Simple Sublinear Algorithms for $(\Delta+1)$ Vertex Coloring via Asymmetric Palette Sparsification

Sepehr Assadi* Helia Yazdanyar†

Abstract

The **palette sparsification theorem (PST)** of Assadi, Chen, and Khanna (SODA 2019) states that in every graph G with maximum degree Δ, sampling a list of $O(\log n)$ colors from $\{1,\ldots,\Delta+1\}$ for every vertex independently and uniformly, with high probability, allows for finding a $(\Delta+1)$ vertex coloring of G by coloring each vertex only from its sampled list. PST naturally leads to a host of **sublinear algorithms** for $(\Delta+1)$ vertex coloring, including in semi-streaming, sublinear time, and MPC models, which are all proven to be nearly optimal, and in the case of the former two are the only known sublinear algorithms for this problem.

While being a quite natural and simple-to-state theorem, PST suffers from two drawbacks. Firstly, all its known proofs require technical arguments that rely on sophisticated graph decompositions and probabilistic arguments. Secondly, finding the coloring of the graph from the sampled lists in an efficient manner requires a considerably complicated algorithm.

We show that a natural *weakening* of PST addresses both these drawbacks while still leading to sublinear algorithms of similar quality (up to polylog factors). In particular, we prove an **asymmetric palette sparsification theorem (APST)** that allows for list sizes of the vertices to have different sizes and only bounds the *average* size of these lists. The benefit of this weaker requirement is that we can now easily show the graph can be $(\Delta+1)$ colored from the sampled lists using the standard greedy coloring algorithm. This way, we can recover nearly-optimal bounds for $(\Delta+1)$ vertex coloring in all the aforementioned models using algorithms that are much simpler to implement and analyze.

*(sepehr@assadi.info) Cheriton School of Computer Science, University of Waterloo. Supported in part by a Sloan Research Fellowship, an NSERC Discovery Grant, a University of Waterloo startup grant, and a Faculty of Math Research Chair grant.

†(hyazdanyar@uwaterloo.ca) Cheriton School of Computer Science, University of Waterloo.

1 Introduction

Let $G = (V, E)$ be any n-vertex graph with maximum degree Δ. A basic graph theory fact is that vertices of G can be colored with $\Delta + 1$ colors such that no edge is monochromatic. This can be done using a textbook greedy algorithm: iterate over vertices of G in any arbitrary order and for each vertex greedily find a color that has not been used in its neighborhood, which is guaranteed to exist by the pigeonhole principle. This algorithm is quite simple and efficient as it runs in linear time and space. But, can we design even more efficient algorithms?

Assadi, Chen, and Khanna [5] addressed this question by designing various *sublinear* algorithms for $(\Delta + 1)$ vertex coloring. These are algorithms whose resource requirements are significantly smaller than their input. Canonical examples include (*a*) *graph streaming* algorithms [12] that process graphs by making one pass over their edges and using a limited space; (*b*) *sublinear time* algorithms [10] that process graphs by querying its adjacency list/matrix and using limited *time*; or (*c*) *Massively Parallel Computation (MPC)* algorithms [23] that process graphs in synchronous rounds and a distributed manner with limited *communication* (see Section 2.1).

The key contribution of [5] was proving the following (purely combinatorial) theorem about $(\Delta + 1)$ coloring, termed the **palette sparsification theorem (PST)**.

THEOREM 1.1. (**Palette Sparsification Theorem (PST)** [5]) *For any graph $G = (V, E)$ with maximum degree Δ, if we sample $O(\log n)$ colors $L(v)$ for each vertex $v \in V$ independently and uniformly at random from the colors $\{1, 2, \ldots, \Delta + 1\}$, then, with high probability, G can be colored so that every vertex v chooses its color from its own list $L(v)$.*

The authors in [5] used PST to design sublinear algorithms in all the models above using a "sparsifying" nature of this result: Firstly, it is easy to see that to color G from the sampled lists, we can ignore all edges $(u, v) \in E$ where $L(u) \cap L(v) = \emptyset$ since they never create any conflict under any coloring. Secondly, for any edge $(u, v) \in E$, the probability that it still remains conflicting—the probability that two independently chosen $O(\log n)$-lists from $(\Delta+1)$ colors intersect—is $O(\log^2(n)/\Delta)$. Combining these with the upper bound of $n\Delta/2$ on the number of edges implies that in expectation there are only $O(n \log^2(n))$ "important" edges that the algorithm should consider. The sublinear algorithms can now be obtained from this in a simple way[1].

