Vizing's Theorem in Near-Linear Time

Sepehr Assadi
University of Waterloo
Waterloo, Canada
sepehr@assadi.info

Soheil Behnezhad
Northeastern University
Boston, USA
s.behnezhad@northeastern.edu

Sayan Bhattacharya
University of Warwick
Coventry, United Kingdom
s.bhattacharya@warwick.ac.uk

Martín Costa
University of Warwick
Coventry, United Kingdom
martin.costa@warwick.ac.uk

Shay Solomon
Tel Aviv University
Tel Aviv, Israel
solo.shay@gmail.com

Tianyi Zhang
ETH Zürich
Zürich, Switzerland
tianyi.zhang@inf.ethz.ch

Abstract

Vizing's theorem states that any n-vertex m-edge graph of maximum degree Δ can be *edge colored* using at most $\Delta + 1$ different colors [Vizing, 1964]. Vizing's original proof is algorithmic and shows that such an edge coloring can be found in $O(mn)$ time. This was subsequently improved to $\tilde{O}(m\sqrt{n})$ time, independently by [Arjomandi, 1982] and by [Gabow et al., 1985].

Very recently, independently and concurrently, using randomization, this runtime bound was further improved to $\tilde{O}(n^2)$ by [Assadi, 2024] and $\tilde{O}(mn^{1/3})$ by [Bhattacharya, Carmon, Costa, Solomon and Zhang, 2024] (and subsequently to $\tilde{O}(mn^{1/4})$ by [Bhattacharya, Costa, Solomon and Zhang, 2024]).

In this paper, we present a randomized algorithm that computes a $(\Delta+1)$-edge coloring in near-linear time—in fact, only $O(m \log \Delta)$ time—with high probability, *giving a near-optimal algorithm for this fundamental problem.*

CCS Concepts

• **Theory of computation → Graph algorithms analysis**.

Keywords

Edge Coloring, Vizing's Theorem

ACM Reference Format:
Sepehr Assadi, Soheil Behnezhad, Sayan Bhattacharya, Martín Costa, Shay Solomon, and Tianyi Zhang. 2025. Vizing's Theorem in Near-Linear Time. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC '25), June 23–27, 2025, Prague, Czechia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3717823.3718265

1 Introduction

Given a simple undirected graph $G = (V, E)$ on n vertices and m edges, as well as an integer $\kappa \in \mathbb{N}^+$, a κ-edge coloring $\chi : E \to \{1, 2, \ldots, \kappa\}$ of G assigns a color $\chi(e)$ to each edge $e \in E$ so that any two adjacent edges receive distinct colors. The minimum possible value of κ for which a κ-edge coloring exists in G is known as the *edge chromatic number* of G. If G has maximum vertex degree Δ, any proper edge coloring would require at least Δ different colors. A classical theorem by Vizing shows that $\Delta + 1$ colors are always sufficient [54]. Moreover, it was proven by [44] that it is NP-complete to distinguish whether the edge chromatic number of a given graph is Δ or $\Delta + 1$, and therefore $\Delta + 1$ is the best bound we can hope for with polynomial time algorithms.

Vizing's original proof easily extends to an $O(mn)$ time algorithm, which was improved to $\tilde{O}(m\sqrt{n})$ in the 1980s by [2] and [39] independently[1]. More recently, the algorithms of [2, 39] were simplified in [52], while shaving off extra logarithmic factors from their runtime complexities, achieving a clean $O(m\sqrt{n})$ runtime bound. Very recently, this longstanding $O(m\sqrt{n})$ time barrier was bypassed in two concurrent works [3] and [11] which improved the runtime bound to two incomparable bounds of $\tilde{O}(n^2)$ and $\tilde{O}(mn^{1/3})$, respectively. In a follow-up work, the $\tilde{O}(mn^{1/3})$ runtime bound of [11] to was further improved to $\tilde{O}(mn^{1/4})$ in [16].

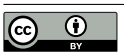
In this work, we resolve the time complexity of randomized $(\Delta + 1)$-edge coloring up to at most a log factor by presenting a near-linear time algorithm for this problem.

Theorem 1.1. *There is a randomized algorithm that, given any simple undirected graph $G = (V, E)$ on n vertices and m edges with maximum degree Δ, finds a $(\Delta + 1)$-edge coloring of G in $O(m \log n)$ time with high probability.*

Remarks: Several remarks on our Theorem 1.1 are in order:

- Our algorithm in Theorem 1.1 does not rely on any of the recent developments in [3, 11, 16] and takes an entirely different path. We present an overview of our approach, as well as a comparison to these recent developments in Section 1.2.
- Our main contribution in this work is to improve the time-complexity of $(\Delta + 1)$-edge coloring by polynomial factors all the way to near-linear. For this reason, as well as for the sake of transparency of our techniques, we focus primarily on presenting an $O(m \log^3 n)$ time randomized algorithm (Theorem 6.1), which showcases our most novel ideas. In the full version [4], we show that a more careful implementation of the same algorithm achieves a clean $O(m \log n)$ runtime.
- We can additionally use a result of [9] to further replace the $\log n$ term in Theorem 1.1 with a $\log \Delta$ term, leading to an algorithm for $(\Delta+1)$-edge coloring in $O(m \log \Delta)$ time with high probability (see full version [4]). This matches the longstanding time bound

*Full version is available on arXiv [4]: https://arxiv.org/abs/2410.05240



STOC '25, Prague, Czechia

ACM ISBN 979-8-4007-1510-5/25/06
https://doi.org/10.1145/3717823.3718265

[1] Throughout, we use $\tilde{O}(f) := O(f \text{ polylog } (n))$ to suppress log-factors in the number of vertices of the graph.

for Δ-edge coloring bipartite graphs [30] (which, to the best of our knowledge, is also the best known runtime for $(\Delta+1)$-edge coloring bipartite graphs). This bound is also related to a recent line of work in [9, 10, 33] that focused on the $\Delta = n^{o(1)}$ case and gave a randomized $(\Delta+1)$-coloring algorithm that runs in $O(m\Delta^4 \log \Delta)$ time with high probability [10].

- Vizing's theorem generalizes for (loop-less) multigraphs, asserting that any multigraph with edge multiplicity μ can be $(\Delta+\mu)$-edge colored [54, 55]. A related result is Shannon's theorem [51] that asserts that any multigraph can be $\lfloor 3\Delta/2 \rfloor$ edge colored independent of μ; both these bounds are tight: see the so-called *Shannon multigraphs* [55]. We show that our techniques extend to these theorems as well, giving $O(m \log \Delta)$ time algorithms for both problems (see full version [4]), which vastly improves upon the previous runtime $O(\min\{mn + m\Delta, n\,\text{poly}(\Delta)\})$ by [32, 51].

1.1 Related Work

In addition to algorithms for Vizing's theorem, there has also been a long line of work on fast algorithms for edge coloring that use more than $\Delta+1$ colors. It was first shown in [45] that an edge coloring can be computed in $\tilde{O}(m)$ time when we have $\Delta + \tilde{O}(\sqrt{\Delta})$ different colors. In addition, there are algorithms which run in linear or near-linear time for $(1+\epsilon)\Delta$-edge coloring [10, 15, 33, 34, 36] when $\epsilon \in (0,1)$ is a constant. Most recently, it was shown in [3] that even a $(\Delta + O(\log n))$-edge coloring can be computed in $O(m \log \Delta)$ expected time.

There are other studies on restricted graph classes. In bipartite graphs, a Δ-edge coloring can be computed in $\tilde{O}(m)$ time [1, 28, 30, 42]. In bounded degree graphs, one can compute a $(\Delta+1)$-edge coloring in $\tilde{O}(m\Delta)$ time [39], and it was generalized recently for bounded arboricity graphs [14]; see also [13, 24, 46] for further recent results on edge coloring in bounded arboricity graphs. Subfamilies of bounded arboricity graphs, including planar graphs, bounded tree-width graphs and bounded genus graphs, were studied in [25, 26, 29].

Beside the literature on classical algorithms, considerable effort has been devoted to the study of edge coloring in various computational models in the past few years, including dynamic [6, 12, 15, 22, 23, 34], online [17–19, 27, 35, 47, 49], distributed [5, 8, 20, 22, 31, 37, 38, 40, 43, 48, 53], and streaming [7, 21, 41, 50] models, among others.

1.2 Technical Overview

We now present an overview of prior approaches to $(\Delta+1)$-edge coloring and describe our techniques at a high level. For the following discussions, we will assume basic familiarity with Vizing's proof and its underlying algorithm.

Prior Approaches. A generic approach for $(\Delta+1)$-edge coloring, dating back to Vizing's proof itself, is to extend a partial $(\Delta+1)$-edge coloring of the graph one edge at a time, possibly by recoloring some edges, until the entire graph becomes colored. As expected, the main bottleneck in the runtime of this approach comes from extending the coloring to the last few uncolored edges. For instance, given a graph $G = (V, E)$ with maximum degree Δ, we can apply Eulerian partitions to divide G into two edge-disjoint subgraphs $G = G_1 \cup G_2$ with maximum degrees at most $\lceil \Delta/2 \rceil$. We then find $(\lceil \Delta/2 \rceil + 1)$-edge colorings of the subgraphs G_1 and G_2 recursively. Directly combining the colorings of G_1 and G_2 gives a coloring of G with $\Delta+3$ colors, so we have to uncolor a $2/(\Delta+3)$ fraction of the edges—amounting to $O(m/\Delta)$ edges—and try to extend the current partial $(\Delta+1)$-edge coloring to these edges. Thus, the "only" remaining part is to figure out a way to color these final $O(m/\Delta)$ edges, and let the above approach take care of the rest.

This task of extending the coloring to the last $\Theta(m/\Delta)$ edges is the common runtime bottleneck of all previous algorithms. Vizing's original algorithm [54] gives a procedure to extend any partial coloring to an arbitrary uncolored edge (u, v) by rotating some colors around u and flipping the colors of an alternating path starting at u. The runtime of this procedure would be proportional to the size of the rotation, usually called a *Vizing fan*, and the length of the alternating path, usually called a *Vizing chain*, which are bounded by Δ and n respectively. As such, the total runtime for coloring the remaining $O(m/\Delta)$ edges using Vizing fans and Vizing chains will be $O(mn/\Delta)$ time.

