Simple vertex coloring algorithms

Jackson Morris
Department of Mathematics, University of California Los Angeles, Google
jrexmo@google.com

Fang Song
Department of Computer Science, Portland State University
fsong@pdx.edu

Abstract

Given a graph G with n vertices and maximum degree Δ, it is known that G admits a vertex coloring with $\Delta + 1$ colors such that no edge of G is monochromatic. This can be seen constructively by a simple greedy algorithm, which runs in time $O(n\Delta)$.

Very recently, a sequence of results (e.g., [Assadi et. al. SODA'19, Bera et. al. ICALP'20, AlonAssadi Approx/Random'20]) show randomized algorithms for $(1+\epsilon)\Delta$-coloring in the query model making $\tilde{O}(n\sqrt{n})$ queries, improving over the greedy strategy on dense graphs. In addition, a lower bound of $\Omega(n\sqrt{n})$ for any $O(\Delta)$-coloring is established on general graphs.

In this work, we give a simple algorithm for C-coloring where $C > \Delta + 1$. This algorithm makes $O(\frac{n^2}{C-\Delta})$ queries. This matches the classical lower bound for $C \geq c\Delta$ with $c \gg 1$ and a new upper bound of $O(n^{\frac{k+2}{k+1}})$ for $C = \Delta^k \geq \Delta^2$. Additionally, it can be readily adapted to a quantum query algorithm making $\tilde{O}(n^{\frac{k+3}{k+2}})$ queries, bypassing the classical lower bound when $C = c\Delta$ for $c \gg 1$. Further, we show that the algorithm presented in Assadi et. al. SODA'19 for $(\Delta+1)$-coloring can be adapted to a quantum algorithm making $\tilde{O}(n^{4/3})$ queries.

We also show initial upper and lower bounds for the very closely related problem of *verifying* whether or not a given graph coloring is valid.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Quantum computation theory

Keywords and phrases graph coloring, quantum algorithms, query model

Digital Object Identifier 10.4230/LIPIcs...

Funding This work is supported by NSF grants CCF-2041841, CCF-2042414, and CCF2054758 (CAREER). Any opinions, findings and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF).

Acknowledgements The authors would like to thank Sepher Assadi for clarifying several points regarding the work that this paper builds on.

1 Introduction

Graph coloring is a fundamental problems in discrete algorithms and graph theory. It has wide applications in resource allocation, compiler optimization, scheduling, and logistics. In this problem, one aims to assign every vertex a color such that no edge is *monochromatic*, i.e., both endpoints having the same color. Finding such a valid coloring with $k \geq 1$ colors is usually called the k-coloring problem. A critical graph parameter $\chi(G)$ is the minimum number of colors necessary to guarantee a valid coloring. In Karp's renowned result, 3-coloring as well as deciding $\chi(G)$ in general are proven NP-complete [12]. On the other hand, it is known that for any graph G, $\chi(G) \leq \Delta + 1$ where Δ is the maximum degree of G. This follows from a simple greedy algorithm based on the basic observation that any valid partial $(\Delta + 1)$-coloring of G can be completed to a $(\Delta + 1)$-coloring on the whole graph. This is because every vertex has at most Δ neighbors, there must be at least one choice of color for that vertex that does not conflict with any of its neighbors.

The $O(n\Delta)$ greedy algorithm had been the best known algorithm for $(\Delta + 1)$ vertex coloring on general graphs until recently. Assadi et al. [2] gave a new $\tilde{O}(n\sqrt{n})$ algorithm in the *query* model, where the graph's adjacency matrix is given as a black-box. This algorithm relies on a tool called *palette sparsification*, which roughly ensures that if one samples $O(\log n)$ colors independently and uniformly at random at every vertex from a palette of $\Delta + 1$ colors, then with high probability there will be a valid coloring of the graph where each vertex's color comes from the $O(\log n)$ colors sampled before. An $\tilde{O}(n\sqrt{n})$ query algorithm follows relatively immediately from this, and the same *time* complexity can be achieved with a few other novel algorithmic ideas [2]. Alon and Assadi [1] later extended this theory of palette sparsification, achieving improved results for when G is triangle free and for $(1 + \epsilon)\Delta$-coloring in general. Another important line of research concerns solving these problems in a space-efficient manner: [2, **?**] develop algorithms for $(\Delta + 1)$-coloring and related problems using a (near) linear amount of space. It is worth pointing out that an almost matching time and query lower bound of $\Omega(n\sqrt{n})$ is also proven in [2].

