number of vertices $n$ in the graph, the parameter $\Delta$ and the vertex set $D$; the only missing information to $\mathcal{A}$ is the identity of the non-edge.

When a vertex $u \in V \setminus D$ is queried by $\mathcal{A}$, the adversary returns an arbitrary vertex from $D$ not returned thus far as an answer to a query on $u$; since at most $\Delta$ queries on $u$ are performed and as $|D| = \Delta$, this strategy is well-defined. When a vertex $u \in D$ is queried, the adversary returns an arbitrary vertex in $V \setminus u$ not returned thus far as an answer to a query on $u$; this too is well-defined, as $n \geq \Delta + 1$ is implied by the condition $\Delta < n/2$ in the lemma statement. Note that any edge returned by the adversary is incident on $D$ with at least one endpoint. Suppose that the algorithm $\mathcal{A}$ returns a sparsifier $G_\Delta$ of approximation ratio better than $\frac{n}{2\Delta}$. By definition, this means that $G_\Delta$ contains a matching of size bigger than $\Delta$, and as such, it must include at least one edge $e$ with both endpoints outside $D$. But the graph $G \in \mathcal{G}_n$ whose non-edge is $e$ (i.e., $\bar{e}(G) = e$) is consistent with all the edges returned by the adversary but it does not include edge $e$, hence $e$ does not belong to $G_\Delta$. In other words, there always exists at least one graph in $\mathcal{G}_n$ for which the output of $\mathcal{A}$ is not feasible, completing the proof. ∎

The lower bound provided by Lemma 2.13 applies only to our construction $G_\Delta$ of sparsifiers. Nonetheless, one can extend the lower bound argument of [8] (see Section 5 therein) to obtain a similar hardness result that applies to any deterministic construction of sparsifiers. Since the focus of our work is on the upper bounds side and as this extension is rather straightforward yet a bit tedious, we omit the details for the sake of conciseness.

Next, allowing randomization, we claim that our construction of sparsifiers $G_\Delta$ cannot preserve the exact size of the MCM with reasonable probability, unless $\Delta$ is close to $n$.

Observation 2.14. *There exist (infinitely many) $n$-vertex graph instances $G$ such that if the randomized sparsifier construction $G_\Delta$ preserves the exact MCM size with probability $p$, then $\Delta$ needs to be as large as $\Omega(p \cdot n)$. Moreover, to achieve a high success probability (more precisely, a success probability of at least $1 - \frac{1}{n^c}$ for an arbitrarily*

*large constant $c > 1$), the sparsifier $G_\Delta$ needs to coincide with the entire graph $G$.*

**Proof:** Consider a graph $G$ that consists of two complete graphs $A := K_{n/2}$ and $B := K_{n/2}$ that span an odd number of vertices each (i.e., $n/2$ is an odd integer) and a single edge that connects a vertex $a$ of $A$ with a vertex $b$ of $B$. Observe that the MCM size in $G$ is $n/2$ and that any MCM must contain edge $(a, b)$, i.e., any matching for $G$ that does not contain edge $(a, b)$ is not of maximum size. Note further that edge $(a, b)$ can be marked only by $a$ and by $b$, thus assuming each vertex marks $\Delta$ neighboring edges uniformly at random, the probability of not marking edge $(a, b)$ is:

$$\left(\frac{\binom{n/2-1}{\Delta}}{\binom{n/2}{\Delta}}\right)^2 = \left(1 - \frac{2\Delta}{n}\right)^2. \tag{5}$$

It follows that the probability of marking edge $e$ is $1 - (1 - \frac{2\Delta}{n})^2$, which is at most $\frac{4\Delta}{n}$. In other words, for the sparsifier to include this edge with probability $p$, $\Delta$ should be set to $\Omega(p \cdot n)$. Moreover, to achieve a high probability (a success probability of at least $1 - \frac{1}{n^c}$ for an arbitrarily large constant $c > 1$), we must take $\Delta$ to be $n/2$, in which case the sparsifier $G_\Delta$ is the entire graph $G$. ∎

## 3 APPLICATIONS

In this section we demonstrate the applicability of our matching sparsifier construction, $G_\Delta$, in several different settings. In addition to these concrete applications, our sparsification algorithm can be used more broadly in computational models where there are local or global memory constraints, such as the massively parallel computation (MPC) model (which is an abstraction of MapReduce-style frameworks, cf. [4, 31]), the streaming model of computation (cf. [3]), and the dynamic distributed model (where some graph structure has to be maintained in a dynamically changing distributed network using low local memory at processors, cf. [7, 27, 56, 75]).

### 3.1 Centralized Sequential Algorithms

Our first application is in the classical centralized sequential setting. Assume for now that our matching sparsifier $G_\Delta$ can be computed within time linear in its size, namely, $O(n \cdot \frac{\beta}{\epsilon} \log \frac{1}{\epsilon})$. We can then run the standard $(1+\epsilon)$-approximate MCM algorithm of [52, 70], which takes $O(m/\epsilon)$ time for any $m$-edge graph, leading to a total runtime of $O(n \cdot \frac{\beta}{\epsilon^2} \log \frac{1}{\epsilon})$, which is sublinear in the graph size for almost the entire regime of parameter $\beta$. Moreover, by Observation 2.10, we actually get a sharper runtime bound of $O(|\mathsf{MCM}(G)| \cdot \frac{\beta}{\epsilon^2} \log \frac{1}{\epsilon})$.

