Computing Square Colorings on Bounded-Treewidth and Planar Graphs

Abstract

A *square coloring* of a graph G is a coloring of the square G^2 of G, that is, a coloring of the vertices of G such that any two vertices that are at distance at most 2 in G receive different colors. We investigate the complexity of finding a square coloring with a given number of q colors. We show that the problem is polynomial-time solvable on graphs of bounded treewidth by presenting an algorithm with running time $n^{2^{\mathsf{tw}+4}+O(1)}$ for graphs of treewidth at most tw. The somewhat unusual exponent 2^{tw} in the running time is essentially optimal: we show that for any $\varepsilon > 0$, there is no algorithm with running time $f(\mathsf{tw})n^{(2-\varepsilon)^{\mathsf{tw}}}$ unless the Exponential-Time Hypothesis (ETH) fails.

We also show that the square coloring problem is NP-hard on planar graphs for any fixed number $q \geq 4$ of colors. Our main algorithmic result is showing that the problem (when the number of colors q is part of the input) can be solved in subexponential time $2^{O(n^{2/3}\log n)}$ on planar graphs. The result follows from the combination of two algorithms. If the number q of colors is small ($\leq n^{1/3}$), then we can exploit a treewidth bound on the square of the graph to solve the problem in time $2^{O(\sqrt{qn}\log n)}$. If the number of colors is large ($\geq n^{1/3}$), then an algorithm based on protrusion decompositions and building on our result for the bounded-treewidth case solves the problem in time $2^{O(n\log n/q)}$.

1 Introduction

The *square* G^2 of a graph G has the same vertex set as G and two vertices in G^2 are adjacent if and only if they are at distance at most 2 in G. A *square coloring* of G is a proper vertex coloring of G^2, or equivalently, an assignment of colors to the vertices of G such that not only adjacent vertices receive different colors, but this is also true for vertices at distance 2. Observe that if G has maximum degree Δ, then a square coloring of G certainly needs at least $\Delta + 1$ colors.

The notion of square coloring appeared in many different forms in the combinatorics and computer science literature. Wegener [48] conjectured that a planar graph can be square colored with $\frac{3}{2}\Delta + 1$ colors (for $\Delta \geq 7$). This conjecture started a long and still active line of research, with the goal of obtaining upper bounds on the square chromatic number of various graph classes [2, 3, 9, 10, 15, 39, 44, 46]. A *strong edge coloring* is the distance-2 version of an edge coloring, i.e, a strong edge coloring of G is a square coloring of the line graph of G. Erdős and Nešetřil [27] conjectured that $\frac{5}{4}\Delta^2$ colors are always sufficient for a strong edge coloring. While the conjecture is still open, there are many partial results towards this bound and on similar bounds for, e.g., planar graphs [4, 12, 17, 29, 38]. Some of these combinatorial upper bounds are algorithmic and give polynomial-time approximation algorithms for computing the square chromatic number of, e.g., planar graphs (see also [2, 34]).

In computer science, graph coloring and its variants are often used as a model for assignment problems where adjacent vertices are in conflict and hence cannot receive the same resource (time slot, frequency, processor, etc.) [11, 13, 18–20, 31, 45, 47]. Then the Square Coloring problem (given a graph G and a number q, determine whether there is a square coloring of G with q colors) is the natural extension where not only adjacent vertices, but even vertices at distance two are in conflict. In an even more general setting, an $L(p, q)$-labeling models a frequency assignment problem: assuming that the colors are the integers, adjacent vertices have to receive colors that have difference at least q, while vertices at distance 2 have to receive colors that have difference at least p. Now a square coloring is precisely a $L(1, 1)$-labeling. The Square Coloring problem also received significant attention also in the field of distributed computing (under the name distance-2 coloring or d2-coloring), as it appears naturally for example in unique naming and derandomization [5, 23, 24, 28, 32].

How does the complexity of Square Coloring differ from usual Vertex Coloring problem? The problem is known to be NP-hard for every fixed $q \geq 4$ [16,34,36]. Note that if $q = 3$, then every YES-instance has maximum degree 2, hence Square Coloring is polynomial-time solvable. The goal of this paper is to look at the exact complexity of solving the Square Coloring problem on certain important classes of graphs. One can observe that the square of an interval graph is always an interval graph [40], hence Square Coloring is also polynomial-time solvable on interval graphs. The square of a tree is not necessarily a tree, but it is chordal [41], hence Square Coloring on trees is also polynomial-time solvable.

Treewidth is a graph invariant that, roughly speaking, measures how close the graph is to having a treelike structure. The combinatorial and algorithmic aspects of treewidth were extensively studied in the literature, both as a standalone goal and in applications to other graph classes (see, e.g., [7]). The algorithmic importance of treewidth comes from the fact that many hard algorithmic problems are polynomial-time solvable on graphs of bounded treewidth. For example, for every fixed tw, the chromatic number problem is linear-time solvable for graphs of treewidth at most tw. More precisely, q-Coloring can be solved in time $q^{O(\mathsf{tw})} \cdot n$, which, together with the fact that a graph of treewidth tw has chromatic number at most $\mathsf{tw} + 1$, implies that a $2^{O(\mathsf{tw} \log \mathsf{tw})} \cdot n$ algorithm for chromatic number.

It was shown by Zhou, Kanari, and Nishizeki [49] that Square Coloring is also polynomial-

time solvable on bounded-treewidth graphs. More precisely, given an n-vertex graph G with treewidth tw, their algorithm decides in time $(q+1)^{2^{8\mathsf{tw}+1}} \cdot n^{O(1)}$ if G admits a square coloring with q colors. Our first result is improving the exponent of the running time from 256^{tw} to roughly 2^{tw}.

Theorem 1.1. Square Coloring *can be solved in time* $(q+1)^{2^{\mathsf{tw}+4}} \cdot n^{O(1)}$ *on graphs of treewidth* tw.

The algorithm follows standard dynamic programming techniques on tree decompositions; however, we make some extra effort to ensure that 2^{tw} appears in the exponent and not c^{tw} for some $c > 2$. Note that this form of running time is somewhat unusual. Typically, when considering a problem on graphs of treewidth tw, then either the problem is fixed-parameter tractable (FPT) parameterized by treewidth (that is, it can be solved in time $f(\mathsf{tw}) \cdot n^{O(1)}$ for some function f) or W[1]-hard, but $n^{O(\mathsf{tw})}$ time is sufficient to solve it. Fiala, Golovach, and Kratochvíl [20] showed that Square Coloring is W[1]-hard parameterized by treewidth, hence tw has to appear in the exponent of n in the running time (observe that n colors are always sufficient to obtain a square coloring of G, i.e., we can always assume $q \leq n$). It may seem inefficient that the exponent of the running time depends on the treewidth in such a drastic way in our algorithm, but we show that the exponential dependence seems to be unavoidable. The lower bound assumes the Exponential-Time Hypothesis (ETH) of Impagliazzo, Paturi, and Zane [30].

Theorem 1.2. *Assuming* ETH*, for any* $\varepsilon > 0$ *and any function* f*, there is no* $f(\mathsf{tw})n^{(2-\varepsilon)^{\mathsf{tw}}}$ *time algorithm solving* Square Coloring *on graphs of treewidth* tw.

That is, our algorithm with 2^{tw} in the exponent is essentially optimal. We are not aware of any other natural problem with a similar dependence on treewidth. Interestingly, for the measure cliquewidth cw, it is known that the best possible running time for Vertex Coloring has 2^{cw} in the exponent (assuming ETH) [25].

Next, we turn our attention to planar graphs. The q-Coloring problem is NP-hard on planar graphs for $q = 3$, but becomes polynomial-time solvable for every $q \geq 4$ (because of the Four Color Theorem). By contrast, we show that Square Coloring is NP-hard on planar graphs for every fixed $q \geq 4$.

Theorem 1.3. Square-q-Coloring *is* NP*-hard on planar graphs for every fixed* $q \geq 4$.

Even though 3-Coloring is NP-hard on planar graphs, it can be still solved more efficiently than on general graphs. It is known that an n-vertex planar graph has treewidth $O(\sqrt{n})$. This combinatorial bound and the $2^{O(\mathsf{tw})} \cdot n$ algorithm for 3-Coloring immediately imply a subexponential $2^{O(\sqrt{n})}$ time algorithm. On the other hand, for general graphs, a $2^{o(n)}$ time algorithm would violate ETH.

Does Square Coloring also admit a subexponential algorithm on planar graphs? As there is no constant bound on the number of colors needed for square coloring planar graphs, we consider the version where the number q of colors is part of the input. The treewidth-based approach does not seem to work. First, even though treewidth is $O(\sqrt{n})$, Theorem 1.1 would give only a double exponential $n^{2^{O(\sqrt{n})}}$ time algorithm. We can try to use the $\mathsf{tw}^{O(\mathsf{tw})} \cdot n$ algorithm on the square of the planar graph. However, the square of an n-vertex planar graph G can have treewidth up to $n-1$ (for example, when G is a star with $n-1$ leaves and hence G^2 is an n-clique). Therefore, this approach would give only a $2^{O(n \log n)}$ time algorithm. Nevertheless, our main algorithmic result is a positive answer to this question:

Theorem 1.4. Square Coloring *can be solved in time* $2^{O(n^{2/3} \log n)}$ *on planar graphs.*

The slightly unusual exponent $2/3$ comes from a trade off between two algorithms with running time $2^{O(\sqrt{qn}\log n)}$ and $2^{O(n\log n/q)}$, respectively. By using the former for $q \leq n^{1/3}$ and the latter for $q \geq n^{1/3}$, the bound $O(n^{2/3}\log n)$ on the exponent follows.

1.1 Our Techniques

In this section, we briefly overview the techniques and main ideas used in the results of the paper.

Algorithm for bounded-treewidth graphs. Let us recall first the definition of tree decompositions (see Section 2 for more details). A tree decomposition of a graph G consists of a rooted tree T and a bag $\beta(t) \subseteq V(G)$ for every node t of T with the following properties: (i) every vertex v of G appears in at least one bag, (ii) for every vertex v of G, the bags containing v correspond to a connected subtree of T, (iii) if two vertices of G are adjacent, then there is at least one bag containing both of them. The width of a tree decomposition is the size of the largest bag minus one, and the treewidth of a graph is the smallest possible width of a decomposition. For a node t of T, let us denote by V_t the union over all bags $\beta(t')$ where t' is a descendant of t (including t itself).

A standard way of designing algorithms on tree decompositions is to define some number of subproblems for each node t of the decomposition, and then solve them in a bottom up way. Typically, these subproblems ask about the existence of partial solutions having a certain type. We classify the partial solutions into some number of types in such a way that if two partial solutions have the same type and one has an extension into a full solution, then the same extension would work for the other solution as well. The subproblems at node t would correspond to finding which types of partial solutions are possible. Finally, we argue that if we have solved every subproblem for every child t' of t, then the subproblems at t can be solved efficiently. For example, for the q-COLORING problem, the partial solutions at node t are proper colorings of the graph $G[V_t]$. We define types by classifying the partial solutions according to how they color $\beta(t)$. As $|\beta(t)| \leq \mathsf{tw}+1$, this gives at most $q^{\mathsf{tw}+1}$ types. Then easy recurrence relations show how to solve these subproblems if all the subproblems are already solved for every child of t.

Let us observe that we cannot define the types of partial solutions the same way in the case of the SQUARE COLORING problem. It very well may be that two colorings of $G[V_t]$ agree on $\beta(t)$, but one has an extension to G, whereas the other one does not. For example, let χ_t be a square coloring of V_t and let u be a vertex not in V_t. Then, whether χ_t can be extended to a square coloring χ of G where $\chi(u)$ is red, depends not only on whether u has a neighbor in $\beta(t)$ that was colored red by χ_t, but also on whether those vertices have a neighbor in $V_t \setminus \beta(t)$ that was colored red by χ_t (see Figure 1). Thus, we cannot define the types of partial solutions at node t based only on how they color $\beta(t)$, but we need to take into account the colors on the neighbors of these vertices. This suggests a more refined way of defining the type of a partial solution based on how each vertex $v \in \beta(t)$ is colored and also which subset of the q colors appears already in the neighborhood of each $v \in \beta(t)$. However, as $\beta(t)$ can be up to $\mathsf{tw}+1$, this definition would result in up to $(q \cdot 2^q)^{\mathsf{tw}+1}$ types. As q can be of order n, this is clearly too many.

Our main insight is that instead of precisely describing which colors appear in the neighborhood of each vertex, we classify the colors according to where they appear and only consider the number of colors in each class. More precisely, the type of a partial solution χ_t of V_t depends on the following.

(I) The coloring χ_t restricted to $\beta(t)$ (up to $q^{\mathsf{tw}+1}$ possibilities).

(II) For each color c that appears on $\beta(t)$ in χ_t, the subset of S_c of $\beta(t)$ that has c in its neighborhood (up to $(2^{\mathsf{tw}+1})^{\mathsf{tw}+1}$ possibilities).

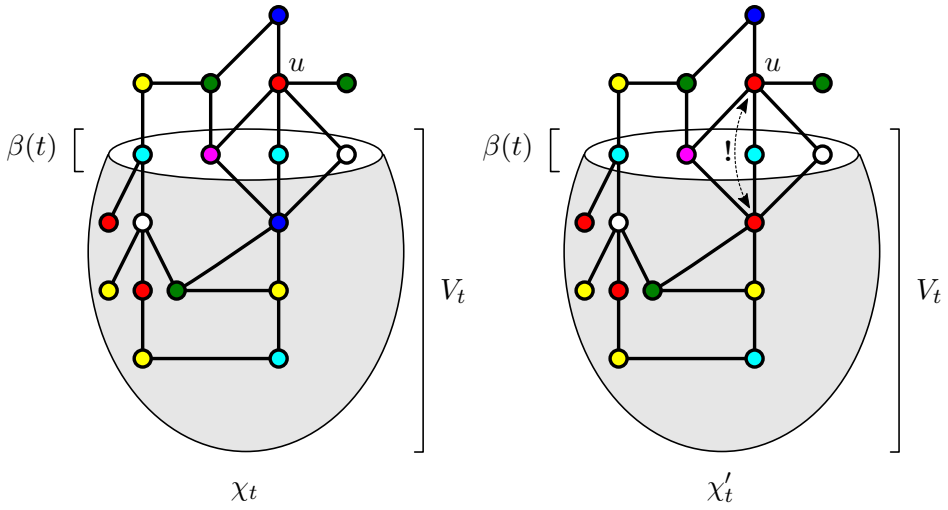


Figure 1: The figure shows two 7-colorings χ_t and χ'_t of V_t that agree on $\beta(t)$, and hence have the same type for the (ordinary) 7-Coloring problem. For example, both can be extended to a coloring of G with the extension shown in the figure. However, if we consider the Square Coloring problem, they are not of the same type: the extension shown in the figure extends χ_t to a proper square coloring, while it is not a valid extension of χ'_t (as it creates two red vertices at distance 2).

(III) For a subset $A \subseteq \beta(t)$, let q_A be the number of colors c that do not appear on $\beta(t)$ in χ_t, but A is exactly the subset of $\beta(t)$ that has c in its neighborhood. Considering every $A \subseteq \beta(t)$, we have $(q+1)^{2^{\mathsf{tw}+1}}$ possibilities for the values of q_A.

All together, this gives roughly $(q+1)^{2^{\mathsf{tw}+1}}$ different types of partial solutions, which matches the running time we would like to achieve.

The process of solving the subproblems in a bottom up manner is fairly standard, but there is one more challenge if we want to achieve the claimed running time. Typically, the main bottleneck in dynamic programming on a tree decomposition appears when handling the *join nodes,* that is, nodes t of the tree decomposition having exactly two children t', t'' and $\beta(t) = \beta(t') = \beta(t'')$ holds. Consider a square coloring χ_t of V_t and its restrictions $\chi_{t'}$ and $\chi_{t''}$ to $V_{t'}$ and $V_{t''}$, respectively. Suppose that a color c appears in the neighborhood of $A' \subseteq \beta(t)$ in $\chi_{t'}$, and on A'' in $\chi_{t''}$. Then in the coloring χ_t, color c appears in the neighborhood of exactly $A' \cup A''$. The main difficulty is that our description of types (as defined in the previous paragraph) does not define where a given color c appears, it only specifies the number of colors that appear in the neighborhood of a given set. This suggests that if we have a description of the type of $\chi_{t'}$ and $\chi_{t''}$, then we need to somehow match up the colors in $\chi_{t'}$ with the colors of $\chi_{t''}$ to determine what type of partial colorings of V_t they can be combined into. This can be formulated as an integer linear programming problem. In order to solve this problem efficiently and have 2^{tw} in the exponent of n, we use another layer of dynamic programming.

Lower bound for bounded-treewidth graphs. As a starting point, we first prove a lower bound for a problem involving vectors, and then reduce that to Square Coloring. In the Vector k-Sum problem, we are given k lists $A_1, \ldots, A_k$, each list containing n integer m-dimensional vectors, and a target vector t. The task is to select exactly one vector from each list such that they sum up to exactly t. Problems of similar flavor were considered before (see, e.g., [1, 33]), but we need

a lower bound with specific parameter settings and restrictions. By a reduction from Subgraph Isomorphism for 3-regular graphs and a known lower bound for this problem [35], we prove that if $k = O(2^w)$, $m = O(2^w)$, then there is no $f(t)n^{(2-\varepsilon)^w}$ time algorithm for any $\varepsilon > 0$, unless ETH fails. Our reduction produces instances of Vector k-Sum with the following additional properties:

1. For every A_i, there are at most 3 coordinates where the vectors in A_i can be non-zero.
2. Every coordinate is non-zero in at most 2 lists.

As we shall see, these properties will be crucial for our lower bound.

Given the lower bound in the previous paragraph, our task is to reduce an instance of Vector k-Sum with $k = O(2^w)$, $m = O(2^w)$ to an instance of Square Coloring with treewidth roughly w. That is, the treewidth of the new instance should be at most *logarithmic* in k and m. Then it follows that $f(\mathsf{tw})n^{(2-\varepsilon)^{\mathsf{tw}}}$ time algorithm for any $\varepsilon > 0$ would violate ETH. While the reduction is highly technical with an elaborate construction of a sequence of gadgets, here we give a brief overview of the main ideas of the proof and in particular how the logarithmic bound for treewidth can be achieved. It will be convenient to describe the reduction to a slight extension of the problem where some vertices are "colorless": they do not need to be colored in the solution, but they are relevant for computing the distances. At the end of the proof, we show how the reduction can be modified if these vertices are also colored.

Figure 2 shows the structure of the constructed instance. On the left, we have $2m$ sets $X_1, \ldots, X_m$, $Y_1, \ldots, Y_m$ where $|X_i| = |Y_i|$ is exactly the i-th coordinate of the target vector t. Let the number q of colors be $\sum_{i=1}^{m} |X_i|$. We want to ensure that in every square coloring with q colors,

1. disjoint sets of colors appear on X_i and X_j for $i \neq j$,
2. the same set of colors appear on X_i and Y_i.