Our Contribution. The tools mentioned above are remarkable but at the same time employ intricate combinatorial and probabilistic arguments, when compared to the vanilla greedy algorithm and looking ahead the algorithms in this paper. In this work, we revisit the graph coloring problem, in both classical and quantum query models. We give a new randomized algorithm for C-coloring which establishes new upper bounds for $C = \Omega(\Delta)$, and a quantum variant of this algorithm follows readily that bypasses the classical lower bound in [2]. The most appealing feature of our algorithms is their simplicity. Basically, we are inspired by the $\Delta + 1$ greedy algorithm, where the algorithms progresses by extending a partial solution *locally*, and it is suitable to speed up "local" search subroutine by quantum unstructured search. In particular, the powerful Palette Sparsification tool is not needed anymore.

Result 1: A randomized algorithm for C-coloring which makes $O(\frac{n^2}{C-\Delta})$ queries in expectation and always returns a valid coloring for $C \geq \Delta + 1$.

This basic idea inherits the same general "local search" strategy. For each vertex, we sample a random color and check if it conflicts with any of its neighbors that has been assigned already. This simple approach is surprisingly efficient We show that this algorithm has expected query complexity $O(\frac{n^2}{C-\Delta})$, and as in existing works we then just need to choose between our algorithm and the vanilla greedy algorithm. In the case of $C = \Delta^k$ which gives

the desired $O(\min\{n\Delta, \frac{n^2}{\Delta^k}\}) = O(n^{\frac{k+2}{k+1}})$ expected query complexity. This is optimal for $C = c\Delta$ with $c \gg 1$ as we show in Corollary 2. This algorithm also serves a template for designing our efficient quantum algorithm. Our new algorithm only needs oracle access to the adjacency matrix (i.e., pair queries), and the vanilla greedy algorithm needs access to the adjacency list (i.e., neighbor queries). Therefore the final algorithm assumes both kinds of oracles are available.

Previously considered algorithms for $(1+\epsilon)\Delta$-coloring and $(\Delta+1)$-coloring do not differ significantly from our approach. For example, the authors of [2] mention how a simple $\tilde{O}(n\sqrt{n})$ *query* algorithm for $(\Delta+1)$-coloring follows immediately from their main theorem (this is mentioned in proof of claim 4.11). Explicitly, this can be done by sampling $O(\log n)$ colors at each vertex then constructing the conflict graph can be done with $\tilde{O}(\frac{n^2}{\Delta})$ queries and from there no further queries are necessary to find the coloring guaranteed by the palette sparsification theorem (w.h.p) - more queries are required if one wishes to find the coloring in $\tilde{O}(n^{3/2})$ time.

Our algorithm avoids the palette sparsification technique, and achieves the efficiency in some settings. In fact it improves the complexity in some other settings not focused before (e.g., sufficiently large C). As opposed to non-adpative queries in algorithms based on palette sparsification, our algorithm makes adaptive queries, which allows for repetition of a remarkably simple local procedure until the graph is fully colored. An additional advantage of our algorithm is that we only need store the current assignment of colors - the global structure of the graph is unimportant, leading to truly optimal space complexity $O(n)$.

An extremely simple subroutine, which we call **Detect-Conflict**, is employed in our algorithm. It takes a vertex v, a set of vertices S with $v \notin S$ and uses the graph oracle to determine if there are any edges of the form (u, v) actually present in $E(G)$ for any of the $u \in S$. Classically, this amounts to an exhaustive search on all such vertices $s \in S$. We notice that this subroutine can be instantiated by (variants of) Grover's quantum search algorithm [10, 15, 6, 11, 16], which we call **Q-Detect-Conflict**, under the standard quantum query model where the adjacency matrix can be queried in quantum superposition. With some additional care in the analysis we get a quantum algorithm bypassing the classical lower bound.

Result 2: A quantum algorithm which makes $\tilde{O}(\sqrt{\frac{n^3}{C-\Delta}})$ queries and returns a valid C-coloring with high probability.

We remark that although we use well-known quantum algorithmic tools, it is crucial that they are applied in the right place. For instance, one may be tempted to perform a search over just the color palette, thus requiring $O(\sqrt{\Delta}\log\Delta)$ queries, but such an approach would need a rather powerful "color validity" oracle for every vertex, which needs to be dynamically updated as various colors become forbidden for some vertices. Maintaining these oracles from standard oracles (adjacency matrix or adjacency list) would incur a significant overhead that would wipe out possible quantum speedup.

As a side product, we also show a "quantized" version of the palette-sparsification based sublinear algorithm of [2] wherein we recover the conflict graph via the same quantum search technique. This results in a quantum algorithm which makes $\tilde{O}(n^{4/3})$ queries and returns a valid $(\Delta+1)$-coloring with high probability. This is discussed further in Appendix B.