As one can see, flipping long alternating paths is the major challenge in Vizing's approach. To improve this part of the runtime, [39] designed an algorithm that groups all uncolored edges into $O(\Delta^2)$ types depending on the two colors of the Vizing chain induced by this edge. Since all the Vizing chains of the same type are vertex-disjoint, they can be flipped simultaneously and their total length is only $O(n)$. This means that the runtime of coloring all edges of a single type can be bounded by $O(n)$ as well. This leads to an $O(n\Delta^2)$ time algorithm for handling all $O(\Delta^2)$ types; a more careful analysis can bound this even with $O(m\Delta)$ time. Finally, balancing the two different bounds of $O(mn/\Delta)$ and $O(m\Delta)$ yields a runtime bound of $O(m\sqrt{n})$ for coloring $O(m/\Delta)$ edges, which leads to an $\tilde{O}(m\sqrt{n})$ time algorithm using the above framework.

There has been some very recent progress that broke through this classical barrier of $O(m\sqrt{n})$ time in [2, 39]. In [11], the authors speed up the extension of the coloring to uncolored edges when these edges admit a small vertex cover. They then show how to precondition the problem so that uncolored edges admit a small vertex cover, leading to a $\tilde{O}(mn^{1/3})$ time algorithm. In [3], the author avoided the need for Eulerian partition and recursion altogether by instead designing a new near-linear time algorithm for $(\Delta + O(\log n))$-edge coloring. This algorithm borrows insights from sublinear matching algorithms in regular bipartite graphs by [42] and is thus completely different from other edge coloring algorithms mentioned above. By using this algorithm, finding a $(\Delta+1)$-edge coloring directly reduces to the color extension problem with $O((m \log n)/\Delta)$ uncolored edges (by removing the colors of a $\Theta(\log n)/\Delta$ fraction of the edges in the $(\Delta + O(\log n))$-edge coloring to obtain a partial $(\Delta+1)$-edge coloring first). Applying Vizing's procedure for these uncolored edges takes additional $\tilde{O}(mn/\Delta) = \tilde{O}(n^2)$ time, leading to an $\tilde{O}(n^2)$ time algorithm for $(\Delta+1)$-edge coloring. Finally, in [16], the authors showed that a $(\Delta+1)$-coloring can be computed in $\tilde{O}(mn^{1/4})$ time, by using the algorithm of [3] for initial coloring of the graph and then presenting an improved color extension subroutine with a runtime of $\tilde{O}(\Delta^2 + \sqrt{\Delta n})$ for coloring each remaining edge; the best previous color extension time bounds were either the trivial $O(n)$ bound or the bound $\tilde{O}(\Delta^4)$ by [8, 10].

Our Approach: A Near-Linear Time Color Extension Algorithm. We will no longer attempt to design a faster color extension for a *single edge*, and instead color them in large *batches* like in [39], which allows for a much better amortization of runtime in coloring multiple edges. This ultimately leads to our main technical contribution: a new randomized algorithm for solving the aforementioned color extension problem for the last $O(m/\Delta)$ edges in $\tilde{O}(m)$ time. With this algorithm at hand, we can follow the aforementioned Eulerian partition approach and obtain a $(\Delta+1)$-edge coloring algorithm whose runtime $T(m)$ follows the recursion $T(m) \leq 2T(m/2) + \tilde{O}(m)$ with high probability; this implies that $T(m) = \tilde{O}(m)$, hence, giving us a near-linear time randomized algorithm for $(\Delta+1)$-edge coloring. This way, we will not even need to rely on the $(\Delta + O(\log n))$-edge coloring algorithm of [3] to color the earlier parts of the graph (although one can use that algorithm instead of Eulerian partition approach to the same effect).

We now discuss the main ideas behind our color extension algorithm. In the following, it helps to think of the input graph as being near-regular (meaning that the degree of each vertex is $\Theta(\Delta)$), and thus the total number of edges will be $m = \Theta(n\Delta)$; this assumption is *not* needed for our algorithm and is only made here to simplify the exposition.

Color Type Reduction. Recall that the runtime of $O(n\Delta^2)$ for the color extension algorithm of [39] is due to the fact that there are generally $O(\Delta^2)$ types of alternating paths in the graph and that the total length of the paths of each color type is bounded by $O(n)$ edges. However, *if* it so happens that the existing partial coloring only involves $O(\Delta)$ color types instead, then the same algorithm will only take $O(n\Delta) = O(m)$ time (by the near-regularity assumption). The underlying idea behind our algorithm is to modify the current partial edge coloring (without decreasing the number of uncolored edges) so that the number of color types reduces from $O(\Delta^2)$ to $O(\Delta)$ only.

To explore this direction, let us *assume for now the input graph is bipartite*, which greatly simplifies the structure of Vizing fans and Vizing chains, thus allowing us to convey the key ideas more clearly (see Section 3 for a more detailed exposition); later we highlight some of the key challenges that arise when dealing with general graphs. We shall note that it has been known since the 80s that one can Δ-edge color bipartite graphs in $\tilde{O}(m)$ time [28, 30]. However, the algorithms for bipartite graphs use techniques that are entirely different from Vizing fans and Vizing chains, and which do not involve solving the color extension problem at all. In particular, prior to this work, it was unclear whether one can efficiently solve the color extension problem in bipartite graphs. Therefore, the assumption of a bipartite input graph does not trivialize our goal of using Vizing fans and Vizing chains for efficiently solving the color extension problem. Additionally, we can assume that in the color extension problem, the last $O(m/\Delta)$ edges to be colored can be partitioned into $O(1)$ matchings (this guarantee follows immediately from the recursive framework we outlined earlier), and that we deal with each of these matchings separately. In other words, we can also *assume that the uncolored edges are vertex-disjoint.*

Let χ be a partial $(\Delta+1)$-edge coloring, and for any vertex $w \in V$, let $\text{miss}_\chi(w) \subseteq [\Delta+1]$ be the set of colors missing from the edges incident to w under χ. Given any uncolored edge (u,v), the color type of this edge (u,v) would be $\{c_u, c_v\}$ for some arbitrary choices of $c_u \in \text{miss}_\chi(u)$ and $c_v \in \text{miss}_\chi(v)$ (it is possible for an edge to be able to choose more than one color type, but we fix one arbitrary choice among them); in other words, if we flip the $\{c_u, c_v\}$-alternating path starting at u, then we can assign $\chi(u,v)$ to be c_v. To reduce the total number of different color types to $O(\Delta)$, we would have to make some color types much more *popular*: at the beginning, a type spans an $\Omega(1/\Delta^2)$ proportion of the uncolored edges but we would like to have a type spanning an $\Omega(1/\Delta)$ proportion. For this purpose, we fix an arbitrary color type $\{\alpha,\beta\}$, and want to modify χ to transform the type of an arbitrary uncolored edge (u,v) from $\{c_u, c_v\}$ to $\{\alpha,\beta\}$ – we call this *popularizing* the edge (u,v). To do this, we can simply flip the $\{\alpha, c_u\}$-alternating path P_u starting at u and the $\{\beta, c_v\}$-alternating path P_v starting at v.

There are two technical issues regarding this path-flipping approach. Firstly, the alternating paths P_u and P_v could be very long, and require a long time for being flipped. More importantly, flipping P_u and P_v could possibly damage other $\{\alpha,\beta\}$-type (uncolored) edges that we popularized before. More specifically, say that we have popularized a set Φ of uncolored edges. When popularizing the next uncolored edge (u,v), it could be the case that the $\{\alpha, c_u\}$-alternating path P_u is ending at a vertex u' for some edge $(u',v') \in \Phi$. If we flip the path P_u, then (u',v') would no longer be of $\{\alpha,\beta\}$-type as α would not be missing at u' anymore. See Figure 1 for an illustration.

Our key observation is that when $|\Phi|$ is relatively small, most choices for an alternating path P_u cannot be ending at edges in Φ. Consider the above bad example where P_u is a $\{c_u, \alpha\}$-alternating path ending at u' for some $(u',v') \in \Phi$. Let us instead look at this from the perspective of the $\{\alpha,\beta\}$-type edge $(u',v') \in \Phi$. For any $(u',v') \in \Phi$ and any color γ, there can be at most one uncolored edge $(u,v) \notin \Phi$ whose corresponding path P_u is the same $\{\gamma,\alpha\}$-alternating path starting at u'; this is because any vertex belongs to at most one alternating path of a fixed type (here, the type $\{\gamma,\alpha\}$) and each vertex belongs to at most one uncolored edge (recall that the uncolored edges form a matching). This is also true for $\{\gamma,\beta\}$-type edges for the same exact reason. As such, ranging over all possible choices of $\gamma \in [\Delta+1]$, there are at most $O(|\Phi|\Delta)$ uncolored edges (u,v) whose alternating paths could damage the set Φ. Therefore, as long as the size of Φ is a $o(1/\Delta)$ fraction (or more precisely, an $O(1/\Delta)$ fraction, for a sufficiently small constant hiding in the O-notation) of all uncolored edges, the following property holds: a constant fraction of uncolored edges $e \notin \Phi$ can be popularized using the above method without damaging Φ. See Figure 2 for an illustration.

This property resolves both technical issues raised earlier simultaneously. Let λ denote the number of uncolored edges. For the second issue, as long as $|\Phi| = o(\lambda/\Delta)$, we can take a random uncolored edge $(u,v) \notin \Phi$ and flip P_u and P_v if this does not damage any already popularized edge in Φ; by the observation above, this happens with constant probability. For the first issue, we can show that the expected length of alternating paths P_u and P_v, for the random uncolored edge picked above, is $O(m/\lambda)$; indeed, this is because the number of edges colored α or β is $O(n)$, hence the total length of all alternating paths with one color being fixed to α or β is $O(m)$. All in all, the total time we spend to popularize a single type $\{\alpha,\beta\}$ to become a $\Theta(1/\Delta)$ fraction of all uncolored edges

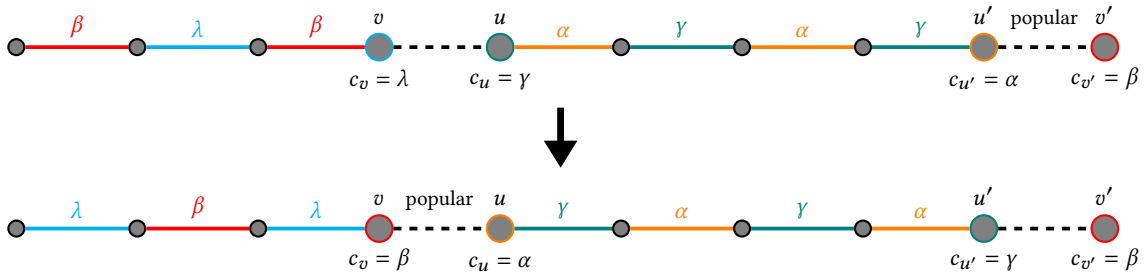


Figure 1: In this picture, we attempt to popularize edge (u, v) by flipping the $\{\alpha, \gamma\}$-alternating path from u and the $\{\beta, \lambda\}$-alternating path from v. However, flipping the $\{\alpha, \gamma\}$-alternating path from u makes a previously popular edge (u', v') unpopular as u' will not miss color α anymore.

is $O(\lambda/\Delta \cdot m/\lambda) = O(m/\Delta)$. Since coloring edges of a single type takes $O(n) = O(m/\Delta)$ time by our earlier discussion, we can color in this way a $\Theta(1/\Delta)$ fraction of all uncolored edges in $O(m/\Delta)$ expected time. As a direct corollary, we can color all uncolored edges in $O(m \log n)$ time, hence solving the color extension problem, in (near-regular) bipartite graphs, in near-linear time. The above argument will be detailed in the proof of Lemma 3.2 in Section 3.