While there are many different ways of achieving this, the following construction ensures the logarithmic bound on the treewidth. We introduce a set S of $r \approx \log m$ colorless vertices. For every $i \in [m]$, we choose a distinct subset $S_i \subseteq S$ of size $r/2 + 1$ and connect every vertex of X_i to every vertex of S_i, and every vertex of Y_i to every vertex of $S \setminus S_i$. Note that as $\binom{r}{r/2+1} \approx 2^r \approx m$, we can choose distinct S_i's. Observe that X_i and Y_i do not have common neighbors, allowing the use of the same colors on these two sets. As $S_i \cap S_j \neq \emptyset$ if $i \neq j$, there is a vertex of S adjacent to every vertex of $X_i \cup X_j$, implying that a color cannot be used on both X_i and X_j. Similarly, $(S \setminus S_i) \cap S_j \neq \emptyset$ if $i \neq j$, implying that a color cannot be used on both Y_i and X_j. Together with the bound on the number q of colors, this implies that Y_i has to use the same set of colors as X_i. Therefore, this gadget defines a partition into m sets of colors, each set having the required size, and each set appearing on two sets of vertices, X_i and Y_i.

Next, we introduce a new colorless vertex x and k *vector selection gadgets* $W_1, \ldots, W_k$, representing the lists $A_1, \ldots, A_k$. We design the gadgets in a way that ensures that their treewidth is some constant $\alpha \leq 20$. Suppose that vectors in A_i are nonzero only in the three coordinates i_1, i_2, i_3. Then gadget W_i is attached to vertex x, and for every $\ell \in \{1, 2, 3\}$, to one of X_{i_ℓ} and Y_{i_ℓ}. As every coordinate is nonzero in the vectors of at most two lists, these attachments can be done in such a way that every X_j or Y_j is used only by one gadget. Therefore, if we remove vertex x and the set S, then the instance falls apart into disjoint gadgets. As each gadget has treewidth at most α, it follows that the constructed graph has treewidth at most $|S| + \alpha + 1 \approx \log m$, as required.

The role of the vertex selection gadgets is the following. Suppose that gadget W_i is attached to $X_{i_1} \cup X_{i_2} \cup X_{i_3}$ and to vertex x. We know that vertices in $X_{i_1} \cup X_{i_2} \cup X_{i_3}$ receive distinct colors and in every coloring W_i "exhibits" some set C_i of colors to x, that is, C_i is the set of colors appearing

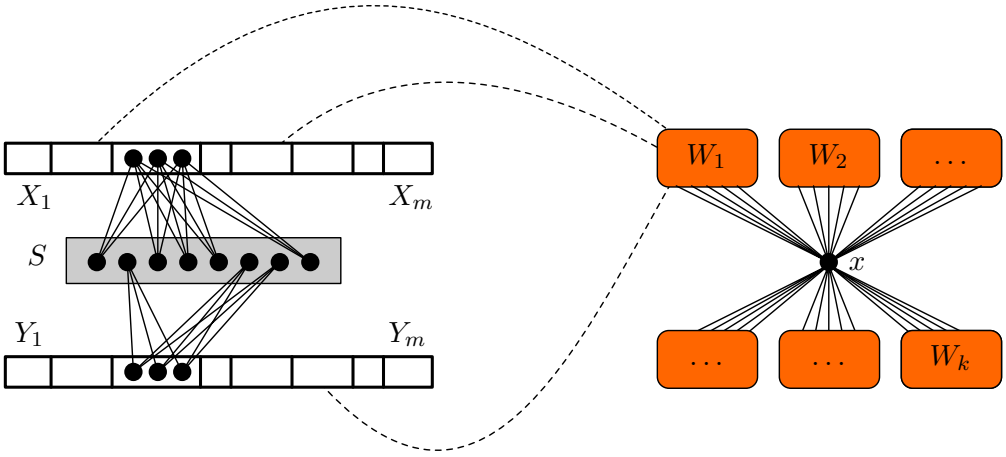


Figure 2: The high-level structure of the constructed instance of Square Coloring in the proof Theorem 1.2. Each W_i is connected to three color classes; here only the connections of W_1 are indicated.

on the neighbors of x in the gadget W_i. The possible square colorings of gadget W_i can be classified into n different "states", corresponding to the n vectors in A_i. If a vector $v_i \in A_i$ has value a_ℓ at coordinate i_ℓ for $\ell \in \{1,2,3\}$, then in the corresponding state exactly a_ℓ colors of X_{i_ℓ} are exhibited to x. The colors on the neighbors of x should be all different, which means that the gadgets together should exhibit at most $|X_{i_\ell}|$ colors of X_{i_ℓ}. In other words, if vector $v_i \in A_i$ corresponds to the state of gadget W_i, then $\sum_{i=1}^{k} v_i$ should be a vector whose i_ℓ-th coordinate is at most $|X_{i_\ell}|$. Observing this for every coordinate shows that $\sum_{i=1}^{k} v_i \leq t$. With additional arguments, this can be extended to show that there is actually equality. Therefore, the possible combination of states of the vector selection gadgets in square colorings of the constructed graph correspond to the solutions of the Vector k-Sum instance.

We remark that the actual proof is somewhat different, for example, the sketch above ignores the fact that the gadget W_i should always exhibit the same number of colors to x. In the proof, we find it more convenient to define the Vector k-Sum problem such that $t = 0$ and hence the vectors may have positive and negative integer values. Then we represent each coordinate with two sets of colors and if a vector has value a_i at the i-th coordinate, then the gadget exhibits $M - a_i$ and $M + a_i$ colors from these sets, respectively. The proof idea described above goes through with appropriate modifications.

Algorithm for planar graphs. The subexponential algorithm in Theorem 1.4 is obtained as a combination of two algorithms (see Figure 3). It is known that an n-vertex planar graph has treewidth $O(\sqrt{n})$, but this bound is obviously not true in general for the square of a planar graph. However, we can obtain a useful bound if we take into account the maximum degree Δ of the graph as well.

Lemma 1.5. *Let G be a planar graph of maximum degree Δ. Then*

$$\mathrm{tw}(G^2) = O\left(\sqrt{n\Delta}\right).$$

We can assume that Δ is at most the number q of colors, otherwise there is no solution. Thus, we can assume that the treewidth of G^2 is $O(\sqrt{nq})$. By using a $q^{O(\mathsf{tw})} \cdot n^{O(1)}$ algorithm for q-coloring a graph of treewidth tw, we obtain the first algorithm:

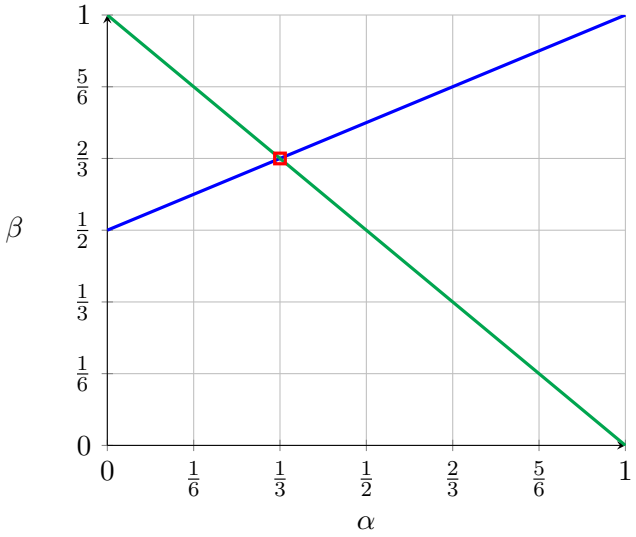


Figure 3: The running time of the algorithms from Lemma 1.6 (blue) and 1.7 (green) as a function of the the number q of colors. The x-axis shows the number of colors in logarithmic scale, i.e., $q = n^\alpha$. The y-axis shows the running time in double logarithmic scale, i.e., the algorithms run in time $n^{O(n^\beta)}$.

Lemma 1.6. Square Coloring *can be solved in time* $q^{O(\sqrt{qn})}$ *on planar graphs.*

Let us now discuss the second algorithm, which has running time $2^{O((n \log n)/q)}$. Let $N_2[v]$ be the set of vertices at distance at most 2 from v. The initial observation is that if $|N_2[v]| \leq q$, then coloring v does not present any difficulty: if the rest of the graph is colored, there is still at least one color c that is unused in the distance-2 neighborhood of v and hence we can assign c to v. Let $U = \{v \mid |N_2[v]| > q\}$. Once we have a partial square coloring of the vertices of U, then it can be extended to a square coloring of G. This implies that G has a square coloring if and only if $G' = G[N[U]]$ has (note that we need to include the neighbors of U into G' to preserve the distance-2 paths between vertices of U). It follows that we can assume that $G = G[N[U]]$, that is, U is a dominating set of G. Using a simple greedy selection argument, we can show that there is a subset $U' \subseteq U$ of size $O(n/q)$ that is a distance-3 dominating set of G, that is, every vertex of G is at distance at most 3 from U'.

Known results show that if a planar graph has a small distance-3 dominating set, then it can be decomposed into a smaller planar graph with small treelike attachments, called protrusions, connected to it. Formally, an (α, δ, k)-*protrusion decomposition* is a tree decomposition where the root bag has size at most α, all the other bags have size at most k, and the root has at most δ children. It follows from earlier work [8] (see also [22]) that if G has a distance-3 dominating set of size $O(n/q)$, then it has an $(O(n/q), O(n/q), O(1))$-protrusion decomposition.

How to solve the Square Coloring problem given such a protrusion decomposition T? An obvious approach would be to try every possible coloring of the root bag ($q^{O(n/q)}$ possibilities, which is feasible in our target running time), and then somehow extend the coloring to all children of the root bag. As the bags below the root have size $O(1)$, this should be very similar to square coloring graphs of treewidth $O(1)$, which is polynomial-time solvable by Theorem 1.1. However, this approach is flawed. If t', t'' are two children of the root r, then two vertices $v' \in \beta(t') \setminus \beta(r)$ and $v'' \in \beta(t'') \setminus \beta(r)$ can be at distance 2 from each other (via a vertex in $\beta(r)$) and hence may need to receive distinct colors. This means that even if the colors of the vertices of $\beta(r)$ are fixed,

we cannot just extend the coloring to the subtree of each child independently, as such conflicts have to be avoided as well.

An natural extension to circumvent this problem is to not only guess the coloring of the root bag, but also guess the type of the coloring of each of the subtrees of the root as described in Items (I) - (III). To be more precise, let r denote the root of the protrusion decomposition T and let $t_1, \ldots, t_\delta$ be the children of r. Let us also denote by V_i, $i \in \{1, \ldots, \delta\}$, the set of all vertices located in bags below t_i (including t_i itself). Since $V_i \cap \beta(r)$ has size $O(1)$ the number of different types for each individual subtree is bounded by $q^{O(1)}$. So in total, there are only $q^{O(\delta)} = q^{O(n/q)}$ possibilities for guessing all the types, which is still feasible in our target running time. However, this does still not solve the problem pointed out above due to Item (III). Let us say a color c is *i-free* if the color c does not appear in $V_i \cap \beta(r)$ (given a fixed coloring of the root bag). For each set $A \subseteq V_i \cap \beta(r)$, Item (III) only provides the number $q_{i,A}$ of i-free colors c so that A is exactly the set that has c in its neighborhood (only considering vertices from $V_i \setminus \beta(r)$). Hence, we need to identify a suitable set of i-free colors $C_{i,A}$ of size $q_{i,A}$ which contains precisely those colors c. The challenge is to do this in a consistent way. Indeed, as already indicated above, we need to assign colors in such a way that $C_{i,A} \cap C_{i',A'} = \emptyset$ for all distinct $(i, A), (i', A')$ such that $A \cap A' \neq \emptyset$, as otherwise a vertex $v \in A \cap A'$ has two neighbors that are assigned the same color, one in the subtree rooted at t_i, and one in the subtree rooted at $t_{i'}$.

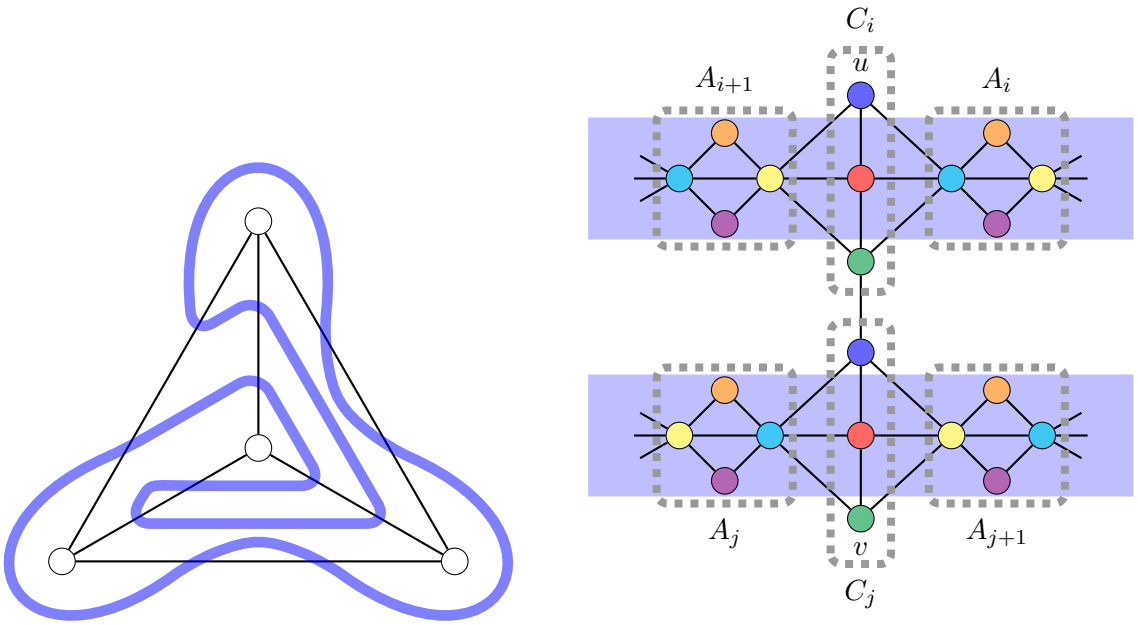
To solve this problem, we yet again rely on dynamic programming. Let $\mathcal{Z}$ be the set of all pairs (i, A) where $i \in \{1, \ldots, \delta\}$ and $A \subseteq V_i \cap \beta(r)$. Note that $|\mathcal{Z}| = O(n/q)$. For each individual color $c \in \{1, \ldots, q\}$ and every subset $\mathcal{Z}' \subseteq \mathcal{Z}$, we can check whether assigning c to exactly those sets $C_{i,A}$ for which $(i, A) \in \mathcal{Z}'$ leads to any color conflict. Note that the number of subsets $\mathcal{Z}' \subseteq \mathcal{Z}$ is bounded by $2^{O(n/q)}$, so this is indeed feasible in the given time frame. Now, checking whether $q_{i,A}$ many i-free colors can be assigned to $C_{i,A}$ for every $(i, A) \in \mathcal{Z}$ in a consistent can be done by a dynamic program that iteratively increases the number of available colors $q' \leq q$ and checks which "demands" on i-free colors can be met by only using colors from $\{1, \ldots, q'\}$. Once we arrive at $q' = q$ colors, we can deduce whether all guesses made so far (i.e., the coloring of the root bag and all the types of partial coloring of V_i) are consistent which completes the algorithm. Overall, we obtain the second algorithm.

Lemma 1.7. Square-q-Coloring *can be solved in time $n^{O(n/q)}$ on planar graphs.*

Finally, Theorem 1.4 follows from applying the algorithm from Lemma 1.6 for $q \leq n^{1/3}$, and the algorithm from Lemma 1.7 for $q \geq n^{1/3}$ (see also Figure 3).

NP-hardness for planar graphs. The NP-hardness of Square Coloring on planar graphs was shown by Lloyd and Ramanathan [34]. However, they only show hardness for $q = 7$ colors. We extend this to any fixed number $q \geq 4$ of colors by reducing from 3-Coloring restricted to planar input graphs.. The NP-hardness of this problem has been proven in [43]. Let G denote a planar graph for which we wish to test 3-colorability. In order to create an equivalent instance of Square Coloring (with q colors) we follow the natural strategy of splitting the set of q colors into a set of 3 "candidate" colors and $q - 3$ "auxiliary" colors. Now, the basic idea is to extend G by certain gadgets that ensure that every original vertex of G needs to be colored by one of the "candidate" colors, and adjacent vertices in the original graph G need to receive different colors. The main challenge of this approach is to devise a method to distribute the information of which colors are the "candidate" colors over the entire graph.

Here, our main insight is that, for every plane graph G (i.e., a planar graph together with its embedding in the plane) of minimum degree 3, one can draw a circle on the plane that crosses every



(a) The planar graph K_4 together with a circle that crosses every edge of K_4 exactly twice.

(b) The constructed instance of Square Coloring for $q = 7$ when zooming in on an edge uv of the original graph G.

Figure 4: Visualization of the reduction from 3-Coloring to Square Coloring on planar graphs.

edge of G exactly twice (and does intersect any vertex of G). An example is given in Figure 4a. Given such a circle, we can distribute the information of which colors are the "candidate" colors and which are the "auxiliary" colors along the circle while preserving planarity. Suppose $q \geq 5$. Figure 4b shows the constructed instance of Square Coloring when zooming in on a single edge $uv \in E(G)$ of the original graph G. In the example, the colors red, blue and green are the "candidate" colors and yellow, orange, violet and cyan are the "auxiliary" colors. This information is forwarded along the blue circle by the construction shown the Figure 4b. To be more precise, we introduce sets of vertices C_i and A_i for all $i \in [2|E(G)|]$ where each C_i contains 3 vertices and each A_i contains $q - 3 \geq 2$ vertices. The sets C_i and A_i are placed alternately on the constructed circle where each C_i is associated with a crossing between the constructed circle and an edge of G (see also Figure 4). For every $i \in [2|E(G)|]$ we introduce edges between A_i and C_i as well as C_i and A_{i+1} as indicated in Figure 4b. We obtain that, in the square graph, both $A_i \cup C_i$ and $C_i \cup A_{i+1}$ form cliques, which means that all q colors have to be used on both sets. In particular, this implies that all sets A_i and A_{i+1} have to be colored by the same set of colors which are declared to be the "auxiliary" colors. Overall, we obtain that all sets A_i are colored by the "auxiliary" colors and all sets C_i are colored by the "candidate" colors.

Looking at Figure 4b, it can also be checked that u and v need to be assigned distinct "candidate" colors. Indeed, the only way to color C_i and C_j in a consistent way is to color them in the same way from top to bottom. So overall, we can thus ensure that all vertices of G are colored by one of the "candidate" colors and adjacent vertices need to receive distinct colors. This almost completes the reduction. As the last remaining step, we only need to ensure that gadgets living on two edges incident to the same vertex do not interfere with one another. However, this can easily be assured by some simple gadgets that are introduced at every vertex of G.

