When Algorithms for Maximal Independent Set and Maximal Matching Run in Sublinear Time

Sepehr Assadi
Department of Computer Science, Princeton University, NJ, USA
sassadi@princeton.edu

Shay Solomon
School of Electrical Engineering, Tel Aviv University, Israel
shayso@post.tau.ac.il

──── **Abstract** ────

Maximal independent set (MIS), maximal matching (MM), and $(\Delta + 1)$-(vertex) coloring in graphs of maximum degree Δ are among the most prominent algorithmic graph theory problems. They are all solvable by a simple linear-time greedy algorithm and up until very recently this constituted the state-of-the-art. In SODA 2019, Assadi, Chen, and Khanna gave a randomized algorithm for $(\Delta + 1)$-coloring that runs in $\widetilde{O}(n\sqrt{n})$ time[1], which even for moderately dense graphs is sublinear in the input size. The work of Assadi et al. however contained a spoiler for MIS and MM: neither problems provably admits a sublinear-time algorithm in general graphs. In this work, we dig deeper into the possibility of achieving sublinear-time algorithms for MIS and MM.

The neighborhood independence number of a graph G, denoted by $\beta(G)$, is the size of the largest independent set in the neighborhood of any vertex. We identify $\beta(G)$ as the "right" parameter to measure the runtime of MIS and MM algorithms: Although graphs of bounded neighborhood independence may be very dense (clique is one example), we prove that carefully chosen variants of greedy algorithms for MIS and MM run in $O(n\beta(G))$ and $O(n \log n \cdot \beta(G))$ time respectively on any n-vertex graph G. We complement this positive result by observing that a simple extension of the lower bound of Assadi et al. implies that $\Omega(n\beta(G))$ time is also necessary for any algorithm to either problem for all values of $\beta(G)$ from 1 to $\Theta(n)$. We note that our algorithm for MIS is deterministic while for MM we use randomization which we prove is unavoidable: any deterministic algorithm for MM requires $\Omega(n^2)$ time even for $\beta(G) = 2$.

Graphs with bounded neighborhood independence, already for constant $\beta = \beta(G)$, constitute a rich family of possibly dense graphs, including line graphs, proper interval graphs, unit-disk graphs, claw-free graphs, and graphs of bounded growth. Our results suggest that even though MIS and MM do not admit sublinear-time algorithms in general graphs, one can still solve both problems in sublinear time for a wide range of $\beta(G) \ll n$.

Finally, by observing that the lower bound of $\Omega(n\sqrt{n})$ time for $(\Delta + 1)$-coloring due to Assadi et al. applies to graphs of (small) constant neighborhood independence, we unveil an intriguing separation between the time complexity of MIS and MM, and that of $(\Delta + 1)$-coloring: while the time complexity of MIS and MM is strictly higher than that of $(\Delta + 1)$ coloring in general graphs, the exact opposite relation holds for graphs with small neighborhood independence.

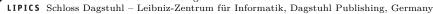[1] Here, and throughout the paper, we define $\widetilde{O}(f(n)) := O(f(n) \cdot \mathrm{polylog}(n))$ to suppress log-factors.

1 Introduction

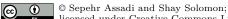Maximal independent set (MIS) and maximal matching (MM) are two of the most prominent graph problems with a wide range of applications in particular to symmetry breaking. Algorithmic study of these problems can be traced back to at least four decades ago in the pioneering work of [33, 41, 1, 32] on PRAM algorithms. These problems have since been studied extensively in various models including distributed algorithms [40, 46, 31, 34, 38, 10, 9, 21, 19], dynamic algorithms [43, 11, 50, 45, 5, 6], streaming algorithms [18, 27, 15, 4], massively parallel computation (MPC) algorithms [37, 22, 12], local computation algorithms (LCA) [48, 2, 21, 20, 39, 23], and numerous others.

In this paper, we consider the **time complexity** of MIS and MM (in the centralized setting) and focus on one of the most basic questions regarding these two problems:

How fast can we solve maximal independent set and maximal matching problems?

At first glance, the answer to this question may sound obvious: there are text-book greedy algorithms for both problems that run in linear time and "of course" one cannot solve these problems faster as just reading the input takes linear time. This answer however is not quite warranted: for the closely related problem of $(\Delta + 1)$-(vertex) coloring, very recently Assadi, Chen, and Khanna [4] gave a randomized algorithm that runs in only $\widetilde{O}(n\sqrt{n})$ time on any n-vertex graph with high probability[2]. This means that even for moderately dense graphs, one can indeed color the graph faster than reading the entire input, i.e., in sublinear time.

The Assadi-Chen-Khanna algorithm hints that one could perhaps hope for sublinear-time algorithms for MIS and MM as well. Unfortunately however, the work of [4] already contained a spoiler: neither MIS nor MM admits a sublinear-time algorithm in general graphs.

In this work, we show that despite the negative result of [4] for MIS and MM, the hope for obtaining sublinear-time algorithms for these problems need not be short lived. In particular, we identify a key parameter of the graph, namely the *neighborhood independence number*, that provides a more nuanced measure of runtime for these problems and show that both problems can be solved much faster when neighborhood independence is small. This in turn gives rise to sublinear-time algorithms for MIS and MM on a rich family of graphs with bounded neighborhood independence. In the following, we elaborate more on our results.

1.1 Our Contributions

For a graph $G(V, E)$, the neighborhood independence number of G, denoted by $\beta(G)$, is defined as the size of the largest independent set in the graph in which all vertices of the independent set are incident on some shared vertex $v \in V$. Our main result is as follows:

> ▷ Result 1. There exist algorithms that given a graph $G(V, E)$ find (i) a maximal independent set of G deterministically in $O(n \cdot \beta(G))$ time, and (ii) a maximal matching of G randomly in $O(n \log n \cdot \beta(G))$ time in expectation and with high probability.

When considering sublinear-time algorithms, specifying the exact data model is important as the algorithm cannot even read the entire input once. We assume that the input graph is presented in the adjacency array representation, i.e., for each vertex $v \in V$, we are given degree $\mathsf{deg}(v)$ of v followed by an array of length $\mathsf{deg}(v)$ containing all neighbors of v in

---

[2] We say an event happens with high probability if it happens with probability at least $1 - 1/\mathrm{poly}(n)$.

arbitrary order. This way, we can access the degree of any vertex v or its i-th neighbor for $i \in [\deg(v)]$ in $O(1)$ time. 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. [25, 26, 44]). We elaborate on several aspects of Result 1 in the following.