Collecting U-Fans in General Graphs. We now discuss the generalization of our scheme above to non-bipartite graphs. The existence of odd cycles in non-bipartite graphs implies that we can no longer assign a color type $\{c_u, c_v\}$ to an uncolored edge (u, v) for $c_u \in \text{miss}_\chi(u), c_v \in \text{miss}_\chi(v)$, and hope that flipping the $\{c_u, c_v\}$-alternating path from u allows us to color the edge (u, v) with c_v (because the path may end at v, and thus after flipping it c_v will no longer be missing from v). This is where Vizing fans and Vizing chains come into play: in non-bipartite graphs, a color type of an uncolored edge (u, v) is $\{c_u, \gamma_{u,v}\}$ where c_u is an arbitrary color in $\text{miss}_\chi(u)$ but $\gamma_{u,v}$ is determined by the Vizing fan around u and the alternating path that we take for coloring (u, v) (or the Vizing chain of (u, v)). Thus, while as before we can switch c_u with some fixed color α, it is unclear how to flip alternating paths to change $\gamma_{u,v}$ to β also, in order to popularize the edge (u, v) to be of some designated type $\{\alpha, \beta\}$, without damaging another popularized edge as a result.

To address this challenge, we rely on the notion of a *u-fan*, introduced by [39], which is the non-bipartite graph analogue of an uncolored edge in bipartite graphs. A u-fan of type $\{\alpha, \beta\}$ is a pair of uncolored edges $(u, v), (u, w)$ such that $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(u) \cap \text{miss}_\chi(v)$. Consider the $\{\alpha, \beta\}$-alternating path starting at u. As at least one of v or w is not the other endpoint of this alternating path (say v), flipping this path allows us to assign the color β to (u, v). Consequently, as u-fans are similar to edges in bipartite graphs, we can still essentially (but not exactly) apply our color type reduction approach if all the uncolored edges are paired as u-fans. Therefore, it suffices to modify χ to pair all uncolored edges together to form u-fans.

In order to pair different uncolored edges together and form u-fans, we should first be able to move uncolored edges around. Such operations already appeared in some previous work on dynamic edge coloring [22, 23, 34]. Basically, for any uncolored edge (u, v), we can modify χ to shift this uncolored edge to any position on its Vizing chain. This naturally leads to a win-win situation: if the Vizing chain is short, then (u, v) can be colored efficiently using Vizing's procedure; otherwise if most Vizing chains are long, then there must be a pair of Vizing chains meeting together after a few steps, so we can shift two uncolored edges to form a u-fan efficiently.

Let us make this a bit more precise. Fix an arbitrary color α and consider the set U_α of all the uncolored edges (u, v) such that $\alpha \in \text{miss}_\chi(u)$ and the respective Vizing chain is of type $\{\alpha, \cdot\}$. Also, assume there are m_α edges colored α under χ. If most $\{\alpha, \cdot\}$-Vizing chains have length larger than $\Omega(m_\alpha/|U_\alpha|)$, then on average, two Vizing chains should meet within the first $O(m_\alpha/|U_\alpha|)$ steps; in this case, we can repeatedly pick two intersecting Vizing chains and create a u-fan by shifting their initial uncolored edges to the intersection of these Vizing chains; see Figure 3 for illustration. Given the length of the chains, this takes $O(m_\alpha/|U_\alpha|)$ time. Otherwise, the average cost of applying Vizing's color extension procedure is $O(m_\alpha/|U_\alpha|)$, and in this case we can directly color all those edges in $O(m_\alpha)$ time. Summing over all different $\alpha \in [\Delta + 1]$ gives a near-linear runtime. The above argument will be stated in Lemma 6.2.

The above discussion leaves out various technical challenges. For instance, moving around uncolored edges as described above breaks the assumption that the uncolored edges form a matching. Handling this requires dedicating *different* colors from $\text{miss}_\chi(u)$ for every uncolored edge incident on a vertex u. This is formalized via the notion of *separability* in Section 5.1. Additionally, we have ignored all algorithmic aspects of (efficiently) finding pairs of intersecting Vizing chains, as well as the corner cases of Vizing fan intersections and fan-chain intersections. We defer the discussions on these details to the actual proofs in subsequent sections.

2 Basic Notation

Let $G = (V, E)$ be graph on n vertices with m edges and maximum degree Δ and let $\chi : E \to [\Delta + 1] \cup \{\perp\}$ be a (partial) $(\Delta + 1)$-edge coloring of G. We refer to edges $e \in E$ with $\chi(e) = \perp$ as *uncolored*. Given a vertex $u \in V$, we denote the set of colors that are not assigned to any edge incident on u by $\text{miss}_\chi(u)$. We sometimes refer to $\text{miss}_\chi(u)$ as the *palette* of u. We say that the colors in $\text{miss}_\chi(u)$ are *missing* (or *available*) at u.

Given a path $P = e_1, \ldots, e_k$ in G, we say that P is an $\{\alpha, \beta\}$-*alternating path* if $\chi(e_i) = \alpha$ whenever i is odd and $\chi(e_i) = \beta$

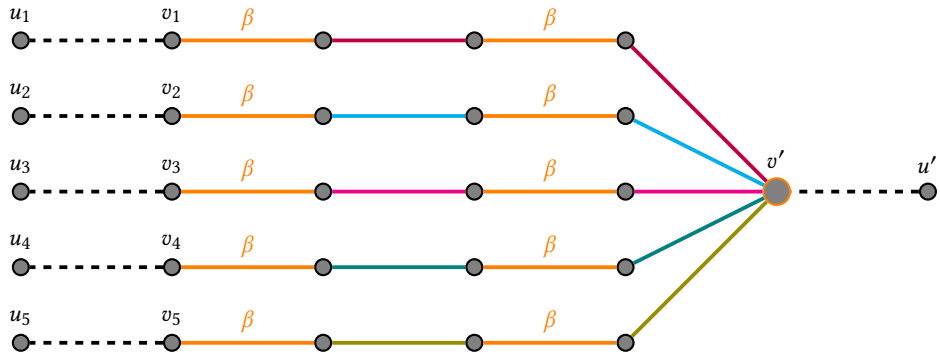


Figure 2: In this picture, $(u', v') \in \Phi$ with $c_{u'} = \alpha, c_{v'} = \beta$. For each uncolored edge (u_i, v_i), flipping the $\{c_i, \beta\}$-alternating path from v_i would damage the property that $c_{v'} = \beta$. Fortunately, there are at most Δ many different such (u_i, v_i) because each of them is at the end of an $\{\beta, \cdot\}$-alternating path starting at v'.

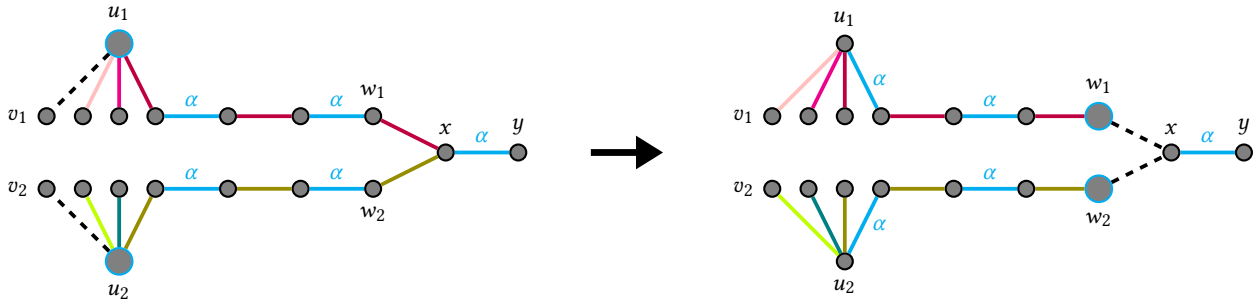


Figure 3: In this picture, we have two different uncolored edges $(u_1, v_1), (u_2, v_2)$ such that $\alpha \in \mathsf{miss}_\chi(u_1) \cap \mathsf{miss}_\chi(u_2)$, and their Vizing chains first intersect at edge (x, y) which currently has color α under χ. Then we can rotate their Vizing fans and flip part of their Vizing chains to shift $(u_1, v_1), (u_2, v_2)$ to $(w_1, x), (w_2, x)$ respectively to form a u-fan; note that $\alpha \in \mathsf{miss}_\chi(w_1) \cap \mathsf{miss}_\chi(w_2)$ after this shifting procedure.

whenever i is even (or vice versa). We say that the alternating path P is *maximal* if one of the colors α or β is missing at each of the endpoints of P. We refer to the process of changing the color of each edge $e_i \in P$ with color α (resp. β) to β (resp. α) as *flipping* the path P. We denote by $|P|$ the length (i.e., the number of edges) of the alternating path P. We define the *length i prefix* of the path P to be the path $P_{\leq i} := e_1, \ldots, e_i$.

Consider a set $U \subseteq E$ of edges that are uncolored under χ, i.e., $\chi(e) = \perp$ for all $e \in U$. We use the phrase *"extending χ to U"* to mean the following: Modify χ so as to ensure that $\chi(e) \neq \perp$ for all $e \in U$, without creating any new uncolored edges. When the set U consists of a single edge e (i.e., when $U = \{e\}$), we use the phrase *"extending χ to the edge e"* instead of *"extending χ to U"*.

Our algorithms will always work by modifying a partial coloring χ; unless explicitly specified otherwise, every new concept we define (such as u-fans and separable collection in Section 5) will be defined with respect to this particular partial coloring χ.

3 Showcase: Our Algorithm Instantiated on Bipartite Graphs

In this section, we instantiate our algorithm on bipartite graphs to showcase some of our key insights, and outline a proof of Theorem 3.1 below.