Further discussion. Figure 1 gives a visual comparison of this work and some existing vector coloring algorithms with at least $\Delta+1$ colors.

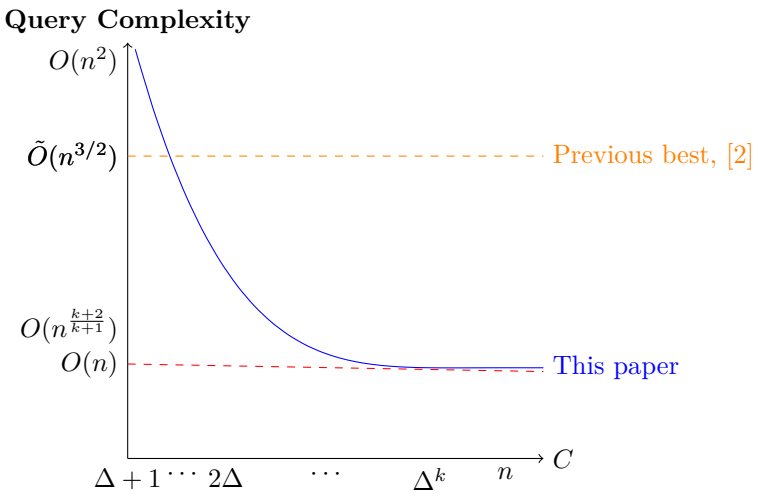

Figure 1 For C sufficiently close to $\Delta+1$, we get roughly $O(n^2)$ query complexity, but once $C = c\Delta$ with $c \gg 1$ our classical algorithm outperforms the Palette Sparsification-based method of [2]. The quantum version of our algorithm also underperforms when $C \approx \Delta+1$, but improves to $\tilde{O}(n^{\frac{k+3}{k+2}})$ for $C = \Delta^k \gg \Delta+1$ and in particular to $\tilde{O}(n^{4/3})$ for $C = c\Delta$ with $c \gg 1$ (in our analysis we choose $C \geq 3\Delta$, but this choice is arbitrary, any $C \gg \Delta+1$ suffices), bypassing the classical lower bound of $\Omega(n^{3/2})$ for $C = O(\Delta)$. When C approaches n, our algorithms approach the (trivial) bound for $C = O(n)$.

One immediate problem left open by our results in to close the missing pieces in various settings such as the proving a matching quantum or classical lower bound. The simple algorithmic ideas and the quantum speedups may also be useful to variants of coloring (defective coloring, edge coloring, etc) and dynamic graph problems.

We mention a few additional results on quantum graph algorithms and. A problem that has received considerable study in the quantum query model is triangle finding. Here one aims to either output a triangle if one exists or determine that the graph is triangle free with bounded error (or other variants such as listing all triangles). For a general graph this may require $\Omega(n^2)$ queries to the adjacency matrix oracle. However, a modified Grover search can be used, searching over triples of vertices, to achieve an $\tilde{O}(n\sqrt{n})$ quantum algorithm for triangle finding as in [8]. More sophisticated arguments in [9] improve upon this resulting, bring the query complexity down to $\tilde{O}(n^{5/4})$. Conversely, the best know lower bound for triangle finding in the quantum query model is the immediate $\Omega(n)$ bound established in [3]. Just as in triangle finding, a gap persists between the best known quantum lower bound and the best known quantum algorithm for $O(\Delta)$ coloring as shown in our work.

In addition, the problem of *maximum matching* in which one wishes to find the largest maximal matching, has been studied in the quantum query model. The very recent results of [13] gives an improved quantum algorithm for maximum matching in the adjacency matrix model which makes $O(n^{7/4})$ queries. However, a lower bound of $\Omega(n^{3/2})$ queries in the adjacency matrix model established in [5] show that a gap persists between the best known algorithm and lower bound for this problem as well. Other structural graph problems such as those relating to connected components and spanning forests have also been studied. A recent work [14], surprisingly, shows exponential quantum speedup for these problem, assuming a

more sophisticated oracle model that can answer "cut queries".