Optimality of Our Bounds. Assadi et al. [4] proved that any algorithm for MIS or MM requires $\Omega(n^2)$ time in general[3]. These lower bounds can be extended in an easy way to prove that $\Omega(n \cdot \beta)$ time is also necessary for both problems on graphs with neighborhood independence $\beta(G) = \beta$. Indeed, independently sample $t := n/\beta$ graphs $G_1, \ldots, G_t$ each on β vertices from the hard distribution of graphs in [4] and let G be the union of these graphs. Clearly, $\beta(G) \leq \beta$ and it follows that since solving MIS or MM on each graph G_i requires $\Omega(\beta^2)$ time by the lower bound of [4], solving t independent copies requires $\Omega(t \cdot \beta^2) = \Omega(n\beta)$ time. As such, our Result 1 is optimal for every β ranging from a constant to $\Theta(n)$ (up to a constant factor for MIS and $O(\log n)$ for MM).

Our Algorithms. Both our algorithms for MIS and MM in Result 1 are similar to the standard greedy algorithms, though they require careful adjustments and implementation. Specifically, the algorithm for MIS is the standard deterministic greedy algorithm (with minimal modification) and for MM we use a careful implementation of the (modified) randomized greedy algorithm (see, e.g. [16, 3, 42, 47]). The novelty of our work mainly lies in the analysis of these algorithms. We show, perhaps surprisingly, that already-known algorithms can in fact achieve an improved performance and run in sublinear-time for graphs with bounded neighborhood independence *even* when the value of $\beta(G)$ is unknown to the algorithms. Combined with the optimality of our bounds mentioned earlier, we believe that this makes neighborhood independence number an ideal parameter for measuring the runtime of MIS and MM algorithms.

Determinism and Randomization. Our MIS algorithm in Result 1 is deterministic which is a rare occurrence in the realm of sublinear-time algorithms. But for MM, we again fall back on randomization to achieve sublinear-time performance. This is not a coincidence however: we prove in Theorem 11 that any deterministic algorithm for MM requires $\Omega(n^2)$ time even on graphs with constant neighborhood independence number. This also suggests a separation in the time complexity of MIS and MM for deterministic algorithms.

Bounded Neighborhood Independence. Our Result 1 is particularly interesting for graphs with constant neighborhood independence as we obtain quite fast algorithms with running time $O(n)$ and $O(n \log n)$ for MIS and MM, respectively. Graphs with constant neighborhood independence capture a rich family of graphs; several illustrative examples are as follows:

- *Line graphs:* For any arbitrary graph G, the neighborhood independence number of its line graph $L(G)$ is at most 2. More generally, for any r-hyper graph $\mathcal{H}$ in which each hyper-edge connects at most r vertices, $\beta(L(\mathcal{H})) \leq r$.

---

[3] We note that the lower bound for MIS only appears in the full version of [4].

- *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 $f(r)$. Bounded-growth graphs in turn capture several intersection graphs of geometrical objects such as proper interval graphs [30], unit-disk graphs [28], quasi-unit-disk graphs [36], and general disc graphs [29].

- *Claw-free graphs:* Graphs with neighborhood independence β can be alternatively defined as β-claw-free graphs, i.e., graphs that do not contain $K_{1,\beta}$ as an induced subgraph. Claw-free graphs have been subject of extensive study in structural graph theory; see the series of papers by Chudnovsky and Seymour, starting with [14], and the survey by Faudree et al. [17].

Above graphs appear naturally in the context of symmetry breaking problems (for instance in the study of wireless networks), and there have been numerous works on MIS and MM in graphs with bounded neighborhood independence and their special cases (see, e.g. [36, 49, 28, 7, 8, 29, 19] and references therein).

1.2   Other Implications

Despite the simplicity of our algorithms in Result 1, they lead to several interesting implications, when combined with some known results and/or techniques:

(a) *Approximate vertex cover and matching:* Our MM algorithm in Result 1 combined with well-known properties of maximal matchings implies an $O(n \log n \cdot \beta(G))$ time 2-approximation algorithm to both maximum matching and minimum vertex cover. For graphs with constant neighborhood independence, our results improve upon the sublinear-time algorithms of [44] that achieve $(2 + \varepsilon)$-approximation to the *size* of the optimal solution to both problems but did not find the actual edges or vertices in $\widetilde{O}_\varepsilon(n)$ time on general graphs.

(b) *Caro-Wei bound and approximation of maximum independent set:* The Caro-Wei bound [13, 51] states that any graph $G(V, E)$ contains an independent set of size at least $\sum_{v \in V} \frac{1}{\deg(v)+1}$, and there is a substantial interest in obtaining independent sets of this size (see, e.g. [27, 29, 15] and references therein). One standard way of obtaining such independent set is to run the greedy MIS algorithm on the vertices of the graph in the increasing order of their degrees. As our Result 1 implies that one can implement the greedy MIS algorithm for *any* ordering of vertices, we can sort the vertices in $O(n)$ time and then run our deterministic algorithm with this order to obtain an independent set with Caro-Wei bound size in $O(n\beta(G))$ time. Additionally, it is easy to see that on graphs with $\beta(G) = \beta$, any MIS is a β-approximation to the *maximum* independent set (see, e.g. [35, 49]). We hence also obtain a constant factor approximation in $O(n)$ time for maximum independent set on graphs with bounded neighborhood independence.

(c) *Separation of $(\Delta + 1)$-coloring with MIS and MM:* Assadi et al. [4] gave an $\widetilde{O}(n\sqrt{n})$ time algorithm for $(\Delta + 1)$ coloring and an $\Omega(n^2)$ time lower bound for MIS and MM on general graphs. It is also shown in [4] that $(\Delta + 1)$ coloring requires $\Omega(n\sqrt{n})$ time and in fact the lower bound holds for graphs with constant neighborhood independence. Together with our Result 1, this implies an interesting separation between the time-complexity of MIS and MM, and that of $(\Delta + 1)$-coloring: while the time complexity of MIS and MM is strictly higher than that of $(\Delta + 1)$ coloring in general graphs, the exact opposite relation holds for graphs with small neighborhood independence number.

(d) *Efficient MM computation via MIS on line graphs:* The line graph $L(G)$ of a graph G contains m vertices corresponding to edges of G and up to $O(mn)$ edges. Moreover, for any graph G, $\beta(L(G)) \leq 2$. As an MIS in $L(G)$ corresponds to an MM in G, our results suggest that despite the larger size of $L(G)$, perhaps surprisingly, computing an MM of G through computing an MIS for $L(G)$ is just as efficient as directly computing an MM of G (assuming direct access to $L(G)$). This observation may come into play in real-life situations where there is no direct access to the graph but rather only to its line graph.