Theorem 3.1. *There is a randomized algorithm that, given a bipartite graph $G = (V, E)$ with maximum degree Δ, returns a Δ-edge coloring of G in $\tilde{O}(m)$ time with high probability.*

As explained in Section 1.2, focusing on bipartite graphs allows us to ignore the technical issues that arise while dealing with Vizing fans. At the same time, this does *not* trivialize the main conceptual ideas underpinning our algorithm. In particular, we prove Theorem 3.1 via Lemma 3.2 below, which gives a specific *color-extension* algorithm on bipartite graphs. Although near-linear time Δ-edge coloring algorithms on bipartite graphs existed since the 1980s [28, 30], to the best of our knowledge there was no known algorithm for Lemma 3.2 prior to our work.

Lemma 3.2. *Let $\chi : E \to [\Delta] \cup \{\perp\}$ be a partial Δ-edge coloring in a bipartite graph $G = (V, E)$, and let $U \subseteq E$ be a matching of size λ such that every edge $e \in U$ is uncolored in χ. Furthermore, suppose that we have access to an "auxiliary data structure", which allows us*

to detect in $\tilde{O}(1)$ time the two least common colors $\alpha, \beta \in [\Delta]$ in χ.[2] *Then there is a randomized algorithm that can extend χ to $\Omega(\lambda/\Delta)$ many edges of U in $\tilde{O}(m/\Delta)$ time with high probability.*

Let us start by showing that Lemma 3.2 implies Theorem 3.1 easily.

Proof of Theorem 3.1. We follow the strategy outlined in Section 1.2. Given a bipartite graph G, we first find an Eulerian partition of the graph to partition the edges of G into two subgraphs of maximum degree $\lceil \Delta/2 \rceil$ each, and color them recursively using different colors. This leads to a $2 \cdot \lceil \Delta/2 \rceil \leq (\Delta + 2)$ edge coloring of G. We then form a partial Δ edge coloring χ by uncoloring the two color classes of this $(\Delta+2)$-edge coloring with the fewest edges assigned to them, which leaves us with two edge-disjoint matchings U_1 and U_2 to color. This is our previously mentioned color extension problem. To solve this problem, we apply Lemma 3.2 to U_1 first to extend χ to $\Omega(1/\Delta)$ fraction of it, and keep applying this lemma to the remaining uncolored edges of U_1 until they are all colored. We then move to U_2 in the same exact way and extend χ to its edges as well, obtaining a Δ-coloring of the entire G as a result.

The correctness of the algorithm follows immediately by induction. The runtime can also be analyzed as follows. When coloring U_1 (or U_2), each invocation of Lemma 3.2 reduces the number of uncolored edges in U_1 (or U_2) by a $(1 - \Omega(1/\Delta))$ factor and thus we apply this lemma a total of $O(\Delta \cdot \log n)$ time. Moreover, each application of Lemma 3.2 takes $\tilde{O}(m/\Delta)$ time with high probability. Thus, with high probability, it only takes $\tilde{O}(m)$ time to extend the coloring χ to U_1 and U_2 in the color extension problem. Hence, the runtime of the algorithm, with high probability, follows the recurrence $T(m, \Delta) \leq 2T(m/2, \Delta/2) + \tilde{O}(m)$, and thus itself is $\tilde{O}(m)$ time.

Finally, note that with $O(m)$ preprocessing time, we can maintain access to the "auxiliary data structure" throughout the repeated invocations of Lemma 3.2 above: All we need to do is maintain a counter for each color $\gamma \in [\Delta]$, which keeps track of how many edges in G are currently assigned the color γ in χ. We maintain these counters in a balanced search tree, and update the relevant counter whenever we change the color of an edge in G. □

The rest of this section is dedicated to the proof of Lemma 3.2.

3.1 Our Bipartite Color Extension Algorithm in Lemma 3.2

At a high level, our algorithm for Lemma 3.2 consists of the following three steps.

(1) Pick the two least common colors $\alpha, \beta \in [\Delta]$ in Δ. This implies that there are at most $O(m/\Delta)$ edges in G that are colored with α or β in χ.
(2) Modify the coloring χ so that $\Omega(\lambda/\Delta)$ of the edges $(u, v) \in U$ either receive a color under χ, or have $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(v)$. While implementing this step, we ensure that the total number of edges with colors α or β remains at most $O(m/\Delta)$.
(3) Let Φ denote the set of edges $(u, v) \in U$ with $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(v)$. We call these edges *popular*. We extend χ to a constant fraction of the edges in Φ, by flipping a set of maximal $\{\alpha, \beta\}$-alternating paths.

We now formalize the algorithm; the pseudocode is provided in Algorithm 1. As input, we are given a bipartite graph $G = (V, E)$, a partial Δ-edge coloring χ of G, and a matching $U \subseteq E$ of uncolored edges of size λ.

The algorithm starts by fixing the two least common colors $\alpha, \beta \in [\Delta]$ in χ. The main part is the **while** loop in Line 3, which runs in *iterations*. In each iteration of the **while** loop, the algorithm samples an edge $e = (u, v)$ from U u.a.r. and *attempts* to either (1) directly extend the coloring χ to e (see Line 10), which adds e to a set $C \subseteq U$ of colored edges in U or (2) modify χ so that $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(v)$—we refer to this as making the edge (u, v) *popular*—, which adds e to a set $\Phi \subseteq U$ of popular edges (see Line 20). The attempt to modifying χ is done by essentially finding a maximal $\{c_u, \alpha\}$-alternating path P_u starting at u and a $\{c_v, \beta\}$-alternating path P_v starting at v for $c_u \in \text{miss}_\chi(u)$ and $c_v \in \text{miss}_\chi(v)$ (see Line 12 and Lines 14 and 15). The modification itself is done only if P_u and P_v do not intersect any other popular edge already in Φ.

We say that the concerned iteration of the **while** loop FAILS if it chooses an already colored edge in C (Line 6), or modifying the color leads to an already popular edge in Φ to no longer remain popular (Line 17); otherwise, we say the iteration *succeeds*. As stated earlier, the algorithm maintains a subset $\Phi \subseteq U$ of popular edges, and a subset of edges $C \subseteq U$ that got colored since the start of the **while** loop. Initially, we have $C = \Phi = \emptyset$. Thus, the quantity $|\Phi| + |C|$ denotes the number of successful iterations of the **while** loop that have been performed so far. The algorithm performs iterations until $|\Phi| + |C| = \Omega(\lambda/\Delta)$, and then it proceeds to extend the coloring χ to at least a constant fraction of the edges in Φ by finding $\{\alpha, \beta\}$-alternating paths for edges in Φ that admits such paths (see the **for** loop in Line 22).

3.2 Analysis of the Bipartite Color Extension Algorithm: Proof of Lemma 3.2

We start by summarizing a few key properties of Algorithm 1.

Claim 3.3. *Throughout the* **while** *loop in Algorithm 1, there are at most $O(m/\Delta)$ edges in G that receive either the color α or the color β, under χ.*

Proof. We start with $O(m/\Delta)$ such edges in G and each successful iteration of the **while** loop increases the number of such edges by $O(1)$, and there are $O(\lambda/\Delta) = O(m/\Delta)$ such iterations. □

Lemma 3.4. *Throughout the execution of the* **while** *loop in Algorithm 1, the following conditions hold: (i) the set C consists of all the edges in U that are colored under χ; (ii) for every edge $(u, v) \in \Phi$, we have $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(v)$.*

Proof. Part (i) of the lemma follows from Line 2 and Line 10. For part (ii), consider an edge $e = (u, v)$ that gets added to Φ. This happens only after flipping the paths P_u and P_v in Line 19. Just before we execute Line 19, the following conditions hold:

[2] Specifically, for every $\gamma \in [\Delta] \setminus \{\alpha, \beta\}$ and $\gamma' \in \{\alpha, \beta\}$, we have $|\{e \in E \mid \chi(e) = \gamma\}| \geq |\{e \in E \mid \chi(e) = \gamma'\}|$.

Algorithm 1: BipartiteExtension(G, χ, U)

1 Let $\alpha, \beta \in [\Delta]$ be the two least common colors in χ // We have $\alpha \neq \beta$
2 Initialize $\Phi \leftarrow \emptyset$, $C \leftarrow \emptyset$, and set $\lambda \leftarrow |U|$
3 **while** $|\Phi| + |C| < \lambda/(10\Delta)$ **do**
4 Sample an edge $e = (u, v) \sim U$ independently and u.a.r.
5 **if** $(u, v) \in \Phi \cup C$ **then**
6 The iteration FAILS
7 go to Line 3
8 Identify (arbitrarily) two colors $c_u \in \text{miss}_\chi(u)$ and $c_v \in \text{miss}_\chi(v)$
9 **if** $c_u = c_v$ **then**
10 Set $\chi(u, v) \leftarrow c_u$ and $C \leftarrow C \cup \{(u, v)\}$
11 go to Line 3
12 **if** $c_u = \beta$ *or* $c_v = \alpha$ **then**
13 Set $(u, v) \leftarrow (v, u)$ // Now $c_u \neq c_v$, $c_u \neq \beta$, $c_v \neq \alpha$ (see Lines 9 and 12)
14 Let P_u be the maximal $\{c_u, \alpha\}$-alternating path starting at u ($P_u = \emptyset$ if $\alpha \in \text{miss}_\chi(u)$)
15 Let P_v be the maximal $\{c_v, \beta\}$-alternating path starting at v ($P_v = \emptyset$ if $\beta \in \text{miss}_\chi(v)$)
16 **if** *either* P_u *or* P_v *ends at a node that is incident on some edge in* $\Phi \setminus \{e\}$ **then**
17 The iteration FAILS
18 **else**
19 Modify χ by flipping the alternating paths P_u and P_v
20 Set $\Phi \leftarrow \Phi \cup \{(u, v)\}$ // Now $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(v)$
21 $\Phi' \leftarrow \Phi$
22 **for** *each edge* $e = (u, v) \in \Phi'$ **do**
23 $\Phi' \leftarrow \Phi' \setminus \{e\}$
24 W.l.o.g., suppose that $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(v)$
25 **if** *there exists a color* $c \in \{\alpha, \beta\}$ *such that* $c \in \text{miss}_\chi(u) \cap \text{miss}_\chi(v)$ **then**
26 $\chi(u, v) \leftarrow c$
27 **else**
28 Let $P_u^\star$ be the maximal $\{\alpha, \beta\}$-alternating path starting at u
29 (Since G is bipartite, $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(v)$, $P_u^\star$ does *not* end at v)
30 Modify χ by flipping the alternating path $P_u^\star$, and set $\chi(u, v) \leftarrow \beta$
31 **if** *the path* $P_u^\star$ *ends at a node that is incident on some edge in* $e' \in \Phi' \setminus \{e\}$ **then**
32 $\Phi' \leftarrow \Phi' \setminus \{e'\}$

- $\alpha \neq \beta$ (see Line 1).
- $c_u \in \text{miss}_\chi(u)$ and $c_v \in \text{miss}_\chi(v)$ (see Line 8).
- $c_u \neq c_v$, $c_u \neq \beta$ and $c_v \neq \alpha$ (see Line 12).
- If $\alpha \in \text{miss}_\chi(u)$ then $P_u = \emptyset$ (see Line 14), and if $\beta \in \text{miss}_\chi(v)$ then $P_v = \emptyset$ (see Line 15).
- The path P_u (resp. P_v) does *not* end at that a vertex that is incident on some edge in $\Phi \setminus \{e\}$ (see Lines 16 and 17), although it might possibly end at v (resp. u).