Observe that the above construction requires $q \geq 5$ since the red vertices in Figure 4b always

have degree 4 (independent of the number of colors q). For $q = 4$, we design a specialized reduction which is slightly easier than the one described above. Here, the main insight is that we only need to distribute one "auxiliary" color which can be done in a more direct way while preserving planarity.

Finally, let us also point out the constructed instance of Square Coloring has $O(qn)$ many vertices (where n denotes the number of vertices of G). As a result, building on known lower bounds for 3-Coloring on planar graphs (see, e.g., [14, Theorem 14.9]), we also obtain the following conditional lower bound for Square Coloring.

Theorem 1.8. *Assuming* ETH*, there is no* $2^{o(\sqrt{n})}$ *time algorithm solving* Square Coloring *on planar graphs.*

In particular, assuming ETH, the algorithm from Lemma 1.6 is essentially optimal for constant number of colors q.

1.2 Structure of the Paper

After providing the necessary preliminaries in the next section, we prove Theorem 1.1 in Section 3. Afterward, we show a matching lower bound (Theorem 1.2) in Section 4. In Section 5, we prove Lemmas 1.6 and 1.7 which together imply Theorem 1.4. Finally, we provide the hardness results for Square Coloring on planar graphs in Section 6 where we show Theorems 1.3 and 1.8.

2 Preliminaries

Basics For an integer $k \geq 1$ we denote $[k] \coloneqq \{1, 2, \ldots, k\}$ and $[k]_0 \coloneqq \{0, 1, \ldots, k\}$. We define $\mathbb{Z}_{>0}$ to be the set of positive integers, and $\mathbb{Z}_{\geq 0}$ to be the set of non-negative integers. For a finite set X we denote by 2^X the powerset of X, i.e., the set of all subsets of X, and $\binom{X}{k}$ denotes the set of all k-element subsets of X. If not otherwise stated, log denotes the logarithm with base 2.

We use standard notation for graphs. A *graph* is pair $G = (V(G), E(G))$ with finite vertex set $V(G)$ and edge set $E(G) \subseteq \binom{V(G)}{2}$. Unless stated otherwise, all graphs considered in this paper are simple (i.e., there are no loops or mulitedges) and undirected. We use uv as a shorthand for edges $\{u, v\} \in E(G)$. The *(open) neighborhood* of a vertex $v \in V(G)$ is denoted by $N_G(v) \coloneqq \{w \in V(G) \mid vw \in E(G)\}$. The *degree* of v is the size of its neighborhood, i.e., $\deg_G(v) \coloneqq |N_G(v)|$. The *closed neighborhood* is $N_G[v] \coloneqq N_G(v) \cup \{v\}$. More generally, for a set $X \subseteq V(G)$, we define $N_G(X) \coloneqq (\bigcup_{v \in X} N_G(v)) \setminus X$ and $N_G[X] \coloneqq N_G(X) \cup X$. We usually omit the index G if it is clear from context.

A *path of length* k between two vertices $v, w \in V(G)$ is a sequence of pairwise distinct vertices $v = u_0, \ldots, u_k = w$ such that $u_{i-1}u_i \in E(G)$ for all $i \in [k]$. The *distance* between v and w, denoted by $\mathrm{dist}_G(v, w)$, is the length of the shortest path between v and w. For $X \subseteq V(G)$ we write $G[X]$ to denote the *induced subgraph* on vertex set X, and $G - X \coloneqq G[V(G) \setminus X]$ denotes the induced subgraph on the complement of X.

Colorings Let G be a graph. For $q \geq 1$ a *(proper) q-coloring* of G is a mapping $\chi \colon V(G) \to [q]$ such that $\chi(v) \neq \chi(w)$ for all edges $vw \in E(G)$. A *square q-coloring* of G is a mapping $\chi \colon V(G) \to [q]$ such that $\chi(v) \neq \chi(w)$ for all distinct $v, w \in V(G)$ such that $\mathrm{dist}_G(v, w) \leq 2$. Observe that a mapping $\chi \colon V(G) \to [q]$ is a square q-coloring of G if and only if χ is a q-coloring of the *square graph* G^2 defined by $V(G^2) \coloneqq V(G)$ and

$$E(G^2) \coloneqq \{vw \mid v \neq w, \mathrm{dist}_G(v, w) \leq 2\}.$$

A *coloring* of G is a q-coloring of G for some number $q \geq 1$. Similarly, a *square coloring* of G is a square q-coloring of G for some number $q \geq 1$. We consider the following computational problem.

Square Coloring
Input: A graph G, and an integer q.
Question: Is there a square q-coloring of G?

Also, we call Square-q-Coloring the variant of the problem where the number of colors is fixed to q.

Given a graph G and an arbitrary mapping $\chi\colon V(G) \to [q]$, we define a *(color) conflict* to be a pair of vertices violating the square coloring constraint, that is, a (color) conflict is pair of distinct vertices $v, w \in V(G)$ such that $\mathrm{dist}_G(v, w) \leq 2$ and $\chi(v) = \chi(w)$.

Tree Decompositions Next, we define tree decompositions and state some basic properties. For a more thorough introduction to treewidth and its many applications, we refer the reader to [14, Chapter 7].

Let G be a graph. A *tree decomposition* of G is a pair (T, β) consisting of a rooted tree T and a mapping $\beta\colon V(T) \to 2^{V(G)}$ such that

(T.1) $\bigcup_{t \in V(T)} \beta(t) = V(G)$,

(T.2) for every edge $vw \in E(G)$ there is some node $t \in V(T)$ such that $\{u, v\} \subseteq \beta(t)$, and

(T.3) for every $v \in V(G)$ the set $\{t \in V(T) \mid v \in \beta(t)\}$ induces a connected subtree of T.

The *width* of a tree decomposition (T, β) is defined as $\max_{t \in V(T)} |\beta(t)| - 1$. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum width of a tree decomposition of G.

The following fact on tree decompositions is well-known.

Observation 2.1. *Let G be a graph and let (T, β) be a tree decomposition of G. Also let $S \subseteq V(G)$ such that $G[S]$ is connected. Then*

$$T_S \coloneqq \{t \in V(T) \mid \beta(t) \cap S \neq \emptyset\}$$

induces a connected subtree of T.

When designing algorithms on graphs of bounded treewidth, one usually builds on nice tree decompositions. Let (T, β) be a tree decomposition and denote $X_t \coloneqq \beta(t)$ for $t \in V(T)$. The decomposition is *nice* if $X_r = \emptyset$ where r denotes the root of T, $X_\ell = \emptyset$ for all leaves $\ell \in V(T)$, and every internal node $t \in V(T)$ has one of the following types:

Introduce: t has exactly one child t' and $X_t = X_{t'} \cup \{v\}$ for some $v \notin X_{t'}$; the vertex v *is introduced at* t,

Forget: t has exactly one child t' and $X_t = X_{t'} \setminus \{v\}$ for some $v \in X_{t'}$; the vertex v *is forgotten at* t, or

Join: t has exactly two children t_1, t_2 and $X_t = X_{t_1} = X_{t_2}$.

It is well-known that every tree decomposition (T, β) of G of width tw can be turned into a nice tree decomposition of the same width tw of size $O(\mathsf{tw} \cdot |V(T)|)$ in time $O(\mathsf{tw}^2 \cdot \max(|V(G)|, |V(T)|))$ (see, e.g., [14, Lemma 7.4]).

When dealing with tree decompositions of graphs, we adopt the standard notion of using the word "node" for nodes of T and the word "vertex" for vertices of G.

For upper-bounding the treewidth of a graph, there is a useful characterization via a graph-theoretic game called *cops and robbers*. The game is parameterized by a number k and a graph G, the playing field. It has two teams, one with k cops and one with a single robber. The rules are as follows: The k cops select their starting vertices in the graph. Then the robber may choose their starting vertex. The cops can always see the robber and adapt their strategy accordingly. Similarly, the robber can see the cops. The game now proceeds in rounds, where in each round, one of the cops chooses an arbitrary destination vertex and takes off via helicopter in the direction of that vertex. While the cop is traveling, the robber sees where they will land and may now move arbitrarily along edges of the graph, as long as they do not pass through stationary cops. When the robber has finished moving, the cop lands. The cops win if they catch the robber after a finite number of moves. Otherwise, the robber wins. We say that *k cops can catch a robber on G* if the k cops have a winning strategy in this game.

Theorem 2.2 ([42]). *A graph G has treewidth at most tw if and only if $\mathsf{tw}+1$ cops can catch a robber on G.*

Planar Graphs A *plane graph* is a pair consisting of a graph G and an embedding of G in the plane (without any edge crossings). A graph G is *planar* if it can be embedded in the plane. Let G be a planar graph and fix some embedding of G in the plane. For each $v \in V(G)$ we can cyclically order the incident edges $E_G(v) \coloneqq \{e \in E(G) \mid v \in e\}$ clockwise according to the fixed embedding. In our constructions of planar, we occasionally require the existence of planar embeddings where this cyclic ordering of incident edges satisfies certain properties. Let us remark at this point that fixing such a cyclic ordering for every $v \in V(G)$ uniquely describes an embedding of G in the plane (if it exists) up to homeomorphisms. Hence, it usually suffices to give such a cyclic ordering for every vertex of G to describe the embedding. This is usually referred to as a *combinatorial embedding of G*. It is well-known that, given a planar graph G, a combinatorial embedding of G can be computed in polynomial time (see, e.g., [37]).

3 Dynamic Programming for Graphs of Bounded Treewidth

In this section, we show that Square Coloring can be solved in polynomial time over graphs of bounded treewidth. More precisely, the main result of this section is the following theorem.

Theorem 3.1 (Theorem 1.1 restated). *There is an algorithm solving* Square Coloring *in time $(q+1)^{2^{(\mathsf{tw}+4)}} \cdot n^{O(1)}$ on input graphs of treewidth tw.*

We will prove this by describing a dynamic programming algorithm on the tree decomposition.

First observe that, using the algorithm from [6], we can compute a tree decomposition of G of width tw in time $2^{O(\mathsf{tw}^3 \log \mathsf{tw})} \cdot n = (q+1)^{2^{(\mathsf{tw}+4)}} \cdot n^{O(1)}$. This can be converted to a nice tree decomposition of the same width with $O(\mathsf{tw} \cdot n)$ bags in time $O(\mathsf{tw}^2 \cdot n)$ [14, Lemma 7.4]. This means that if we can show the existence of an algorithm with running time $(q+1)^{2^{(\mathsf{tw}+4)}} \cdot n^{O(1)}$ for the same problem where an optimal nice tree decomposition is already given, we are done, since that running time will dominate both aforementioned running times.

Hence, it suffices to show that such an algorithm exists. For the remainder of this section let us fix an input graph G, a number of colors $q \leq n$, and a nice tree decomposition (T, β) of width tw such that $|V(T)| = O(\mathsf{tw} \cdot n)$. For $t \in V(T)$ we denote $X_t \coloneqq \beta(t)$ and V_t denotes the union of all bags located in the subtree rooted at node t (including t itself).

We will use dynamic programming (DP) over the nice tree decomposition. The obvious approach would be to have a DP table where a state of a bag encodes for each vertex of the bag what its color is, as well as a list of what colors are within distance one. This way we can always check whether a coloring of a vertex is valid by asserting that none of its neighbors have the same color and that none of its neighbors have the color in their list. However, this introduces a factor of $2^{\mathsf{tw} \cdot q}$ in the running time, where q is the number of colors. Since the necessary number of colors may be as large as n, this can be exponential in n, which we do not want.

Instead, we will track equivalence classes of colors. For a fixed node $t \in V(T)$ and a coloring $\bar{\chi} : V_t \to [q]$, we say that a vertex $v \in X_t$ is *cone-adjacent* to a color $c \in [q]$ if there exists a vertex $u \in N(v) \cap (V_t \setminus X_t)$ such that $\bar{\chi}(u) = c$. For a fixed color $c \in [q]$, we call the set of vertices $v \in X_t$ which are cone-adjacent to c the *bag-adjacent* set of c, i.e.,

$$A_t(\bar{\chi}, c) \coloneqq \{v \in X_t \mid v \text{ is cone-adjacent to } c\}.$$

Now, for a specific $t \in V(T)$ and a coloring $\bar{\chi} : V_t \to [q]$, we say that two colors $c_1, c_2 \in [q]$ are *equivalent* if $A_t(\bar{\chi}, c_1) = A_t(\bar{\chi}, c_2)$. We naturally identify an equivalence class by the bag-adjacent set $A \in 2^{X_t}$ that is shared by all colors the equivalence class contains.

The main insight in our dynamic programming algorithm is that apart from the coloring of the vertices in a bag X_t, the only relevant information that gets transmitted from one side of the bag to the other is the size of each of the equivalence classes. Specifically, we will store the number of *free colors* in each equivalence class, i.e., colors that do not also appear in the bag.

Now let us describe the dynamic programming algorithm in detail. We begin by specifying the structure and content of the dynamic programming table.

Definition 3.2 (Entries of the DP table)**.** For each node $t \in V(T)$, each $\chi : X_t \to [q]$, each $\xi : X_t \to 2^{X_t}$ and each mapping $\rho : 2^{X_t} \to [q]_0$, there is an entry $D[t][\chi][\xi][\rho]$ which is set to true ($\top$) if there exists a square q-coloring $\bar{\chi}$ of $G[V_t]$ such that

(DP.1) $\chi(v) = \bar{\chi}(v)$ for all $v \in X_t$,

(DP.2) $\xi(v) = A_t(\bar{\chi}, \chi(v))$ for all $v \in X_t$, i.e., the color of v is contained in the equivalence class $\xi(v)$, and

(DP.3)
$$\rho(A) = |\{c \in [q] \mid A_t(\bar{\chi}, c) = A\} \setminus \{\chi(v) \mid v \in X_t\}|$$
for all $A \in 2^{X_t}$, i.e., $\rho(A)$ gives the number of free colors (i.e., colors in $[q] \setminus \chi(X_t)$) that are in the equivalence class A.

Note that for every $t \in V(T)$ and every coloring $\bar{\chi}$ of V_t there exist corresponding induced mappings χ, ξ and ρ. In particular, for every square q-coloring of $G[V_t]$, there is a corresponding entry in the partial table $D[t]$ that is set to true.

Also observe that the DP table defined above has

$$|D[t]| \leq q^{\mathsf{tw}+1} \cdot (2^{\mathsf{tw}+1})^{\mathsf{tw}+1} \cdot (q+1)^{2^{(\mathsf{tw}+1)}} \leq (q+1)^{\mathsf{tw}+1+(\mathsf{tw}+1)^2+2^{\mathsf{tw}+1}} \tag{1}$$

many entries for each node $t \in V(T)$.

The following lemma is the main technical result of this section.

Lemma 3.3. *There is an algorithm that, given a graph G, a number q and a nice tree decomposition (T, β) of G of width tw, computes all table entries $D[t][\chi][\xi][\rho]$ in time $|V(T)| \cdot (q+1)^{2^{(\mathsf{tw}+4)}} \cdot n^{O(1)}$.*

Before diving into the proof of Lemma 3.3, let us first derive Theorem 3.1 assuming the lemma holds true.

Proof of Theorem 3.1. Let G denote the input graph and q the number of colors. The algorithm first computes a tree decomposition of G with $O(n)$ bags and width $\mathsf{tw} = \mathrm{tw}(G)$ in time $2^{O(\mathsf{tw}^3 \log \mathsf{tw})} \cdot n = (q+1)^{2^{(\mathsf{tw}+4)}} \cdot n^{O(1)}$ using the algorithm from [6]. This decomposition can be turned into a nice tree decomposition with $O(\mathsf{tw} \cdot n)$ bags and width tw in time $O(\mathsf{tw}^2 \cdot n)$.

The algorithm computes all entries $D[t][\chi][\xi][\rho]$ of the DP table in time $(q+1)^{2^{(\mathsf{tw}+4)}} \cdot n^{O(1)}$. After having calculated all entries, we take the root r of T (which, by definition of a nice tree decomposition, satisfies $X_r = \emptyset$) and output YES if and only if $D[r][\varepsilon_1][\varepsilon_2][\rho] = \top$ for the two empty functions $\varepsilon_1 : \emptyset \to [q], \varepsilon_2 : \emptyset \to \{\emptyset\}$ and $\rho : \{\emptyset\} \to [q]_0$ with $\rho(\emptyset) = q$.

The correctness of this algorithm follows directly from Definition 3.2. Indeed, $D[r][\varepsilon_1][\varepsilon_2][\rho] = \top$ if and only if there exists a coloring of $V_r = V(G)$ such that for the (empty) bag of the root, we have that (DP.1) the coloring coincides with the empty coloring for this bag, (DP.2) for each vertex of the empty bag, its color is contained in the equivalence class defined by ξ, and (DP.3) for each $S \in 2^{\emptyset}$, we have that $\rho(S)$ gives the number of free colors that are in the equivalence class S – in this case, we check that all q colors are indeed in the equivalence class of the empty set. In other words, $D[r][\varepsilon_1][\varepsilon_2][\rho] = \top$ if and only if there exists a square coloring of G.

To complete the proof, observe that the overall running time of the algorithm is bounded by $(q+1)^{2^{(\mathsf{tw}+4)}} \cdot n^{O(1)}$. □

We now turn to the proof of Lemma 3.3. Here, the following definition turns out to be useful which formulates basic, local conditions for a table entry to evaluate to true.

Definition 3.4 (Local validity)**.** We say that a 4-tuple (t, χ, ξ, ρ) is *locally invalid* if any of the following conditions are met:

(L.1) there exist two adjacent vertices $u, v \in X_t$ such that $\chi(u) = \chi(v)$ (i.e., the coloring in the bag is invalid due to two vertices with distance one having the same color),

(L.2) there exist three vertices $u, v, w \in X_t$ such that $uv, vw \in E(G)$ and $\chi(u) = \chi(w)$ (i.e., the coloring in the bag is invalid due to two vertices with distance two having the same color), or

(L.3) there exist two adjacent vertices $u, v \in X_t$ such that $v \in \xi(u)$ (i.e., a vertex u has a color which occurs in the set of cone-adjacent colors of one of its neighbors v and is thus within distance two).

It is easy to check that $D[t][\chi][\xi][\rho]$ is false for every locally invalid tuple (t, χ, ξ, ρ).

Also, we shall use another dynamic programming algorithm as a subroutine to solve certain integer linear programs efficiently. More precisely, we rely on the following subroutine.

Lemma 3.5. *Given an ILP instance of the form $\{Ax = b; \forall i : 0 \leq x_i \leq \|b\|_\infty;\ x \text{ integer}\}$ where A and b have non-negative entries, we can determine feasibility of that instance in time $O(V \cdot C \cdot (\|b\|_\infty + 1)^{C+1})$, where V is the number of variables and C is the number of constraints (i.e., A is a $C \times V$ matrix).*

Proof. Let $x = (x_1, \ldots, x_V)$. We show that there exists a dynamic programming algorithm that achieves this running time. We will have a DP table D structured as follows: For each $0 \leq i \leq V$ and each $(d_1, \ldots, d_C) \in \{0, \ldots, \|b\|_\infty\}^C$, we have that $D[i][d_1][d_2] \ldots [d_C] = \top$ (here, $\top$ refer to true) if there is an assignment to the variables $x_1, \ldots, x_V$ such that

1. $\forall j > i : x_j = 0$ and
2. $\forall j \in [C] : A_j \cdot x = d_j$ (where A_j is the j-th row in A)

and $\perp$ otherwise. Intuitively, the entry reflects whether there exists a partial assignment to the first i variables such that for each j, the left-hand side of the jth equality constraint has value d_j. Trivially, the solution to the entire instance will then be stored in $D[V][b_1][b_2]\dots[b_C]$.