Preliminaries and Notation

For a graph $G(V, E)$ and vertex $v \in V$, $N(v)$ and $\deg(v)$ denote the neighbor-set and degree of a vertex v, respectively. For a subset $U \subseteq V$, $\deg_U(v)$ denotes the degree of v to vertices in U. Denote by $\beta(G)$ the neighborhood independence number of graph G.

2 Technical and Conceptual Highlights

Our first (non-technical) contribution is in identifying the neighborhood independence number as the "right" measure of time-complexity for both MIS and MM. We then show that surprisingly simple algorithms for these problems run in sublinear-time on graphs with bounded $\beta(G)$.

The textbook greedy algorithm for MIS works as follows: scan the vertices in an arbitrary order and add each scanned vertex to a set $\mathcal{M}$ iff it does not already have a neighbor in $\mathcal{M}$. Clearly the runtime of this algorithm is $\Theta(\sum_{v \in V} \deg(v)) = \Theta(m)$ and this bound does not improve for graphs with small β. We can slightly tweak this algorithm by making every vertex that joins $\mathcal{M}$ to *mark* all its neighbors and simply ignore scanning the already marked vertices. This tweak however is not useful in general graphs as the algorithm may waste time by repeatedly marking the same vertices over and over again without making much further progress (the complete bipartite graph is an extreme example). The same problem manifests itself in other algorithms, including those for MM, and is at the root of the lower bounds in [4] for sublinear-time computation of MIS and MM.

We prove that this issue cannot arise in graphs with bounded neighborhood independence. Noting that the runtime of the greedy MIS algorithm that uses "marks" is $\Theta(m_\mathcal{M})$, where we define $m_\mathcal{M} := \sum_{v \in \mathcal{M}} \deg(v)$, a key observation is that $m_\mathcal{M}$ is much smaller than m when β is small. Indeed, as the vertices of $\mathcal{M}$ form an independent set, all the edges incident on $\mathcal{M}$ lead to $V \setminus \mathcal{M}$, and so if $m_\mathcal{M}$ is large, then the *average* degree of $V \setminus \mathcal{M}$ to $\mathcal{M}$ cannot be "too small"; however, the latter average degree cannot be larger than β as otherwise there is some vertex in $V \setminus \mathcal{M}$ that is incident to more than β independent vertices, a contradiction. This is all we need to conclude that the runtime of the greedy MIS algorithm that uses marks is bounded by $O(n \cdot \beta)$.

Both the MIS algorithm and its analysis are remarkably simple, and *in hindsight*, this is not surprising since this parameter β is in a sense "tailored" to the MIS problem. Although MM and MIS problems are intimately connected to each other, the MM problem appears to be much more intricate for graphs with bounded neighborhood independence. Indeed, while the set U of unmatched vertices in any MM forms an independent set and hence total number m_U of edges incident on U cannot be too large by the above argument, the runtime of greedy or any other algorithm cannot be bounded in terms of m_U (as m_U can simply be zero). In fact, it is provably impossible to adjust our argument for MIS to the MM problem due to our lower bound for deterministic MM algorithms (Theorem 11) that shows that any such algorithm must incur a runtime of $\Omega(n^2)$ even for $\beta = 2$.

The main technical contribution of this paper is thus in obtaining a fast randomized MM algorithm for graphs with bounded β. Our starting point is the modified randomized greedy (MRG) algorithm of [16, 3] that finds an MM by iteratively picking an *unmatched* vertex u uniformly at random and matching it to a uniformly at random chosen *unmatched* neighbor $v \in N(u)$. On its own, this standard algorithm does not benefit from small values of β: while picking an unmatched vertex u is easy, finding an unmatched neighbor v for u is too time consuming in general. We instead make the following simple but crucial modification: instead of picking v from unmatched neighbors of u, we simply sample v from the set of *all* neighbors of u and only match it to u if it is also unmatched; otherwise we sample another vertex u and continue like this (additional care is needed to ensure that this process even terminates but we postpone the details to Section 4).

To analyze the runtime of this modified algorithm, we leverage the above argument for MIS and take it to the next step to prove a basic structural property of graphs with bounded neighborhood independence: for any set P of vertices, a constant fraction of vertices are such that their inner degree inside P is "not much smaller" than their total degree (depending both on β and size of P). Letting P to be the set of unmatched vertices in the above algorithm allows us to bound the number of iterations made by the algorithm before finding the next matching edge, and ultimately bounding the overall runtime of the algorithm by $O(n \log n \cdot \beta)$ both in expectation and with high probability.

Technical Comparison with Prior Work

Our work is most closely related to the $\widetilde{O}(n\sqrt{n})$-time $(\Delta + 1)$-coloring algorithm of Assadi, Chen, and Khanna [4] (and their $\Omega(n^2)$ time lower bounds for MIS and MM on general graphs), as well as the series of work by Goel, Kapralov, and Khanna [25, 24, 26] on finding perfect matchings in regular bipartite graphs that culminated in an $O(n \log n)$ time algorithm.

The coloring algorithm of [4] works by *non-adaptively sparsifying* the graph into $O(n \log^2(n))$ edges in $\widetilde{O}(n\sqrt{n})$ time in such a way that a $(\Delta + 1)$ coloring of the original graph can be found quickly from this sparsifier. The algorithms in [25, 24] were also based on the high-level idea of sparsification but the final work in this series [26] instead used a *(truncated) random walk* approach to speedup augmenting path computations in regular graphs. The sparsification methods used in [4, 25, 24] as well as the random walk approach of [26] are all quite different from our techniques in this paper that are tailored to graphs with bounded neighborhood independence. Moreover, even though every perfect matching is clearly maximal, our results and [25, 24, 26] are incomparable as d-regular bipartite graphs and graphs with bounded neighborhood independence are in a sense the exact opposite of each other: for a d-regular bipartite graph, $\beta(G) = d$ which is the largest possible for graphs with maximum degree d.

3   Maximal Independent Set

The standard greedy algorithm for MIS works as follows: Iterate over vertices of the graph in an arbitrary order and insert each one to an initially empty set $\mathcal{M}$ if none of its neighbors have already been inserted to $\mathcal{M}$. By the time all vertices have been processed, $\mathcal{M}$ clearly provides an MIS of the input graph. See Algorithm 1 for a pseudo-code.

We prove that this algorithm is fast on graphs with bounded neighborhood independence.

▶ **Theorem 1.** *The greedy MIS algorithm (as specified by Algorithm 1) computes a maximal independent set of a graph G given in adjacency array representation in $O(n \cdot \beta(G))$ time.*

Algorithm 1: The (Deterministic) Greedy Algorithm for Maximal Independent Set.