Since its introduction [5], PST and its variants and generalizations have been studied in sublinear algorithms [4, 6–9], distributed computing [13, 14, 17], and discrete mathematics [2, 3, 19, 21, 22].

Despite its long list of applications, and perhaps even due to its generality, PST suffers from two drawbacks. Firstly, the proof of PST is considerably complicated and technical; it goes through the so-called sparse-dense decomposition of graphs due to [24] (and the variant proved in [5] itself, building on [18]), and then a detailed three-phase approach for coloring each part in the decomposition differently using various probabilistic and random graph theory arguments[2]. Secondly, PST, as stated, is a purely combinatorial theorem and not an algorithmic one, and thus to obtain the coloring of G, one needs to run a rather complicated and non-greedy post-processing algorithm to find the coloring of G from the sampled lists (given also the decomposition of the graph; see [5]).

Our contribution. We show that introducing a simple *asymmetry* in the definition of PST leads to the same colorability guarantee but this time using a much simpler proof and algorithm.

Asymmetric Palette Sparsification Theorem (APST). For any graph $G = (V, E)$ with maximum degree Δ, there is a distribution on list-sizes $\ell : V \to \mathbb{N}$ (depending only on vertices V and not edges E) such that an average list size is deterministically $O(\log^2(n))$ and the following holds. With high probability, if we sample $\ell(v)$ colors $L(v)$ for each vertex $v \in V$ independently and uniformly at random from colors $\{1, 2, \ldots, \Delta + 1\}$, then, with high probability, the greedy coloring algorithm that processes vertices in the increasing order of list-sizes finds a proper coloring of G by coloring each v from its own list $L(v)$.

[1] For instance, for a streaming algorithm, first sample all the $O(n \log n)$ colors, and during the stream whenever an edge arrives, see if it is an important edge and if so store it. At the end, use Theorem 1.1 to color the graph using these stored edges. This leads to a single-pass streaming algorithm with $O(n \log^2(n))$ space for $(\Delta + 1)$ coloring.

[2] All known proofs of PST for $(\Delta + 1)$ coloring (or so-called (deg +1)-(list) coloring) [2, 5, 15, 17, 21] follow this decomposition plus three-phase approach and it was even pointed out in [21] that: "something of this type [...] seems more or less unavoidable".

The benefit of our APST compared to the original PST of [5] (and all its other alternative variants in [2, 15, 17, 21]) is twofold: it admits a *significantly* simpler proof, while also allowing for coloring from the sampled list using the standard greedy algorithm itself. At the same time, it is also weaker than the original PST on two fronts: it requires $O(\log^2(n))$ list sizes per vertex as opposed to $O(\log n)$ (which is similar to [15, 17]) but much more importantly, it allows the vertices to have much larger list sizes in the worst-case and only bounds the *average* list sizes (this is different than all prior work on PST, and was inspired by the recent work of [16] on the communication complexity of $(\Delta+1)$ coloring). We note that asymmetric list sizes are *necessary* if we like to color the graph *greedily* from the sampled colors[3].

Finally, APST allows for recovering the same type of sublinear algorithms as in [5] with only an extra $\text{polylog}(n)$ overhead in their resources. In particular, we obtain the following sublinear algorithms for $(\Delta+1)$ vertex coloring:

- A randomized **graph streaming** algorithm with $\widetilde{O}(n)$ space[4] and a single pass over the input.
- A randomized **sublinear time** algorithm with $\widetilde{O}(n^{1.5})$ time, given (non-adaptive) query access to both adjacency list and matrix of the input (also called the general query model).
- A randomized **MPC** algorithm with $\widetilde{O}(n)$ memory per machine and $O(1)$ rounds.