It remains to justify the assumption that $G_\Delta$ can be computed within time linear in its size (which, in turn, is sublinear in general in the size of the original graph $G$). It is straightforward to achieve this runtime with high probability, as will be shown next, but our goal is to achieve this runtime bound *deterministically*; this is possible since the bounds on the size and arboricity of $G_\Delta$ are deterministic (the only use of randomization is for achieving the bound $1 + \epsilon$ on the approximation factor, which holds with high probability). Before proceeding further, we stress that it is crucial to specify the exact data model when considering sublinear-time algorithms, as such algorithms cannot even read the entire input. We shall assume that the input graph is given in the *adjacency array representation*, which means that for each vertex $v \in V$, we are given the degree $\deg(v)$ of $v$, followed by an array of length $\deg(v)$ containing all neighbors of $v$ in an arbitrary order. This way, we can determine the degree of any vertex $v$ or its $i$-th neighbor for $i \in [\deg(v)]$ in $O(1)$ time. Typically the algorithm has read-only access to the adjacency arrays of vertices, i.e., one cannot write to these arrays. We also make the common assumption that a random number from 1 to $n$ can be generated in $O(1)$ time. This is a standard input representation for graph problems and is commonly used in the area of sublinear-time algorithms (see, e.g. [8, 41, 42, 72]).

The only nontrivial issue in constructing $G_\Delta$ within time linear in its size arises in the implementation of the random edge samplings. Recall that for each vertex $v$, we mark $\Delta$ neighboring edges uniformly at random (without repetitions), and the sparsifier includes all the marked edges. Consider an arbitrary vertex $v$. A straightforward randomized approach would be to generate a random number from 1 to $\deg(v)$, say $i$, mark the edge corresponding to the $i$-th neighbor of $v$ if unmarked, and repeat until $\min\{\Delta, \deg(v)/2\}$ edges incident to $v$ have been marked. There is a small technical issue with vertices of degree lower than $2\Delta$, so for this to work, we tweak the construction of $G_\Delta$ so that we will mark all the neighbors of any vertex of degree at most $2\Delta$ (rather than $\Delta$ as before); this tweak will increase the size and arboricity bounds of the sparsifier by at most a factor of 2, which is fine. It is now readily verified that this sampling method leads to $O(\Delta)$ time in expectation for each vertex $v$, and we obtain the required runtime bound of $O(n \cdot \Delta) = O(n \cdot \frac{\beta}{\epsilon} \log \frac{1}{\epsilon})$ with high probability by applying a standard Chernoff bound.

Instead of achieving a probabilistic bound on the runtime that holds with high probability, we next show that the same runtime bound can be achieved deterministically by designing a more careful edge sampling method. In other words, we shall obtain an algorithm with deterministic runtime bound and probabilistic bound on the approximation guarantee.

The naïve sampling method that leads to $O(\Delta)$ deterministic time per vertex $v$ is to sample a random integer $i$ between 1 and $\deg(v)$, and then swap the $i$-th element of the adjacency array with the element at position $\deg(v)$. (In this discussion, we may identify the $i$-th neighbor of $v$ with the corresponding edge incident on $v$, when this should not lead to a misunderstanding.) In the next step, we sample an integer between 1 and $\deg(v) - 1$, replace it with the element at position $\deg(v) - 1$, and so on, until we have sampled $\min\{\Delta, \deg(v)\}$ elements. At the end, the last $\min\{\Delta, \deg(v)\}$ members of the adjacency array correspond to the marked edges incident to $v$ (and will be included in the sparsifier). The problem with this approach is that it requires the algorithm to write to the adjacency arrays, while, as mentioned above, sublinear algorithms typically have read-only access to these arrays.

To overcome this problem, the first idea that comes to mind is to create a copy of the adjacency array of each vertex and work on the copy array instead of the read-only array. Unfortunately, the runtime required for creating copies of all the adjacency arrays is linear in the input size. (Recall that our goal is to work within time linear in the size of the sparsifier $G_\Delta$, but not in time linear in the size of the entire input graph $G$.) We resolve this issue by introducing an additional array for each vertex $v$ of the graph,

denoted by $pos_v$, which will represent the *positions* of the vertices in the adjacency array of $v$. The key insight here is that, instead of copying the entire content of an adjacency array to a new array, the new array $pos_v$ that we introduce can be initialized in constant time *uniformly*, i.e., to the same initial value. Specifically, we can choose to initialize $pos_v$ with zeroes, meaning that each element is at its original position. This is possible using the so-called "sparse array" data structure, which supports all the usual operations of array and, on top of that, supports initialization to a single arbitrary value in constant time; see [2] for further details on this data structure.

Using the $pos_v$ array, we will emulate the naïve sampling method described above without ever writing to the adjacency array of $v$. Specifically, instead of actually moving the elements of the adjacency array of $v$, we will "implicitly" move them by changing the entries of $pos_v$, so that the $i$-th entry of $pos_v$ (denoted by $pos_v[i]$) represents the actual position of the $i$-th element in the adjacency array of $v$ (i.e., its position as if we have made the actual moves). A zero entry means that the element has not been moved yet. More formally, we will maintain the following invariant: $pos_v[i]$ is zero iff the element has not been moved yet and otherwise it represents the actual position of the $i$-th element in the adjacency array of $v$. As before, we will sample a random integer $i_1$ between 1 and $\deg(v)$, and emulate the changes to the adjacency array by making the following modifications to the $pos_v$ array. We change $pos_v[i_1]$ to $i_1$ and $pos_v[\deg(v)]$ to $\deg(v)$. Next we simply replace $pos_v[i_1]$ with $pos_v[\deg(v)]$, thereby emulating the consequences of the actual swap. It is easy to see that by doing that, the aforementioned invariant will hold. In the next step, we generate a random number $i_2$ between 1 and $\deg(v) - 1$. If $pos_v[i_2]$ is zero, we set it to $i_2$ and if $pos_v[\deg(v) - 1]$ is zero we set it to $\deg(v) - 1$ and finally swap $pos_v[i_2]$ with $pos_v[\deg(v) - 1]$. And so forth. In this way the invariant always holds and after $\min\{\Delta, \deg(v)\}$ sampling steps, the last $\min\{\Delta, \deg(v)\}$ entries of the array $pos_v$ represent the actual positions of the elements in the adjacency array after the (implicit) changes have been performed. Each of the above operations takes constant time, hence we can perform the edge sampling for a single vertex in $O(\Delta)$ time, and the entire sparsifier can thus be built in $O(n \cdot \Delta) = O\left(n \cdot \frac{\beta}{\epsilon} \log \frac{1}{\epsilon}\right)$ time, as required.