---

1 Input: An n-vertex graph $G(V, E)$ given in adjacency array representation.

2 Output: An MIS $\mathcal{M}$ of G.

3 Initialize $\mathcal{M} = \emptyset$ and $\mathtt{mark}[v_i] = \text{FALSE}$ for all vertices $v_i \in V$ where
$V := \{v_1, \ldots, v_n\}$.

4 for $i = 1$ **to** n **do**

5   **if** $\mathtt{mark}[v_i] = \text{FALSE}$ **then**

6   add v_i to $\mathcal{M}$ and set $\mathtt{mark}[u] = \text{TRUE}$ for all $u \in N(v_i)$.

7   **end**

8 end

9 Return $\mathcal{M}$.

Proof. Let $G(V, E)$ be an arbitrary graph. Suppose we run Algorithm 1 on G and obtain $\mathcal{M}$ as the resulting MIS. To prove Theorem 1, we use the following two simple claims.

▷ **Claim 2.** The time spent by Algorithm 1 on a graph $G(V, E)$ is $O(n + \sum_{v \in \mathcal{M}} \deg(v))$.

Proof. Iterating over vertices in the **for** loop takes $O(n)$ time. Beyond that, for each vertex joining the MIS $\mathcal{M}$, we spend time that is linear in its degree to mark all its neighbors. ◁

▷ **Claim 3.** For any independent set $I \subseteq V$ in G, $\sum_{v \in I} \deg(v) \leq n \cdot \beta(G)$.

Proof. Let $E(I)$ denote the edges incident on vertices in the independent set I. Since I is an independent set, these edges connect vertices of I with vertices of $V \setminus I$. Suppose towards a contradiction that $|E(I)| = \sum_{v \in I} \deg(v) > n \cdot \beta(G)$. By a double counting argument, there must exist a vertex v in $V \setminus I$ with at least $|E(I)| / |V \setminus I| > \beta(G)$ neighbors in I. But since I is an independent set, this means that there exists an independent set of size $> \beta(G)$ in the neighborhood of v, which contradicts the fact that $\beta(G)$ is the neighborhood independence number of G. ◁

Theorem 1 now follows from Claims 2 and 3 as $\mathcal{M}$ is an independent set of G. ◀

4   Maximal Matching

We now consider the maximal matching (MM) problem. Similar to MIS, a standard greedy algorithm for MM is to iterate over the vertices in arbitrary order and match each vertex to one of their unmatched neighbors (if any). However, as we show in Section 5 this and any other *deterministic* algorithm for MM, cannot run in sublinear-time even when $\beta(G) = 2$.

We instead consider the following variant of the greedy algorithm, referred to as the *(modified) randomized greedy algorithm*, put forward by [16, 3] and extensively studied in the literature primarily with respect to its approximation ratio for the maximum matching problem (see, e.g. [42, 47] and the references therein). Pick an unmatched vertex u uniformly at random; pick an unmatched vertex v incident on u uniformly at random and add (u, v) to the matching M; repeat as long as there is an unmatched edge left in the graph. It is easy to see that at the end of the algorithm M will be an MM of G.

As it is, this algorithm is not suitable for our purpose as finding an unmatched vertex v incident on u is too costly. We thus instead consider the following variant which samples the set v from all neighbors of u and only match it to u if v is also unmatched (we also change the final check of the algorithm for maximality of M with a faster computation). See Algorithm 2 for a pseudo-code after proper modifications.

Algorithm 2: The (Modified) Randomized Greedy Algorithm for Maximal Matching.

---

1 Input: An n-vertex graph $G(V, E)$ given in adjacency array representation.

2 Output: A maximal matching M of G.

3 Initialize $M = \emptyset$ and $U = V$.

4 while $U \neq \emptyset$ **do**

5   Define the threshold $\tau := \tau(U) = \frac{4n \cdot \beta(G)}{|U|}$.

6   Sample a vertex u uniformly at random from U.

7   **if** $\deg(u) < \tau$ **then**

8   Choose a random vertex v from $N(u) \cap U$ (if non-empty), add (u, v) to M and set $U \leftarrow U \setminus \{u, v\}$. If $N(u) \cap U = \emptyset$, set $U \leftarrow U \setminus \{u\}$.

9   **else**

10   Sample a vertex v uniformly at random from $N(v)$.

11   **if** $v \in U$ **then**

12   add (u, v) to M and set $U \leftarrow U \setminus \{u, v\}$.

13   **end**

14 end

15 Return M.

We remark that the first **if** condition in Algorithm 2 is used to remove the costly operation of checking if any unmatched edge is left in the graph. It is easy to see that this algorithm always output an MM.

We prove that Algorithm 2 is fast both in expectation and with high probability on graphs with bounded neighborhood independence. We also note that as stated, Algorithm 2 actually assumes knowledge of $\beta(G)$ (needed for the definition of the threshold parameter τ). However, we show at the end of this section that this assumption can be lifted easily and obtain a slight modification of Algorithm 2 with the same asymptotic runtime that does *not* require any knowledge of $\beta(G)$.

▶ **Theorem 4.** *The modified randomized greedy MM algorithm (as specified by Algorithm 2) computes a maximal matching of a graph G given in adjacency array representation in $O(n \log n \cdot \beta(G))$ time in expectation and with high probability.*

Let t denote the number of iterations of the **while** loop in Algorithm 2. We can bound the runtime of this algorithm based on t as follows.

▷ Claim 5.   Algorithm 2 can be implemented in $O(n \log n \cdot \beta(G) + t)$ time.

Proof. First, we would like to store the set U in a data structure that supports random sampling and deletion of a vertex from U, as well as determining whether a vertex is currently in U or not, in constant time. This data structure can be easily implemented using two

arrays A_1 and A_2; we provide the rather tedious details for completeness. The arrays are initialized as $A_1[i] = A_2[i] = i$ for all $i = 1, \ldots, n$, where $A_2[i]$ holds the index of the cell in A_1 where v_i is stored, or -1 if v_i is not in U, while A_1 stores the vertices of U in its first $|U|$ cells, identifying each v_i with index i. When any unmatched vertex v_i is removed from U, we first use $A_2[i]$ to determine the cell where v_i is stored in A_1, then we move the unmatched vertex stored at the last cell in A_1, $A_1[|U|]$, to the cell currently occupied by v_i by setting $A_1[A_2[i]] = A_1[|U|]$, and finally set $A_2[A_1[|U|]] = A_2[i], A_2[i] = -1$. Randomly sampling a vertex from U and determining whether a vertex belongs to U can now be done in $O(1)$ time.