2 Preliminaries

In this paper the notation $[n]$ will refer to the set $\{1, 2, \dots n\}$ for any positive integer n. For a positive integer L, the L-coloring problem is as follows: given a graph $G = (V, E)$ and $n = |V|$ we wish to find an assignment $c : [L] = \{1, \dots L\} \to V$ such that for any edge $(x, y) \in E$ we have $c(x) \neq c(y)$. L will be referred to as the palette throughout. Such an assignment is called a *valid L-coloring of G*. Let Δ be the maximum degree of any vertex in G. As previously stated a simple greedy algorithm can be shown to always produce a valid coloring and runs in time $O(n\Delta)$. Rather than time complexity, the results in this paper will mostly be stated in terms of query complexity. The main models of computation in this paper are the graph query model and the quantum graph query model. In the standard graph query model, we assume that n and the degrees of the vertices is the only knowledge available about G before any queries have been made. The standard query model for graphs supports the following types of queries:

- Pair Queries: we can query the oracle if the edge (v_i, v_j) is present in the graph for any $i, j \in \{1, \dots n\}$. This will be denoted as

$$O_M : (i, j) \mapsto \begin{cases} 1 & (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

  This is equivalent to granting oracle access to the adjacency matrix $(M_{i,j})$.
- Neighbor Queries: we can query the oracle for the jth neighbor of vertex v_i for any $i \in \{1, \dots n\}$ and $j \in \{1, \dots \deg v_i\}$. When $j > \deg v_i$ the query returns $Null$. Similarly this is equivalent to granting oracle access to the adjacency list of the graph.

The classic greedy $\Delta + 1$-coloring algorithm can easily be implemented in the graph query model via neighbor queries. Explicitly, we can determine which colors are valid at any vertex $v \in V$ with $\deg v$ neighbor queries.

In the quantum setting, we assume standard quantum query access to the adjacency matrix. Namely we allow "pair queries" in superposition. More precisely, we encode each vertex i in a $O(\log n)$-qubit register $|i\rangle$, and assume a black-box unitary operation

$$|O_M\rangle : |i, j\rangle \mapsto (-1)^{M(i,j)}|i, j\rangle, \quad \forall i, j \in V(G)\,.$$

Each application of this unitary transformation will be referred to as a quantum query.

Quantum "neighbor query" oracle can be defined similarly (i.e., black-box unitary computing the adjacency list). However, all our algorithms will only require *classical* neighbor queries, and hence we do not specify the quantum oracle explicitly.

Throughout this paper we use the phrase *with high probability* to mean with probability at least $1 - \frac{1}{n^k}$ for a sufficiently large constant k.

A useful subroutine we will need solves the following problem. Given a vertex $v \in V$ and a collection of vertices $S \subset V \setminus v$ determine if $\{(u, v) : u \in S\} \cap E(G) = \emptyset$ given oracle access to the graph's adjacency matrix M. In other words, does v have any neighbors in S? Classically, we need to pay $O(|S|)$ queries to O_M, and we call this procedure **Detect-Conflict**. Once we have quantum access to $|O_M\rangle$, there exists a quantum search algorithm that can solve the problem with a quadratic speedup. We state it below[1].

[1] In our algorithm we need to search for some "marked" item without prior knowledge of the number of "marked" items. Such quantum search algorithms are well established in the literature (e.g. [6, 7, 11, 16]).

▷ Claim 1. There exists a quantum algorithm **Q-Detect-Conflict** that decides whether a given vertex set of vertices S has non-empty intersection with a given vertex v with high probability and uses $O(\log(n)\sqrt{|S|})$ queries (here, high probability will mean at least $1 - \frac{1}{\mathbf{poly}(n)}$).

3 Classical Algorithmic Results

In this section we give a simple randomized algorithm for C-Coloring where $\Delta + 1 < C \leq n$. This algorithm proceeds in three steps: 1) **seed** the vertices of the graph with randomly chosen colors, 2) **prune** the graph of the conflicts created during the previous step, and finally 3) **recolor** the pruned vertices.

Algorithm 1 C-color G

Initialize $\chi(c) = \emptyset$ for all $c \in [C]$
for each uncolored vertex v
  Sample $c \in [C]$ uniformly at random, and run **Detect-Conflict**$(v, \chi(c), O_M)$;
 repeat until the output is NO (i.e., no conflicts).
  Color v with c by appending v to $\chi(c)$, and move to a new vertex
return The coloring assignment $\{\chi(1), \ldots \chi(C)\}$

▶ **Theorem 2.** *The above algorithm produces a valid C-coloring of G and makes $O(\frac{n^2}{C-\Delta})$ queries in expectation.*

Proof of Theorem 1. For any $v \in V$ let Q_v be the number of queries made in attempting to color v. Clearly, the total number of queries this algorithm makes is equal to $\sum_{v \in V} Q_v$ and via linearity of expectation the total expected number of queries is $\sum_{v \in} \mathbb{E}[Q_v]$.