We compute these values bottom-up. The base case $D[0][0][0]\dots[0] = \top$ is obvious. Now for each value of i from 1 to V, we do the following. For each choice of $x_i \leftarrow v_i$ where $0 \leq v_i \leq \|b\|_\infty$, we compute the product $w := A \cdot v$, where v is a vector with v_i in the i-th position and zeros everywhere else. Then, for each value of $d = (d_1, \dots, d_C) \in \{0, \dots, \|b\|_\infty\}^C$, we test whether $D[i-1][d_1][d_2]\dots[d_C]$ is true. If so, we also set $D[i][d_1+w_1][d_2+w_2]\dots[d_C+w_C]$ to true (assuming $d_j + w_j \leq \|b\|_\infty$ for all $j \in [V]$, otherwise we just continue with the next d).

It is easy too see that the algorithm is correct. For running time, we do $V+1$ iterations in the outermost loop over values of i, $\|b\|_\infty + 1$ iterations in the second loop over values of v_i and $(\|b\|_\infty + 1)^C$ iterations in the innermost loop over values of d. In each iteration of the innermost loop we use $O(C)$ many steps to update a single DP table entry. Hence, the running time is $O(V \cdot C \cdot (\|b\|_\infty + 1)^{C+1})$. □

Now, we are ready to give the proof of Lemma 3.3.

Proof of Lemma 3.3. We calculate the entries of the DP table D bottom-up starting with the leaf nodes. Before executing the main algorithm, we initialize all entries of the table to false.

Leaf: Suppose $t \in V(T)$ is a leaf node. Since (T, β) is a nice tree decomposition, it holds that $X_t = \emptyset$. Thus, the only possible values for χ and ξ are the empty functions $\varepsilon_1, \varepsilon_2$, and ρ only has the empty set in its domain. For each $\rho\colon \{\emptyset\} \to [q]_0$, we define

$$D[t][\varepsilon_1][\varepsilon_2][\rho] = \begin{cases} \top & \text{if } \rho(\emptyset) = q \\ \perp & \text{otherwise} \end{cases}.$$

The correctness of this base cases is obvious. For the running time, note that, for a fixed leaf node $t \in V(T)$, we only need to compute $q+1$ entries of the DP table which can be done in time $n^{O(1)}$.

Introduce: Suppose $t \in V(T)$ is an introduce node with the unique child $t' \in V(T)$. Hence, there is some $v \notin X_{t'}$ such that $X_t = X_{t'} \cup \{v\}$.

We iterate over all triples (χ', ξ', ρ') such that $D[t'][\chi'][\xi'][\rho']$ is true. For each possible extension χ of χ' (i.e., for all $\chi\colon X_t \to [q]$ such that $\chi|_{X_{t'}} = \chi'$, or equivalently all colorings of v), we do the following. First, we define an extension ξ of ξ'. If $\chi(v)$ was not a free color in χ', i.e., there is some $v' \in X_{t'}$ such that $\chi(v') = \chi(v)$, we define

$$\xi(u) := \begin{cases} \xi'(u) & \text{if } u \neq v \\ \xi'(v') & \text{if } u = v \end{cases}$$

for all $u \in X_t$. Note that this is well-defined since $\xi'(v') = \xi(v'')$ for all $v', v'' \in X_{t'}$ such that $\chi(v') = \chi(v'')$ by Condition (DP.2).

In the other case $\chi(v)$ is a free color, and we also iterate over all possible choices of the equivalence class $A_v \subseteq X_{t'}$ such that $\rho'(A_v) > 0$ and define

$$\xi(u) := \begin{cases} \xi'(u) & \text{if } u \neq v \\ A_v & \text{if } u = v \end{cases}$$

for all $u \in X_t$. Finally, we define ρ by setting

$$\rho(A) \coloneqq \begin{cases} 0 & \text{if } v \in A \\ \rho'(A) - 1 & \text{if } \chi(v) \text{ is a free color in } \chi' \text{ and } A = A_v \\ \rho'(A) & \text{otherwise} \end{cases}$$

for all $A \in 2^{X_t}$. Now, for each such (χ, ξ, ρ) such that (t, χ, ξ, ρ) is locally valid, we set $D[t][\chi][\xi][\rho]$ to true.

(**Correctness.**) We need to show that we correctly compute the all entries of $D[t]$ according to Definition 3.2.

First, suppose the algorithm sets $D[t][\chi][\xi][\rho]$ to true in iteration (χ', ξ', ρ'). In particular, this means that $D[t'][\chi'][\xi'][\rho']$ is true. Hence, by induction, there exists a square q-coloring $\chi_{V_{t'}}$ of $G[V_{t'}]$ satisfying the conditions of Definition 3.2.

We first argue that we may assume without loss of generality that

$$A_{t'}(\chi_{V_{t'}}, \chi(v)) = \xi(v). \tag{2}$$

Indeed, if there exists some $v' \in X_{t'}$ such that $\chi(v) = \chi(v') = \chi'(v')$ then $A_{t'}(\chi_{V_{t'}}, \chi(v)) = A_{t'}(\chi_{V_{t'}}, \chi'(v')) = \xi'(v') = \xi(v)$ by Condition (DP.2) and the definition of ξ. Otherwise, $\xi(v) = A_v$ for some $A_v \subseteq X_{t'}$ such that $\rho'(A_v) > 0$. Since $\rho'(A_v) > 0$, there is some color $c \in [q] \setminus \{\chi'(u) \mid u \in X_{t'}\}$ such that $A_v = A_{t'}(\chi_{V_{t'}}, c)$. By renaming colors, we may assume without loss of generality that $c = \chi(v)$ which implies that $A_{t'}(\chi_{V_{t'}}, \chi(v)) = A_v = \xi(v)$.

Now, we extend $\chi_{V_{t'}}$ to a coloring χ_{V_t} of V_t by setting

$$\chi_{V_t}(u) = \begin{cases} \chi(u) & \text{if } u = v \\ \chi_{V_{t'}}(u) & \text{otherwise} \end{cases}.$$

We first argue that χ_{V_t} is a square coloring of $G[V_t]$. Recall that we only set $D[t][\chi][\xi][\rho]$ to true if (t, χ, ξ, ρ) is locally valid, i.e., if Conditions (L.1) - (L.3) are satisfied. Since $\chi_{V_{t'}}$ is a square coloring of $G[V_{t'}]$, any potential color conflict has to involve the vertex v introduced at t, i.e., either there is a vertex $u \in V_{t'}$ such that $\chi_{V_t}(u) = \chi_{V_t}(v)$ and $\mathrm{dist}_{G[V_t]}(u, v) \leq 2$, or there are $u, w \in N_{G[V_t]}(v)$ such that $\chi_{V_t}(u) = \chi_{V_t}(w)$. Note that by the definition of nice tree decompositions v is not adjacent to any vertices in $V_t \setminus X_t$. Hence there are only four cases we need to consider:

1. There is some $u \in X_t \cap N(v)$ such that $\chi(u) = \chi_{V_t}(u) = \chi_{V_t}(v) = \chi(v)$. However, this is prevented by (L.1).

2. There are two vertices $u, w \in X_t \cap N(v)$ such that $\chi(u) = \chi_{V_t}(u) = \chi_{V_t}(w) = \chi(w)$. However, this is prevented by (L.2).

3. There are vertices $w \in X_t \cap N(v)$ and $v \neq u \in X_t \cap N(w)$ such that $\chi(u) = \chi_{V_t}(u) = \chi_{V_t}(v) = \chi(v)$. However, this is also prevented by (L.2).

4. There are vertices $w \in X_t \cap N(v)$ and $u \in N(w) \cap (V_t \setminus X_t)$ such that $\chi_{V_{t'}}(u) = \chi_{V_t}(u) = \chi_{V_t}(v) = \chi(v)$. This means that $w \in A_{t'}(\chi_{V_{t'}}, \chi(v)) = \xi(v)$ using Equation (2). However, this is prevented by (L.3).

Next, we show that χ_{V_t} satisfies Conditions (DP.1) - (DP.3). Clearly, (DP.1) is satisfied since $\chi|_{X_{t'}} = \chi'$. For (DP.2), observe again that v is not adjacent to any vertices in $V_t \setminus X_t$. This implies

that $\xi(u) = \xi'(u) = A_{t'}(\chi_{V'_t}, \chi'(u)) = A_t(\chi_{V_t}, \chi(u))$ for all $u \in X_{t'}$. Also, $\xi(v) = A_{t'}(\chi_{V'_t}, \chi(v)) = A_t(\chi_{V_t}, \chi(u))$ by Equation (2).

To show (DP.3), let $A \in 2^{X_t}$. If $v \in A$ then $A \neq A_t(\chi_{V_t}, c)$ for all $c \in [q]$ since v is not adjacent to any vertices in $V_t \setminus X_t$. So $\rho(A) = 0 = |\{c \in [q] \mid A_t(\chi_{V_t}, c) = A\} \setminus \{\chi(v) \mid v \in X_t\}|$ as desired. Otherwise, $A \in 2^{X_{t'}}$ and, similar as above, we have that $\{c \in [q] \mid A_t(\chi_{V_t}, c) = A\} = \{c \in [q] \mid A_{t'}(\chi_{V_{t'}}, c) = A\}$. Since $A_{t'}(\chi_{V_{t'}}, \chi(v)) = \xi(v) = A_v$ in the case where $\chi(v)$ is a free color, Condition (DP.3) follows.

Conversely, suppose there exists a square q-coloring χ_{V_t} of $G[V_t]$ satisfying (DP.1) - (DP.3). We show that the algorithm sets $D[t][\chi][\xi][\rho]$ to true. Certainly, we can restrict χ_{V_t} to a square coloring $\chi_{V_{t'}} := \chi_{V_t}|_{V_{t'}}$ of $G[V_{t'}]$, and indeed it is easy to see (analogous to the arguments for (DP.1) - (DP.3) above) that this coloring satisfies the conditions of Definition 3.2 for the corresponding tuple (t', χ', ξ', ρ'). Hence, by induction, the algorithm sets $D[t'][\chi'][\xi'][\rho']$ to true. So to prove that the algorithm sets $D[t][\chi][\xi][\rho]$ to true, it suffices to show that (t, χ, ξ, ρ) is locally valid.

However, this is easy to see via the following argument. Invalidity conditions (L.1) and (L.2) cannot be fulfilled, because χ_{V_t} is a valid square coloring and $\chi = \chi_{V_t}|_{X_t}$. Also, (L.3) is also not fulfilled by Condition (DP.2). So (t, χ, ξ, ρ) is locally valid which concludes the proof.

(**Running Time.**) Let us analyze the running time of computing all DP table entries for t. For each possible choice of (χ', ξ', ρ'), we iterate over $q \leq n$ choices of the color of v and at most $2^{\mathsf{tw}+1}$ many possible choices for the set A_v (in the case that $\chi(v)$ is a free color). The time for computing χ, ξ and ρ, and then indexing the correct $D[t][\chi][\xi][\rho]$ can be naively bounded by $2^{\mathsf{tw}} \cdot n^{O(1)}$.

So overall, the running time can be bounded by

$$|D[t']| \cdot 2^{2\mathsf{tw}} \cdot n^{O(1)} = (q+1)^{\mathsf{tw}+1+(\mathsf{tw}+1)^2+2^{\mathsf{tw}+1}} \cdot 2^{2\mathsf{tw}} \cdot n^{O(1)} = (q+1)^{2^{\mathsf{tw}+2}} \cdot n^{O(1)}$$

using Equation (1).

Forget: Next, suppose $t \in V(T)$ is a forget node with unique child $t' \in V(T)$. So there is some $v \in X_t$ such that $X_t = X_{t'} \setminus \{v\}$. We iterate over all triples (χ', ξ', ρ') such that $D[t'][\chi'][\xi'][\rho']$ is true.

We define $\chi := \chi'|_{X_t}$, and also define ξ via

$$\xi(u) := \begin{cases} \xi'(u) \setminus \{v\} & \text{if } \chi'(u) \neq \chi'(v) \\ \xi'(u) \setminus \{v\} \cup (N(v) \cap X_t) & \text{if } \chi'(u) = \chi'(v) \end{cases}$$

for all $u \in X_t$. Finally, we define $\rho : 2^{X_t} \to [q]_0$ via

$$\rho(A) := \begin{cases} \rho'(A) + \rho'(A \cup \{v\}) + 1 & \text{if } \chi'(v) \notin \chi'(X_t) \text{ and } \xi(v) = A \\ \rho'(A) + \rho'(A \cup \{v\}) & \text{otherwise} \end{cases}$$

for all $A \in 2^{X_t}$. We set $D[t][\chi][\xi][\rho]$ to true.

(**Correctness.**) First suppose the algorithm sets $D[t][\chi][\xi][\rho]$ to true in iteration (χ', ξ', ρ'). In particular, this means that $D[t'][\chi'][\xi'][\rho']$ is true. Hence, by induction, there exists a square q-coloring $\chi_{V_{t'}}$ of $G[V_{t'}]$ satisfying the conditions of Definition 3.2 (with respect to (t', χ', ξ', ρ')).

We show that choosing $\chi_{V_t} := \chi_{V_{t'}}$ as a coloring for V_t suffices, i.e., that it also satisfies the conditions of Definition 3.2 with respect to (t, χ, ξ, ρ). Since $V_t = V_{t'}$ it immediately follows that χ_{V_t} is a square coloring of $G[V_t]$.

Condition (DP.1) is clearly satisfied since $\chi = \chi'|_{X_t}$. So consider (DP.2) and let $u \in X_t$. If $\chi'(u) \neq \chi'(v)$ then

$$\xi(u) = \xi'(u) \setminus \{v\} = A_{t'}(\chi_{V_{t'}}, \chi'(u)) \setminus \{v\} = A_{t'}(\chi_{V_t}, \chi(u)) \setminus \{v\} = A_t(\chi_{V_t}, \chi(u))$$

as desired. Otherwise, $\chi'(u) = \chi'(v)$ and

$$\begin{aligned}\xi(u) &= \xi'(u) \setminus \{v\} \cup (N(v) \cap X_t) \\ &= A_{t'}(\chi_{V_{t'}}, \chi'(u)) \setminus \{v\} \cup (N(v) \cap X_t) \\ &= A_{t'}(\chi_{V_t}, \chi(u)) \setminus \{v\} \cup (N(v) \cap X_t) \\ &= A_t(\chi_{V_t}, \chi(u)).\end{aligned}$$

Finally, to see that (DP.3) is satisfied, consider that for any $A \in 2^{X_t}$, the equivalence classes A and $A \cup \{v\}$ are conflated. The off-by-one in the case $\chi'(v) \notin \chi(X_t) \wedge \xi(v) = A$ is due to the color $\chi'(v)$ becoming a free color again (and in compliance with Definition 3.2 we only count free colors in ρ).

Conversely, suppose that there exists a square q-coloring χ_{V_t} of $G[V_t]$ satisfying the conditions in Definition 3.2. We show that the algorithm sets $D[t][\chi][\xi][\rho]$ to true. Certainly, $\chi_{V_{t'}} = \chi_{V_t}$ is a valid square coloring of $G[V_{t'}]$, and, since ξ and ρ are correctly updated (as argued in the last paragraph), it satisfies the conditions of Definition 3.2. Hence, by induction, $D[t'][\chi'][\xi'][\rho']$ is set to true. So the algorithm sets $D[t][\chi][\xi][\rho]$ to true.

(**Running Time.**) Let us again analyze the running time of computing all DP table entries for t. For each possible choice of (χ', ξ', ρ'), we calculate (χ, ξ, ρ) in time $2^{\mathsf{tw}} n^{O(1)}$ and set the corresponding entry $D[t][\chi][\xi][\rho]$ to true. So overall, we get a running time that is bounded

$$|D[t']| \cdot 2^{\mathsf{tw}} \cdot n^{O(1)} = (q+1)^{\mathsf{tw}+1+(\mathsf{tw}+1)^2+2^{\mathsf{tw}+1}} \cdot 2^{\mathsf{tw}} \cdot n^{O(1)} = (q+1)^{2^{\mathsf{tw}+2}} \cdot n^{O(1)}$$

using Equation (1).

Join: Finally, assume $t \in V(T)$ is a join node, i.e., t has exactly two children $t', t'' \in V(T)$ and $X_t = X_{t'} = X_{t''}$. Consider an entry $D[t][\chi][\xi][\rho]$ of the DP table. To determine whether $D[t][\chi][\xi][\rho]$ is true we iterate over all triples (χ', ξ', ρ') as well as (χ'', ξ'', ρ'') such that $D[t'][\chi'][\xi'][\rho']$ and $D[t''][\chi''][\xi''][\rho'']$ are both true. We set $D[t][\chi][\xi][\rho]$ to true if

(i) $\chi = \chi' = \chi''$,

(ii) $\xi'(u) \cap \xi''(u) = \emptyset$ for all $u \in X_t$,

(iii) $\xi(u) = \xi'(u) \cup \xi''(u)$ for all $u \in X_t$, and

(iv) there exists a function $\eta : 2^{X_{t'}} \times 2^{X_{t''}} \to [q]_0$ such that

(J.1)
$$\forall A' \in 2^{X_{t'}}, A'' \in 2^{X_{t''}} : A' \cap A'' \neq \emptyset \implies \eta(A', A'') = 0,$$

(J.2)
$$\forall A' \in 2^{X_{t'}} : \sum_{A'' \in 2^{X_{t''}}} \eta(A', A'') = \rho'(A') \quad \text{and} \quad \forall A'' \in 2^{X_{t''}} : \sum_{A' \in 2^{X_{t'}}} \eta(A', S'') = \rho''(A''),$$

(J.3)
$$\forall A \in 2^{X_t} : \rho(A) = \sum_{\substack{A',A'' \in 2^{X_t} \\ A' \cup A'' = A}} \eta(A', A'').$$

Let us provide some intuition on these conditions. Condition (i) is self-explanatory. Condition (ii) ensures that there are no conflicts when merging partial square-colorings for the two subcones $V_{t'}$ and $V_{t''}$ for non-free colors (i.e., colors from $\{\chi(v) \mid v \in X_t\}$). Indeed, if $v \in \xi'(u) \cap \xi''(u)$, this means that v is adjacent to some $v' \in V_{t'} \setminus X_t$ of color $\chi(u)$, and v is also adjacent to some $v'' \in V_{t''} \setminus X_t$ of the same color $\chi(u)$. Condition (iii) ensures that ξ is updated correctly: if the color of u is cone-adjacent to the vertices in $\xi'(u)$ in the first subcone, and to the vertices in $\xi''(u)$ in the second subcone, it will be cone-adjacent exactly to the vertices in $\xi'(u) \cup \xi''(u)$ in the entire cone V_t. Finally, for Condition (iv), we need to check that ρ is correctly obtained from ρ' and ρ'' which is achieved by an auxiliary mapping η describing the number of shared free colors between two equivalence classes from the two different subcones. More precisely, for two equivalence classes $A' \in 2^{X_{t'}}, A'' \in 2^{X_{t''}}$, we interpret the value $\eta(A', A'')$ as the number of free colors that occur in both the equivalence classes identified with A' and A''.