Summarizing, we have proved the following theorem.

Theorem 3.1. *A $(1+\epsilon)$-approximate matching can be computed in the centralized sequential setting in $O(n \cdot \frac{\beta}{\epsilon^2} \log \frac{1}{\epsilon})$ deterministic time, for any $0 < \epsilon < 1$ and $\beta = O\left(\frac{\epsilon n}{\log n}\right)$, where the bound on the approximation factor holds with high probability. Moreover, the sharper runtime bound of $O(|\mathtt{MCM}(G)| \cdot \frac{\beta}{\epsilon^2} \log \frac{1}{\epsilon})$ holds as well.*

**Remark.** Assadi and Solomon [8] presented a randomized algorithm for computing a maximal matching (and thus a 2-approximate MCM) whose runtime is bounded by $O(n \log n \cdot \beta)$ with high probability. Our algorithm achieves an approximation factor of $1 + \epsilon$, and its runtime shaves a factor of $\log n$ from the runtime of [8] in the entire regime of parameter $\beta = O\left(\frac{n}{\log n}\right)$, and for any constant $\epsilon$. As mentioned, the complementary regime of $\beta = \omega\left(\frac{n}{\log n}\right)$ is irrelevant in terms of the improvement over [8], since in that regime the runtime of [8] is inferior to the naïve $O(n^2)$ runtime of the greedy maximal matching algorithm. Finally, our runtime is *tight* for all $\beta = O\left(\frac{n}{\log n}\right)$ (for constant $\epsilon$), matching the lower bound of $\Omega(n \cdot \beta)$ due to [5, 8]. Consequently, we answer Question 1.1 in the affirmative, and furthermore provide an optimal (up to the $\epsilon$-dependence) resolution in the regime $\beta = O\left(\frac{n}{\log n}\right)$.

## 3.2 Distributed Algorithms

We focus on the $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models of communication (cf. [76]), which are two standard models in distributed computing that capture the essence of spatial locality and congestion. In these models, all the processors wake up simultaneously and computation proceeds in fault-free synchronous rounds in which every processor exchanges messages of either unbounded size (in the $\mathcal{LOCAL}$ model) or of $O(\log n)$-bit size (in the $\mathcal{CONGEST}$ model). Furthermore, since our sparsifier relies on a simple sampling primitive that does not require the knowledge of processors' identifiers, it can be constructed in the $KT_0$ model.

It is immediate that our matching sparsifier $G_\Delta$ can be implemented in distributed networks using a single communication round in the $\mathcal{LOCAL}$ model. In particular, if messages are transmitted in a *broadcast* manner, i.e., any piece of information sent from a processor reaches *all* its neighbors (rather than a subset of its neighbors), then a single round of communication will naively require messages of size $O(\log n \cdot \Delta)$ (where $\Delta$ is the number of random neighboring edges "marked" locally by a processor, and it grows linearly with $\beta$). Many distributed message-passing systems, however, support *unicast* or *multicast* transmissions, meaning that any processor can send a message to a subset of its neighbors. In such systems, which we shall refer to as *unicast communication systems*, our sparsifier can be implemented in a single round using 1-bit messages in a straightforward way. Indeed, in a single round each processor locally marks randomly $\Delta$ of its neighboring edges, and can then send a 1-bit message only along its "marked" edges.

From that stage onwards, one can work on the sparsifier $G_\Delta$ rather than the original graph $G$. It is desirable in general, and for our purposes in particular (as demonstrated below), to work with bounded degree graphs. Alas, we cannot upper bound the maximum degree of $G_\Delta$ by $\Delta$, and a-priori there may exist graph instances $G$ for which the degree of $G_\Delta$ is much greater than $\Delta$. In fact, the question of upper bounding the maximum degree of $G_\Delta$ in general graphs, as a function of $n$, $\beta$ and $\epsilon$, is likely to coincide with deep mathematical questions regarding the distribution of the maximum degree in special types of random graphs. Therefore, instead of trying to upper bound the maximum degree of $G_\Delta$, we resort to the more robust sparsity parameter of *arboricity*, which roughly speaking measures how uniformly sparse the graph is. As we have shown in Observation 2.12, the arboricity of our sparsifier is upper-bounded by $2\Delta$.

In ITCS'18, Solomon [81] gave a construction of $(1+\epsilon)$-sparsifiers with bounded degree in graphs of bounded arboricity. Specifically, given a graph with arboricity bounded by $\alpha$, one can construct a $(1+\epsilon)$-matching sparsifier for it with maximum degree bounded by $\Delta_\alpha = \Theta(\alpha/\epsilon)$, as follows. For each vertex we mark $\Delta_\alpha = \Theta(\alpha/\epsilon)$ arbitrary neighboring edges, and then take to the sparsifier only edges that are marked by both endpoints. As with our sparsifier,

the sparsifier of [81] can be constructed in a single communication round. Clearly, the maximum degree of that sparsifier is at most $\Delta_\alpha = \Theta(\alpha/\epsilon)$. The sparsifier of [81] is different than ours in two aspects. First, it is deterministic, so marking any set of $\Delta_\alpha$ arbitrary neighboring edges per vertex will do the job in bounded arboricity graphs, whereas, as shown by Lemma 2.13, in bounded neighborhood independence graphs randomization is essential. Second, the sparsifier of [81] includes only edges that are marked by both endpoints, which leads to a degree upper bound of $\Delta_\alpha$, but the same trick fails in bounded neighborhood independence graphs.