When a color is randomly chosen for v let $p = \mathbb{P}[c \text{ is valid for } v]$. Note that when this color is valid for v (i.e. when coloring v with c would not introduce any conflicts between v and its neighbors) the algorithm will query all potential conflicts, find none, and color v with c. If the color is not valid, there must be some $w \in \chi(c)$ that is a neighbor of v. Hence, in this case we will make at most $|\chi(c)|$ queries before recognizing the conflict and choosing a new color, starting the process from scratch. Thus, $\mathbb{E}[Q_v]$ can be computed as

$$\begin{aligned}\mathbb{E}[Q_v] &= p\mathbb{E}[|\chi(c)| : c \text{ is valid for } v] \\ &\quad + (1-p) \cdot \mathbb{E}[|\chi(c)| : c \text{ is not valid for } v] + \mathbb{E}[Q_v] \\ &= \mathbb{E}[|\chi(c)|] + (1-p)\mathbb{E}[Q_v]\end{aligned}$$

This gives $\mathbb{E}[Q_v] = \frac{\mathbb{E}[|\chi(c)|]}{p}$. Since c is chosen uniformly from the set $[C]$ and the C color classes partition some subset of vertices (namely, those which have been colored) we have that $\sum_{i \in [C]} |\chi(c)| \leq n$ and thus $\mathbb{E}[|\chi(c)|] \leq \frac{n}{C}$. Also, recall that for any v there are at most Δ invalid colors c, so $p \geq \frac{C-\Delta}{C}$. This gives

$$\mathbb{E}[Q_v] = \frac{\mathbb{E}[|\chi(c)|]}{p} \leq \frac{C}{C-\Delta} \cdot \frac{n}{C} \leq \frac{n}{C-\Delta}.$$

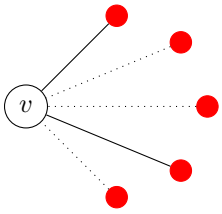

Figure 2 Here, we query all edges whose existence would prevent us from coloring v red

This implies that $\mathbb{E}[Q_v] \leq \frac{n}{C-\Delta}$ for all v. Thus, the total expected number of queries can be computed as

$$\sum_{v \in V} \mathbb{E}[Q_v] \leq \sum_{v \in V} \frac{n}{C-\Delta} = O\left(\frac{n^2}{C-\Delta}\right).$$

◀

Clearly, this algorithm fails to be efficient when $C = \Delta + O(1)$ (in particular, it gives a quadratic runtime); finding a simple to analyze and efficient algorithm for this regime remains as an open problem.

▶ Remark 3. When $C = \Delta^k \geq 2\Delta$ the previous algorithm has query complexity $O(\frac{n^2}{\Delta^k})$. When used in combination with the greedy algorithm, we can achieve query complexity $O(min(\frac{n^2}{\Delta^k}, n\Delta)) = O(n^{\frac{k+2}{k+1}})$. In particular, this is optimal for $C = O(\Delta)$, via the lower bounds given in [2]. We suspect that there is a class of matching lower bounds for larger values of k, as it seems to fit nicely with the $\Omega(n^{3/2})$ bound for the case of $\Delta + 1$ colors.

4 Simple quantum coloring algorithm

Our simple randomized algorithm admits readily an adaptation by invoking a quantum search algorithm to detect conflicts (**Q-Detect-Conflict**). We present the quantum-enhanced algorithm next. Let $T = 16 \log^2 n \sqrt{\frac{n^3}{C-\Delta}}$ and assume that $C \geq 3\Delta$ (this value is chosen to simplify the analysis, the same proof will work for $C = c\Delta$ when $c \ll 1$).

Algorithm 2 Quantum-Color(C, G)

Initialize $\chi(c) = \emptyset$ for all $c \in [C]$
While T or fewer queries have been made
  for each uncolored vertex v
    Repeat choosing $c \in [C]$ uniformly at random, until $\chi(c) < \frac{4n}{C-\Delta}$ and **Q-Detect-Conflict**$(v, \chi(c), O_M)$ outputs NO (i.e., no conflicts).
    Color v with c by appending v to $\chi(c)$, and move to a new vertex
return The coloring assignment $\{\chi(1), \ldots \chi(C)\}$

▶ **Theorem 4.** *The above algorithm makes* $\tilde{O}\left(\frac{n^{3/2}}{\sqrt{(C-\Delta)}}\right)$ *queries and returns a valid coloring with probability at least* $1 - O(1/n)$.

The main differences between algorithms 1 and 2 is that in the latter we place artificial limits on the size of color classes which we check for conflicts as well as on the total number

of queries made. We want to keep a fixed total number of queries in order to limit our algorithm's randomness to only the probability of success rather than in the number of queries made - this artificial upper bound is cleaner. The small color class limitation is made to help simplify the analysis and is not central to how this algorithm works.