The number of passes in the streaming algorithms as well as the rounds in the MPC one are clearly optimal. It was also proven in [5] that the runtime of the sublinear time algorithm and the space of the streaming algorithm are nearly optimal up to $\text{polylog}\,(n)$ factors (for the streaming algorithm, storing the coloring itself requires this much space anyway). These constitute the simplest known sublinear algorithms for $(\Delta+1)$ vertex coloring.[5]

In conclusion, we find our APST as a more "algorithmic friendly" version of PST that still allows for recovering nearly optimal sublinear algorithms for $(\Delta+1)$ coloring. In addition, we hope its proof can act as a gentle warmup and introduction to the original PST itself.

2 Preliminaries

A **hypergeometric** random variable with parameters N, K, and M is a discrete random variable in $\mathbb{N}$ distributed as follows: we have N elements, K of them are marked as 'good', and we sample M elements uniformly at random and *without* replacement and count the number of good samples. We use a standard result on the concentration of hypergeometric random variables.

PROPOSITION 2.1. (CF. [20, THEOREM 2.10]) *Suppose X is a hypergeometric random variable with parameters N, K, M and thus the expectation $\mathbb{E}[X] = M \cdot K/N$. Then, for any $t \geqslant 0$*

$$\Pr(X \leqslant \mathbb{E}[X] - t) \leqslant \exp\left(-\frac{t^2}{2\,\mathbb{E}[X]}\right).$$

2.1 Sublinear Models Considered in this Paper

For completeness, we present a quite brief definition of each of the sublinear algorithms models that we consider. We refer the interested reader to [5] for more background on these models.

Graph streaming. In this model, the input graph $G = (V, E)$ is presented to the algorithm as a stream of its edges ordered in some arbitrary manner. The algorithm makes a single pass (or sometimes multiple ones) over this stream while using $\widetilde{O}(n)$ memory and at the end of the stream, outputs a solution to the problem on G.

[3]Consider coloring a $(\Delta+1)$-clique. When coloring the last vertex v of the clique in the greedy algorithm, there is only one color that can be assigned to v but to ensure this color is sampled by v even with a constant probability, we need $L(v)$ to have size $\Omega(\Delta)$. Our APST addresses this by allowing vertices that are colored later in the greedy algorithm to also sample more colors, while earlier vertices need to stick with smaller list sizes.

[4]Throughout, we use $\widetilde{O}(f) := O(f \cdot \text{poly}\log(f))$ to suppress polylog factors.

[5]To our knowledge, no other streaming nor sublinear time algorithms beside [5] have been developed for this problem but for MPC algorithms, [9, 11] have subsequently presented other algorithms for this problem.

Sublinear time. In this model, the input graph $G = (V, E)$ is presented to the algorithm via query access to its adjacency list and matrix or alternatively speaking, using degree-, neighbor-, and pair-queries. The algorithm can make its queries adaptively or non-adaptively and then at the end outputs a solution to the problem on G.

MPC. In this model, the input graph $G = (V, E)$ is originally partitioned across multiple machines, each with $\widetilde{O}(n)$ memory. Computation happens in synchronous rounds wherein each machine can send and receive $\widetilde{O}(n)$-size messages. After the last round, one designated machine outputs a solution to the problem on G.

3 Asymmetric Palette Sparsification

We present our asymmetric palette sparsification theorem and its simple proof in this section.

THEOREM 3.1. (**Asymmteric Palette Sparsification Theorem**) *Let $G = (V, E)$ be any n-vertex graph with maximum degree Δ. Sample a random permutation $\pi : V \to [n]$ uniformly and define*

$$\ell(v) := \min\left(\Delta + 1\,,\, \frac{40\, n \ln n}{\pi(v)}\right),$$

for every $v \in V$ as the size of the lists of colors to be sampled for vertex v. Then,

- **List sizes:** *Deterministically, $\sum_{v \in V} \ell(v) = O(n \log^2(n))$, and for any fixed vertices $u \neq v \in V$,*

$$\mathbb{E}\left[\ell(v)\right] = O(\log^2(n)) \quad \textit{and} \quad \mathbb{E}\left[\ell(u) \cdot \ell(v)\right] = O(\log^4(n)).$$

- **Colorability:** *If for every vertex $v \in V$, we sample a list $L(v)$ of $\ell(v)$ colors from $[\Delta + 1]$ uniformly and independently, then, with high probability (over the randomness of ℓ and sampled lists) the greedy algorithm that iterates over vertices $v \in V$ in the increasing order of $\ell(v)$ finds a proper list-coloring of G from the lists $\{L(v) \mid v \in V\}$.*

We establish the advertised properties in the theorem in the following claims.