By composing the two sparsifiers, we obtain a clean and simple reduction for the problem of distributed $(1+\epsilon)$-approximate MCM from graphs of bounded neighborhood independence to bounded degree graphs, which requires only two communication rounds. Concretely, we construct a matching sparsifier of bounded degree for a graph $G$ as follows: In the first round we construct our $(1+\epsilon)$-matching sparsifier $G_\Delta$ with arboricity at most $2\Delta = O(\frac{\beta}{\epsilon}\log\frac{1}{\epsilon})$, and in the second round we construct the $(1+\epsilon)$-matching sparsifier of [81] on top of ours. The resulting sparsifier, denoted by $\tilde{G}_\Delta$, achieves approximation factor at most $(1+\epsilon)(1+\epsilon) \leq (1+3\epsilon)$ (assuming $\epsilon < 1$) and a maximum degree of $O(2\Delta/\epsilon) = O(\frac{\beta}{\epsilon^2}\log\frac{1}{\epsilon})$, and as before we can reduce the approximation factor to $1+\epsilon$ by a scaling argument.

Equipped with the bounded degree matching-sparsifier $\tilde{G}_\Delta$ for the original (of possibly huge degree) graph $G$, we can apply a distributed algorithm that runs efficiently for bounded degree graphs. Specifically, Even et al. [34] presented a distributed algorithm for computing a $(1+\epsilon)$-approximate MCM in $\Delta^{O(1/\epsilon)} + O(\frac{1}{\epsilon^2})\cdot\log^* n$ rounds, for graphs of maximum degree $\Delta$. Running the algorithm of Even et al. on top of the sparsifier $\tilde{G}_\Delta$ yields a $(1+O(\epsilon))$-approximate MCM in $\left(\frac{\beta}{\epsilon}\right)^{O(1/\epsilon)} + O\left(\frac{1}{\epsilon^2}\right)\cdot\log^* n$ communication rounds; yet again, the approximation factor can be reduced to $1+\epsilon$ by scaling.

Summarizing, we have proved the following theorem.

Theorem 3.2. *There is a randomized distributed algorithm for computing a $(1+\epsilon)$-approximate matching in $\left(\frac{\beta}{\epsilon}\right)^{O(1/\epsilon)} + O\left(\frac{1}{\epsilon^2}\right)\cdot\log^* n$ rounds, for any $0<\epsilon<1$ and $\beta = O\left(\frac{\epsilon n}{\log n}\right)$, where the bound on the approximation factor holds with high probability.*

**Remark.** When $\beta$ and $\epsilon$ are constants, Theorem 3.2 gives rise to $O(\log^* n)$ communication rounds, which provides an improvement over the (deterministic) algorithm of Barenboim and Oren [16, 17] that requires the same number of rounds but achieves a $(2+\epsilon)$-approximation. (Barenboim and Oren do not analyze the performance of their algorithm [16, 17] for super-constant $\beta$ and $\epsilon$.)

*3.2.1 Sublinear Communication.* The *round complexity* and the *message complexity* are perhaps the two most basic quality measures of distributed algorithms, where the former is a measure of "runtime" and the latter can be viewed as a measure of "total work". The message complexity is almost always assumed to be at least linear in the graph size (i.e., $\Omega(m)$, where $m$ is the number of edges in the graph), since for almost any nontrivial distributed graph problem, every processor needs to send and receive at least one message along each of its incident edges. If all messages are transmitted in a broadcast manner, then a sublinear message complexity cannot be achieved unless sufficiently many processors do not participate in the distributed algorithm (but that rules out essentially any nontrivial distributed problem). We shall therefore restrict our attention to unicast communication systems; in such systems there is hope to design algorithms with a sublinear message complexity, yet very few examples of such algorithms are known (see [9, 40, 60, 63, 68, 69, 74], and the references therein).

Our work adds a new example to this very small pool, showing that one can solve the problem of distributed $(1+\epsilon)$-approximate MCM using a sublinear (depending, of course, on the neighborhood independence number $\beta$) message complexity. As mentioned already, constructing the random matching sparsifier $G_\Delta$ in unicast communication systems can be done in a single round, where each processor locally marks randomly $\Delta$ of its neighboring edges, and then sends a 1-bit message only along its marked edges. Clearly, the total number of messages sent is at most twice the size of the resulting sparsifier $G_\Delta$, namely, $O(n\cdot\frac{\beta}{\epsilon}\log\frac{1}{\epsilon})$. We can then run any approximate MCM algorithm *as a black-box* on top of our sparsifier to achieve a low (depending on $\beta$) message complexity. Specifically, if the round complexity of the black-box matching algorithm is $T(n)$, then running it on top of our sparsifier will give rise to an algorithm whose message complexity is at most $T(n)\cdot O(n\cdot\frac{\beta}{\epsilon}\log\frac{1}{\epsilon})$.

Summarizing, we have proved the following result.