Proof of Theorem 4. We will first show that the algorithm succeeds in producing a valid coloring with high probability. Note that there are only two ways to fail in producing a valid coloring:

- Not all vertices are colored by the time that T queries have been made
- Some vertex is assigned an incorrect color

We will analyze these cases separately and show either occurs with low sufficiently low probability.

Let Q_v be the number of queries made when attempting to color v. Additionally, for $r \geq 1$ define

$$A_r = \begin{cases} 1 & \text{if the first } r \text{ colors tested for } v \text{ are not valid} \\ 0 & \text{otherwise} \end{cases}$$

Note that any call to **Q-Detect-Conflict** will use at most $F = 4 \log n \sqrt{\frac{n}{C-\Delta}}$ queries since it will only ever get called on color classes of size $\frac{4n}{C-\Delta}$ or smaller. Thus,

$$Q_v \leq F\left(1 + \sum_{r=1}^{\infty} A_r\right)$$

Now, let $p = \mathbb{P}[\chi(c) \leq \frac{4n}{\Delta - c} \wedge c \text{ is valid for } v]$. There are at most $\frac{\Delta - C}{4}$ large color classes and at most Δ invalid classes for any v, so there are at least $C - \Delta - \frac{C-\Delta}{4} = \frac{3(C-\Delta)}{4}$ small valid color classes for any vertex v. Hence, a randomly chosen color class is both valid and sufficiently small with probability $p \geq \frac{3(C-\Delta)}{4C}$ and since $C \geq 3\Delta$, $p \geq \frac{9}{16}$.

For now, assume that **Q-Detect-Conflict** never returns an incorrect result. Under this assumption we can see that $\mathbb{P}[A_r = 1] = (1-p)^r \leq 2^{-r}$. Hence,

$$\begin{aligned} \mathbb{P}[Q_v \geq 12 \log^2 nF] &\leq \mathbb{P}[1 + \sum_{r=1}^{\infty} A_r \geq 3 \log n] \\ &\leq \mathbb{P}[A_{2 \log n} = 1] \\ &\leq 2^{-2 \log n} \leq \frac{1}{n^2} \end{aligned}$$

So, assuming the correctness of **Q-Detect-Conflict** we can see that any given vertex v gets colored correctly using at most $12F \log^2 n$ queries with probability at least $1 - \frac{1}{n^2}$.

Note that the number of queries used by this algorithm, T is bounded by n^2 for sufficiently large values of n, so we can use n^2 as an upper bound on the number of times **Q-Detect-Conflict** is called in total (since each call makes a non-zero number of queries) each call returns an incorrect result with probability at most $\frac{1}{n^3}$. A simple union bound shows that all calls to **Q-Detect-Conflict** are correct with probability at least $1 - 1/n$. Another simple union bound shows that each vertex is assigned a valid color using at most $12F \log^2 n$ queries with probability at least $1 - 1/n$. Hence, the probability that each vertex is assigned a valid color before T queries are used is at least $1 - \frac{2}{n} = 1 - O(1/n)$. This concludes our proof of Theorem 4. ◀

▶ Remark 5. When $C = \Delta^k > 2\Delta$ the previous algorithm makes $\tilde{O}\left(\sqrt{\frac{n^3}{\Delta^k}}\right)$ queries - further, when one trades this query complexity off with the classical greedy algorithm one can achieve a bound of $\tilde{O}(n^{\frac{k+3}{k+2}})$ (roughly, when $\Delta > n^{\frac{1}{k+2}}$ this algorithm performs better and the classical performs better otherwise).

▶ Remark 6. The previous proof can be reworked for any sufficiently large constant choice of $\epsilon \gg 1$ such that $C \geq \epsilon\Delta$, here we use $\epsilon = 3$ to simplify the analysis. The main restriction is ensuring that there are enough small and valid color classes such that the probability of choosing one at random is not so low that too many ($\Omega(C)$) colors must be selected.

▶ Remark 7. The palette sparsification method for sublinear $(\Delta + 1)$-coloring of [2] can be adapted to a quantum setting to achieve $\tilde{O}(n^{4/3})$ query complexity. We defer it to Appendix A.