With this interpretation in mind, (J.1) checks that no color is shared between equivalence classes that have a non-empty shared neighborhood, i.e., no color used in one of the subcones is within distance two of that same color in the other subcone (this is the counterpart of Condition (ii) for free colors). Condition (J.2) checks that all numbers add up in the correct way. And finally, knowing the number shared colors between two equivalence classes, we can deduce ρ using (J.3). Indeed, any fixed free color c occurs in some equivalence class A' in the first subcone and an equivalence class A'' in the second subcone (note that the free colors in both subcones are identical since $\chi' = \chi''$). Hence, c is a shared color of A' and A'' and is counted in (and only in) $\eta(A', A'')$. The color c will be in the equivalence class corresponding to $A' \cup A'' = A$ in the entire subcone V_t.

Since $\chi, \xi, \rho, \chi', \xi', \rho', \chi'', \xi'', \rho''$ are fixed in each iteration, Conditions (i) and (iii) can trivially be checked in polynomial time. For Condition (iv), we view Conditions (J.1) - (J.3) as an integer linear program (ILP) with unknowns $\eta(A', A'')$, $A \in 2^{X_{t'}}$, $A'' \in 2^{X_{t''}}$. Now, we can check the feasibility of this ILP using Lemma 3.5 and set $D[t][\chi][\xi][\rho]$ accordingly.

(**Correctness.**) First suppose the algorithm sets $D[t][\chi][\xi][\rho]$ to true via triples (χ', ξ', ρ') and (χ'', ξ'', ρ'') and the mapping η that satisfies (J.1) - (J.3). Since $D[t'][\chi'][\xi'][\rho']$ and $D[t''][\chi''][\xi''][\rho'']$ are true, we know by induction that there exist square q-colorings $\chi_{V_{t'}}$ of $G[V_{t'}]$ and $\chi_{V_{t''}}$ of $G[V_{t''}]$ which satisfy the conditions in Definition 3.2. We show that if the algorithm sets $D[t][\chi][\xi][\rho]$ to true, then we can combine $\chi_{V_{t'}}$ and $\chi_{V_{t''}}$ into a square coloring χ_{V_t} for $G[V_t]$ using the function η. Towards this end, we start by showing the following claim.

Claim 3.1. There is a square q-colorings $\lambda_{V_{t''}}$ of $G[V_{t''}]$ such that

1. $\lambda_{V_{t''}}$ satisfies (DP.1) - (DP.3) with respect to $(t'', \chi'', \xi'', \rho'')$, and
2. for every $A' \in 2^{X_{t'}}$ and $A'' \in 2^{X_{t''}}$ it holds that

$$\eta(A', A'') = |\{c \in [q] \mid A_{t'}(\chi_{V_{t'}}, c) = A' \wedge A_{t''}(\lambda_{V_{t''}}, c) = A''\} \setminus \{\chi(v) \mid v \in X_t\}|.$$

Proof. We obtain $\lambda_{V_{t''}}$ from $\chi_{V_{t''}}$ by renaming colors. More precisely, we define a bijection $\pi \colon [q] \to [q]$ and define $\lambda_{V_{t''}}(u) \coloneqq \pi(\chi_{V_{t''}}(u))$ for all $u \in V_{t''}$.

First, we choose the identity for all colors in the bag X_t, i.e., $\pi(c) = c$ for all $c \in \{\chi''(v) \mid v \in X_t\}$.

In the second step, we will use η to permute free colors. For each pair of equivalence classes $A' \in 2^{X_{t'}}, A'' \in 2^{X_{t''}}$, we arbitrarily match $\eta(A', A'')$ free colors from the equivalence class A' to free

colors from the equivalence class A'', i.e., we define π in such a way that

$$\eta(A', A'') = |\{c \in [q] \mid A_{t'}(\chi_{V_{t'}}, c) = A' \wedge A_{t''}(\chi_{V_{t'}}, \pi^{-1}(c)) = A''\} \setminus \{\chi(v) \mid v \in X_t\}|$$

Note that this is always possible, and indeed that it achieves a full matching, by condition (J.2). Also, $\lambda_{V_{t''}}$ still satisfies (DP.1) - (DP.3) with respect to $(t'', \chi'', \xi'', \rho'')$ since only free colors are renamed. ⌟

Using the last claim, we may assume without loss of generality that $\chi_{V_{t''}} = \lambda_{V_{t''}}$. We define a coloring χ_{V_t} of V_t via

$$\chi_{V_t}(v) = \begin{cases} \chi_{V_{t'}}(v) & \text{if } v \in V_{t'} \\ \chi_{V_{t''}}(v) & \text{if } v \in V_{t''} \end{cases}.$$

First observe that χ_{V_t} is well-defined by Condition (i).

We show that χ_{V_t} is a square coloring of $G[V_t]$. Let $u, w \in V_t$ be distinct vertices such that $\text{dist}_{G[V_t]}(u, w) \leq 2$. If $uw \in E(G)$ then $u, w \in V_{t'}$ or $u, w \in V_{t''}$, and the respective square coloring $\chi_{V_{t'}}$ or $\chi_{v_{t''}}$ prohibits that u and v have the same color.

Otherwise, there is some $v \in V_t$ such that $uv, vw \in E(G)$. As before, if $u, v, w \in V_{t'}$ or $u, v, w \in V_{t''}$, the respective square coloring $\chi_{V_{t'}}$ or $\chi_{v_{t''}}$ prohibits that u and v have the same color. Since u, v and v, w are adjacent, each pair of vertices must be in the same subcone. So without loss of generality the only other case we need to consider is $u \in V_{t'} \setminus X_t$, $v \in X_t$ and $w \in V_{t''} \setminus X_t$. Suppose without loss of generality that $\chi_{V_t}(u) = \chi_{V_t}(w)$. First suppose there is some $u' \in X_t$ such that $\chi_{V_t}(u) = \chi(u')$. Then $v \in \xi'(u)$ and $v \in \xi''(w)$ which contradicts Condition (ii).

Otherwise, let $A_u \coloneqq A_t(\chi_{V_t}, \chi(u))$ and $A_w \coloneqq A_t(\chi_{V_t}, \chi(w))$. We have $v \in A_u$ and $v \in A_w$ by definition and hence, $A_u \cap A_w \neq \emptyset$. So $\eta(A_u, A_w) = 0$ by condition (J.1). This means that

$$\{c \in [q] \mid A_{t'}(\chi_{V_{t'}}, c) = A_u \wedge A_{t''}(\chi_{V_{t''}}, c) = A_w\} \setminus \{\chi(v) \mid v \in X_t\} = \emptyset$$

by Claim 3.1. But $c \coloneqq \chi_{V_t}(u) = \chi_{V_t}(w)$ is contained in the first set, but not the second one. So this gives again a contradiction. Overall, we get that χ_{V_t} is a square coloring of $G[V_t]$.

It remains to show that χ_{V_t} satisfies the conditions in Definition 3.2. Condition (DP.1) immediately follows from (i).

For $u \in X_t$ we have that

$$\begin{aligned} \xi(u) &= \xi'(u) \cup \xi''(u) \\ &= A_{t'}(\chi_{V_{t'}}, \chi'(u)) \cup A_{t''}(\chi_{V_{t''}}, \chi''(u)) \\ &= A_{t'}(\chi_{V_{t'}}, \chi(u)) \cup A_{t''}(\chi_{V_{t''}}, \chi(u)) \\ &= \Big\{ v \in X_t \;\Big|\; \Big(\exists w \in V_{t'} \setminus X_{t'} \colon vw \in E(G) \wedge \chi_{V_{t'}}(w) = \chi(u)\Big) \\ &\qquad\qquad \vee \Big(\exists w \in V_{t''} \setminus X_{t''} \colon vw \in E(G) \wedge \chi_{V_{t''}}(w) = \chi(u)\Big) \Big\} \\ &= A_t(\chi_{V_t}, \chi(u)) \end{aligned}$$

which implies Condition (DP.2).

For Condition (DP.3), we first observe that

$$A_t(\chi_{V_t}, c) = A_{t'}(\chi_{V_{t'}}, c) \cup A_{t''}(\chi_{V_{t''}}, c)$$

for all colors $c \in [q]$ using the same arguments as above. Hence, Condition (J.3) and Claim 3.1 imply that

$$\rho(A) = \sum_{\substack{A', A'' \in 2^{X_t} \\ A' \cup A'' = A}} \eta(A', A'') = |\{c \in [q] \mid A_t(\chi_{V_t}, c) = A\} \setminus \{\chi(v) \mid v \in X_t\}|$$

for all $A \in 2^{X_t}$ as desired.

Conversely, suppose there exists a square q-coloring χ_{V_t} of $G[V_t]$ satisfying the conditions in Definition 3.2. We show that the algorithm sets $D[t][\chi][\xi][\rho]$ to true.

To do this, we consider $\chi_{V_t}|_{V_{t'}}$ and $\chi_{V_t}|_{V_{t''}}$. Certainly, these colorings are square colorings of $G[V_{t'}]$ and $G[V_{t''}]$, respectively. They immediately induce triples (χ', ξ', ρ') and (χ'', ξ'', ρ'') by choosing these mappings in such a way that $\chi_{V_t}|_{V_{t'}}$ witnesses that $D[t', \chi', \xi', \rho']$ is true. The colorings $\chi_{V_t}|_{V_{t'}}$ and $\chi_{V_t}|_{V_{t''}}$ also define η by setting

$$\eta(A', A'') \coloneqq |\{c \in [q] \mid A_{t'}(\chi_{V_t}, c) = A' \wedge A_{t''}(\chi_{V_t}, c) = A''\} \setminus \{\chi(v) \mid v \in X_t\}|.$$

Now, it is easy to check that Conditions (i) - (iv) are satisfied. So the algorithm sets $D[t][\chi][\xi][\rho]$ to true.

(**Running Time.**) We analyze the running time of computing all DP table entries for t.

Iterating over all choices of $\chi, \xi, \rho, \chi', \xi', \rho', \chi'', \xi'', \rho''$ takes

$$|D[t]| \cdot |D[t']| \cdot |D[t'']| \leq (q+1)^{3 \cdot (\mathsf{tw}+1+(\mathsf{tw}+1)^2+2^{\mathsf{tw}+1})}$$

many iterations by Equation (1). For each iteration, we only require polynomial time to check Conditions (i) - (iii). For the ILP used to check Condition (iv), the number of unknowns is at most $2^{2(\mathsf{tw}+1)}$. Note that (J.1) just says that certain unknowns are zero. We can hence ignore them and delete them from all equations. The number of remaining linear equations from (J.2) and (J.3) is at most $(2^{\mathsf{tw}+1} + 2^{\mathsf{tw}+1}) + 2^{\mathsf{tw}+1} = 3 \cdot 2^{\mathsf{tw}+1}$. The unknowns are constrained by $0 \leq \eta(S', S'') \leq q$. Hence, the ILP can be solved in time in time $O(2^{2 \cdot (\mathsf{tw}+1)} \cdot 2^{\mathsf{tw}+1} \cdot (q+1)^{3 \cdot 2^{\mathsf{tw}+1}+1})$ by Lemma 3.5.

So overall, the running time can be bounded by

$$\begin{aligned} &(q+1)^{3 \cdot (\mathsf{tw}+1+(\mathsf{tw}+1)^2+2^{\mathsf{tw}+1})} \cdot 2^{3 \cdot (\mathsf{tw}+1)} \cdot (q+1)^{3 \cdot 2^{\mathsf{tw}+1}+1} \cdot n^{O(1)} \\ &= (q+1)^{6 \cdot (\mathsf{tw}+1)+3 \cdot (\mathsf{tw}+1)^2+6 \cdot 2^{\mathsf{tw}+1}+1} \cdot n^{O(1)} \\ &= (q+1)^{2^{\mathsf{tw}+4}} \cdot n^{O(1)}. \end{aligned}$$

We complete the proof by observing that, for every node t, all entries of $D[t]$ can be computed in time $(q+1)^{2^{\mathsf{tw}+4}} \cdot n^{O(1)}$. So overall, the running time is bounded by $|V(T)| \cdot (q+1)^{2^{\mathsf{tw}+4}} \cdot n^{O(1)}$. □

4 Lower Bound for Graphs of Bounded Treewidth

We have seen an algorithm solving Square Coloring in time $(q+1)^{2^{\mathsf{tw}+4}} \cdot n^{O(1)}$. We now show that assuming the Exponential Time Hypothesis (ETH), this running time is essentially optimal. More specifically, we show:

Theorem 1.2 (restated). *Assuming* ETH*, for any $\varepsilon > 0$ and any function f, there is no $f(\mathsf{tw}) n^{(2-\varepsilon)^{\mathsf{tw}}}$ time algorithm solving* Square Coloring *on graphs of treewidth* tw*.*

4.1 Problems Used in the Reduction

In our proof of Theorem 1.2, we will use two other problems, which we define below. The first will be the starting point of our reduction, the second will be an intermediate problem.

4.1.1 The (Colored) Subgraph Isomorphism Problem

The starting point for the hardness result is the well-known Subgraph Isomorphism problem:

> Subgraph Isomorphism
> **Input:** A pattern graph H and a host graph G
> **Question:** Does there exist a subgraph of G that is isomorphic to H?

We will actually use a colored variant of this, in which the vertices of H are colored using $|V(H)|$ unique colors. The graph G is also colored arbitrarily using these $|V(H)|$ colors. The problem now has the additional constraint that the isomorphism must preserve colors. We define a convenient version of this as follows:

> Colored Subgraph Isomorphism
> **Input:** A pattern graph H with k vertices, a host graph G with $k \cdot n$ vertices, and a function $f : V(G) \to V(H)$ such that $\forall h \in V(H) : |f^{-1}(h)| = n$
> **Question:** Does there exist a subgraph G' of G such that $\forall h \in V(H) : |f^{-1}(h) \cap V(G')| = 1$ and G' is isomorphic to H?

4.1.2 The (Restricted) Vector k-Sum Problem

We first define the basic Vector k-Sum problem, before stating a restricted version that we use in our reduction.

> Vector k-Sum
> **Input:** k lists $A_1, \ldots, A_k$ of m-dimensional vectors, each of size n, and an m-dimensional target vector t
> **Question:** Is there a way to pick one vector from each list such that the sum of these vectors is exactly t? That is,
>
> $$\exists v_1 \in A_1, \ldots, v_k \in A_k : \sum_{i=1}^{k} v_i = t \quad ?$$

Definition 4.1. Let parameters $m, n \in \mathbb{Z}_{>0}$ be arbitrary. We define a set of m-dimensional vectors A to be a *node-representing vector list* if there exist $D^+, D^- \subseteq [m]$ with $|D^+ \cup D^-| = |D^+| + |D^-| = 3$ such that

$$\forall a \in A : \forall j \in [m] : a_j \in \begin{cases} [1, n^2] & \text{if } j \in D^+ \\ [-n^2, -1] & \text{if } j \in D^- \\ \{0\} & \text{otherwise} \end{cases}$$

We call the elements of D^+ the *positive dimensions*, the elements of D^- the *negative dimensions* and the elements of $D^+ \cup D^-$ the *non-zero dimensions*.

RESTRICTED VECTOR k-SUM

Definition: Let parameters $m, k, n \in \mathbb{Z}_{>0}$ be arbitrary. We define a RESTRICTED VECTOR k-SUM instance to be a VECTOR k-SUM instance that satisfies all of the following conditions:

1. The target vector of the instance is $(0, ..., 0) \in \mathbb{Z}^m$.
2. There are k vector lists, each of size exactly n^4.
3. All vector lists are node-representing vector lists with parameters m, n.
4. Each $j \in [m]$ is a non-zero dimension in exactly two of the n^4 vector lists.

4.2 The Reduction Chain

To prove Theorem 1.2, we rely on the hardness of certain instances of COLORED SUBGRAPH ISOMORPHISM under ETH, and construct a two-step reduction from those instances of COLORED SUBGRAPH ISOMORPHISM to instances of SQUARE COLORING.

For the former, we use a result that follows directly from [35]. It shows the hardness of COLORED SUBGRAPH ISOMORPHISM with cubic pattern graphs parameterized by its number of edges. Recall that a graph H is *cubic* if every vertex has degree exactly 3.

Theorem 4.2. *Let $m_0 > 0$ be arbitrary. The* COLORED SUBGRAPH ISOMORPHISM *problem, restricted to cubic pattern graphs with $m \geq m_0$ edges, cannot be solved in time $f(m)n^{o(m/\log m)}$ for any function f, unless* ETH *fails.*

This follows from [35, Corollary 6.1] when using as a graph family $\mathcal{G}$ a family of cubic expander graphs with at least m_0 edges. It is well-known that cubic expander graphs have treewidth $\Omega(m)$ (where m denotes the number of edges) [26].

Now, our two-step reduction consists of a reduction from COLORED SUBGRAPH ISOMORPHISM with cubic pattern graphs to RESTRICTED VECTOR k-SUM instances, and then from those instances to SQUARE COLORING instances. More precisely, we prove the following theorems:

Theorem 4.3. *There exists a polynomial-time algorithm $\mathcal{A}$ that, given a* COLORED SUBGRAPH ISOMORPHISM *instance with a k-vertex cubic pattern graph H (which has $m = 3k/2$ edges) and a $(k \cdot n)$-vertex host graph G, outputs an equivalent* RESTRICTED VECTOR k-SUM *instance with parameters m, k, n.*

Theorem 4.4. *There exists an algorithm $\mathcal{B}$ such that for every $\varepsilon > 0$ there exists an $m_\varepsilon \in \mathbb{Z}_{>0}$ such that for all $m > m_\varepsilon$, the following holds. Let $k \in \mathbb{Z}_{>0}$ be arbitrary. Given a* RESTRICTED VECTOR k-SUM *instance with parameters m, k, n, the algorithm $\mathcal{B}$ produces an equivalent* SQUARE COLORING *instance with treewidth at most $(1+\varepsilon)\log(m) + O(1)$ and with $(k+m+n)^{O(1)}$ vertices.*

Moreover, the algorithm runs in time $(k+m+n)^{O(1)}$.

Assuming both theorems hold, we can already prove Theorem 1.2, i.e., assuming ETH, SQUARE COLORING cannot be solved in time $f(\mathsf{tw})n^{(2-\varepsilon)^{\mathsf{tw}}}$ for any $\varepsilon > 0$.