Theorem 3.3. *Suppose there is a distributed algorithm for computing a $\gamma$-approximate MCM in $T(n)$ rounds in general $n$-vertex graphs, for any parameter $\gamma \geq 1$ and "runtime function" $T$. Then there is also a distributed algorithm for computing a $(1+\epsilon)\gamma$-approximate MCM in $T(n)+1$ rounds and using $T(n)\cdot O(n\cdot\frac{\beta}{\epsilon}\log\frac{1}{\epsilon})$ messages, for any $0<\epsilon<1$ and $\beta = O\left(\frac{\epsilon n}{\log n}\right)$, provided that the distributed system supports unicast or multicast transmissions. The bound on the approximation factor holds with high probability.*

Applying Theorem 3.3 in conjunction with Theorem 3.2 yields a distributed algorithm for computing a $(1+\epsilon)$-approximate MCM in $\left(\frac{\beta}{\epsilon}\right)^{O(1/\epsilon)} + O\left(\frac{1}{\epsilon^2}\right)\cdot\log^* n$ rounds and using

$$n\cdot\left(\left(\frac{\beta}{\epsilon}\right)^{O(1/\epsilon)} + O\left(\frac{\beta}{\epsilon^3}\log\frac{1}{\epsilon}\right)\cdot\log^* n\right)$$

messages. When both $\beta$ and $\epsilon$ are constants, the number of rounds and messages is reduced to $O(\log^* n)$ and $O(n\log^* n)$, respectively.

## 3.3 Dynamic Graph Algorithms

In this section we show that a $(1+\epsilon)$-approximate MCM can be maintained in the standard fully dynamic setting with a worst-case update time of $O\left(\Delta/\epsilon^2\right) = O(\frac{\beta}{\epsilon^3}\log\frac{1}{\epsilon})$. As mentioned in Section 1.2, our update time bound holds *deterministically* and the approximation factor of $(1+\epsilon)$ holds with high probability against an *adaptive adversary*. This is a rare example of a randomized algorithm that does not make the *oblivious adversary* assumption.

If we were to make the oblivious adversary assumption, life would be much simpler. In particular, in this case it is straightforward to maintain the sparsifier $G_\Delta$ with a worst-case update time of $O(\Delta)$. Indeed, in such a case, following every edge update $(u,v)$,

we can simply remove from the sparsifier the (at most) $2\Delta$ edges marked due to $u$ and due to $v$, and then add in their place new randomly sampled edges, where we mark $\Delta$ random neighboring edges due to $u$ and then another $\Delta$ random neighboring edges due to $v$, and add the newly marked edges to $G_\Delta$ in place of those that got removed. As for the analysis, marking $\Delta$ random neighboring edges for a vertex can be done in time $O(\Delta)$ deterministically, e.g., by employing the argument of Section 3.1 (for the centralized sequential setting), hence the worst-case update time of the algorithm is trivially bounded by $O(\Delta)$. As for the approximation factor, since the adversary is oblivious to the random bits used by the algorithm, the proof of Theorem 2.1 remains valid. Now that we are equipped with the dynamic sparsifier, and recalling (from Observation 2.12) that its arboricity is at most $2\Delta$, we can simply run on it the dynamic $(1+\epsilon)$-approximate MCM algorithm due to [77], to get a worst-case update time of $O(\Delta/\epsilon^2)$, as required.

Our goal, however, is to cope with an adaptive adversary. The suggestion above would fail in this case, since the probabilistic argument used in the proof of Theorem 2.1 for bounding the approximation factor of $G_\Delta$ makes critical use of the fact that the random neighboring edges marked for any vertex are completely independent of random choices made due to other vertices. To overcome this challenge, we take a completely different approach, which is based on a scheme for dynamic approximate matchings due to [44].

The scheme of [44] exploits a basic *stability* property of matchings: the MCM size changes by at most 1 following each update step. Thus if we have a $\gamma$-approximate MCM, for any parameter $\gamma \geq 1$, the approximation factor of the matching will remain close to $\gamma$ throughout a long update sequence. (Here we focus on the regime of $\gamma \approx 1+\epsilon$.)

Lemma 3.4 (Lemma 3.1 from [44]). *Let $\epsilon, \epsilon' \leq 1/2$. Suppose that $M_i$ is a $(1+\epsilon)$-approximate MCM for $G_i$. For $j = i, i+1, \ldots, i + \lfloor \epsilon' \cdot |M_i| \rfloor$, let $M_i^{(j)}$ denote the matching $M_i$ after removing from it all edges that got deleted during the updates $i+1, \ldots, j$. Then $M_i^{(j)}$ is a $(1+2\epsilon+2\epsilon')$-approximate MCM for the graph $G_j$.*

Next, we adapt the argument of [44] for maintaining a $(1+\epsilon)$-approximate MCM to fully dynamic graphs of neighborhood independence number bounded by $\beta$. One can compute a $(1+\epsilon/4)$-approximate MCM $M_t$ at a certain update step $t$, and then re-use the same matching $M_t^{(i)}$ throughout all update steps $i = t, t+1, \ldots, t' = t + \lfloor \epsilon/4 \cdot |M_t| \rfloor$ (after removing from it all edges that got deleted from the graph between steps $t$ and $i$). By Lemma 3.4, assuming $\epsilon \leq 1/2$, $M_t^{(i)}$ provides a $(1+\epsilon)$-approximate MCM for all graphs $G_i$. Next compute a fresh $(1+\epsilon/4)$-approximate MCM $M_{t'}$ following update step $t'$ and re-use it throughout all update steps $t', t'+1, \ldots, t' + \lfloor \epsilon/4 \cdot |M_{t'}| \rfloor$, and repeat. In this way the static time complexity of computing a $(1+\epsilon)$-approximate MCM $M$ is *amortized* over $1 + \lfloor \epsilon/4 \cdot |M| \rfloor = \Omega(\epsilon \cdot |M|)$ update steps. By Theorem 3.1, the static computation time of a $(1+\epsilon)$-approximate MCM is $O(|M| \cdot \frac{\beta}{\epsilon^2} \cdot \log \frac{1}{\epsilon})$, hence the *amortized* update time is $O(\frac{\beta}{\epsilon^3} \cdot \log \frac{1}{\epsilon})$.