A Quantum extension of [2]

In their breakthrough work, the authors of [2] introduce the the Palette Sparsification sparsification theorem, a surprising statement about how small individual palettes (the set of colors available to a vertex) can be shrunk down without losing the ability to color the graph in a valid way. This is a purely combinatorial statement which they use as the basis for several vertex coloring algorithms. The theorem is as follows:

▶ **Theorem 8** (Theorem 10 (Restatement of Theorem 1 from [2])). *Given a graph G with maximum degree Δ, one can sample a set of colors $L(v) \subset [\Delta + 1]$ of size $\Theta(\log n)$ uniformly at random for each vertex v and with high probability find a valid coloring of G in which v is colored with some member of $L(v)$.*

While this result is elegant on its own, it is also quite useful for the design of vertex coloring algorithms.

Algorithm 3 Palette Sparsification-based Coloring

1. Sample $\Theta(\log(n))$ colors uniformly at random from $[\Delta + 1]$, $L(v)$, for each vertex
2. For each color $c \in [\Delta + 1]$ define $\chi(c) = \{v \in V \mid c \in L(v)\}$
3. Define $E_{conflict}$ as the set of all edges (u, v) where both $u, v \in \chi(c)$ for some $c \in [\Delta + 1]$
4. Construct the conflict graph $G_{conflict}(V, E_{conflict})$
5. Find a proper list-coloring of $G_{conflict}$ where the color for a vertex v comes from $L(v)$.

The step in this algorithm that In the case of their classical algorithm, one must query all edges (u, v) where $u, v \in \chi(c)$ in order to construct the conflict graph. However, in the quantum case we can make use of **Find-Conflict**.

▶ **Theorem 9.** *$G_{conflict}$ can be constructed with high probability using $\tilde{O}(n^{4/3})$ quantum queries with high probability.*

Proof sketch of Theorem 11. Our goal is to discover all edges $(u, v) \in E(G)$ such that $L(v) \cap L(u) \neq \emptyset$. Note that **Find-Conflicts** can be modified to return the conflicting edge that it finds, given that one exists (we can have it return null otherwise). Sparing some details, we will use the following lemma:

▶ **Lemma 10.** *There exists a quantum algorithm which takes $E' \subset V \times V$ and returns an edge $e \in E' \cap E(G)$ if it exists with probability at least $1 - \frac{1}{n^2}$ and makes $\tilde{O}(\sqrt{E'} \log n)$ queries.*

Essentially, the above can be achieved via a Grover search-like procedure. Hence, we can find all "true edges" among a set of candidates via $\tilde{O}(T\sqrt{E'} \log n)$ queries where $T = |E' \cap E_{conflict}|$. With high probability, $T \leq 4n \log^2 n$ since each edge is conflicting with probability $\frac{\log^2 n}{\Delta+1}$ and there are at most $n\Delta$ edges. Thus, we can amplify as before to ensure that all $T \leq 4n \log^2 n$ edges are found with probability at least $1 - \frac{1}{n}$.

Note that Step 3 in the Algorithm 4 is the only one in which we actually make queries to the graph, so plugging the method from Theorem 11 in for step 4 gives the desired quantum algorithm. ◀

B Verifying a Coloring

The problem of verifying whether a given coloring is valid seems to be a natural dual to the problem studied in this paper of finding a valid graph coloring. The problem can be roughly stated as follows:

Problem 1: Given access (query, streaming, or otherwise) to a graph G and an arbitrary C-coloring $(c : V(G) \to [C])$ verify whether c is valid for G.

First, consider the restricted computational model in which only pair queries and pair queries in super position are allowed.

A rather simple example gives optimal lower bounds:

▶ **Theorem 11.** *$\Theta(n^2)$ queries are necessary and sufficient for deciding whether an arbitrary $C-$coloring is valid in the classical pair query only model.*

Proof. Consider a coloring, c, in which all vertices are assigned the same color. The only n-node graph for which c is valid has zero edges. Hence, deciding if c is valid reduces to deciding if G has any edges and in the pair-query only model this requires us to perform all $\Omega(n^2)$ queries of the form $(i, j) \in \binom{[n]}{2}$.

The upper bound can be achieved via an exhaustive search for a conflict among all pairs of vertices. ◀

Note that an algorithm which iteratively queries the neighbors of each vertex would also verify the coloring and require $O(n\Delta)$ queries. One might expect that an algorithm similar to 3 would perform well for dense graphs e.g. $O(n^2/\Delta)$ for a given $\Delta + 1$-coloring, but this is not the case since an arbitrary coloring may have some color c for which $|\chi(c)| = \Omega(n)$ - which would require $\Omega(n^2)$ queries to be verefied (if only pair-queries are to be used). This only explains that the dense-sparse meta-algorithm approach will not work for this problem, but there may well be a hybrid algorithm which makes $o(n^2)$ total queries.