Using these arrays each iteration of the **while** loop in which $\deg(u) \geq \tau$ can be carried out within $O(1)$ time. Iterations for which $\deg(u) < \tau$ are more costly, due to the need to determine $N(u) \cap U$. Nonetheless, in each such iteration we spend at most $O(\tau)$ time while at least one vertex is removed from U, hence the time required by all such iterations is bounded by $\sum_{k=1}^{n} O(\frac{n \cdot \beta(G)}{k}) = O(n \log n \cdot \beta(G))$. It follows that the total runtime of the algorithm is $O(n \log n \cdot \beta(G) + t)$. ◁

The main ingredient of the analysis is thus to bound the number t of iterations. Before proceeding we introduce some definition. We say that an iteration of the **while** loop *succeeds* iff we remove at least one vertex from U in this iteration. Clearly, there can be at most n successful iterations. We prove that each iteration of the algorithm is successful with a sufficiently large probability, using which we bound the total number of iterations.

To bound the success probability, we shall argue that for sufficiently many vertices u in U, the number of its neighbors in U, referred to as its *internal* degree, is proportionate to the number of its neighbors outside U, referred to as its *external* degree; for any such vertex u, a random neighbor v of u has a good chance of belonging to U. This is captured by the following definition.

▶ **Definition 6** (Good vertices). *For a parameter $\delta \in (0, 1)$, we say a vertex $u \in U$ is δ-good iff $\deg_U(u) \geq \delta \cdot \deg_{V \setminus U}(u)$ or $\deg(u) < 1/\delta$.*

Clearly, if we sample a δ-good vertex $u \in U$ in some iteration, then with probability $\geq \delta$ that iteration succeeds. It thus remains to show that many vertices in U are good for an appropriate choice of δ. We use the bounded neighborhood independence property (in a more sophisticated way than it was used in the proof of Theorem 1) to prove the following lemma, which lies at the core of the analysis.

▶ **Lemma 7.** *Fix any choice of U in some iteration and let $\delta := \delta(U) = 1/\tau(U)$ (for the parameter $\tau(U)$ defined in Algorithm 2). Then at least half the vertices in U are δ-good in this iteration.*

Proof. Let us say that a vertex $u \in U$ is *bad* iff it is not δ-good (for the parameter δ in the lemma statement). Let B denote the set of bad vertices and let $b := |B|$.

▷ **Claim 8.** There exists an independent set $I \subseteq B$ with at least $\frac{b}{2\delta}$ edges leading to $V \setminus U$.

Proof. We prove this claim using a probabilistic argument. Pick a random permutation σ of vertices in B and add each vertex $v \in B$ to an initially empty independent set $I = I_\sigma$ iff v appears before all its neighbors in B according to σ. Clearly, the resulting set I is an independent set inside B.

Let $E(I)$ denote the set of edges that connect vertices of I with vertices of $V \setminus U$. For any vertex $v \in B$, define a random variable $D_v \in \left\{ 0, \deg_{V \setminus U}(v) \right\}$ which takes value equal to

the external degree of v iff v is added to I. Clearly, $|E(I)| = \sum_{v \in V} D_v$. We have,

$$\mathbb{E}\,|E(I)| = \sum_{v \in B} \mathbb{E}\,[D_v] = \sum_{v \in B} \mathbf{Pr}\,(v \in I) \cdot \deg_{V \setminus U}(v)$$

$$= \sum_{v \in B} \frac{1}{\deg_B(v) + 1} \cdot \deg_{V \setminus U}(v)$$

(v is only chosen in I iff it is ranked first by σ among itself and its $\deg_B(v)$ neighbors)

$$\geq \sum_{v \in B} \frac{1}{\delta \cdot \deg_{V \setminus U}(v) + 1} \cdot \deg_{V \setminus U}(v) \qquad \text{(as } B \subseteq U \text{ and vertices in } B \text{ are all bad)}$$

$$\geq \sum_{v \in B} \frac{1}{2\delta \cdot \deg_{V \setminus U}(v)} \cdot \deg_{V \setminus U}(v) = \frac{b}{2\delta}.$$

$$\text{(as } \deg(v) \geq \tfrac{1}{\delta} \text{ and hence } \delta \cdot \deg_{V \setminus U}(v) \geq 1)$$

It follows that there exists a permutation σ for which the corresponding independent set $I = I_\sigma \subseteq B$ has at least $\frac{b}{2\delta}$ edges leading to $V \setminus U$, finalizing the proof. ◁

We next prove Lemma 7 using an argument akin to Claim 3 in the proof of Theorem 1. Let I be the independent set guaranteed by Claim 8. As at least $\frac{b}{2\delta}$ edges are going from I to $V \setminus U$, a double counting argument implies that there exists a vertex $v \in V \setminus U$ with degree to I satisfying:

$$\deg_I(v) \geq \frac{b}{2\delta \cdot |V \setminus U|} \geq \frac{b \cdot \tau(U)}{2n} = \frac{2b \cdot \beta(G)}{|U|}, \tag{1}$$

where the equality is by the choice of $\tau(U)$. Suppose towards a contradiction that $b > |U|/2$. This combined with Eq (1) implies that $\deg_I(v) > \beta(G)$. Since I is an independent set, it follows that $N(v)$ contains an independent set of size larger than $\beta(G)$, a contradiction. Hence $b \leq |U|/2$, and so at least half the vertices in U are δ-good, as required. ◀

We now use Lemma 7 to bound the expected number of iterations t in Algorithm 2. Let us define the random variables $X_1, \ldots, X_n$, where X_k denotes the number of iterations spent by the algorithm when $|U| = k$. Clearly, the total number of iterations $t = \sum_{k=1}^n X_k$. We use these random variables to bound the expected value of t in the following claim. The next claim then proves a concentration bound for t to obtain the high probability result.

▷ **Claim 9.** The number of iterations in Algorithm 2 is in expectation $\mathbb{E}\,[t] \leq 8\beta(G) \cdot n \log n$.

Proof. As stated above, $\mathbb{E}\,[t] = \sum_{k=1}^n \mathbb{E}\,[X_k]$ and hence it suffices to bound each $\mathbb{E}\,[X_k]$. Fix some $k \in [n]$ and consider the case when $|U| = k$. Recall the function $\delta(U)$ in Lemma 7. As $\delta(U)$ is only a function of size of k, we slightly abuse the notation and write $\delta(k)$ instead of $\delta(U)$ where k is the size of U.