CLAIM 3.1. (**List sizes**) *Deterministically, $\sum_{v \in V} \ell(v) = O(n \ln^2(n))$, and for all $u \neq v \in V$,*

$$\mathbb{E}\left[\ell(v)\right] = O(\ln^2(n)) \qquad \textit{and} \qquad \mathbb{E}\left[\ell(u) \cdot \ell(v)\right] = O(\ln^4(n)).$$

Proof. By the definition of $\ell(v)$,

$$\sum_{v \in V} \ell(v) \leqslant \sum_{v \in V} \frac{40\, n \ln n}{\pi(v)} = \sum_{i=1}^{n} \frac{40\, n \ln n}{i} = O(n \ln^2 n),$$

by the sum of the Harmonic series. This proves the deterministic bound on the total size of the lists as well as the first inequality for the expected size of each list, given the distribution of list sizes is symmetric across vertices. For the second inequality, we have

$$\mathbb{E}\left[\ell(u) \cdot \ell(v)\right] \leqslant \sum_{i \neq j} \Pr(\pi(v) = i \wedge \pi(u) = j) \cdot \frac{(40\, n \ln(n))^2}{i \cdot j} = \frac{1600\, n^2 \ln^2(n)}{\binom{n}{2}} \cdot \sum_{i=1}^{n} \sum_{j=1}^{i-1} \frac{1}{i \cdot j} = O(\ln^4(n)),$$

again, by the sum of the Harmonic series. This concludes the proof. ∎

Establishing the colorability property is the main part of the proof. We first need some notation. Iterating over vertices in the increasing order of $\ell(\cdot)$ is the same as the decreasing order of $\pi(\cdot)$. With this in mind, for any vertex $v \in V$, define $N_\pi^<(v)$ as the neighbors u of v with $\pi(u) < \pi(v)$, namely, the ones that are processed *after* v by the greedy algorithm. Let $\deg_\pi^<(v) := |N_\pi^<(v)|$.

An easy observation is that when coloring any vertex v by the greedy algorithm, there are still at least $\deg_\pi^<(v) + 1$ colors not used in the neighborhood of v. Thus, lower bounding $\deg_\pi^<(v)$ allows us to later prove that $L(v)$ contains an available color for v with high probability. We only need such a bound for a subset of vertices captured in the following claim.

CLAIM 3.2. *For all $v \in V$ with $\pi(v) > 40n\ln(n)/(\Delta+1)$ and $\deg(v) \geqslant 3\Delta/4$,*

$$\Pr\left(\deg_\pi^<(v) < \frac{\Delta \cdot \pi(v)}{4n}\right) \leqslant n^{-2.5}.$$

Proof. Fix any vertex v as above and condition on $\pi(v) = i+1$ for some $i \in \{0, 1, \ldots, n-1\}$. We pick an arbitrary subset of $3\Delta/4$ neighbors of v and for $j \in [3\Delta/4]$, define $X_j \in \{0,1\}$ as the indicator random variable which is 1 iff the j-th neighbor of v appears in the first i vertices of π. For X being the sum of these X_j-variables, we have,

$$\mathbb{E}\left[\deg_\pi^<(v) \mid \pi(v) = i+1\right] \geqslant \mathbb{E}[X] = \sum_{j=1}^{3\Delta/4} \mathbb{E}[X_j] = \frac{3\Delta}{4} \cdot \frac{i}{n-1},$$

as each neighbor of v appears in the first i position of π with probability $i/(n-1)$ conditioned on $\pi(v) = i+1$. Random variable X is distributed as a hypergeometric random variable with parameters $N = n-1$, $K = i$, and $M = 3\Delta/4$. Thus, by Proposition 2.1 for $t := \Delta \cdot i/(2n)$,