Proof of Theorem 1.2. Without loss of generality let $0 < \varepsilon \leq 1$ (otherwise it is trivial). Choose $0 < \widetilde{\varepsilon} \leq 1$ such that $1 - \widetilde{\varepsilon} = \log(2 - \varepsilon)$; note that this is always possible. Now assume that there exists an algorithm $\mathcal{C}$ solving SQUARE COLORING in time $f(\mathsf{tw})N^{(2-\varepsilon)^{\mathsf{tw}}} = f(\mathsf{tw})N^{2^{(1-\widetilde{\varepsilon})\mathsf{tw}}}$ on graphs of treewidth at most tw. Without loss of generality, we may assume that f is monotonically increasing.

First, we use Theorem 4.4 with $\varepsilon' = \widetilde{\varepsilon}$, obtaining an algorithm $\mathcal{B}$ and a constant $m_{\varepsilon'} > 0$. Our goal is to design an algorithm with running time $h(m)n^{o(m/\log m)}$ for COLORED SUBGRAPH

Isomorphism, where h is some arbitrary function and where the pattern graphs are restricted to cubic graphs with $m > m_{\varepsilon'}$ edges. By Theorem 4.2, this implies that ETH is false.

Hence let such a Colored Subgraph Isomorphism instance be given. We use algorithm $\mathcal{A}$ from Theorem 4.3 to convert this into an equivalent Restricted Vector k-Sum instance with parameters m, k, n. This conversion runs in time $(n+m)^{O(1)}$.

For all $m > m_{\varepsilon'}$, the algorithm $\mathcal{B}$ from above converts a Restricted Vector k-Sum instance with parameters m, k, n into a Square Coloring instance with treewidth $\mathsf{tw} \leq (1+\widetilde{\varepsilon})\log(m)+\alpha$, where α denotes a suitable absolute constant, and with $N = (k+m+n)^{O(1)}$ vertices. Note that $m = 3k/2$, so $N = (m+n)^{O(1)}$. This conversion runs in time $(k+m+n)^{O(1)} = (m+n)^{O(1)}$.

Now, we use algorithm $\mathcal{C}$ on this newly constructed Square Coloring instance. Let us define $g(m) := f((1+\widetilde{\varepsilon})\log(m)+\alpha)$. Observe that $f(\mathsf{tw}) \leq g(m)$. Then algorithm $\mathcal{C}$ runs in time

$$f(\mathsf{tw})N^{2^{(1-\widetilde{\varepsilon})\mathsf{tw}}} = g(m)\left((m+n)^{O(1)}\right)^{2^{(1-\widetilde{\varepsilon})((1+\widetilde{\varepsilon})\log(m)+\alpha)}}.$$

We rewrite this as

$$g(m)(m+n)^{O(2^{\xi})}$$

where

$$\xi := (1-\widetilde{\varepsilon})((1+\widetilde{\varepsilon})\log(m)+\alpha).$$

Setting $\delta := \widetilde{\varepsilon}^2 > 0$, we get that $\xi = (1-\delta)\log(m) + O(1)$. Hence $2^{\xi} = O(m^{(1-\delta)})$. So the running time can be written as

$$g(m)(m+n)^{O(2^{\xi})} = g(m)m^{O(m^{(1-\delta)})}n^{O(m^{(1-\delta)})} = h(m)n^{o(m/\log m)}$$

for some suitable $h(m) = g(m)\cdot m^{O(m^{(1-\delta)})}$.

Note that we have an additional running time of $(m+n)^{O(1)}$ for both of the two reduction steps, but this is dominated by the above running time. Hence we have an algorithm for Colored Subgraph Isomorphism running in total time $h(m)n^{o(m/\log m)}$ as desired. □

In the next two sections, we will construct the reductions that prove Theorems 4.3 and 4.4.

4.3 From Subgraph Isomorphism to Restricted Vector Sum

First we prove Theorem 4.3, which provides the reduction from Colored Subgraph Isomorphism to Restricted Vector k-Sum. We restate it below:

Theorem 4.3 (restated). *There exists a polynomial-time algorithm $\mathcal{A}$ that, given a* Colored Subgraph Isomorphism *instance with a k-vertex cubic pattern graph H (which has $m = 3k/2$ edges) and a $(k{\cdot}n)$-vertex host graph G, outputs an equivalent* Restricted Vector k-Sum *instance with parameters m, k, n.*

Proof. Let a Colored Subgraph Isomorphism instance with a k-vertex m-edge cubic pattern graph H, along with a host graph G be given. In the following, we construct an equivalent Restricted Vector k-Sum instance.

First, we purge redundant vertices from H. That is, for each $u_0 \in V(H)$ with neighbours u_1, u_2, u_3 such that $\nexists (u_0', u_1', u_2', u_3') \in f^{-1}(u_0)\times f^{-1}(u_1)\times f^{-1}(u_2)\times f^{-1}(u_3) : \{u_0'u_1', u_0', u_2', u_0'u_3'\} \subseteq E(G)$, we remove u_0 from $V(H)$ and all vertices in $f^{-1}(u_0)$ from $V(G)$.

Having done that, we define our target vector to be $(0,\ldots,0) \in \mathbb{Z}^m$. We rename the vertices of H to $\{h_1,\ldots,h_k\}$. We also define an arbitrary bijection $b_E : E(H) \to [m]$.

Structure of the Vectors We now create k vector lists $A_{h_1}, \ldots, A_{h_k}$. The vector list A_{h_i} is responsible for the node $h_i \in V(H)$. All vectors have dimension m. Each dimension represents one of the edges of H. Specifically, for a vector a, $a[b_E(e)]$ is the entry of the vector corresponding to the edge $e \in E(H)$.

Now pick an arbitrary vertex $u_0 \in V(H)$ and let it be connected to three other vertices $u_1, u_2, u_3 \in V(H)$. In the following, we construct the list A_{u_0} of vertex u_0. The list has at most n^4 vectors for now – we will later fill the list with dummy vectors to achieve exactly n^4 vectors.

Specifically, for each choice of $(u'_0, u'_1, u'_2, u'_3) \in f^{-1}(u_0) \times f^{-1}(u_1) \times f^{-1}(u_2) \times f^{-1}(u_3)$ such that $u'_0u'_1, u'_0u'_2, u'_0u'_3 \in E(G)$, we create one vector $a_{u'_0,\{u'_1,u'_2,u'_3\}} \in A_{u_0}$. Note that the vector is identified by the choice of central vertex u'_0 and the set of its three neighbouring vertices $\{u'_1, u'_2, u'_3\}$.

Entries of the Vectors We now construct the entries of these vectors. First, we define an arbitrary ordering $\preceq$ on $V(H)$.

For each edge $vw \in E(H)$, we denote by $E(G)_{vw}$ the set of edges in G that connect a vertex in $f^{-1}(v)$ to a vertex in $f^{-1}(w)$. Furthermore, we define an arbitrary bijection $b_{vw} : E(G)_{vw} \to [n^2]$.

Fix a node $u_0 \in V(H)$ and let u_0 have neighbours u_1, u_2, u_3. We go through all choices of $(u'_0, u'_1, u'_2, u'_3) \in f^{-1}(u_0) \times f^{-1}(u_1) \times f^{-1}(u_2) \times f^{-1}(u_3)$ such that $u'_0u'_1, u'_0u'_2, u'_0u'_3 \in E(G)$. For each such choice and each $j \in [m]$, we set

$$a_{u'_0,\{u'_1,u'_2,u'_3\}}[j] := \begin{cases} b_{u_0u_i}(u'_0u'_i) & \text{if } \exists i \in [3] : j = b_E(u_0u_i) \wedge u_0 \preceq u_i \\ -b_{u_0u_i}(u'_0u'_i) & \text{if } \exists i \in [3] : j = b_E(u_0u_i) \wedge u_0 \succeq u_i \\ 0 & \text{otherwise} \end{cases}$$

Dummy vectors This almost concludes the construction. We still have to guarantee that each vector list has exactly n^4 vectors. So far, each vector list has at most n^4 vectors. Recall that we purged redundant vertices from $V(H)$ at the beginning of the reduction, thus each vector list must have at least one vector. Hence, we can simply fill each vector list by duplicating any of its vectors until the list has exactly n^4 vectors.

Proof of Running Time Note that in the construction, for each node $u_0 \in V(H)$ with neighbours u_1, u_2, u_3, we iterate over all 4-tuples $(u'_0, u'_1, u'_2, u'_3) \in f^{-1}(u_0) \times f^{-1}(u_1) \times f^{-1}(u_2) \times f^{-1}(u_3)$ and each $j \in [m]$. Since we do a constant number of operations for each such choice, this takes time $O(kn^4m)$. We have $m = 3k/2$, thus this can be rewritten as $O(m^2n^4)$. This is obviously the dominating term in the running time of the reduction.

Proof of Correct Format We show that the instance we constructed fulfills all conditions for being a Restricted Vector k-Sum instance. Conditions 1 and 2 are obviously satisfied.

For condition 3, fix a node $u_0 \in V(H)$ and let u_0 have neighbours u_1, u_2, u_3. Consider the vector list A_{u_0}. In order for A_{u_0} to be a node-representing vector list, we must present its positive dimensions D^+ and negative dimensions D^-. Indeed, for each $i \in [3]$, we define

$$b_E(u_0u_i) \in \begin{cases} D^+ & \text{if } u_0 \preceq u_i \\ D^- & \text{otherwise} \end{cases} \tag{3}$$

This obviously shows $|D^+ \cup D^-| = |D^+| + |D^-| = 3$. Furthermore, if we rewrite this definition, it becomes $D^+ = \{j \mid \exists i \in [3] : j = b_E(u_0u_i) \wedge u_0 \preceq u_i\}$ and $D^- = \{j \mid \exists i \in [3] : j = b_E(u_0u_i) \wedge u_i \succeq u_0\}$.

Hence all that remains to be shown is that

$$\forall a \in A : \forall j \in [m] : a_j \in \begin{cases} [1, n^2] & \text{if } \exists i \in [3] : j = b_E(u_0u_i) \wedge u_0 \preceq u_i \\ [-n^2, -1] & \text{if } \exists i \in [3] : j = b_E(u_0u_i) \wedge u_0 \succeq u_i \\ \{0\} & \text{otherwise} \end{cases}$$

However, this can be seen directly from the definition of the vector entries above.

For condition 4, fix $j \in [m]$. We must show that j is a non-zero dimension in exactly two vector lists. Since b_E is a bijection, there must be $u, v \in V(H)$ such that $b_E(uv) = j$. Without loss of generality, assume $u \preceq v$. Then by the definition of the non-zero dimension in formula 3 above, it must be true that j is a positive dimension in the vector list A_u and a negative dimension in the vector list A_v. Furthermore, there do not exist any other vector lists in which j is a non-zero dimension.

Proof of Equivalence Note that a choice of a vector in A_{u_0}, where $N(u_0) = \{u_1, u_2, u_3\}$, induces a choice of vertices from $f^{-1}(u_0), f^{-1}(u_1), f^{-1}(u_2)$ and $f^{-1}(u_3)$. Now let us look at our construction of vector entries, and why they lead to equivalence of this new instance to the original Colored Subgraph Isomorphism instance.

Suppose there exists a solution $g : V(H) \mapsto V(G)$ for the Colored Subgraph Isomorphism instance. For each $u_0 \in V(H)$ with neighbours $u_1, u_2, u_3 \in V(H)$, we choose from A_{u_0} the vector $a_{g(u_0),\{g(u_1),g(u_2),g(u_3)\}}$. Note that this vector exists because it is guaranteed that the edges $g(u_0)g(u_1), g(u_0)g(u_2), g(u_0)g(u_3) \in E(G)$ exist by the fact that g is a solution. We obtain a choice of vectors that we will prove to be a solution to the Restricted Vector k-Sum instance.

Now fix an edge $u_0v_0 \in E(H)$ with $u_0 \preceq v_0$ and let u_0 have neighbours u_1, u_2 and v_0 have neighbours v_1, v_2. We consider the dimension $b_E(u_0v_0)$ responsible for the edge u_0v_0. We know that $a_{g(u_0),\{g(v_0),g(u_1),g(u_2)\}}[b_E(u_0v_0)] = b_{u_0v_0}(g(u_0)g(v_0))$ and that $a_{g(v_0),\{g(u_0),g(v_1),g(v_2)\}}[b_E(u_0v_0)] = -b_{u_0v_0}(g(u_0)g(v_0))$. Hence their sum is 0. Furthermore, these two vectors are from the only two lists which have non-zero entries is this dimension. Hence the sum of all chosen vectors has a 0 in this dimension. Since this is true for all dimensions, we have proven that the vector choices are a solution to the Restricted Vector k-Sum instance.

Conversely, suppose there exists a choice of vectors that is a solution to the Restricted Vector k-Sum instance. We now show that the choice of vertices it induces is well-defined and that it is a solution to the Colored Subgraph Isomorphism instance.

Observe the dimension corresponding to an arbitrary edge $uv \in E(H)$ with $u \preceq v$, namely dimension $b_E(uv)$. Of the chosen vectors, only two can have a non-zero entry in this dimension, namely the chosen vectors $a_u \in A_u$ and $a_v \in A_v$. Let a_u induce a selection of $u' \in f^{-1}(u)$ and $v' \in f^{-1}(v)$ and let a_v induce a selection of $u'' \in f^{-1}(u)$ and $v'' \in f^{-1}(v)$. By construction, a_u will have a value of $b_{uv}(u'v')$ in dimension $b_E(uv)$, while a_v will have a value of $-b_{uv}(u''v'')$ in dimension $b_E(uv)$. Since our target vector is $(0, \ldots, 0)$, this means that we have the constraint $(b_{uv}(u'v')) + (-b_{uv}(u''v'')) = 0$. Since b_{uv} is a bijection this implies the constraint $u'v' = u''v''$. Hence the induced choices u' and u'' as well as v' and v'' coincide. This shows that the vertex choice induced by the vector solution is well-defined.

Let us summarize the induced vector choices into a map $g : V(H) \mapsto V(G)$. It remains to show that this map is a solution to the Colored Subgraph Isomorphism instance with which we started.

However, this is trivial: For every edge $uv \in E(H)$, suppose u has other neighbours u_1, u_2. We must have chosen a vector $a_{g(u),\{g(v),g(u_1),g(u_2)\}}$ from A_u. Since $a_{g(u),\{g(v),g(u_1),g(u_2)\}}$ exists, it must be true that $g(u)g(v), g(u)g(u_1), g(u)g(u_2) \in E(G)$. For us, it is enough that $g(u)g(v) \in E(G)$.

Since this is true for every edge $uv \in E(H)$, all edge constraints of the Colored Subgraph Isomorphism instance are satisfied. □

4.4 From Restricted Vector Sum to Square Coloring

We now show Theorem 4.4, which provides a way to convert a Restricted Vector k-Sum instance to an equivalent Square Coloring instance of low treewidth. We restate it below:

Theorem 4.4 (restated). *There exists an algorithm $\mathcal{B}$ such that for every $\varepsilon > 0$ there exists an $m_\varepsilon \in \mathbb{Z}_{>0}$ such that for all $m > m_\varepsilon$, the following holds. Let $k \in \mathbb{Z}_{>0}$ be arbitrary. Given a* Restricted Vector k-Sum *instance with parameters m, k, n, the algorithm $\mathcal{B}$ produces an equivalent* Square Coloring *instance with treewidth at most $(1+\varepsilon)\log(m) + O(1)$ and with $(k+m+n)^{O(1)}$ vertices.*

Moreover, the algorithm runs in time $(k+m+n)^{O(1)}$.

Our proof of this theorem is very long and highly technical. It spans the rest of this section.

Let $\varepsilon > 0$ be given. We choose m_ε in such a way that

$$\forall m > m_\varepsilon : m^\varepsilon > 2(1+\varepsilon)\log m + 6 \tag{4}$$

which is certainly possible. We will explain this choice later. Continuing, let $m > m_\varepsilon$ also be given. Finally, let a Restricted Vector k-Sum instance with parameters m, k, n be given, for $k, n \in \mathbb{Z}_{>0}$. Name the vector lists $A_1, \ldots, A_k$.

4.4.1 Preparation

Before we describe the reduction algorithm, we note a few key definitions and details used in the reduction.

Colorless and forced-color vertices We use colorless vertices in our construction. These are normal vertices that simply won't be assigned a color. Note that neighbouring nodes of colorless vertices must still have pairwise distinct colors. We will later describe how to get rid of these colorless vertices again.

Similarly, we will use vertices with a predetermined color, e.g. when we say that we create a "red vertex". These are normal vertices that we will connect in such a way that they are forced to assume the color we assign (red, in this case). Of course, red is not a fixed color, but will be identified as being the color that is used to color a certain other vertex somewhere in the graph.

Colors used in reduction We now define our target number of colors q. That is, the result of the reduction will be a YES-instance if and only if it will be colorable using q colors.

The colors we use are divided into three categories:

Counting Colors: There are a total of $2m \cdot 2n^6$ so-called *counting colors*, which are grouped into color classes. There are $2m$ of these color classes, one pair of color classes for each dimension of the vectors from the Restricted Vector k-Sum instance. We number the color classes $1, \ldots, 2m$ and say that dimension $i \in [m]$ corresponds to color classes $2i-1$ and $2i$. Each color class has $2n^6$ colors.

Logic Colors: We have an additional three colors which we name red, green and blue. They are used for local logic computations in some of the gadgets.

Neutral Colors: Finally, we have some number $q_{\text{colorless}}$ of colors which are used in the final step of the reduction, where we replace the colorless vertices with colored vertices. We will later define $q_{\text{colorless}}$ to be the total number of colorless vertices in our constructed graph, but for now we leave it as a variable.

Hence in total, we have $q = 2m \cdot 2n^6 + 3 + q_{\text{colorless}}$ many colors.

The central vertex x The output graph of our reduction will have a special colorless vertex named x. It is special because most important conditions are encoded via its neighbourhood. Accordingly, most gadgets we construct for our reduction will have an essential part of their "output" be how they control the coloring of neighbors of x.

Relations Defining Gadgets To describe some of the gadgets used in the reduction, we define a specific notion of a gadget, along with a specific notation. Our gadgets will have a set of input vertices V_{in}, a set of output vertices V_{out} and a set of vertices V_x that are adjacent to the central vertex x. In the input, we distinguish between single vertices and sets of vertices which we call "color class inputs". Each gadget will have some number d of these color class inputs, and we will call these sets of vertices $I_1, \ldots, I_d$.

The behaviour of the gadget will then be defined by a relation R of input colorings, output colorings and vector specifications of colorings of V_x. We refer to the latter as "vector output". Informally, the relation defines what combinations of output colorings and vector outputs are allowed for what inputs.