By Lemma 7, at least half the vertices in U are $\delta(k)$-good. Hence, in each iteration, with probability at least half, we sample a $\delta(k)$-good vertex u from U. Conditioned on this event, either $\deg(u) < 1/\delta(k)$ which means this iteration succeeds with probability 1 or $\deg_U(u) \geq \delta(k) \cdot \deg_{V \setminus U}(u)$, and hence with probability at least $\delta(k)$ the sampled vertex v belongs to U and again this iteration succeeds. As a result, as long as U has not changed, each iteration has probability at least $\delta(k)/2$ to succeed.

By the above argument, X_k statistically dominates a Poisson distribution with parameter $\delta(k)/2$ and hence $\mathbb{E}[X_k] \leq 2/\delta(k)$. To conclude,

$$\mathbb{E}[t] = \sum_{k=1}^{n} \mathbb{E}[X_k] \leq \sum_{k=1}^{n} 2/\delta(k) = 8\beta(G) \cdot n \cdot \sum_{k=1}^{n} \frac{1}{k} \leq 8\beta(G) \cdot n \log n.$$

which finalizes the proof.   ◁

Claim 9 combined with Claim 5 is already enough to prove the expected runtime bound in Theorem 4. We now prove a concentration bound for t to obtain the high probability bound. We note that it seems possible to prove the following claim by using standard concentration inequalities; however doing so requires taking care of several boundary cases for the case when $|U|$ become $o(\log n)$, and hence we instead prefer to use the following direct and more transparent proof.

▷ **Claim 10.** The number of iterations in Algorithm 2 is $t = O(\beta(G) \cdot n \log n)$ with high probability.

Proof. Recall from the proof of Claim 9 that $t = \sum_{k=1}^{n} X_k$ and that each X_k statistically dominates a Poisson distribution with parameter $\delta(k)/2$ (as defined in Claim 9). Define $Y_1, \ldots, Y_n$ as independent random variables where Y_k is distributed according to exponential distribution with mean $\mu_k := \frac{2}{\delta(k)}$. For any $x \in \mathbb{R}^+$,

$$\mathbf{Pr}(X_k \geq x) \leq \left(\frac{\delta(k)}{2}\right)^x = \mathbf{Pr}(Y_k \geq x).$$

As such, the random variable $Y := \sum_{k=1}^{n} Y_k$ statistically dominates the random variable t for number of iterations. Moreover by Claim 9, $\mu := \mathbb{E}[Y] = 8\beta(G) \cdot n \log n$ (the equality for Y follows directly from the proof).

In the following, we prove that with high probability Y does not deviate from its expectation by much. The proof follows the standard moment generating function idea (used for instance in the proof of Chernoff-Hoeffding bound). Let $y \in \mathbb{R}^+$. For any $s > 0$,

$$\mathbf{Pr}(Y \geq y) = \mathbf{Pr}\left(e^{sY} \geq e^{sy}\right) \leq \frac{\mathbb{E}[e^{sY}]}{e^{sy}}, \tag{2}$$

where the inequality is simply by Markov bound. Additionally, since $Y = \sum_{k=1}^{n} Y_k$ and and Y_k's are independent, we have for any $s > 0$,

$$\frac{\mathbb{E}[e^{sY}]}{e^{sy}} = \frac{\mathbb{E}\left[e^{s \cdot \sum_{k=1}^{n} Y_k}\right]}{e^{sy}} = \frac{\prod_{k=1}^{n} \mathbb{E}[e^{s \cdot Y_k}]}{e^{sy}}. \tag{3}$$

Recall that for every $i \in [n]$, $\mathbb{E}[Y_k] = \mu_k$ and Y_k is distributed according to exponential distribution. Thus, for any $s < 1/\mu_k$,

$$\mathbb{E}[e^{sY_k}] = \int_{y=0}^{\infty} e^{sy} \cdot \mathbf{Pr}(Y_k = y)\, dy = \frac{1}{\mu_k} \int_{y=0}^{\infty} e^{sy} \cdot e^{-y/\mu_k} dy = \frac{1}{1 - s\mu_k}. \tag{4}$$

Recall that $\mu_k = 2/\delta(k)$ for every $k \in [n]$ and $\delta(1) < \delta(2) < \ldots < \delta(n)$ by definition. Pick $s^* = 1/2\mu_1$ and so $s^* < 1/\mu_k$ for all $k \in [n]$. By plugging in the bounds in Eq (4) for $s = s^*$

into Eq (3), we have,

$$\frac{\mathbb{E}\left[e^{s^*Y}\right]}{e^{s^*y}} = \frac{\prod_{k=1}^n \mathbb{E}\left[e^{s^*\cdot Y_k}\right]}{e^{s^*y}} = \prod_{k=1}^n \frac{e^{-s^*y}}{1-s^*\mu_k} \le e^{-s^*y}\cdot\exp\left(2\sum_{k=1}^n s^*\mu_k\right)$$
$$\text{(as } 1-x \ge e^{-2x} \text{ for } x \in (0,1/2])$$
$$= \exp\left(-s^*y+2s^*\mu\right) = \exp\left(-y/2\mu_1+\mu/\mu_1\right).$$

We now plug in this bound into Eq (2) with the choice of $y = 4\mu$ to obtain that,

$$\mathbf{Pr}\left(Y \ge 4\mu\right) \le \exp\left(-4\mu/2\mu_1+\mu/\mu_1\right) = \exp\left(-\mu/\mu_1\right) = \exp\left(-\log n\right) = 1/n,$$

where we used the fact that $\mu/\mu_1 \ge \log n$. This means that with high probability, Y is only 4 times larger than its expectation, finalizing the proof. ◁

The high probability bound in Theorem 4 now follows from Claim 5 and Claim 10, concluding the whole proof of this theorem.

Unknown $\beta(G)$

We next show that our algorithm can be easily adjusted to the case when $\beta(G)$ is unknown. The idea is simply to "guess" $\beta(G)$ in powers of two, starting from $\beta = 2$ and ending at $\beta = n$, and each time to (sequentially) run Algorithm 2 under the assumption that $\beta(G) = \beta$. For each choice of β, we shall only run the algorithm for at most $t = O(n \log n \cdot \beta(G))$ iterations (where the constant hiding in the O-notation should be sufficiently large, in accordance with that in the proof of Claim 10) and if at the end of a run the set U in the algorithm has not become empty, we start a new run from scratch with the next (doubled) value of β. (For $\beta = n$, we do not terminate the algorithm prematurely and instead run it until U is empty.)