From these conditions, it follows that the paths P_u and P_v are edge-disjoint, and after we flip them in Line 19, we have $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(v)$ in Line 20.

In subsequent iterations of the **while** loop, the only places where we change the coloring χ are Lines 10 and 19. Since the edges in U form a matching, changing the coloring in Line 10 cannot affect whether or not the edge $(u, v) \in \Phi$ remains popular (i.e., has $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(v)$). Finally, during a subsequent iteration of the **while** loop where we sample an edge $(u', v') \sim U$, we flip the paths $P_{u'}, P_{v'}$ in Line 19 only if their endpoints are *not* incident on any edges in $\Phi \setminus \{(u', v')\}$ (see Line 16), and in particular, on u or v. Thus, this operation cannot change what colors are available at u and v, and so cannot change whether or not the edge $(u, v) \in \Phi$ remains popular. □

We use the following lemma to bound the number of iterations of the **while** loop in Algorithm 1.

Lemma 3.5. *Each iteration of the* **while** *loop in Algorithm 1 increases the value of* $|\Phi| + |C|$ *by an additive one, with probability at least* $1/2$ *and otherwise keep it unchanged.*

Proof. Fix any given iteration of the **while** loop. At the start of this iteration, we sample an edge from U u.a.r. We say that an edge $e \in U$ is *bad* if the iteration FAILS when we sample e (see Line 6 and Line 17), and *good* otherwise. Note that if we sample a good edge $e \in U$, then the iteration either adds one edge to the set Φ (see Line 20), or adds one edge to the set C (see Line 10). In other words, if we sample a good (resp. bad) edge $e \in U$ at the start of the iteration, then this increases in the value of $|\Phi| + |C|$ by one (resp. keeps the value of $|\Phi| + |C|$ unchanged). We will show that at most $\lambda/2$ edges in U are bad. Since $|U| = \lambda$, this will imply the lemma.

To see why this claimed upper bound on the number of bad edges holds, first note that there are $(|\Phi|+|C|)$ many bad edges that cause the iteration to FAIL in Line 6. It now remains to bound the number of bad edges which cause the iteration to FAIL in Line 17.

Towards this end, note that for each edge $(u', v') \in \Phi$, there are at most 4Δ many maximal $\{\alpha, \cdot\}$- or $\{\beta, \cdot\}$-alternating paths that end at either u' or v'. Furthermore, each such alternating path has its other endpoint incident on at most one edge in U since the edges in U form a matching. Thus, for each edge $(u', v') \in \Phi$, there are at most 4Δ many edges $f_{(u',v')} \in U$ that satisfy the following condition: Some alternating path constructed by the algorithm after sampling $f_{(u',v')}$ ends at either u' or v' (see Line 14 and Line 15). Each such edge $f_{(u',v')}$ is a bad edge which causes the iteration to FAIL in Line 17, and moreover, only such edges are causing the iteration to FAIL in Line 17. Thus, the number of such bad edges is at most $|\Phi| \cdot 4\Delta$.

To summarize, the total number of bad edges is at most $(|\Phi| + |C|) + |\Phi| \cdot 4\Delta < \lambda/2$, where the last inequality holds since $|\Phi|+|C| < \lambda/(10\Delta)$ (see Line 3). This concludes the proof. □

Similarly, we can bound the expected runtime of each iteration of the **while** loop in Algorithm 1.

Lemma 3.6. *Each iteration of the* **while** *loop in Algorithm 1 takes $\tilde{O}(m/\lambda)$ time in expectation, regardless of the outcome of previous iterations.*

Proof. Alternating path flips can be done in time proportional to the path lengths using standard data structures, so we only need to analyze the path lengths. Fix any given iteration of the **while** loop. At the start of this iteration, we can classify the edges in U into one of the following three categories: An edge $e \in U$ is of "Type I" if the iteration ends at Line 7 when we sample e, is of "Type II" if the iteration ends at Line 11 when we sample e, and is of "Type III" otherwise. Let λ_1, λ_2 and λ_3 respectively denote the total number of Type I, Type II and Type III edges, with $\lambda_1 + \lambda_2 + \lambda_3 = \lambda$. For every Type III edge $e = (u, v) \in U$, we refer to the alternating paths P_u and P_v (see Line 14 and Line 15) as the "characteristic alternating paths" for e. Let $\mathcal{P}_3$ denote the collection of characteristic alternating paths of all Type III edges. Since the set of Type III edges is a subset of U, they form a matching, and hence different paths in $\mathcal{P}_3$ have different starting points. Furthermore, every path in $\mathcal{P}_3$ is either a maximal $\{\alpha, \cdot\}$-alternating path or a maximal $\{\beta, \cdot\}$-alternating path. Accordingly, Theorem 3.3 implies that the total length of all the paths in $\mathcal{P}_3$ is at most $O((m/\Delta) \cdot \Delta) = O(m)$.

Now, if at the start of the iteration, we happen to sample either a Type I or a Type II edge $e \in U$, then the concerned iteration takes $O(1)$ time. In the paragraph below, we condition on the event that the edge $e = (u, v) \in U$ sampled at the start of the iteration is of Type III.

Using appropriate data structures, the time taken to implement the concerned iteration is proportional (up to $\tilde{O}(1)$ factors) to the lengths of the alternating paths P_u and P_v (see Line 14 and Line 15). The key observation is that for each $x \in \{u, v\}$, the path P_x is sampled almost uniformly (i.e., with probability $\Theta(1/\lambda_3)$) from the collection $\mathcal{P}_3$. Since the total length of all the paths in $\mathcal{P}_3$ is $O(m)$, it follows that the expected length of each of the paths P_u, P_v is $O(m/\lambda_3)$.

To summarize, with probability λ_3/λ, we sample a Type III edge at the start of the concerned iteration of the **while** loop, and conditioned on this event the expected time spent on that iteration is $\tilde{O}(m/\lambda_3)$. In contrast, if we sample a Type I or a Type II edge at the start of the concerned iteration, then the time spent on that iteration is $O(1)$. This implies that we spend at most $\tilde{O}(m/\lambda_3) \cdot (\lambda_3/\lambda) + O(1) = \tilde{O}(m/\lambda)$ expected time per iteration of the **while** loop. □

Finally, we show that in the very last step of the algorithm, the **for** loop in Line 22, the algorithm succeeds in coloring a constant fraction of popular edges.

Lemma 3.7. *The* **for** *loop in Line 22 extends the coloring χ to at least half of the edges in Φ.*

Proof. Consider any given iteration of the **for** loop where we pick an edge $e = (u, v) \in \Phi'$ in Line 22, where w.l.o.g. $\alpha \in \text{miss}_\chi(u)$ and $\beta \in \text{miss}_\chi(v)$. It is easy to verify that during this iteration, we successfully extend the coloring χ to e, either in Line 26 or in Line 30. In the latter case, we crucially rely on the fact that the graph G is bipartite (see Line 29), and hence the maximal $\{\alpha, \beta\}$-alternating path $P_u^\star$ starting at u cannot end at v; in fact, this is the only place where we rely on the biparteness of G. Lines 31 and 32 ensure that the following invariant is satisfied: For every edge $e' = (u', v') \in \Phi'$, we have $\alpha \in \text{miss}_\chi(u')$ and $\beta \in \text{miss}_\chi(v')$, i.e., the edge e' is popular; indeed, Lemma 3.4 implies that this invariant holds just before the **for** loop starts (see Line 21), and any edge $e' \in \Phi'$ that may violate this invariant at a later stage, which may only occur due to flipping an alternating path that ends at a node incident on e' (in Line 31), is removed from Φ' (in Line 32).

Now, the lemma follows from the observation that each time we successfully extend the coloring to one edge e in Φ', we remove at most one other edge $e' \neq e$ from Φ' (see Line 32), due to the vertex-disjointness of the edges in $U \supseteq \Phi \supseteq \Phi'$. □

We are now ready to conclude the running time analysis of Algorithm 1 and establish the required lower bound on the number of newly colored edges in U under χ by this algorithm.

Lemma 3.8. *Algorithm 1 takes $\tilde{O}(m/\Delta)$ time in expectation and extends χ to $\Omega(\lambda/\Delta)$ new edges.*

Proof. We start with the runtime analysis:

- Line 1 can be implemented in $\tilde{O}(1)$ time using the auxiliary data structure, and Lines 2 and 21 take constant time.
- Next, we bound the running time of the **while** loop (Line 3). For any integer $k \geq 0$, let $T(k)$ denote the expected runtime of **while** loop if we start the loop under the condition that $|\Phi|+|C| = k$. We are interested in $T(0)$ and we know that $T(\lambda/10\Delta) = O(1)$ by the termination condition of the loop. By Lemma 3.5 and Lemma 3.6, for any $0 < k < \lambda/10\Delta$, we have,
$$T(k) \leq \tilde{O}(m/\lambda) + \frac{1}{2} \cdot T(k) + \frac{1}{2} \cdot T(k+1),$$
where we additionally used the monotonicity of $T(\cdot)$, as well as the fact that each while-loop of Algorithm 1 has expected runtime $\tilde{O}(m/\lambda)$ regardless of previous iterations, according to Lemma 3.6. Thus, $T(k) \leq T(k+1) + \tilde{O}(m/\lambda)$ and hence $T(0) \leq \lambda/10\Delta \cdot \tilde{O}(m/\lambda) = \tilde{O}(m/\Delta)$.
- Finally, since the total number of edges with colors α and β just before Line 22 is $O(m/\Delta)$ (see Theorem 3.3), the **for** loop can be implemented in $\tilde{O}(m/\Delta)$ time deterministically in a straightforward manner.

Thus, the total runtime is $\tilde{O}(m/\Delta)$ in expectation.