More formally, the relation R will be a subset of tuples $(\chi_{\text{in}}, \chi_{\text{out}}, v_x)$ where $\chi_{\text{in}} : V_{\text{in}} \to [q]$ and $\chi_{\text{out}} : V_{\text{out}} \to [q]$, and where $v_x \in \mathbb{Z}^d_{\geq 0}$. For each $j \in [d]$, the jth entry of v_x corresponds to the j-th input color class. Informally said, it will contain the number of output vertices in V_x that are colored using colors from $\chi_{\text{in}}(I_j)$. More formally, each gadget has a corresponding function $p : (V_x \to [q]) \to \mathbb{Z}^d_{\geq 0}$ which maps output colorings to output vectors. For a given coloring $\chi_{\text{in}} : V_{\text{in}} \to [q]$ (which will always be clear from context) and any coloring $\chi_x : V_x \to [q]$, the jth entry of $p(\chi_x)$ is then given by

$$p(\chi_x)_j := |\chi_{\text{in}}(I_j) \cap \chi_x(V_x)|$$

Our definitions of R will always ensure that $\sum_{j=1}^{d} (v_x)_j = |V_x|$, i.e. that all colors on the x-adjacent vertices must be distinct.

The relation R will be defined by two sets of constraints, C_{in} and C_R. Informally, C_{in} will contain constraints on the input, meaning it will specify for what colorings of V_{in} we even care about the behaviour of the gadget, and C_R will contain constraints that specify the behaviour for those inputs. Formally, for any colorings $\chi_{\text{in}} : V_{\text{in}} \to [q], \chi_{\text{out}} : V_{\text{out}} \to [q]$ and any $v_x \in \mathbb{Z}^d_{\geq 0}$, we have that $(\chi_{\text{in}}, \chi_{\text{out}}, v_x) \in R$ if and only if χ_{in} satisfies all constraints contained in C_{in} and $(\chi_{\text{in}}, \chi_{\text{out}}, v_x)$ satisfies all constraints contained in C_R. We write this as $\chi_{\text{in}} \vdash C_{\text{in}}$ and $(\chi_{\text{in}}, \chi_{\text{out}}, v_x) \vdash C_R$.

We will formally define each gadget in a *specification table* containing the parameters the construction and behaviour of the gadget depends on, a description of the sets of interface vertices $V_{\text{in}}, V_{\text{out}}$ and V_x, as well as a description of the sets of conditions C_{in} and C_R from which R is constructed.

For each gadget we use, we provide a construction and then prove that this construction does indeed conform to the behaviour described. We define the latter formally as follows:

Definition 4.5. We say that a gadget G *behaves according to* a relation R if it satisfies both of the following:

Output Guarantee: . For any square q-coloring χ of the gadget such that $\chi|_{V_{\text{in}}} \vdash C_{\text{in}}$, it must also hold that $(\chi|_{V_{\text{in}}}, \chi|_{V_{\text{out}}}, p(\chi|_{V_x})) \vdash C_R$ (and, as a direct corollary, that $(\chi|_{V_{\text{in}}}, \chi|_{V_{\text{out}}}, p(\chi|_{V_x})) \in R$).

Existence Guarantee: For any $(\chi_{\text{in}}, \chi_{\text{out}}, v_x) \in R$ and any vector $S = (S_1, \ldots, S_d)$ with $S_1 \subseteq \chi_{\text{in}}(I_1), S_2 \subseteq \chi_{\text{in}}(I_2), \ldots, S_d \subseteq \chi_{\text{in}}(I_d)$ such that $\forall j \in [d] : |S_j| = (v_x)_j$, there exists a square q-coloring χ of the gadget such that $\chi|_{V_{\text{in}}} = \chi_{\text{in}}$, $\chi|_{V_{\text{out}}} = \chi_{\text{out}}$ and $\forall j \in [d] : S_j \subseteq \chi(V_x)$.

Furthermore, we say that a gadget G *satisfies a specification table* if G behaves according to the the relation defined by the table entries specifying C_{in} and C_R.

We are now ready to describe our reduction.

4.4.2 The Top-Level Structure

In this section, we describe and prove the reduction on a very high level. Specifically, we will assume the existence of three gadgets that behave in a certain way and then use them to construct the reduction output. Finally, we prove that this output is equivalent to the Restricted Vector k-Sum instance that was the reduction input. The construction of the gadgets we use – which is very involved – will be done in the next section.

The gadgets we will use here are called the subset gadget, the color class copy gadget and the vector selection gadget.

The Subset Gadget As mentioned in the last section, we specify the inputs, outputs and behaviour of gadgets via a gadget specification table. Table 1 does this for the subset gadget.

Subset Gadget	
parameter(s)	$\alpha, \beta \in \mathbb{Z}_{>0}$ with $\alpha \geq \beta$
V_{in}	α vertices
V_{out}	β vertices
V_x	$\emptyset$
C_{in}	$\lvert\chi_{\text{in}}(V_{\text{in}})\rvert = \lvert V_{\text{in}}\rvert$
C_R	$\chi_{\text{out}}(V_{\text{out}}) \subseteq \chi_{\text{in}}(V_{\text{in}})$ and $\lvert\chi_{\text{out}}(V_{\text{out}})\rvert = \lvert V_{\text{out}}\rvert$

Table 1: The specification table of the subset gadget

To build intuition on gadget specification tables, let us explicitly state what the table expresses. The first row tells us that the construction and behaviour of the subset gadget depend on two parameters $\alpha, \beta \in \mathbb{Z}_{>0}$ with $\alpha \leq \beta$. The next two rows that the gadget has α input vertices and β output vertices. Note that in all our gadgets, the sets of input and output vertices always have an empty intersection. The fourth row tells us that none of the vertices of the subset gadget are adjacent to the central vertex x (as defined in the section on preparation). The fifth row, which specifies the constraints C_{in}, tells us that we only care about the behaviour of the subset gadget if the input coloring colors the input vertices with pairwise distinct colors. Finally, the last row specifies the constraints C_R, namely that the output coloring must use pairwise distinct colors that must be a subset of the colors used in the input coloring. Stated more intuitively, the output colors must be a subset of the input colors. Note that if $\alpha = \beta$, the output coloring must use exactly the input colors. In this case, we may also call the gadget an *equality gadget*.

If we look back on how we defined gadgets, the subset gadget must satisfy both the output guarantee and the existence guarantee. In this case, the output guarantee states that as long as the

input consists of pairwise distinct colors, the output must be a subset of those colors. The existence guarantee states that for any input coloring such that its colors are pairwise distinct and any output colorings such that its colors are a subset of the input colors, there exists a square q-coloring of the subset gadget that expands on those two colorings.

We will use the subset gadget a lot in our reduction, mainly as equality gadget for copying around colors or groups of colors.

The Color Class Copy Gadget Next, we describe the color class copy gadget. It groups the color classes into the color classes and creates a copy of each color class. The gadget is formally specified by Table 2.

Color Class Copy Gadget	
parameter(s)	$m, n \in \mathbb{Z}_{>0}$
V_{in}	vertex groups $X_1, \ldots, X_{2m}$, each of size $2n^6$; define $X := \bigcup_{i=1}^{2m} X_i$
V_{out}	vertex groups $Y_1, \ldots, Y_{2m}$, each of size $2n^6$; define $Y := \bigcup_{i=1}^{2m} Y_i$
V_x	$\emptyset$
C_{in}	$\nexists u, v \in X : \chi_{\text{in}}(u) = \chi_{\text{in}}(v)$
C_R	$\forall i \in [2m] : \chi_{\text{in}}(X_i) = \chi_{\text{out}}(Y_i)$

Table 2: The specification table of the color class copy gadget

Basically, this gadget takes as input a grouping of distinct colors on $X_1, \ldots, X_{2m}$, and guarantees that this same grouping is also output on the vertex groups $Y_1, \ldots, Y_{2m}$. The reader may note that this gadget could be constructed by simply, for each i, connecting X_i to Y_i via a subset gadget. So why introduce a gadget for this? The construction we will describe in a later section will actually be slightly more involved, with the aim being to achieve a lower treewidth in the reduction graph.

Our reduction output will only contain one instance of the color class gadget.

The Vector Selection Gadget Finally, we come to the vector selection gadget. Our reduction will contain k of these gadgets, one for each of the vector groups $A_1, \ldots, A_k$. Its construction and that of its subgadgets is very involved, and will be the longest part in the description of the reduction.

More formally, the purpose of this gadget is to simulate the selection of a vector from a node-representing vector list A as defined in Definition 4.1. Recall that that means that there exist positive dimensions D^+ and negative dimensions D^- such that $|D^+ \cup D^-| = |D^+| + |D^-| = 3$, and that we call $D^+ \cup D^-$ the non-zero dimensions. We define $D^+ \cup D^- =: \{z_1, z_2, z_3\}$.

We remove zero-entries from each $a_j \in A$ and denote the result $\text{NZ}(a_j) = ((a_j)_{z_1}, (a_j)_{z_2}, (a_j)_{z_3})$.

The vector selection gadget will have six color class inputs, and as such we have $d = 6$ for the dimension d of the vector output. We call a vector output $v_x \in \mathbb{Z}^6$ of this gadget a *vector-generated output* of A if $\exists t \in [n^4] : v_x = (n^6 + \text{NZ}(a_t)_1, n^6 - \text{NZ}(a_t)_1, n^6 + \text{NZ}(a_t)_2, n^6 - \text{NZ}(a_t)_2, n^6 + \text{NZ}(a_t)_3, n^6 - \text{NZ}(a_t)_3)$. In that case we say that v_x is generated by a_t (or $\text{NZ}(a_t)$).

The behaviour of the gadget is then specified by Table 3.

This specification is slightly more involved, so let us go through it one by one. The first row simply tells us that the construction and behaviour of the gadget depends on a node-representing A – this is the vector list from which we will simulate the selection of a vector. The second and third rows tell us that the gadget has six color class inputs and three inputs for the logic colors red, green and blue. Furthermore, it has three logic color outputs. If we peek at the first condition of C_R in the last row of the table, the gadget will actually guarantee that it outputs its logic color inputs

Vector Selection Gadget	
parameter(s)	a node-representing vector list A with parameters $m, n \in \mathbb{Z}_{>0}$
V_{in}	• Six groups of $2n^6$ vertices $I_1, I_2, I_3, I_4, I_5, I_6$ (the color class inputs), and • three "logic" input vertices r, g, b.
V_{out}	three "logic" output vertices r', g', b'.
V_x	a total of $6n^6$ vertices
C_{in}	• $\lvert\chi_{\text{in}}(I_1 \cup I_2 \cup I_3 \cup I_4 \cup I_5 \cup I_6 \cup \{r, g, b\})\rvert =$ $\lvert I_1 \cup I_2 \cup I_3 \cup I_4 \cup I_5 \cup I_6 \cup \{r, g, b\}\rvert$ • $\chi_{\text{in}}(r) = \text{red}, \chi_{\text{in}}(g) = \text{green}, \chi_{\text{in}}(b) = \text{blue}$
C_R	1. $\chi_{\text{out}}(r') = \chi_{\text{in}}(r)$, $\chi_{\text{out}}(g') = \chi_{\text{in}}(g)$ and $\chi_{\text{out}}(b') = \chi_{\text{in}}(b)$, 2. It must hold that v_x is a vector-generated output of A, as defined above.

Table 3: The specification table of the vector selection gadget

unchanged, meaning the logic color outputs r', g', b' receive the same colors as the logic color inputs r, g, b, respectively. So the interesting behaviour doesn't happen in the coloring of the output, but rather in the vector output of the gadget – i.e. in the coloring of the neighbours of x. For this, we look at the fourth row of the table, which specifies that V_x consists of some $6n^6$ vertices. If we peek at the second condition of C_R, we see that the vector output encoded on these vertices must be a vector-generated output. Being vector-generated means that it encodes a certain vector from the vector list A, but in a slightly modified form as specified in the definition. Finally, the fifth row simply tells us that we only care about the behaviour of the gadget if all inputs are colored with pairwise different colors and if r, g, b are colored with the logic colors red, green and blue.

Connecting the Gadgets We provide a sketch of the reduction in Figure 5. Before we describe how we fit the gadgets described above together, a quick word on the intuition behind this construction.

To reduce the Restricted Vector k-Sum instance to a Square Coloring instance, we will need a concept of numbers, vectors, summation and equality constraints in the context of square q-colorings. The vector selection gadgets already gives us a concept of vectors and numbers via its vector output by coloring neighbours of x with a certain number of colors from each of its input color classes. If two vector selection gadgets get the same color class as one of their color class inputs, and they both expose a certain number of colors from this class to x, this can be viewed as summation: the total number of colors from that color class exposed to x is the sum of colors from that color class that are exposed to x by each of the two vector selection gadgets. Note that the sets of colors that are exposed must be pairwise different, since all neighbours of x are trivially within distance two. This directly gives us a concept of lesser-than-or-equal constraints: The sum of the number of colors from a certain color class that are exposed by vector selection gadgets must be less than the total number of colors from that color class. Finally, we can encode an equality constraints $x = y$ by encoding it as the two lesser-than-or-equal constraints $x \leq y$ and $y \leq x$.

With this in mind, let us connect the gadgets we have so far.

First, we create the central vertex x and a color class copy gadget. We denote the latter by $G^{(\text{cpy})}$. Now we create k vector selection gadgets, which we denote by $G^{(\text{sel},1)}, \ldots, G^{(\text{sel},k)}$. For each $i \in [k]$, we define the parameter of $G^{(\text{sel},i)}$ to be the vector list A_i.

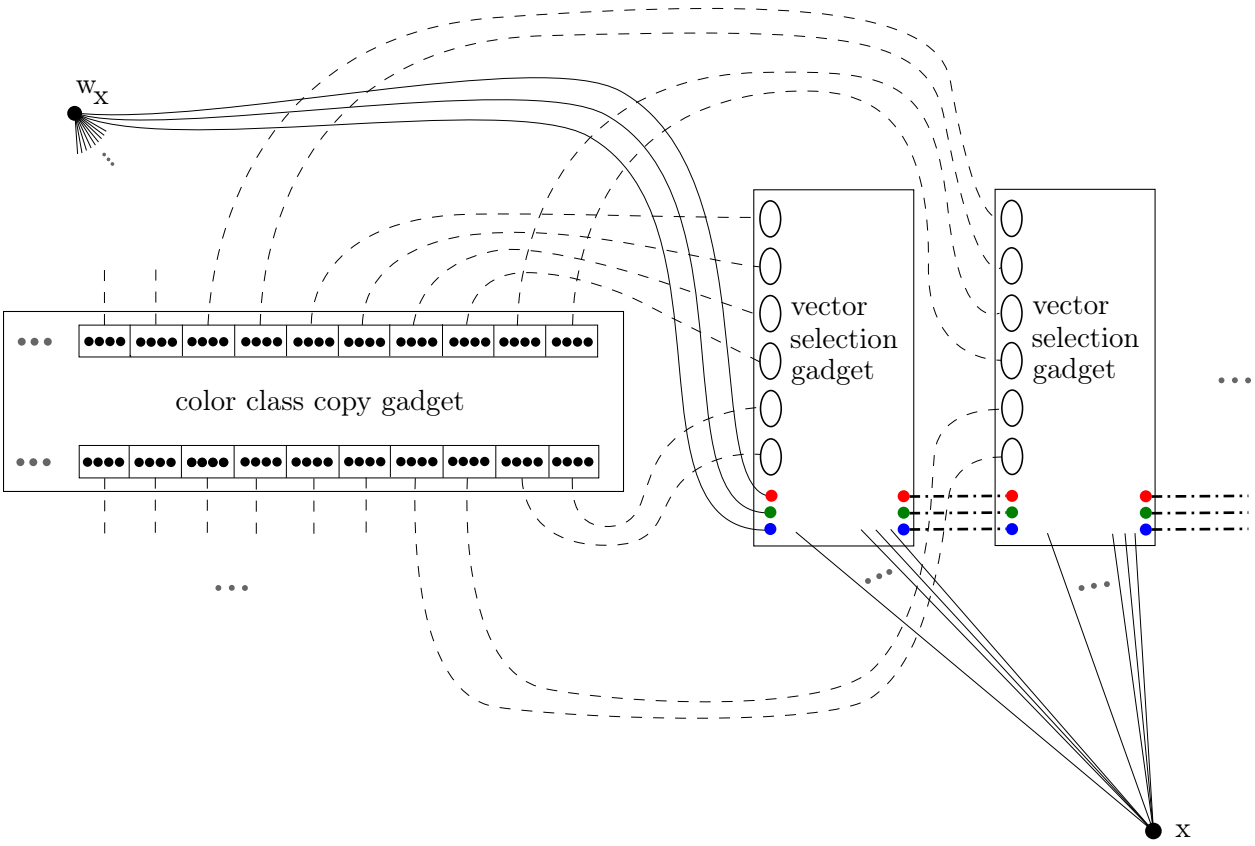


Figure 5: Overview of the construction. Dashed edges represent connections via equality gadgets.

We identify any of the elements (e.g. sets, vertices, conditions) of $G^{(\text{sel},i)}$ or $G^{(\text{cpy})}$ by superscripting them with (sel, i) or (cpy), respectively. E.g. the input vertex set V_{in} of $G^{(\text{sel},1)}$ and the set X_1 of $G^{(\text{cpy})}$ become $V_{\text{in}}^{(\text{sel},1)}$ and $X_1^{(\text{cpy})}$, respectively.

We assume that for each $i \in [k]$, all vertices in $V_x^{(\text{sel},i)}$ are already connected to x via an edge. In fact, these are the only vertices in the entire reduction connected to x.

We create a colorless vertex w_X and connect it to all vertices in $V_{in}^{(\text{cpy})}$, as well as to the vertices $r^{(\text{sel},1)}, g^{(\text{sel},2)}, b^{(\text{sel},3)}$. This ensures that all colors used in the color classes, as well as the logic colors, are all pairwise distinct. We also ensure that the logic colors are passed to each vector selection gadget by using one equality gadget each to connect, for each $i \in [k-1]$, the vertex $(r')^{(\text{sel},i)}$ with $r^{(\text{sel},i+1)}$, $(g')^{(\text{sel},i)}$ with $g^{(\text{sel},i+1)}$ and $(b')^{(\text{sel},i)}$ with $b^{(\text{sel},i+1)}$. Note that connecting two vertices via an equality gadget simply means that one of the vertices is the input and the other is the output.

All that remains is to route the color classes from the color class copy gadget to the vector selection gadgets. Recall that in a Restricted Vector k-Sum instance, each $j \in [m]$ is a non-zero dimension in exactly two of the vector lists. Also, each vector list is a node-representing vector list and thus has exactly three non-zero dimensions. Finally, recall that dimension $i \in [m]$ corresponds to color classes $2i-1$ and $2i$.

For each $i \in [k]$, let vector list A_i have non-zero dimensions $D^+ \cup D^- =: \{z_1^{(i)}, z_2^{(i)}, z_3^{(i)}\}$. The corresponding color class indices are the elements of the ordered set $L^{(i)} := (l_1^{(i)}, \ldots, l_6^{(i)}) := (2z_1^{(i)}-1, 2z_1^{(i)}, 2z_2^{(i)}-1, 2z_2^{(i)}, 2z_3^{(i)}-1, 2z_3^{(i)})$. We wish to connect the color classes with indices from $L^{(i)}$ to the inputs of $G^{(\text{sel},i)}$ in some way. Note that by the fact that each dimension is a non-zero dimension in at most two vector lists, this means that each color class is an input to at most two vector selection gadgets. In other words, $\forall \ell \in [2m] : |\{i \mid \ell \in L^{(i)}\}| = 2$.