**A Worst-Case Update time.** To achieve a low worst-case update time, a standard tweak (used in [44]) is to simulate the static approximate matching computation within a "time window" of $1 + \lfloor \epsilon/4 \cdot |M| \rfloor$ consecutive update steps, so that following each update step the algorithm simulates only

$$O\left(|M| \cdot \frac{\beta}{\epsilon^2} \cdot \log \frac{1}{\epsilon}\right) / (1 + \lfloor \epsilon/4 \cdot |M| \rfloor) \;=\; O\left(\frac{\beta}{\epsilon^3} \cdot \log \frac{1}{\epsilon}\right)$$

steps of the static computation. During this time window the gradually-computed matching, denoted by $M'$, is useless, so the previously-computed matching $M$ is re-used as the output matching. This means that each matching is re-used throughout a time window of twice as many update steps, hence the approximation factor increases from $1+\epsilon$ to $1+2\epsilon$, but we can reduce it back to $1+\epsilon$ by a scaling argument. (Note that the gradually-computed matching does not include edges that got deleted from the graph during the time window.)

While the bound on the update time is deterministic, the bound $1+\epsilon$ on the approximation factor of the maintained matching is not; indeed, it is determined by that of the static approximate matching algorithm (provided by Theorem 3.1), which, in turn, is determined by that of the random matching sparsifier $G_\Delta$ (provided by Theorem 2.1), which holds with high probability. Note, however, that the stability property, which implies that the same approximation factor (up to an additive error of $O(\epsilon)$) provided by the static computation will continue to hold until the next static computation, is deterministic. As a direct consequence, the guarantee on the approximation factor holds with high probability *against an adaptive adversary.*

To summarize, we have proved the following statement.

Theorem 3.5. *A $(1+\epsilon)$-approximate matching can be maintained in fully dynamic graphs of neighborhood independence number bounded by $\beta$ with a worst-case update time of $O(\frac{\beta}{\epsilon^3} \log \frac{1}{\epsilon})$, for any $0 < \epsilon < 1$ and $\beta = O\left(\frac{\epsilon n}{\log n}\right)$. The bound on the update time holds deterministically while the bound on the approximation factor holds with high probability against an adaptive adversary.*

**Remark.** The previous (deterministic) algorithm by Barenboim and Maimon [14] achieves approximation factor 2 (via a maximal matching) with a higher update time of $O(\sqrt{\beta n})$; specifically, the update time due to [14] is higher than ours by a factor of $\sqrt{\frac{n}{\beta}}$, for constant $\epsilon$. In particular, when $\beta$ and $\epsilon$ are constants, our update time is constant and the update time due to [14] is $O(\sqrt{n})$.

## ACKNOWLEDGMENTS

The second-named author thanks Sepehr Assadi for fruitful discussions. Research was partially supported by Israel Science Foundation grant 1991/19 and by Len Blavatnik and the Blavatnik Family foundation.

## REFERENCES


[1] Kook Jin Ahn and Sudipto Guha. 2013. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.* 222 (2013), 59–79.

[2] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. 1974. *The Design and Analysis of Computer Algorithms.* Addison-Wesley.

[3] Noga Alon, Yossi Matias, and Mario Szegedy. 1996. The Space Complexity of Approximating the Frequency Moments. In *Proc. of the 28th STOC*. 20–29.

[4] Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. 2019. Coresets Meet EDCS: Algorithms for Matching and Vertex Cover on Massive Graphs. In *Proc. of the 13th SODA*. 1616–1635.

[5] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. 2019. Sublinear Algorithms for (Δ + 1) Vertex Coloring. In *Proc. of the 30th SODA*. 767–786.

[6] Sepehr Assadi and Sanjeev Khanna. 2017. Randomized Composable Coresets for Matching and Vertex Cover. In *Proc. of the 29th SPAA*. 3–12.

[7] Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. 2018. Fully dynamic maximal independent set with sublinear update time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*. 815–826.

[8] Sepehr Assadi and Shay Solomon. 2019. When Algorithms for Maximal Independent Set and Maximal Matching Run in Sublinear Time. In *Proc. of the 46th ICALP*. 17:1–17:17.

[9] John Augustine, Anisur Rahaman Molla, and Gopal Pandurangan. 2018. Sublinear Message Bounds for Randomized Agreement. In *Proc. of PODC*, Calvin Newport and Idit Keidar (Eds.). 315–324.

[10] Leonid Barenboim and Michael Elkin. 2011. Distributed deterministic edge coloring using bounded neighborhood independence. In *Proc. of the 30th PODC*. 129–138.

[11] Leonid Barenboim, Michael Elkin, and Tzalik Maimon. 2017. Deterministic Distributed (Delta + o(Delta))-Edge-Coloring, and Vertex-Coloring of Graphs with Bounded Diversity. In *Proc. of PODC*. 175–184.

[12] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. 2012. The Locality of Distributed Symmetry Breaking. In *Proc. of the 53rd FOCS*. 321–330.

[13] Leonid Barenboim and Tzalik Maimon. 2018. Distributed Symmetry Breaking in Graphs with Bounded Diversity. In *Proc. of IPDPS*. 723–732.

[14] Leonid Barenboim and Tzalik Maimon. 2019. Fully Dynamic Graph Algorithms Inspired by Distributed Computing: Deterministic Maximal Matching and Edge Coloring in Sublinear Update-Time. *ACM Journal of Experimental Algorithmics* 24, 1 (2019), 1.14:1–1.14:24.

[15] Leonid Barenboim and Tzalik Maimon. 2020. Simple Distributed Spanners in Dense Congest Networks. In *Proc. of SOFSEM*. 260–272.

[16] Leonid Barenboim and Gal Oren. 2020. Distributed Backup Placement in One Round and its Applications to Maximum Matching Approximation and Self-Stabilization. In *Proc. of the 3rd SOSA@SODA*. 99–105.

[17] Leonid Barenboim and Gal Oren. 2020. Fast Distributed Backup Placement in Sparse and Dense Networks. In *Proc. of the 1st APOCS@SODA*. 90–104.