By Theorem 4, for the *first* choice of β for which $\beta \ge \beta(G)$, the algorithm must terminate with high probability within $O(n \log n \cdot \beta(G))$ time (as $\beta \le 2\beta(G)$ also). Moreover, the runtime of every previous run is bounded *deterministically* by $O(n \log n \cdot \beta(G))$. Consequently, the total runtime is

$$O(n \log n) \cdot \sum_{\{2^i | 2^i \le 2\beta(G)\}} 2^i = O(n \log n \cdot \beta(G)),$$

where this bound holds with high probability. In this way we get an algorithm that uses no prior knowledge of $\beta(G)$ and achieves the same asymptotic performance as Algorithm 2.

5 A Lower Bound for Deterministic Maximal Matching

We prove that randomization is necessary to obtain a sublinear time algorithm for MM even on graphs with bounded neighborhood independence.

▶ **Theorem 11.** *Any deterministic algorithm that finds a maximal matching in every given graph G with neighborhood independence $\beta(G) = 2$ (known to the algorithm) presented in the adjacency array representation requires $\Omega(n^2)$ time.*

For every integer $n = 8k$ for $k \in \mathbb{N}$, we define $\mathcal{G}_n$ as the family of all graphs obtained by removing a perfect matching from a clique K_n on n vertices. For a graph G in $\mathcal{G}_n$ we refer to the removed perfect matching of size $5k$ as the *non-edge matching* of G and denote it by $\overline{M}(G)$. Clearly, every graph in $\mathcal{G}_n$ has neighborhood independence $\beta(G) = 2$. Moreover, any MM in G can have at most 2 unmatched vertices.

Let $\mathcal{A}$ be a deterministic algorithm for computing an MM on every graph in $\mathcal{G}_n$. We prove Theorem 11 by analyzing a game between $\mathcal{A}$ and an adaptive adversary that answers the probes of $\mathcal{A}$ to the adjacency array of the graph. In particular, whenever $\mathcal{A}$ probes a *new* entry of the adjacency array for some vertex $v \in V$, we can think of $\mathcal{A}$ making a *query* $Q(v)$ to the adversary, and the adversary outputs a vertex v that had not been so far revealed in the neighborhood of u (as degree of all vertices in G is exactly $n-1$, $\mathcal{A}$ knows the degree of all vertices and we assume it never queries a vertex u more than $n-1$ times).

We now show that there is a strategy for the adversary to answer the queries of $\mathcal{A}$ in a way that ensures $\mathcal{A}$ needs to make $\Omega(n^2)$ queries before it can output an MM of the graph.

Adversary's Strategy. The adversary picks an arbitrary set of $2k$ vertices D referred to as *dummy* vertices. We refer to remaining vertices as *core* vertices and denote them by $C := V \setminus D$. The adversary also fixes a non-edge matching of size k between vertices in D, denoted by $\overline{M}_D$. The non-edge matching of G consists of $\overline{M}_D$ and a non-edge matching of size $4k$ between vertices in C, denoted by $\overline{M}_C$, which unlike $\overline{M}_D$ is constructed adaptively by the adversary. We assume $\mathcal{A}$ knows the partitioning of V into D and C, as well as the non-edge matching $\overline{M}_D$. Hence, the only missing information to $\mathcal{A}$ is identity of $\overline{M}_C$.

To answer a query $Q(u)$ for a dummy vertex $u \in D$, the adversary simply returns any arbitrary vertex in V (not returned so far as an answer to $Q(u)$) except for the pair of u in $\overline{M}_D$ which cannot be neighbor to u. To answer the queries $Q(w)$ for core vertices $w \in C$, the adversary maintains a partitioning of C into C_{used} and C_{free}. Initially all core vertices belong to C_{free} and hence C_{used} is empty. Throughout we only move vertices from C_{free} to C_{used}. The adversary also maintains a counter for every vertex in C_{free} on how many times they have been queried so far. Whenever a vertex $w \in C_{\mathsf{free}}$ is queried, as long as this vertex has been queried at most k times, the adversary returns an arbitrary dummy vertex u from D as the answer to $Q(w)$ (which is possible because of size of D is k). Once a vertex $w \in C_{\mathsf{free}}$ is queried for its $(k+1)$-th time, we pick another vertex w' from C_{free} also, add the pair (w, w') to the non-edge matching $\overline{M}_C$ and move both w and w' to C_{used}, and then answer $Q(w)$ for the case $w \in C_{\mathsf{used}}$ as described below.

Recall that for any vertex $w \in C_{\mathsf{used}}$, by construction, there is another fixed vertex w' in C_{used} (joined at the same time with w) where $(w, w') \in \overline{M}_C$. For any query for $w \in C_{\mathsf{used}}$, the adversary answers $Q(w)$ by returning an arbitrary vertex from $C \setminus \{w'\}$. This concludes the description of the strategy of the adversary.

We have the following basic claim regarding the correctness of the adversary's strategy.

▷ **Claim 12.** The answers returned by the adversary for any sequence of queries are always consistent with at least one graph G in $\mathcal{G}_n$.

Proof. We can append the current sequence of queries by the sequence that ensures all vertices are queried $n-1$ times. Thus, the adversary would eventually construct the whole non-edge matching $\overline{M}_C$ also. There exists a unique graph G in $\mathcal{G}_n$ where $\overline{M}(G) = \overline{M}_D \cup \overline{M}_C$, hence proving the claim (note that before appending the sequence, there can be more than one graph consistent with the original sequence). ◁

We now prove the following lemma which is the key step in the proof of Theorem 11.

▶ **Lemma 13.** *Suppose $\mathcal{A}$ makes at most k^2 queries to the adversary and outputs a matching M using only these queries. Then, there exists some graph G in $\mathcal{G}_n$ where G is consistent with the answers returned by the adversary to $\mathcal{A}$ and $\overline{M}(G) \cap M \neq \emptyset$.*

Proof. Since $\mathcal{A}$ makes at most k^2 queries, there can only be $2k$ vertices in C_{used} by the time the algorithm finishes its queries (as each pair of vertices in C_{used} consumed k queries). Consider the maximal matching M. There are at most $2k$ edges of M that are incident on C_{used} and k more edges incident on D. This implies that at most $3k$ vertices in C_{free} are matched to vertices outside C_{free}. As $|C_{\mathsf{free}}| \geq 5k$, it means that there are at least $2k$ vertices in C_{free} that are not matched by M to vertices outside C_{free}. However, as M is maximal, and since G is in $\mathcal{G}_n$, there should be at least $k-1$ edges in M between these vertices in C_{free}. As the adversary has not committed to the non-edge matching $\overline{M}_C$ entirely at this point and this non-edge matching pairs vertices in C_{free} to each other, the adversary can simply let $\overline{M}_C$ contains M and still obtain a graph G in $\mathcal{G}_n$. For this graph, $\overline{M}(G) \cap M \neq \emptyset$, finalizing the proof.   ◀

Theorem 11 now follows immediately from Lemma 13, as it state that unless the algorithm makes $\Omega(n^2)$ queries, there always exists at least one graph in $\mathcal{G}_n$ for which the output matching of the algorithm is not feasible. As graphs in $\mathcal{G}_n$ all have $\beta(G) = 2$, we obtain the final result.