We now establish the bound on the number of newly colored edges. When the **while** loop terminates, we have $|\Phi| + |C| \geq \lambda/(10\Delta)$ (see Line 3), and all the edges in C are colored under χ (see Lemma 3.4). Next, by Lemma 3.7, the **for** loop in Line 22 further extends the coloring χ to a constant fraction of the edges in Φ, by only flipping $\{\alpha, \beta\}$-alternating paths. Consequently, we get at least $\Omega(\lambda/\Delta)$ newly colored edges in U under χ. This concludes the proof. □

We can now conclude the proof of Lemma 3.2. To achieve the algorithm in this lemma, we simply run Algorithm 1 in parallel $\Theta(\log n)$ times and use the coloring of whichever one finishes first (and terminate the rest at that point). This ensures the high probability guarantee of Lemma 3.2 still in $\tilde{O}(m/\Delta)$ runtime. This concludes the entire proof.

3.3 Extension to General Graphs

In our Lemma 3.2, we crucially need the graph G to be bipartite while executing Lines 28 to 30 in Algorithm 1. Otherwise, if G contains odd cycles, then the maximal $\{\alpha, \beta\}$-alternating path $P_u^\star$ starting from u can end at v. In that case, the color β will no longer be missing at v once we flip the path $P_v^\star$, and so we will not be able to extend the coloring χ to the edge (u, v) via $\chi(u, v) \leftarrow \beta$. We shall emphasize that this is not a minor technicality, but rather the key reason general graphs are not necessarily Δ edge colorable and rather require $(\Delta + 1)$ colors.

The standard machinery to deal with this issue is the Vizing fan (Section 4). However, if we try to use Vizing fans inside the framework of Algorithm 1 in a naive manner, then we lose control over one of the colors in the alternating path being flipped while extending the coloring to an edge, leading to a weaker averaging argument and a running time of $\tilde{O}(\Delta m)$ instead of $\tilde{O}(m)$.

To address this bottleneck, one of our key conceptual contributions is to focus on Vizing fans with respect to an object called a *separable collection of u-components* (see Section 5). Using this concept, in Section 6 we present our algorithmic framework in general graphs. Our main result (see Theorem 6.1) relies upon two fundamental subroutines. The second subroutine (see Lemma 6.3) generalizes Algorithm 1 presented in this section. In contrast, the first subroutine (see Lemma 6.2) either efficiently extends the current coloring to a constant fraction of the uncolored edges, or changes the colors of some edges in the input graph so as to create a situation whereby we can invoke Lemma 6.3. Due to page limit, the proofs of Lemmas 6.2 and 6.3 are deferred to the full version of our paper [4].

4 Preliminaries: Vizing Fans and Vizing Chains

We now define the notion of *Vizing fans*, which has been used extensively in the edge coloring literature [39, 52, 54].

Definition 4.1 (Vizing fan). A *Vizing fan* is a sequence

$$\mathsf{F} = (u, \alpha), (v_1, c_1), \ldots, (v_k, c_k)$$

where $u, v_1, \ldots, v_k$ are distinct vertices and $c_1, \ldots, c_k$ are colors such that

(1) $\alpha \in \mathsf{miss}_\chi(u)$ and $c_i \in \mathsf{miss}_\chi(v_i)$ for all $i \in [k]$.
(2) $v_1, \ldots, v_k$ are distinct neighbours of u.
(3) $\chi(u, v_1) = \perp$ and $\chi(u, v_i) = c_{i-1}$ for all $i > 1$.
(4) Either $c_k \in \mathsf{miss}_\chi(u)$ or $c_k \in \{c_1, \ldots, c_{k-1}\}$.

We say that the Vizing fan $\mathsf{F} = (u, \alpha), (v_1, c_1), \ldots, (v_k, c_k)$ is *α-primed*, has *center* u and *leaves* $v_1, \ldots, v_k$. We refer to c_i as the color of v_i within F. A crucial property is that we can *rotate* colors around the Vizing fan F by setting $\chi(u, v_1) \leftarrow c_1, \ldots, \chi(u, v_{i-1}) \leftarrow c_{i-1}, \chi(u, v_i) \leftarrow \perp$ for any $i \in [k]$. We say that F is a *trivial* Vizing fan if $c_k \in \mathsf{miss}_\chi(u)$. Note that, if F is trivial, we can immediately extend the coloring χ to (u, v_1) by rotating all the colors around F and setting $\chi(u, v_k) \leftarrow c_k$.

Algorithm 2 describes the standard procedure used to construct Vizing fans. As input, it takes a vertex u and a color $\alpha \in \mathsf{miss}_\chi(u)$, and returns an α-primed Vizing fan with center u.

The Algorithm Vizing. We now describe the algorithm Vizing that, given a Vizing fan $\mathsf{F} = (u, \alpha), (v_1, c_1), \ldots, (v_k, c_k)$ as input, extends the coloring χ to the edge (u, v_1) by building a Vizing chain. Algorithm 3 gives a formal description of this procedure.

Algorithm 2: `Vizing-Fan`(u, v, α)

```
For each x ∈ V, let clr(x) ∈ miss_χ(x)
k ← 1 and v_1 ← v
c_1 ← clr(v_1)
while c_k ∉ {c_1, ..., c_{k-1}} and c_k ∉ miss_χ(u) do
    Let (u, v_{k+1}) be the edge with color χ(u, v_{k+1}) = c_k
    c_{k+1} ← clr(v_{k+1})
    k ← k + 1
return (u, α), (v_1, c_1), ..., (v_k, c_k)
```

Algorithm 3: Vizing(F)

```
if F is trivial then
    χ(u, v_i) ← c_i for all i ∈ [k]
    return
Let P denote the maximal {α, c_k}-alternating path starting at u
Extend χ to (u, v_1) by flipping the path P and rotating colors in F (details in Lemma 4.2)
```

Thus, running Vizing(F) extends the coloring χ to the uncolored edge (u, v_1) by rotating colors in the Vizing fan F and flipping the colors of the alternating path P. We sometimes refer to the process of running Vizing(F) as *activating* the Vizing fan F.

Lemma 4.2. *Algorithm 3 extends the coloring χ to the edge (u, v_1) in time $O(\Delta + |P|)$.*

Given a Vizing fan F, we denote the path P considered by Algorithm 3 by `Vizing-Path`(F). If the Vizing fan F is trivial, then `Vizing-Path`(F) denotes an empty path $\emptyset$.

5 Basic Building Blocks

In this section, we introduce the notation of u-fans, u-edges and separable collections, which are the definitions that work as the basic building blocks for our algorithms.

5.1 U-Fans and U-Edges

We begin by defining the notion of *u-fans* that was used by [39].[3]

Definition 5.1 (u-fan, [39]). A *u-fan* is a tuple $\mathsf{f} = (u, v, w, \alpha, \beta)$ where u, v and w are distinct vertices and α and β are distinct colors such that:

(1) (u, v) and (u, w) are uncolored edges.
(2) $\alpha \in \mathsf{miss}_\chi(u)$ and $\beta \in \mathsf{miss}_\chi(v) \cap \mathsf{miss}_\chi(w)$.

We say that u is the *center* of f and that v and w are the *leaves* of f. We also say that the u-fan f is *$\{\alpha, \beta\}$-primed* and define $c_\mathsf{f}(u) := \alpha$, $c_\mathsf{f}(v) := \beta$ and $c_\mathsf{f}(w) := \beta$ (i.e. given a vertex $x \in \mathsf{f}$, $c_\mathsf{f}(x)$ is the available color that f 'assigns' to x).

[3] The term 'u-fan' was originally introduced by [39] as an abbreviation for 'uncolored fan'.

Activating U-Fans. Let f be an $\{\alpha, \beta\}$-primed u-fan with center u and leaves v and w. The key property of u-fans is that at most one of the $\{\alpha, \beta\}$-alternating paths starting at v or w ends at u. Say that the $\{\alpha, \beta\}$-alternating path starting at v does not end at u. Then, after flipping this $\{\alpha, \beta\}$-alternating path, both u, v are missing color α. Thus, we can extend the coloring χ by assigning $\chi(u, v) \leftarrow \alpha$. We refer to this as *activating* the u-fan f.

We also define the notion of a *u-edge* similarly to u-fans.

Definition 5.2 (u-edge). A *u-edge* is a tuple $\mathsf{e} = (u, v, \alpha)$ where (u, v) is an uncolored edge and α is a color such that $\alpha \in \text{miss}_\chi(u)$.

We say that u is the *center* of e and that α is the *active color* of e. For notational convenience, we also say that the u-edge e is *α-primed* and define $c_{\mathsf{e}}(u) := \alpha$ and $c_{\mathsf{e}}(v) = \perp$.[4]

Collections of U-Components. While working with both u-fans and u-edges simultaneously, we sometimes refer to some g that is either a u-fan or a u-edge as a *u-component*. Throughout this paper, we often consider collections of u-components $\mathcal{U}$. We only use the term 'collection' in this context, so we abbreviate 'collection of u-components' by just 'collection'. We will be particularly interested in collections satisfying the following useful property, which we refer to as *separability*.

Definition 5.3 (Separable Collection). Let χ be a partial $(\Delta+1)$-edge coloring of G and $\mathcal{U}$ be a collection of u-components (i.e. u-fans and u-edges). We say that the collection $\mathcal{U}$ is *separable* if the following holds:

(1) All of the u-components in $\mathcal{U}$ are edge-disjoint.
(2) For each $x \in V$, the colors in the multi-set $C_{\mathcal{U}}(x) := \{c_{\mathsf{g}}(x) \mid \mathsf{g} \in \mathcal{U},\ x \in \mathsf{g}\}$ are distinct.

We remark that the second property of this definition is rather important because we need to ensure that different u-components are not interfering with each other when they share common vertices. Check Figure 4 for an illustration.

Damaged U-Components. Suppose we have a partial $(\Delta+1)$-edge coloring χ and a separable collection $\mathcal{U}$ w.r.t. χ. Now, suppose that we modify the coloring χ. We say that a u-component $\mathsf{g} \in \mathcal{U}$ is *damaged* if it is no longer a u-component w.r.t. the new coloring χ.

We note that this can only happen for one of the following two reasons: (1) one of the uncolored edges in g is now colored, or (2) there is a vertex $x \in \mathsf{g}$ such the color $c_{\mathsf{g}}(x)$ that g assigns to x is no longer missing at x.

The following lemma shows that flipping the colors of an alternating path cannot damage many u-components in a separable collection $\mathcal{U}$.

Lemma 5.4. *Let χ be a partial $(\Delta+1)$-edge coloring of a graph G, $\mathcal{U}$ a separable collection and P a maximal alternating path in χ. Then flipping the colors of the alternating path P damages at most 2 u-components in $\mathcal{U}$ (corresponding to the two endpoints of the path).*

[4]Whenever we refer to an "uncolored edge e", we are referring to an edge $e \in E$ such that $\chi(e) = \perp$, whereas a 'u-edge e' always refers to the object from Definition 5.2 and is denoted in bold.