$$\begin{aligned}\Pr\left(X \leqslant \frac{\Delta \cdot i}{4n}\right) &\leqslant \Pr\left(X \leqslant \mathbb{E}[X] - t\right) \leqslant \exp\left(-\frac{t^2}{2\,\mathbb{E}[X]}\right) \\ &= \exp\left(-\frac{\Delta^2 \cdot i^2 \cdot 4 \cdot (n-1)}{4 \cdot n^2 \cdot 3 \cdot \Delta \cdot i}\right) \leqslant \exp\left(-\frac{\Delta \cdot i}{6 \cdot n}\right) \leqslant \exp\left(-\frac{40\ln(n)}{12}\right) \ll n^{-3},\end{aligned}$$

where in the second line, we used the values of N, K, M, and t first, then the bound $n > 1$, and subsequently the lower bound on i in the claim statement combined with $\Delta \geqslant 1$. Given that $\deg_\pi^<(v) \geqslant X$ and the above bound holds for all choices of $\pi(v)$, we obtain the proof. ∎

CLAIM 3.3. (**Colorability**) *With high probability, when coloring each vertex $v \in V$ in the greedy algorithm, at least one of the colors sampled in $L(v)$ has not been used in the neighborhood of v; that is, the greedy algorithm can color this vertex.*

Proof. We condition on the choice of π and by union bound obtain that with high probability, the complement of the event in Claim 3.2 holds for all vertices. For any vertex $v \in V$, we say a color is **available** to v iff it is not assigned to any neighbor of v by the time we process v in the greedy algorithm. Let $a(v)$ denote the number of available colors and note that $a(v) \geqslant \deg_\pi^<(v) + 1 \geqslant 1$.

In the following, we fix a vertex $v \in V$ and further condition on the randomness of $L(u)$ for all vertices u with $\pi(u) > \pi(v)$. The conditionings so far fix the set of available colors and the values of $a(v)$ and $\ell(v)$, but $L(v)$ is still a random $\ell(v)$-subset of the colors.

If $\pi(v) \leqslant 40n\ln(n)/(\Delta+1)$, we have $\ell(v) = \Delta+1$ which means all of the available colors counted in $a(v)$ are sampled in $L(v)$, thus proving the claim for this vertex.

For the remaining vertices, we have,

$$\Pr\left(\text{no available color of } v \text{ is sampled in } L(v)\right) \leqslant (1 - \frac{a(v)}{\Delta+1})^{\ell(v)} \leqslant \exp\left(-\frac{a(v) \cdot \ell(v)}{\Delta+1}\right).$$

Now, if $\deg(v) < 3\Delta/4$, we have $a(v) \geqslant \Delta/4$ and we always have $\ell(v) \geqslant 40\ln n$, thus, in this case,

$$\Pr\left(\text{no available color of } v \text{ is sampled in } L(v)\right) \leqslant \exp\left(-\frac{\Delta}{4} \cdot \frac{40\ln n}{\Delta+1}\right) \leqslant n^{-5}.$$

Otherwise, $\deg(v) \geqslant 3\Delta/4$ and we can use the bounds of Claim 3.2 and have,

$$\Pr\left(\text{no available color of } v \text{ is sampled in } L(v)\right) \leqslant \exp\left(-\frac{\Delta \cdot \pi(v)}{4n} \cdot \frac{40n\ln n}{\pi(v)} \cdot \frac{1}{\Delta+1}\right) \leqslant n^{-5}.$$

Thus, in both cases, with high probability, we can color v from $L(v)$ in the greedy algorithm. Taking a union bound over all vertices concludes the proof. ∎

Theorem 3.1 follows immediately from Claim 3.1 and Claim 3.3.