▶ **Theorem 12.** *In the quantum pair query only model $\Omega(n)$ queries are necessary and $\tilde{O}(n)$ queries suffice to verify an arbitrary k-coloring.*

Proof. In the quantum case, the same example can be seen as an instance of a decision variant of Grover search over all $\binom{n}{2}$ potential edges, which we know requires $\Omega(n)$ queries from [4].

For the upper bound consider the following simple procedure:

Algorithm 4 Quantum Verification

for $b \in [C]$
  $S_b \leftarrow \{(u, v) \mid (u, v) \in \binom{\chi(b)}{2}\}$
  if **Q-Detect-Conflict**(S_b) returns **true**
    return **false**
return **true**

The correctness is given by the validity of **Q-Detect-Conflict** - any existing conflicts will be detected with high probability.

The complexity analysis is quite straightforward: fix a color $b \in [C]$, on the bth iteration can see that **Q-Detect-Conflict** will make $\tilde{O}(\sqrt{|S_b|})$ queries. Hence, the total number of queries made by the algorithm will be:

$$\begin{aligned}\sum_{b\in[C]} \tilde{O}(\sqrt{|S_b|}) &= \tilde{O}\Big(\sum_{b\in[C]} \sqrt{\binom{|\chi(b)|}{2}}\Big) \\ &= \tilde{O}\Big(|\chi(1)| + \cdots + |\chi(C)|\Big) \\ &= \tilde{O}(n)\end{aligned}$$

◀

Note that the verification upper bounds apply to full graph query model and the lower bounds do not. We leave the of challenges of finding verification algorithms and lower bounds in the full graph query model (both quantum and classical) for future work.

References

1 Noga Alon and Sepehr Assadi. Palette Sparsification Beyond (Δ+1) Vertex Coloring. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, volume 176 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:22. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.6.

2 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for (Δ+ 1) vertex coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 767–786. SIAM, 2019. doi:10.1137/1.9781611975482.48.

3 Aleksandrs Belovs and Ansis Rosmanis. On the power of non-adaptive learning graphs. *2013 IEEE Conference on Computational Complexity*, Jun 2013. URL: http://dx.doi.org/10.1109/CCC.2013.14, doi:10.1109/ccc.2013.14.

4 Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, Oct 1997. URL: http://dx.doi.org/10.1137/S0097539796300933, doi:10.1137/s0097539796300933.

5 Aija Berzina, Andrej Dubrovsky, Rusins Freivalds, Lelde Lace, and Oksana Scegulnaja. Quantum query complexity for some graph problems. In Peter Van Emde Boas, Jaroslav Pokorný, Mária Bieliková, and Július Štuller, editors, *SOFSEM 2004: Theory and Practice of Computer Science*, pages 140–150, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

6 Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, Jun 1998. URL: http://dx.doi.org/10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P, doi:10.1002/(sici)1521-3978(199806)46:4/5<493::aid-prop493>3.0.co;2-p.

7 H. Buhrman, R. Cleve, R. De Wolf, and C. Zalka. Bounds for small-error and zero-error quantum algorithms. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pages 358–368, 1999. doi:10.1109/SFFCS.1999.814607.

8 Harry Buhrman, Christoph Dürr, Mark Heiligman, Peter Høyer, Frédéric Magniez, Miklos Santha, and Ronald de Wolf. Quantum algorithms for element distinctness. *SIAM Journal on Computing*, 34(6):1324–1330, Jan 2005. URL: http://dx.doi.org/10.1137/S0097539702402780, doi:10.1137/s0097539702402780.

9 Francois Le Gall. Improved quantum algorithm for triangle finding via combinatorial arguments. *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, Oct 2014. URL: http://dx.doi.org/10.1109/FOCS.2014.31, doi:10.1109/focs.2014.31.

10 Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

11 Lov K Grover. Fixed-point quantum search. *Physical Review Letters*, 95(15):150501, 2005.

12 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972. doi:10.1007/978-1-4684-2001-2_9.

13 Shelby Kimmel and R. Teal Witter. A query-efficient quantum algorithm for maximum matching on general graphs. 2020. arXiv:2010.02324.

14 Troy Lee, Miklos Santha, and Shengyu Zhang. Quantum algorithms for graph problems with cut queries. arXiv:2007.08285, 2020. https://arxiv.org/abs/2007.08285.

15 Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via quantum walk. *SIAM journal on computing*, 40(1):142–164, 2011. doi:10.1137/090745854.

16 Theodore J Yoder, Guang Hao Low, and Isaac L Chuang. Fixed-point quantum search with an optimal number of queries. *Physical review letters*, 113(21):210501, 2014.