5.2 Vizing Fans in Separable Collections

Within our algorithm, we only construct Vizing fans and Vizing chains in a setting where there is some underlying separable collection $\mathcal{U}$. To ensure that activating and rotating colors around Vizing fans does not damage too many u-components, we choose the missing colors involved in Vizing fan constructions so that they 'avoid' the colors assigned to the u-components in $\mathcal{U}$.

Definition 5.5. Let $\mathcal{U}$ be a separable collection and

$$\mathsf{F} = (u, \alpha), (v_1, c_1), \ldots, (v_k, c_k)$$

be a Vizing fan. We say that the Vizing fan F is *$\mathcal{U}$-avoiding* if $c_i \in \text{miss}_\chi(v_i) \setminus C_{\mathcal{U}}(v_i)$ for each leaf $v_i \in \mathsf{F}$.

We say that a Vizing fan F *is a Vizing fan of the u-edge* $\mathsf{e} = (u, v, \alpha)$ if F is α-primed, has center u and its first leaf is v. The following lemma shows that we can always find a $\mathcal{U}$-avoiding Vizing fan for a u-edge.

Lemma 5.6. *Given a u-edge $\mathsf{e} \in \mathcal{U}$, there exists a $\mathcal{U}$-avoiding Vizing fan F of e. Furthermore, we can compute such a Vizing fan in $O(\Delta)$ time.*

The lemma below describes some crucial properties of $\mathcal{U}$-avoiding Vizing fans.

Lemma 5.7. *Let χ be a $(\Delta+1)$-edge coloring of a graph G and $\mathcal{U}$ be a separable collection. For any u-edge $\mathsf{e} = (u, v, \alpha) \in \mathcal{U}$ with a $\mathcal{U}$-avoiding Vizing fan F, we have the following:*

(1) Rotating colors around F does not damage any u-component in $\mathcal{U} \setminus \{\mathsf{e}\}$.
(2) Calling Vizing(F) *damages at most one u-component in $\mathcal{U} \setminus \{\mathsf{e}\}$. Furthermore, we can identify the damaged u-component in $O(1)$ time.*

6 The Main Algorithm

We are now ready to present our main technical result, which is a slightly weaker version of Theorem 1.1, and focuses on achieving a near-linear time algorithm for $(\Delta+1)$ edge coloring (instead of the exact $O(m \log n)$ time in Theorem 1.1; see the remark after that theorem).

Theorem 6.1. *There is a randomized algorithm that, given any simple undirected graph $G = (V, E)$ on n vertices and m edges with maximum degree Δ, finds a $(\Delta+1)$-edge coloring of G in $\tilde{O}(m)$ time with high probability.*

Our main algorithm consists of two main components. The first component is an algorithm called Construct-U-Fans that takes a partial $(\Delta+1)$-edge coloring χ with λ uncolored edges and either extends χ to $\Omega(\lambda)$ of these edges or modifies the coloring to construct a separable collection of $\Omega(\lambda)$ u-fans. Lemma 6.2 summarizes the behavior of this algorithm.

Lemma 6.2. *Given a graph G, a partial $(\Delta+1)$-edge coloring χ of G and a set of λ uncolored edges U, the algorithm* Construct-U-Fans *does one of the following in $O(m + \Delta\lambda)$ time:*

(1) Extends the coloring to $\Omega(\lambda)$ uncolored edges.
(2) Modifies χ to obtain a separable collection of $\Omega(\lambda)$ u-fans $\mathcal{U}$.

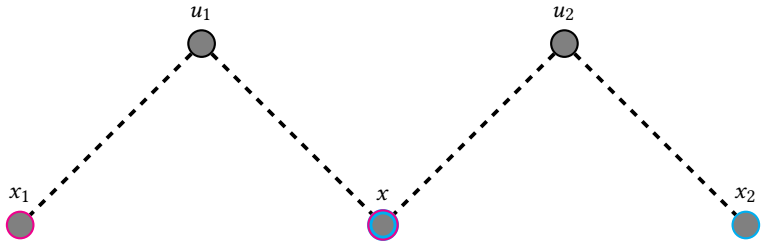


Figure 4: This picture shows two u-fans $(u_1, x_1, x, *, \beta_1)$ and $(u_2, x_2, x, *, \beta_2)$ sharing a common vertex x. The separable condition requires that $\beta_1 \neq \beta_2$; for instance β_1, β_2 could be magenta and cyan as shown here.

The second component is an algorithm called Color-U-Fans that takes a collection of λ u-fans and extends the coloring to $\Omega(\lambda)$ of the edges in the u-fans. Lemma 6.3 summarizes the behavior of this algorithm. The reader may find it helpful to keep in mind that the algorithm for Lemma 6.3 is a generalization of algorithm for Lemma 3.2 in Section 3.

Lemma 6.3. *Given a graph G, a partial $(\Delta+1)$-edge coloring χ of G and a separable collection of λ u-fans $\mathcal{U}$, the algorithm* Color-U-Fans *extends χ to $\Omega(\lambda)$ edges in $O(m \log n)$ time w.h.p.*

In the full version [4], we prove Lemmas 6.2 and 6.3 respectively. Using these lemmas, we now show how to efficiently extend an edge coloring χ to the remaining uncolored edges.

Lemma 6.4. *Given a graph G and a partial $(\Delta+1)$-edge coloring χ of G with λ uncolored edges U, we can extend χ to the remaining uncolored edges in time $O((m+\Delta\lambda)\log^2 n)$ w.h.p.*

Proof. Let U denote the set of edges that are uncolored by χ. We can then apply Lemma 6.2 to either extend χ to a constant fraction of the edges in U or construct a separable collection of $\Omega(\lambda)$ u-fans $\mathcal{U}$ in $O(m+\Delta\lambda)$ time. In the second case, we can then apply Lemma 6.3 to color $\Omega(\lambda)$ of the edges contained in these u-fans in $O(m \log n)$ time w.h.p. Thus, we can extend χ to a constant fraction of the uncolored edges in $O(m \log n + \Delta\lambda)$ time w.h.p. After repeating this process $O(\log \lambda) \leq O(\log n)$ many times, no edges remain uncolored. Thus, we can extend the coloring χ to the entire graph in $O((m+\Delta\lambda)\log^2 n)$ time w.h.p. □

We now use this lemma to prove Theorem 6.1.

Proof of Theorem 6.1. We prove this by applying Lemma 6.4 to the standard Euler partition framework [39, 52]. Given a graph G, we partition it into two edge-disjoint subgraphs G_1 and G_2 on the same vertex set such that $\Delta(G_i) \leq \lceil \Delta/2 \rceil$ for each G_i, where $\Delta(G_i)$ denotes the maximum degree of G_i. We then recursively compute a $(\Delta(G_i)+1)$-edge coloring χ_i for each G_i. Combining χ_1 and χ_2, we obtain a $(\Delta+3)$-edge coloring χ of G. We then uncolor the two smallest color classes in χ, which contain $O(m/\Delta)$ edges, and apply Lemma 6.4 to recolor all of the uncolored edges in χ using only $\Delta+1$ colors in $O(m \log^2 n)$ time w.h.p.

To show that the total running time of the algorithm is $O(m \log^3 n)$, first note that the depth of the recursion tree is $O(\log \Delta)$. Next, consider the i^{th} level of the recursion tree, for an arbitrary $i = O(\log \Delta)$: we have 2^i edge-disjoint subgraphs $G_1, \ldots, G_{2^i}$ such that $\Delta(G_j) \leq O(\Delta/2^i)$ and $\sum_{j=1}^{2^i} |E(G_j)| = m$. Since the total running time at recursion level i is $O(m \log^2 n)$ and the depth of the recursion tree is $O(\log \Delta)$, it follows that the total running time is $O(m \log^3 n)$ w.h.p. □

Acknowledgments

Part of this work was conducted while Sepehr Assadi and Soheil Behnezhad were visiting the Simons Institute for the Theory of Computing as part of the Sublinear Algorithms program.

Sepehr Assadi is supported in part by a Sloan Research Fellowship, an NSERC Discovery Grant (RGPIN-2024-04290), and a Faculty of Math Research Chair grant from University of Waterloo. Soheil Behnezhad is funded by an NSF CAREER award CCF-2442812 and a Google Faculty Research Award. Martín Costa is supported by a Google PhD Fellowship. Shay Solomon is funded by the European Union (ERC, DynOpt, 101043159). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Shay Solomon is also funded by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, and the United States National Science Foundation (NSF). Tianyi Zhang is supported by funding from the starting grant "A New Paradigm for Flow and Cut Algorithms" (no. TMSGI2_218022) of the Swiss National Science Foundation.

References

[1] Noga Alon. 2003. A simple algorithm for edge-coloring bipartite multigraphs. *Inform. Process. Lett.* 85, 6 (2003), 301–302. doi:10.1016/S0020-0190(02)00446-5

[2] Eshrat Arjomandi. 1982. An efficient algorithm for colouring the edges of a graph with $\Delta+1$ colours. *INFOR: Information Systems and Operational Research* 20, 2 (1982), 82–101. doi:10.1080/03155986.1982.11731850

[3] Sepehr Assadi. 2025. Faster Vizing and Near-Vizing Edge Coloring Algorithms. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. doi:10.1137/1.9781611978322.165

[4] Sepehr Assadi, Soheil Behnezhad, Sayan Bhattacharya, Martín Costa, Shay Solomon, and Tianyi Zhang. 2024. Vizing's Theorem in Near-Linear Time. *CoRR* abs/2410.05240 (2024). doi:10.48550/ARXIV.2410.05240 arXiv:2410.05240

[5] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. 2022. Distributed edge coloring in time polylogarithmic in Δ. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*. 15–25. doi:10.1145/3519270.3538440

[6] Leonid Barenboim and Tzalik Maimon. 2017. Fully-Dynamic Graph Algorithms with Sublinear Time Inspired by Distributed Computing. In *International Conference on Computational Science (ICCS) (Procedia Computer Science, Vol. 108)*. Elsevier, 89–98. doi:10.1016/j.procs.2017.05.098

[7] Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Marina Knittel, and Hamed Saleh. 2019. Streaming and Massively Parallel Algorithms for Edge Coloring. In *27th Annual European Symposium on Algorithms (ESA) (LIPIcs, Vol. 144)*. 15:1–15:14. doi:10.4230/LIPIcs.ESA.2019.15

[8] Anton Bernshteyn. 2022. A fast distributed algorithm for $(\Delta+1)$-edge-coloring. *J. Comb. Theory, Ser. B* 152 (2022), 319–352. doi:10.1016/j.jctb.2021.10.004

[9] Anton Bernshteyn and Abhishek Dhawan. 2023. Fast algorithms for Vizing's theorem on bounded degree graphs. *CoRR* abs/2303.05408 (2023).