REMARK 3.4. *We note that almost the same exact proof of Theorem 3.1 implies the following stronger result also: we can assume each vertex $v \in V$ starts with a list $S(v)$ of $\deg(v)+1$ arbitrary colors which can be different across the vertices and we sample $\ell(v)$ colors $L(v)$ from $S(v)$ instead. Then, with high probability, the greedy algorithm can still color G from the sampled lists (the proof is virtually identical since the greedy algorithm also works for this problem). In other words, the $(\deg+1)$-list coloring version of APST also holds (the PST version of this result is also proven in [17]; see also [2]). We omit repeating this proof here.*

4 Sublinear Algorithms from Asymmetric Palette Sparsification

We now show how our asymmetric palette sparsification theorem in Theorem 3.1 can be used to obtain sublinear algorithms for $(\Delta+1)$ vertex coloring. These algorithms are more or less identical to the (exponential-time) sublinear algorithms of [5] from their original palette sparsification theorem and we claim no novelty in this part[6]. Instead, we merely point out how the "asymmetry" in list-sizes in Theorem 3.1 does not weaken the performance of the resulting sublinear algorithms beyond some polylog(n)-factors, but instead leads to much simpler post-processing algorithms for finding the coloring of the graph from the sampled lists.

THEOREM 4.1. *There exists randomized sublinear algorithms that given any graph $G = (V, E)$ with maximum degree Δ with high probability output a $(\Delta+1)$ vertex coloring of G using:*

- **Graph streaming:** *a single pass over the edges of G in any order and $\widetilde{O}(n)$ space;*
- **Sublinear time:** *$\widetilde{O}(n^{1.5})$ time and non-adaptive queries to adjacency list and matrix of G;*
- **Massively parallel computation (MPC):** *$O(1)$ rounds with machines of $\widetilde{O}(n)$ memory.*

As stated earlier, the proof of Theorem 4.1 follows the same exact strategy as the (exponential-time) sublinear algorithms of [5]. To do so, we need the following definition.

Conflict graphs. Let $G = (V, E)$ be any graph with maximum degree Δ and $\mathcal{L} := \{L(v) \mid v \in V\}$ be a set of lists of colors sampled for vertices of G according to the distribution of Theorem 3.1. We define the **conflict graph** $G_{\mathcal{L}} = (V, E_{\mathcal{L}})$ of G and $\mathcal{L}$ as the spanning subgraph of G consisting of all edges $(u, v) \in E$ such that the sampled lists $L(u)$ and $L(v)$ intersect with each other.

The next observation allows us to use conflict graphs in our sublinear algorithms as a proxy to the graph G.

OBSERVATION 4.1. *The greedy algorithm in Theorem 3.1 outputs the same exact coloring when run over the conflict graph $G_{\mathcal{L}}$ instead of the original graph G.*

Proof. The only edges that affect the greedy algorithm of Theorem 3.1 are edges $(u, v) \in E$ such that $L(u) \cap L(v)$ is non-empty. These edges are identical in G and $G_{\mathcal{L}}$. ∎

The following easy claim also allows us to bound the size of the conflict graphs.

CLAIM 4.2. *The list of colors $\mathcal{L}$ consists of $O(n \log^2 n)$ colors deterministically and the expected number of edges in $G_{\mathcal{L}} = (V, E_{\mathcal{L}})$ is $\mathbb{E}\,|E_{\mathcal{L}}| = O(n \log^4 n)$.*

Proof. From Claim 3.1, we already know that $\mathcal{L}$ consists of $O(n \log^2 n)$ colors deterministically. For the second part, for any edge (u, v), we have,

$$\Pr\left((u,v) \text{ is in } E_{\mathcal{L}}\right) \leqslant \mathop{\mathbb{E}}_{\ell(u),\ell(v)}\left[\Pr\left(L(u) \cap L(v) \neq \emptyset \mid \ell(u), \ell(v)\right)\right] \leqslant \mathbb{E}\left[\frac{\ell(u) \cdot \ell(v)}{\Delta+1}\right] = \frac{O(\log^4(n))}{\Delta},$$

[6] We do note that however, their time-efficient algorithms are considerably more complex. They first need to find a so-called sparse-dense decomposition of the input graph via sublinear algorithms. Then, this decomposition is used to color the final graph from the sampled colors using an algorithmic version of the proof of their palette sparsification theorem which in particular requires a highly non-greedy and not-so-simple approach.

where the first inequality is by the law of conditional expectation (by conditioning on list-sizes first), the second is by union bound, and the third is by the list-size properties of Theorem 3.1. Since the total number of edges in G is at most $n\Delta/2$, we can conclude the proof. ∎

We now prove Theorem 4.1 for each family of sublinear algorithms separately. In the following, we prove the resource guarantees of the algorithms only in expectation instead of deterministically. However, using the standard trick of running $O(\log n)$ copies of the algorithm in parallel, terminating any copy that uses more than twice the expected resources, and returning the answer of any of the remaining ones, we obtain the desired algorithms in Theorem 4.1 as well (this reduction only increases space/query/memory of algorithms with an $O(\log n)$ multiplicative factor and the error probability with a $1/\text{poly}(n)$ additive factor).