References

1 Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms*, 7(4):567–583, 1986.

2 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, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1132–1139, 2012.

3 Jonathan Aronson, Martin E. Dyer, Alan M. Frieze, and Stephen Suen. Randomized Greedy Matching II. *Random Struct. Algorithms*, 6(1):55–74, 1995.

4 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear Algorithms for $(\Delta + 1)$ Vertex Coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019.*, pages 767–786, 2019. `arXiv:1807.08886`.

5 Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. 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*, pages 815–826, 2018.

6 Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully Dynamic Maximal Independent Set with Sublinear in n Update Time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1919–1936, 2019.

7 Leonid Barenboim and Michael Elkin. Distributed deterministic edge coloring using bounded neighborhood independence. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 129–138, 2011.

8 Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2013.

9 Leonid Barenboim, Michael Elkin, and Fabian Kuhn. Distributed (Delta+1)-Coloring in Linear (in Delta) Time. *SIAM J. Comput.*, 43(1):72–95, 2014.

10 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The Locality of Distributed Symmetry Breaking. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 321–330, 2012.

11 Surender Baswana, Manoj Gupta, and Sandeep Sen. Fully Dynamic Maximal Matching in O(log n) Update Time. *SIAM J. Comput.*, 44(1):88–113, 2015.

12   Soheil Behnezhad, MohammadTaghi Hajiaghayi, and David G. Harris. Exponentially Faster Massively Parallel Maximal Matching. *CoRR*, abs/1901.03744, 2019.

13   Yair Caro. New results on the independence number. Technical report, Technical Report, Tel-Aviv University, 1979.

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

15   Graham Cormode, Jacques Dark, and Christian Konrad. Independent Sets in Vertex-Arrival Streams. *CoRR*, abs/1807.08331, 2018.

16   Martin E. Dyer and Alan M. Frieze. Randomized Greedy Matching. *Random Struct. Algorithms*, 2(1):29–46, 1991.

17   Ralph Faudree, Evelyne Flandrin, and Zdeněk Ryjáček. Claw-free graphs?a survey. *Discrete Mathematics*, 164(1-3):87–147, 1997.

18   Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. `doi:10.1016/j.tcs.2005.09.013`.

19   Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. Deterministic Distributed Edge-Coloring via Hypergraph Maximal Matching. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 180–191, 2017.

20   Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local Conflict Coloring. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 625–634, 2016.

21   Mohsen Ghaffari. An Improved Distributed Algorithm for Maximal Independent Set. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 270–277, 2016.

22   Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 129–138, 2018.

23   Mohsen Ghaffari and Jara Uitto. Sparsifying Distributed Algorithms with Ramifications in Massively Parallel Computation and Centralized Local Computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1636–1653, 2019.

24   Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect Matchings in\˜ O (n^{1.5}) Time in Regular Bipartite Graphs. *arXiv preprint*, 2009. `arXiv:0902.1617`.

25   Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings via uniform sampling in regular bipartite graphs. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 11–17, 2009.

26   Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings in o(n log n) time in regular bipartite graphs. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 39–46, 2010.

27   Bjarni V. Halldórsson, Magnús M. Halldórsson, Elena Losievskaja, and Mario Szegedy. Streaming Algorithms for Independent Sets in Sparse Hypergraphs. *Algorithmica*, 76(2):490–501, 2016.

28   Magnús M. Halldórsson. Wireless Scheduling with Power Control. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, pages 361–372, 2009.

29   Magnús M. Halldórsson and Christian Konrad. Distributed Large Independent Sets in One Round on Bounded-Independence Graphs. In *Distributed Computing - 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings*, pages 559–572, 2015.

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

31   Michal Hanckowiak, Michal Karonski, and Alessandro Panconesi. On the Distributed Complexity of Computing Maximal Matchings. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA.*, pages 219–225, 1998.

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

33   Richard M. Karp and Avi Wigderson. A Fast Parallel Algorithm for the Maximal Independent Set Problem. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 266–272, 1984.

34   Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 300–309, 2004.

35   Fabian Kuhn, Tim Nieberg, Thomas Moscibroda, and Roger Wattenhofer. Local approximation schemes for ad hoc and sensor networks. In *Proceedings of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing, Cologne, Germany, September 2, 2005*, pages 97–103, 2005.

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

37   Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 85–94, 2011. doi:10.1145/1989493.1989505.

38   Christoph Lenzen and Roger Wattenhofer. MIS on trees. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 41–48, 2011.

39   Reut Levi, Ronitt Rubinfeld, and Anak Yodpinyanee. Local Computation Algorithms for Graphs of Non-constant Degrees. *Algorithmica*, 77(4):971–994, 2017.

40   Nathan Linial. Locality in Distributed Graph Algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

41   Michael Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 1–10, 1985.

42   Zevi Miller and Dan Pritikin. On randomized greedy matchings. *Random Struct. Algorithms*, 10(3):353–383, 1997.

43   Ofer Neiman and Shay Solomon. Simple deterministic algorithms for fully dynamic maximal matching. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 745–754, 2013.

44   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, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1123–1131, 2012.

45   Krzysztof Onak, Baruch Schieber, Shay Solomon, and Nicole Wein. Fully Dynamic MIS in Uniformly Sparse Graphs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 92:1–92:14, 2018.

46   Alessandro Panconesi and Aravind Srinivasan. On the Complexity of Distributed Network Decomposition. *J. Algorithms*, 20(2):356–374, 1996.

47   Matthias Poloczek and Mario Szegedy. Randomized Greedy Algorithms for the Maximum Matching Problem with New Analysis. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 708–717, 2012.

48 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast Local Computation Algorithms. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 223–238, 2011.

49 Johannes Schneider and Roger Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*, pages 35–44, 2008.

50 Shay Solomon. Fully Dynamic Maximal Matching in Constant Update Time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 325–334, 2016.

51 VK Wei. A lower bound on the stability number of a simple graph. Technical report, Bell Laboratories Technical Memorandum 81-11217-9, Murray Hill, NJ, 1981.