[10] Anton Bernshteyn and Abhishek Dhawan. 2024. A linear-time algorithm for $(1+\epsilon)\Delta$-edge-coloring. *arXiv preprint arXiv:2407.04887* (2024).

[11] Sayan Bhattacharya, Din Carmon, Martín Costa, Shay Solomon, and Tianyi Zhang. 2024. Faster $(\Delta+1)$-Edge Coloring: Breaking the $m\sqrt{n}$ Time Barrier. In *65th IEEE Symposium on Foundations of Computer Science (FOCS)*. doi:10.1109/FOCS61266.2024.00128

[12] Sayan Bhattacharya, Deeparnab Chakrabarty, Monika Henzinger, and Danupon Nanongkai. 2018. Dynamic Algorithms for Graph Coloring. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 1–20. doi:10.1137/1.9781611975031.1

[13] Sayan Bhattacharya, Martín Costa, Nadav Panski, and Shay Solomon. 2024. Arboricity-Dependent Algorithms for Edge Coloring. In *19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT) (LIPIcs, Vol. 294)*. 12:1–12:15. doi:10.4230/LIPIcs.SWAT.2024.12

[14] Sayan Bhattacharya, Martín Costa, Nadav Panski, and Shay Solomon. 2024. Density-Sensitive Algorithms for $(\Delta+1)$-Edge Coloring. In *32nd Annual European Symposium on Algorithms, ESA 2024 (LIPIcs, Vol. 308)*. 23:1–23:18. doi:10.4230/LIPIcs.ESA.2024.23

[15] Sayan Bhattacharya, Martín Costa, Nadav Panski, and Shay Solomon. 2024. Nibbling at Long Cycles: Dynamic (and Static) Edge Coloring in Optimal Time. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. doi:10.1137/1.9781611977912.122

[16] Sayan Bhattacharya, Martín Costa, Shay Solomon, and Tianyi Zhang. 2025. Even Faster $(\Delta+1)$-Edge Coloring via Shorter Multi-Step Vizing Chains. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. doi:10.1137/1.9781611978322.167

[17] Sayan Bhattacharya, Fabrizio Grandoni, and David Wajc. 2021. Online Edge Coloring Algorithms via the Nibble Method. In *Proceedings of theACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2830–2842. doi:10.1137/1.9781611976465.168

[18] Joakim Blikstad, Ola Svensson, Radu Vintan, and David Wajc. 2024. Online Edge Coloring is (Nearly) as Easy as Offline. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*. ACM. doi:10.1145/3618260.3649741

[19] Joakim Blikstad, Ola Svensson, Radu Vintan, and David Wajc. 2025. Deterministic Online Bipartite Edge Coloring. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. doi:10.1137/1.9781611978322.49

[20] Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. 2020. Distributed Edge Coloring and a Special Case of the Constructive Lovász Local Lemma. *ACM Trans. Algorithms* 16, 1 (2020), 8:1–8:51. doi:10.1145/3365004

[21] Shiri Chechik, Doron Mukhtar, and Tianyi Zhang. 2024. Streaming Edge Coloring with Subquadratic Palette Size. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia (LIPIcs, Vol. 297)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 40:1–40:12. doi:10.4230/LIPIcs.ICALP.2024.40

[22] Aleksander Bjørn Grodt Christiansen. 2023. The Power of Multi-step Vizing Chains. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1013–1026. doi:10.1145/3564246.3585105

[23] Aleksander B. G. Christiansen. 2024. Deterministic Dynamic Edge-Colouring. *CoRR* abs/2402.13139 (2024). https://doi.org/10.48550/arXiv.2402.13139

[24] Aleksander B. G. Christiansen, Eva Rotenberg, and Juliette Vlieghe. 2024. Sparsity-Parameterised Dynamic Edge Colouring. In *19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT) (LIPIcs, Vol. 294)*. 20:1–20:18. doi:10.4230/LIPIcs.SWAT.2024.20

[25] Marek Chrobak and Takao Nishizeki. 1990. Improved edge-coloring algorithms for planar graphs. *Journal of Algorithms* 11, 1 (1990), 102–116. doi:10.1016/0196-6774(90)90032-A

[26] Marek Chrobak and Moti Yung. 1989. Fast algorithms for edge-coloring planar graphs. *Journal of Algorithms* 10, 1 (1989), 35–51. doi:10.1016/0196-6774(89)90022-9

[27] Ilan Reuven Cohen, Binghui Peng, and David Wajc. 2019. Tight Bounds for Online Edge Coloring. In *60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 1–25. doi:10.1109/FOCS.2019.00010

[28] Richard Cole and John Hopcroft. 1982. On edge coloring bipartite graphs. *SIAM J. Comput.* 11, 3 (1982), 540–546. doi:10.1137/0211043

[29] Richard Cole and Łukasz Kowalik. 2008. New linear-time algorithms for edge-coloring planar graphs. *Algorithmica* 50, 3 (2008), 351–368. doi:10.1007/s00453-007-9044-3

[30] Richard Cole, Kirstin Ost, and Stefan Schirra. 2001. Edge-Coloring Bipartite Multigraphs in $O(E \log D)$ Time. *Comb.* 21, 1 (2001), 5–12. doi:10.1007/s004930170002

[31] Peter Davies. 2023. Improved Distributed Algorithms for the Lovász Local Lemma and Edge Coloring. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 4273–4295. doi:10.1137/1.9781611977554.ch163

[32] Abhishek Dhawan. 2024. Edge-Coloring Algorithms for Bounded Degree Multigraphs. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, David P. Woodruff (Ed.). SIAM, 2120–2157. doi:10.1137/1.9781611977912.77

[33] Abhishek Dhawan. 2024. A Simple Algorithm for Near-Vizing Edge-Coloring in Near-Linear Time. *arXiv preprint arXiv:2407.16585* (2024).

[34] Ran Duan, Haoqing He, and Tianyi Zhang. 2019. Dynamic edge coloring with improved approximation. In *30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. doi:10.1137/1.9781611975482.117

[35] Aditi Dudeja, Rashmika Goswami, and Michael Saks. 2025. Randomized Greedy Online Edge Coloring Succeeds for Dense and Randomly-Ordered Graphs. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. doi:10.1137/1.9781611978322.168

[36] Michael Elkin and Ariel Khuzman. 2024. Deterministic Simple $(1+\epsilon)$-Edge-Coloring in Near-Linear Time. *arXiv preprint arXiv:2401.10538* (2024).

[37] Michael Elkin, Seth Pettie, and Hsin-Hao Su. 2014. $(2\Delta-1)$-Edge-Coloring is Much Easier than Maximal Matching in the Distributed Setting. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 355–370. doi:10.1137/1.9781611973730.26

[38] Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. 2017. Deterministic distributed edge-coloring via hypergraph maximal matching. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 180–191. doi:10.1109/FOCS.2017.25

[39] Harold N Gabow, Takao Nishizeki, Oded Kariv, Daneil Leven, and Osamu Terada. 1985. Algorithms for edge coloring. *Technical Rport* (1985).

[40] Mohsen Ghaffari, Fabian Kuhn, Yannic Maus, and Jara Uitto. 2018. Deterministic distributed edge-coloring with fewer colors. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 418–430. doi:10.1145/3188745.3188906

[41] Prantar Ghosh and Manuel Stoeckl. 2024. Low-Memory Algorithms for Online Edge Coloring. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia (LIPIcs, Vol. 297)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 71:1–71:19. doi:10.4230/LIPIcs.ICALP.2024.71

[42] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. 2010. Perfect matchings in $O(n \log n)$ time in regular bipartite graphs. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*. 39–46. doi:10.1145/1806689.1806697

[43] Jan Grebík and Oleg Pikhurko. 2020. Measurable versions of Vizing's theorem. *Advances in Mathematics* 374 (2020), 107378. doi:10.1016/j.aim.2020.107378

[44] Ian Holyer. 1981. The NP-completeness of edge-coloring. *SIAM Journal on computing* 10, 4 (1981), 718–720. doi:10.1137/0210055

[45] Howard J Karloff and David B Shmoys. 1987. Efficient parallel algorithms for edge coloring problems. *Journal of Algorithms* 8, 1 (1987), 39–52. doi:10.1016/0196-6774(87)90026-5

[46] Lukasz Kowalik. 2024. Edge-Coloring Sparse Graphs with Δ Colors in Quasilinear Time. In *32nd Annual European Symposium on Algorithms, ESA 2024 (LIPIcs, Vol. 308)*. 81:1–81:17. doi:10.4230/LIPIcs.ESA.2024.81

[47] Janardhan Kulkarni, Yang P. Liu, Ashwin Sah, Mehtaab Sawhney, and Jakub Tarnawski. 2022. Online edge coloring via tree recurrences and correlation decay. In *54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. ACM, 104–116. doi:10.1145/3519935.3519986

[48] Alessandro Panconesi and Romeo Rizzi. 2001. Some simple distributed algorithms for sparse networks. *Distributed computing* 14, 2 (2001), 97–100. doi:10.1007/PL00008932

[49] Amin Saberi and David Wajc. 2021. The Greedy Algorithm Is not Optimal for On-Line Edge Coloring. In *48th International Colloquium on Automata, Languages, and Programming (ICALP) (LIPIcs, Vol. 198)*. 109:1–109:18. doi:10.4230/LIPIcs.ICALP.2021.109

[50] Mohammad Saneian and Soheil Behnezhad. 2024. Streaming Edge Coloring with Asymptotically Optimal Colors. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia (LIPIcs, Vol. 297)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 121:1–121:20. doi:10.4230/LIPIcs.ICALP.2024.121

[51] Claude E Shannon. 1949. A theorem on coloring the lines of a network. *Journal of Mathematics and Physics* 28, 1-4 (1949), 148–152. doi:10.1002/sapm1949281148

[52] Corwin Sinnamon. 2019. Fast and simple edge-coloring algorithms. *arXiv preprint arXiv:1907.03201* (2019).

[53] Hsin-Hao Su and Hoa T. Vu. 2019. Towards the locality of Vizing's theorem. In *51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, Moses Charikar and Edith Cohen (Eds.). doi:10.1145/3313276.3316393

[54] V. G. Vizing. 1964. On an estimate of the chromatic class of a p-graph. *Discret Analiz* 3 (1964), 25–30.

[55] Vadim G Vizing. 1965. The chromatic class of a multigraph. *Cybernetics* 1, 3 (1965), 32–41. doi:10.1007/BF01885700

Received 2024-11-04; accepted 2025-02-01