Graph streaming. At the beginning of the stream, sample the colors $\mathcal{L}$ using Theorem 3.1 and during the stream, only store the edges that belong to $G_{\mathcal{L}}$. In the end, run the greedy algorithm on $G_{\mathcal{L}}$ and return the coloring. Theorem 3.1 ensures that with high probability G is (list-)colorable from the sampled lists which leads to a $(\Delta+1)$ coloring of the entire graph. Observation 4.1 ensures that we only need to work with $G_{\mathcal{L}}$ at the end of the stream and not all of G, and Claim 4.2 bounds the space of the algorithm with $\widetilde{O}(n)$ space in expectation.

We can implement this algorithm in dynamic streams also by recovering the conflict graph using a sparse recovery sketch instead of explicitly storing each of its edges in the stream. See [1] for more on dynamic streams.

Sublinear time. Sample the colors $\mathcal{L}$ using Theorem 3.1 and query the edges between *all* pairs of vertices $u \neq v \in V$ with $L(u) \cap L(v)$ being non-empty to find the edges of $G_{\mathcal{L}}$. The same analysis as in Claim 4.2 applied to the $\binom{n}{2}$ vertex-pairs (instead of $\leqslant n\Delta/2$ edges), ensures that the expected number of queries is $\widetilde{O}(n^2/\Delta)$. We can then color $G_{\mathcal{L}}$ using the greedy algorithm in $\widetilde{O}(n)$ time. The correctness follows from Theorem 3.1 and Observation 4.1 as before. This algorithm only requires adjacency matrix access to G and we run it when $\Delta \geqslant \sqrt{n}$ to obtain $\widetilde{O}(n\sqrt{n})$ time/query algorithm. When $\Delta \leqslant \sqrt{n}$, we instead run the standard greedy algorithm using $O(n\Delta) = \widetilde{O}(n\sqrt{n})$ time and queries to the adjacency list of G instead.

MPC. The algorithm is almost identical to the semi-streaming one. Suppose we have access to public randomness. Then, we can sample the lists $\mathcal{L}$ publicly, and each machine that has an edge in $G_{\mathcal{L}}$ can send it to a designated machine. This way, a single machine receives all of $G_{\mathcal{L}}$ and can color it greedily. The correctness follows from Theorem 3.1 and Observation 4.1 and the memory needed for this designated machine will be $\widetilde{O}(n)$ in expectation by Observation 4.2. Finally, we can remove the public randomness by having one machine do the sampling first on its own and share it with all the remaining machines in $O(1)$ rounds using the standard MPC primitives of search and sort.

Acknowledgement

We would like to thank Soheil Behnezhad, Yannic Maus, and Ronitt Rubinfeld for many helpful discussions and for their encouragement toward finding simpler sublinear algorithms for $(\Delta+1)$ vertex coloring. Sepehr Assadi is additionally grateful to Yu Chen and Sanjeev Khanna for their prior collaboration in [5] that formed the backbone of this work.

Part of this work was conducted while the first named author was visiting the Simons Institute for the Theory of Computing as part of the Sublinear Algorithms program.

References

[1] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In Y. Rabani, editor, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pages 459–467. SIAM, 2012.

[2] N. Alon and S. Assadi. Palette sparsification beyond $(\Delta+1)$ vertex coloring. In J. Byrka and R. Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020 (Virtual Conference)*, volume 176 of *LIPIcs*, pages 6:1–6:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[3] J. Anderson, A. Bernshteyn, and A. Dhawan. Coloring graphs with forbidden almost bipartite subgraphs. *arXiv preprint arXiv:2203.07222*, 2022.

[4] S. Assadi, A. Chakrabarti, P. Ghosh, and M. Stoeckl. Coloring in graph streams via deterministic and adversarially robust algorithms. In F. Geerts, H. Q. Ngo, and S. Sintos, editors, *Proceedings of ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2023*, pages 141–153. ACM, 2023.