[18] Surender Baswana, Manoj Gupta, and Sandeep Sen. 2011. Fully Dynamic Maximal Matching in O (log n) Update Time. In *Proc. of the 52nd FOCS*. 383–392.

[19] Aaron Bernstein, Jacob Holm, and Eva Rotenberg. 2018. Online Bipartite Matching with Amortized Replacements. In *Proc. of the 29th SODA*. 947–959.

[20] Aaron Bernstein and Cliff Stein. 2016. Faster Fully Dynamic Matchings with Small Approximation Ratios. In *Proc. of the 27th SODA*. 692–711.

[21] Sayan Bhattacharya, Deeparnab Chakrabarty, and Monika Henzinger. 2017. Deterministic Fully Dynamic Approximate Vertex Cover and Fractional Matching in O(1) Amortized Update Time. In *Proc. of the 19th IPCO*. 86–98.

[22] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. 2018. Deterministic Fully Dynamic Data Structures for Vertex Cover and Matching. *SIAM J. Comput.* 47, 3 (2018), 859–887.

[23] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. 2016. New deterministic approximation algorithms for fully dynamic matching. In *Proc. of the 48th STOC*. 398–411.

[24] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. 2016. New deterministic approximation algorithms for fully dynamic matching. In *Proc. of the 48th STOC*. 398–411.

[25] Sayan Bhattacharya and Janardhan Kulkarni. 2019 (to appear). Deterministically Maintaining a $(2+\epsilon)$-Approximate Minimum Vertex Cover in $O(1/\epsilon^2)$ Amortized Update Time. In *Proc. of the 30th SODA*. Also posted on CoRR, abs/1805.03498, 2018.

[26] Bartlomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych. 2014. Online Bipartite Matching in Offline Time. In *Proc. of the 55th FOCS*. 384–393.

[27] Keren Censor-Hillel, Elad Haramaty, and Zohar S. Karnin. 2016. Optimal Dynamic Distributed MIS. In *Proc. of the PODC*. 217–226.

[28] Moses Charikar and Shay Solomon. 2018. Fully Dynamic Almost-Maximal Matching: Breaking the Polynomial Worst-Case Time Barrier. In *Proc. of the 45th ICALP*. 33:1–33:14.

[29] Nicos Christofides. 1976. *Worst-case analysis of a new heuristic for the travelling salesman problem.* Technical Report. Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group.

[30] Maria Chudnovsky and Paul D. Seymour. 2005. The structure of claw-free graphs. In *Surveys in Combinatorics, 2005 [invited lectures from the Twentieth British Combinatorial Conference, Durham, UK, July 2005]*. Cambridge University Press, 153–171.

[31] Artur Czumaj, Jakub Lacki, Aleksander Madry, Slobodan Mitrovic, Krzysztof Onak, and Piotr Sankowski. 2018. Round compression for parallel matching algorithms. In *Proc. of the 50th STOC*.

[32] Thomas E Easterfield. 1946. A combinatorial algorithm. *Journal of the London Mathematical Society* 1, 3 (1946), 219–226.

[33] Jack Edmonds. 1965. Paths, trees, and flowers. *Canadian Journal of Mathematics* 17 (1965), 449–467.

[34] Guy Even, Moti Medina, and Dana Ron. 2014. Distributed Maximum Matching in Bounded Degree Graphs. *CoRR* abs/1407.7882 (2014).

[35] Ralph J. Faudree, Evelyne Flandrin, and Zdenek Ryjácek. 1997. Claw-free graphs - A survey. *Discrete Mathematics* 164, 1-3 (1997), 87–147.

[36] Zvi Galil and Victor Y. Pan. 1985. Improved Processor Bounds for Algebraic and Combinatorial Problems in RNC. In *Proc. of the 26th FOCS*. 490–495.

[37] Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. 2019. Online Matching with General Arrivals. In *Proc. of the 60th FOCS*. 26–37.

[38] Beat Gfeller and Elias Vicari. 2007. A randomized distributed algorithm for the maximal independent set problem in growth-bounded graphs. In *Proc. of the 26th PODC*. 53–60.

[39] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. 2018. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. In *Proc. of PODC*. 129–138.

[40] Mohsen Ghaffari and Fabian Kuhn. 2018. Distributed MST and Broadcast with Fewer Messages, and Faster Gossiping. In *Proc. of the 32nd DISC*. 30:1–30:12.

[41] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. 2009. Perfect matchings via uniform sampling in regular bipartite graphs. In *Proc. of the 20th SODA*. 11–17.

[42] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. 2010. Perfect matchings in $o(n \log n)$ time in regular bipartite graphs. In *Proc. of the 42nd STOC*. 39–46.

[43] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. 2012. On the communication and streaming complexity of maximum bipartite matching. In *Proc. of the 23rd SODA*. 468–485.

[44] Manoj Gupta and Richard Peng. 2013. Fully Dynamic (1+ e)-Approximate Matchings. In *Proc. of the 54th FOCS*. 548–557.

[45] Frank Hadlock. 1975. Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Comput.* 4, 3 (1975), 221–225.

[46] Magnús M. Halldórsson. 2009. Wireless Scheduling with Power Control. In *Proc. of 17th ESA*, Amos Fiat and Peter Sanders (Eds.). 361–372.

[47] Magnús M. Halldórsson and Christian Konrad. 2015. Distributed Large Independent Sets in One Round on Bounded-Independence Graphs. In *Proc. of DISC*. 559–572.

[48] Magnús M. Halldórsson, Guy Kortsarz, and Hadas Shachnai. 2003. Sum Coloring Interval and k-Claw Free Graphs with Application to Scheduling Dependent Jobs. *Algorithmica* 37, 3 (2003), 187–209.