[5] S. Assadi, Y. Chen, and S. Khanna. Sublinear algorithms for $(\Delta + 1)$ vertex coloring. In T. M. Chan, editor, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 767–786. SIAM, 2019.

[6] S. Assadi, P. Kumar, and P. Mittal. Brooks' theorem in graph streams: a single-pass semi-streaming algorithm for Δ-coloring. In S. Leonardi and A. Gupta, editors, *Proceedings of ACM SIGACT Symposium on Theory of Computing, STOC 2022*, pages 234–247. ACM, 2022.

[7] S. K. Bera, A. Chakrabarti, and P. Ghosh. Graph coloring via degeneracy in streaming and other space-conscious models. In A. Czumaj, A. Dawar, and E. Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020 (Virtual Conference)*, volume 168 of *LIPIcs*, pages 11:1–11:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[8] A. Chakrabarti, P. Ghosh, and M. Stoeckl. Adversarially robust coloring for graph streams. In M. Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPIcs*, pages 37:1–37:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[9] Y.-J. Chang, M. Fischer, M. Ghaffari, J. Uitto, and Y. Zheng. The complexity of $(\Delta+1)$ coloring in congested clique, massively parallel computation, and centralized local computation. In *Proceedings of ACM Symposium on Principles of Distributed Computing, PODC 2019*, pages 471–480, 2019.

[10] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001. Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 2001.

[11] A. Czumaj, P. Davies, and M. Parter. Simple, deterministic, constant-round coloring in congested clique and MPC. *SIAM J. Comput.*, 50(5):1603–1626, 2021.

[12] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004. Proceedings*, volume 3142 of *Lecture Notes in Computer Science*, pages 531–543. Springer, 2004.

[13] M. Fischer, M. M. Halldórsson, and Y. Maus. Fast distributed brooks' theorem. In N. Bansal and V. Nagarajan, editors, *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 2567–2588. SIAM, 2023.

[14] M. Flin, M. Ghaffari, M. M. Halldórsson, F. Kuhn, and A. Nolin. Coloring fast with broadcasts. In K. Agrawal and J. Shun, editors, *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2023*, pages 455–465. ACM, 2023.

[15] M. Flin, M. Ghaffari, M. M. Halldórsson, F. Kuhn, and A. Nolin. A distributed palette sparsification theorem. In D. P. Woodruff, editor, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*, pages 4083–4123. SIAM, 2024.

[16] M. Flin and P. Mittal. $(\Delta + 1)$ vertex coloring in $O(n)$ communication. In R. Gelles, D. Olivetti, and P. Kuznetsov, editors, *Proceedings of ACM Symposium on Principles of Distributed Computing, PODC 2024*, pages 416–424. ACM, 2024.

[17] M. M. Halldórsson, F. Kuhn, A. Nolin, and T. Tonoyan. Near-optimal distributed degree+1 coloring. In S. Leonardi and A. Gupta, editors, *Proceedings of ACM SIGACT Symposium on Theory of Computing, STOC 2022*, pages 450–463. ACM, 2022.

[18] D. G. Harris, J. Schneider, and H. Su. Distributed $(\Delta + 1)$-coloring in sublogarithmic rounds. In D. Wichs and Y. Mansour, editors, *Proceedings of ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 465–478. ACM, 2016.

[19] D. Hefetz and M. Krivelevich. Colouring graphs from random lists. *arXiv preprint arXiv:2402.09998*, 2024.

[20] S. Janson, T. Luczak, and A. Rucinski. *Random graphs*. John Wiley & Sons, 2011.

[21] J. Kahn and C. Kenney. Asymptotics for palette sparsification. *arXiv preprint arXiv:2306.00171*, 2023.

[22] J. Kahn and C. Kenney. Asymptotics for palette sparsification from variable lists. *arXiv preprint arXiv:2407.07928*, 2024.

[23] H. J. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In M. Charikar, editor, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pages 938–948. SIAM, 2010.

[24] B. Reed. ω, Δ, and χ. *Journal of Graph Theory*, 27(4):177–212, 1998.