[49] Michal Hanckowiak, Michal Karonski, and Alessandro Panconesi. 1998. On the Distributed Complexity of Computing Maximal Matchings. In *Proc. of the 9th SODA*. 219–225.

[50] Frank L Hitchcock. 1941. The distribution of a product from several sources to numerous localities. *Journal of mathematics and physics* 20, 1-4 (1941), 224–230.

[51] John E. Hopcroft and Richard M. Karp. 1973. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.* 2, 4 (1973), 225–231.

[52] John E. Hopcroft and Richard M. Karp. 1973. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.* 2, 4 (1973), 225–231.

[53] Amos Israeli and Alon Itai. 1986. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Inf. Process. Lett.* 22, 2 (1986), 77–80.

[54] Lester Randolph Ford Jr. and Delbert Ray Fulkerson. 1965. *Flows in Networks.* Princeton University Press, Princeton, New Jersey.

[55] Leonid V Kantorovich. 1942. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, Vol. 37. 199–201.

[56] Haim Kaplan and Shay Solomon. 2018. Dynamic Representations of Sparse Distributed Networks: A Locality-Sensitive Approach. In *Proc. of the 30th SPAA*. 33–42.

[57] Michael Kapralov. 2013. Better bounds for matchings in the streaming model. In *Proc. of the 24th SODA*. 1679–1697.

[58] Richard M. Karp, Eli Upfal, and Avi Wigderson. 1985. Constructing a Perfect Matching is in Random NC. In *Proc. of the 17th STOC*. 22–32.

[59] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. 1990. An Optimal Algorithm for On-line Bipartite Matching. In *Proc. of the 22nd STOC*. 352–358.

[60] Valerie King, Shay Kutten, and Mikkel Thorup. 2015. Construction and Impromptu Repair of an MST in a Distributed Network with o(m) Communication. In *Proc. of PODC*. 71–80.

[61] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 2004. What cannot be computed locally!. In *Proc. of the 23rd PODC*. 300–309.

[62] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. 2008. Ad hoc networks beyond unit disk graphs. *Wireless Networks* 14, 5 (2008), 715–729.

[63] Shay Kutten, Gopal Pandurangan, David Peleg, Peter Robinson, and Amitabh Trehan. 2013. Sublinear Bounds for Randomized Leader Election. In *Proc. of the 14th ICDCN (Lecture Notes in Computer Science)*. 348–362.

[64] Eugene L Lawler. 1976. Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston. *New York* (1976).

[65] Christoph Lenzen and Roger Wattenhofer. 2010. Minimum Dominating Set Approximation in Graphs of Bounded Arboricity. In *Proc. of DISC*. 510–524.

[66] G. Lev. 1980. *Size bounds and parallel algorithms for networks.* Technical Report CST-8-80. Dept. of Computer Science, Univ. of Edinburgh.

[67] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. 2015. Improved Distributed Approximate Matching. *J. ACM* 62, 5 (2015), 38:1–38:17.

[68] Ali Mashreghi and Valerie King. 2018. Broadcast and Minimum Spanning Tree with o(m) Messages in the Asynchronous CONGEST Model. In *Proc. of the 32nd DISC*. 37:1–37:17.

[69] Ali Mashreghi and Valerie King. 2019. Brief Announcement: Faster Asynchronous MST and Low Diameter Tree Construction with Sublinear Communication. In *Proc. of the 33rd DISC*. 49:1–49:3.

[70] Silvio Micali and Vijay V. Vazirani. 1980. An O(sqrt(|v|) |E|) Algorithm for Finding Maximum Matching in General Graphs. In *Proc. of the 21st FOCS*. IEEE Computer Society, 17–27.

[71] Ofer Neiman and Shay Solomon. 2013. Simple deterministic algorithms for fully dynamic maximal matching. In *Proc. of the 45th STOC*. 745–754.

[72] Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. 2012. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proc. of the 23rd SODA*. 1123–1131.

[73] GI Orlova. 1972. Finding the maximum cut in a graph. *Engineering Cybernetics* 10 (1972), 502–506.

[74] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. 2017. A time- and message-optimal distributed algorithm for minimum spanning trees. In *Proc. of the 49th STOC*. 743–756.

[75] Merav Parter, David Peleg, and Shay Solomon. 2016. Local-on-Average Distributed Tasks. In *Proc. of the 27th SODA*. 220–239.

[76] D. Peleg. 2000. *Distributed Computing: A Locality-Sensitive Approach.* SIAM.

[77] David Peleg and Shay Solomon. 2016. Dynamic $(1+\epsilon)$-Approximate Matchings: A Density-Sensitive Approach. In *Proc. of the 27th SODA*. 712–729.

[78] Johannes Schneider and Roger Wattenhofer. 2008. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *Proc. of the 27th PODC*. 35–44.

[79] Johannes Schneider and Roger Wattenhofer. 2010. An optimal maximal independent set algorithm for bounded-independence graphs. *Distributed Computing* 22, 5-6 (2010), 349–361.

[80] Shay Solomon. 2016. Fully Dynamic Maximal Matching in Constant Update Time. In *Proc. of the 57th FOCS*. 325–334.

[81] Shay Solomon. 2018. Local Algorithms for Bounded Degree Sparsifiers in Sparse Graphs. In *Proc. of the 9th ITCS*. 52:1–52:19.

[82] Robert L Thorndike. 1950. The problem of classification of personnel. *Psychometrika* 15, 3 (1950), 215–235.

[83] Vijay V. Vazirani. 2012. An Improved Definition of Blossoms and a Simpler Proof of the MV Matching Algorithm. *CoRR* abs/1210.4594 (2012).

[84] David Wajc. 2020. Rounding Dynamic Matchings Against an Adaptive Adversary. In *STOC*. To appear. Also in CoRR, abs/1911.05545.