Hence we go through each $i \in [k]$ and each $\ell_j^{(i)} \in L^{(i)}$. Define $\ell := \ell_j^{(i)}$ for simplicity. If $\ell \notin \bigcup_{i' \in [i-1]} L^{(i')}$, we connect $I_j^{(\text{sel},i)}$ to $X_\ell^{(\text{cpy})}$ via a subset gadget. Otherwise, we connect $I_j^{(\text{sel},i)}$ to $Y_\ell^{(\text{cpy})}$ via a subset gadget. Again, connecting two vertex groups via a subset gadget simply means that one vertex group is the input of the subset gadget and the other is the output.

This concludes the top-level construction of the reduction output instance.

Correctness Let us argue why this new Square Coloring instance is equivalent to the Restricted Vector k-Sum instance. In other words, we show the following lemma:

Lemma 4.6. *Assume that there exist gadgets satisfying the specification tables 1, 2 and 3, and that they are used in the construction above.*

Then the input Restricted Vector k-Sum *instance has a solution $a^{(1)} \in A_1, \ldots, a^{(k)} \in A_k$ if and only if the* Square Coloring *instance constructed above has a solution coloring χ.*

For the proof, we need the following property, which is easy to verify for each of our gadget constructions.

Property 4.7. For any gadget G in our reduction except for the color class copy gadget, and for any vertex $v \in V(G)$ that is adjacent to any vertex from $V_{\text{in}} \cup V_{\text{out}} \cup V_x$ of G, we have that v is colorless.

This property holds even for the lower-level gadgets which we will specify later. Stated more intuitively, its says that for all but one gadget, any outwardly exposed vertices are only connected to colorless vertices within the gadget. This makes our correctness proofs easier because we do not need to worry about color conflicts between two inner vertices (i.e. non-input and non-output vertices) that are from different gadgets.

We are now ready to prove Lemma 4.6.

Proof. For the *first direction*, assume the RESTRICTED VECTOR k-SUM instance has a solution $a^{(1)} \in A_1, \ldots, a^{(k)} \in A_k$ such that $\sum_{i=1}^{k} a^{(i)} = (0, \ldots, 0) \in \mathbb{Z}^m$. We must show that our newly constructed SQUARE COLORING instance also has a solution coloring χ.

Let us construct this coloring χ. First, we color $V_{\text{in}}^{(\text{cpy})}$ using all of the $2m \cdot 2n^6$ counting colors. We also color $r^{(\text{sel},1)}, g^{(\text{sel},1)}, b^{(\text{sel},1)}$ using the logic colors red, green and blue, respectively. Note that we have now already used each available color once.

We can now arbitrarily color $V_{\text{out}}^{(\text{cpy})}$ such that the coloring satisfies $C_R^{(\text{cpy})}$, i.e. such that the color groupings are preserved. We may then use the existence guarantees of $G^{(\text{cpy})}$ to find a coloring for all vertices of $G^{(\text{cpy})}$. Now recall that each color class input of each vector seleciton gadget is connected via a subset gadget to either a $X_i^{(\text{cpy})}$ for some i or a $Y_i^{(\text{cpy})}$ for some i. We color it such that the condition C_R of the subset gadget is fulfilled, i.e. such that the color class input gets assigned all colors of that color class. We can then use the existence guarantee of each subset gadget to find a coloring for the vertices of the gadget. We also color, for each $i \in [k] \setminus \{1\}$, the vertices $r^{(\text{sel},i)}, g^{(\text{sel},i)}, b^{(\text{sel},i)}$ with red, green and blue respectively. We do the same with $(r')^{(\text{sel},i)}, (g')^{(\text{sel},i)}, (b')^{(\text{sel},i)}$ for each $i \in [k-1]$. We can now use the existence guarantees for the subset gadgets that connect these logic inputs and logic outputs.

All that remains is to expand the coloring χ we constructed so far to the vertices of the vector selection gadgets. Note that for each $i \in [k]$, the input and output vertices of vector selection gadget $G^{(\text{sel},i)}$ are already colored. In particular, for each $\ell := \ell_j^{(i)} \in L^{(i)}$ (with $j \in [6]$), we know that $I_j^{(\text{sel},i)}$ is connected to either $X_\ell^{(\text{cpy})}$ or $Y_\ell^{(\text{cpy})}$ via a subset gadget, and is hence colored using the color class with index ℓ. To use the existence guarantees, it suffices to specify an output vector $v_x^{(\text{sel},i)}$ and a vector $S^{(\text{sel},i)}$ that satisfy the conditions of the existence guarantees. In other words, we need that $(\chi|_{V_{\text{in}}^{(\text{sel},i)}}, \chi|_{V_{\text{out}}^{(\text{sel},i)}}, v_x^{(\text{sel},i)}) \in R^{(\text{sel},i)}$ and $S^{(\text{sel},i)} = (S_1^{(\text{sel},i)}, \ldots, S_6^{(\text{sel},i)})$ with $S_1^{(\text{sel},i)} \subseteq \chi(I_1^{(\text{sel},i)}), \ldots, S_6^{(\text{sel},i)} \subseteq \chi(I_6^{(\text{sel},i)})$ such that $\forall j \in 6 : |S_j^{(\text{sel},i)}| = (v_x^{(\text{sel},i)})_j$. The former means that the tuple must satisfy the conditions in $C_{\text{in}}^{(\text{sel},i)}$ and $C_R^{(\text{sel},i)}$. The conditions in $C_{\text{in}}^{(\text{sel},i)}$ and the first condition of $C_R^{(\text{sel},i)}$ are obviously satisfied, hence it suffices to choose $v_x^{(\text{sel},i)}$ such that it is a vector-generated output of A_i. We choose it to be the vector-generated output generated by $a^{(i)}$ from the solution of the RESTRICTED VECTOR k-SUM instance, that is we choose it to be $(n^6 + \text{NZ}(a^{(i)})_1, n^6 - \text{NZ}(a^{(i)})_1, n^6 + \text{NZ}(a^{(i)})_2, n^6 - \text{NZ}(a^{(i)})_2, n^6 + \text{NZ}(a^{(i)})_3, n^6 - \text{NZ}(a^{(i)})_3)$.

Now for our choice of $S^{(\text{sel},i)}$. We want all neighbours of the central vertex x to have distinct colors, hence we want that $\forall (i,j), (i',j') \in [k] \times [6] : (i,j) \neq (i',j') \implies S_j^{(\text{sel},i)} \cap S_{j'}^{(\text{sel},i')} = \emptyset$. To do this, we go through all $d \in [m]$. Let $i, i' \in [k]$ with $i < i'$ be the indices of the two vector lists in which d is a non-zero dimension, i.e. $d = z_j^{(i)}$ and $d = z_{j'}^{(i')}$ for some $j, j' \in [3]$. We know that $X_{2d-1}^{(\text{cpy})}$ is connected via subset gadget to $I_{2j-1}^{(\text{sel},i)}$ and $X_{2d}^{(\text{cpy})}$ is connected to $I_{2j}^{(\text{sel},i)}$. Furthermore $Y_{2d-1}^{(\text{cpy})}$ is connected to $I_{2j'-1}^{(\text{sel},i')}$ and $Y_{2d}^{(\text{cpy})}$ is connected to $I_{2j'}^{(\text{sel},i')}$.

We have $(v_x^{(\text{sel},i)})_{2j-1} = n^6 + \text{NZ}(a^{(i)})_d$ and $(v_x^{(\text{sel},i)})_{2j} = n^6 - \text{NZ}(a^{(i)})_d$ and $(v_x^{(\text{sel},i')})_{2j-1} = n^6 + \text{NZ}(a^{(i')})_d$ and $(v_x^{(\text{sel},i')})_{2j} = n^6 - \text{NZ}(a^{(i')})_d$. Furthermore, we have that by the fact that $a^{(1)}, \ldots, a^{(k)}$ is a solution to the RESTRICTED VECTOR k-SUM, we must have $\forall d'' \in [m] : \sum_{i''=1}^{k} a_{d''}^{(i'')} = 0$. Since i, i' are the only two non-zero dimensions of d, we hence have $a_d^{(i)} + a_d^{(i')} = 0$. Thus, $(v_x^{(\text{sel},i)})_{2j-1} + (v_x^{(\text{sel},i')})_{2j-1} = 2n^6$ and $(v_x^{(\text{sel},i)})_{2j} + (v_x^{(\text{sel},i')})_{2j} = 2n^6$.

Note that each color class has exactly $2n^6$ colors. Hence we can choose $S_{2j-1}^{(\text{sel},i)}$ as any arbitrary $(v_x^{(\text{sel},i)})_{2j-1}$ colors from $\chi(I_{2j-1}^{(\text{sel},i)}) = \chi(X_{2d-1}^{(\text{cpy})})$. We use the rest of the colors from $\chi(X_{2d-1}^{(\text{cpy})}) =$

$\chi(I_{2j-1}^{(\text{sel},i')})$ for $S_{2j-1}^{(\text{sel},i')}$. We furthermore choose $S_{2j}^{(\text{sel},i)}$ as any arbitrary $(v_x^{(\text{sel},i)})_{2j}$ colors from $\chi(I_{2j}^{(\text{sel},i)}) = \chi(X_{2d}^{(\text{cpy})})$ and choose $S_{2j-1}^{(\text{sel},i')}$ as the rest of the colors from $\chi(X_{2d}^{(\text{cpy})}) = \chi(I_{2j-1}^{(\text{sel},i')})$.

Note that apart from guaranteeing that the neighbours of x are colored using distinct colors, it is easy to see that actually, for each counting color, there exists exactly one neighbour of x that is colored using that color.

We can now finally use the existence guarantees of the vector selection gadgets to find colorings for them. Hence χ is now a complete coloring of our graph.

It is obvious that there are no color conflicts between neighbours of w_X. We have also already argued above that there are no color conflicts between neighbours of x. Furthermore, by the existence guarantees of the gadgets we used, no color conflicts can exist between two vertices from the same gadget. Hence the only color conflicts that could still exist would be between two vertices from different gadgets. However, such conflicts are prevented by Property 4.7. Hence χ is a valid square q-coloring.

Now for the *second direction*. Assume our new Square Coloring instance has a solution χ. We must show that the Restricted Vector k-Sum instance also has a solution $a^{(1)}, \ldots, a^{(k)}$.

We know that the neighbours of w_X must be colored using pairwise different colors, meaning the vertices in $V_{in}^{(\text{cpy})}$ and $r^{(\text{sel},1)}, g^{(\text{sel},1)}, b^{(\text{sel},1)}$ all have different colors. Hence we can use the output guarantees of $G^{(\text{cpy})}$ to conclude that $\forall i \in [2m] : \chi(X_i^{(\text{cpy})}) = \chi(Y_i^{(\text{cpy})})$. We also use the output guarantees of all subset gadgets connecting vertex groups of the color class copy gadgets to color class inputs of vector selection gadgets. As a simple corollary, we get that for all $i \in [k]$ and each $l := l_j^{(i)} \in L^{(i)}$, we have that $\chi(I_j^{(\text{sel},i)}) = \chi(X_\ell^{(\text{cpy})})$.

Call $\chi(r^{(\text{sel},1)}), \chi(g^{(\text{sel},1)}), \chi(b^{(\text{sel},1)})$ red, green and blue, respectively. We now go through each $i \in [k-1]$. For each such i, we know that the first condition of $C_R^{(\text{sel},i)}$ is that $\chi((r')^{(\text{sel},i)}) = \chi(r^{(\text{sel},i)})$, $\chi((g')^{(\text{sel},i)}) = \chi(g^{(\text{sel},i)})$ and $\chi((b')^{(\text{sel},i)})) = \chi(b^{(\text{sel},i)})$. And indeed, we can use the output guarantees of $G^{(\text{sel},i)}$ to show that this holds. Furthermore, we can use the output guarantees of the subset gadgets connecting the logic color outputs of $G^{(\text{sel},i)}$ to the logic color inputs of $G^{(\text{sel},i+1)}$ to show that $\chi(r^{(\text{sel},i+1)}) = \chi((r')^{(\text{sel},i)})$, $\chi(g^{(\text{sel},i+1)}) = \chi((g')^{(\text{sel},i)})$ and $\chi(b^{(\text{sel},i+1)})) = \chi((b')^{(\text{sel},i)})$. Hence, by transitivity, all these vertices are colored red, green and blue, respectively.

For each $i \in [k]$, we now use the output guarantees of $G^{(\text{sel},i)}$ again. We know that the second condition of $C_R^{(\text{sel},i)}$ is that $v^{(i)} := p(\chi|_{V_x^{(\text{sel},i)}})$ is a vector-generated output of A_i. We choose $a^{(1)}, \ldots, a^{(k)}$ such that for all $i \in [k]$, we have that $v^{(i)}$ is the vector-generated output generated by $a^{(i)}$. In other words, we have $v^{(i)} = (n^6 + \text{NZ}(a^{(i)})_1, n^6 - \text{NZ}(a^{(i)})_1, n^6 + \text{NZ}(a^{(i)})_2, n^6 - \text{NZ}(a^{(i)})_2, n^6 + \text{NZ}(a^{(i)})_3, n^6 - \text{NZ}(a^{(i)})_3)$.

We now go through each $d \in [m]$. Let $i, i' \in [k]$ with $i < i'$ be the indices of the two vector lists in which d is a non-zero dimension, i.e. $d = z_j^{(i)}$ and $d = z_{j'}^{(i')}$ for some $j, j' \in [3]$.

We know that $\chi(I_{2j-1}^{(\text{sel},i)}) = \chi(I_{2j'-1}^{(\text{sel},i')}) = \chi(X_{2d-1}^{(\text{cpy})})$ and $\chi(I_{2j}^{(\text{sel},i)}) = \chi(I_{2j'}^{(\text{sel},i')}) = \chi(X_{2d}^{(\text{cpy})})$. Hence we know that in $G^{(\text{sel},i)}$, a total of $n^6 + \text{NZ}(a^{(i)})_j$ neighbours of x are colored using colors from $\chi(X_{2d-1}^{(\text{cpy})})$ and a total of $n^6 - \text{NZ}(a^{(i)})_j$ neighbours of x are colored using colors from $\chi(X_{2d}^{(\text{cpy})})$. Furthermore, in $G^{(\text{sel},i')}$, a total of $n^6 + \text{NZ}(a^{(i')})_{j'}$ neighbours of x are colored using colors from $\chi(X_{2d-1}^{(\text{cpy})})$ and a total of $n^6 - \text{NZ}(a^{(i')})_{j'}$ neighbours of x are colored using colors from $\chi(X_{2d}^{(\text{cpy})})$. Hence, $(n^6 + \text{NZ}(a^{(i)})_j) + (n^6 + \text{NZ}(a^{(i')})_{j'})$ neighbours of x are colored using colors from $\chi(X_{2d-1}^{(\text{cpy})})$, and $(n^6 - \text{NZ}(a^{(i)})'_j) + (n^6 + \text{NZ}(a^{(i')})_{j'})$ neighbours are colored using colors from $\chi(X_{2d}^{(\text{cpy})})$.

Recall that $\forall r \in [2m] : |X_r^{(\text{cpy})}| = 2n^6$. Since χ must color neighbours of x using pairwise distinct colors, we must thus have $(n^6 + \text{NZ}(a^{(i)})_j) + (n^6 + \text{NZ}(a^{(i')})_{j'}) \leq 2n^6$ and $(n^6 - \text{NZ}(a^{(i)})_j) + (n^6 -$

$\text{NZ}(a^{(i')})_{j'}) \leq 2n^6$. Rewriting, we have $\text{NZ}(a^{(i)})_j + \text{NZ}(a^{(i')})_{j'} \leq 0$ and $\text{NZ}(a^{(i)})_j + \text{NZ}(a^{(i')})_{j'} \geq 0$. Hence, $\text{NZ}(a^{(i)})_j + \text{NZ}(a^{(i')})_{j'} = 0$.

Recall that $d = z_j^{(i)} = z_{j'}^{(i')}$. Thus, $a_d^{(i)} = \text{NZ}(a^{(i)})_j$ and $a_d^{(i')} = \text{NZ}(a^{(i')})_{j'}$, meaning we can rewrite the above as $a_d^{(i)} + a_d^{(i')} = 0$. Since A_i and $A_{i'}$ were the only two vector lists in which d is a non-zero dimension, we get that $\sum_{r=1}^{k} a_d^{(r)} = 0$.

Since we proved the above for all $d \in [m]$, we can conclude $\sum_{r=1}^{k} a^{(r)} = 0$, which means $a^{(1)}, \ldots, a^{(k)}$ is a solution for the Restricted Vector k-Sum instance. □

4.4.3 Constructing the Top-Level Gadgets

We will now see how to actually construct the three gadgets we used in our top-level construction.

The construction of the subset gadget and color class copy gadget are relatively short. However, the construction of the vector selection gadget is very involved and requires us to first build a long sequence of subgadgets.

Subset Gadget, Equality Gadget and Busses We begin with the most basic gadget, the (α, β) *subset gadget*. It has some number α of input vertices, as well as $\beta < \alpha$ output vertices, and it ensures that the colors used to color the output vertices are a subset of the colors used to color the input vertices.

We have already specified the subset gadget via Table 1. A γ *equality gadget* is a subset gadget with $\gamma := \alpha = \beta$. It ensures that the set of input colors is the same as the set of output colors.

The construction of the subset gadget is very simple. The α input vertices in V_{in} are all connected to a colorless vertex a. The vertex a is then connected to $q - \alpha$ new vertices, which we call the *complement vertices* C. The complement vertices C are all connected to another colorless vertex b, which is also connected to all output vertices V_{out}.

We wish to prove that the subset gadget actually encodes the subset constraint, that is:

Lemma 4.8. *The gadget constructed above satisfies the specification table of the subset gadget, as shown in Table 1.*

Proof. **Output Guarantees:** Let a square q-coloring χ such that $\chi|_{V_{\text{in}}} \vdash C_{\text{in}}$ be given.

Since the input vertices V_{in}, the complement vertices and a form a star with q rays, χ must be injective on $V_{\text{in}} \cup C$. However, since there are α input vertices and $q - \alpha$ complement vertices, we must have that the complement vertices are colored using all colors which do not appear on the input vertices. Similarly, since the complement vertices C, the output vertices C and b form a star, the output vertices must be colored using pairwise distinct colors which do not appear on the complement vertices – in other words, using a subset of the colors of the input vertices, with no duplicate colors.

Hence, $(\chi|_{V_{\text{in}}}, \chi|_{V_{\text{out}}}, p(\chi|_{V_x})) \vdash C_R$, where $p(\chi|_{V_x}) = ()$ is the zero-dimensional vector.

Existence Guarantees: Similarly, let $(\chi_{\text{in}}, \chi_{\text{out}}, ()) \in R$ and $S = ()$ be given. It suffices to extend χ_{in} and χ_{out} to a coloring χ by coloring the complement vertices C using the $q - \alpha$ colors not appearing on the input. It is easy to see this is a valid square q-coloring.

□

Color Class Copy Gadget Now for the color class copy gadget, which was specified in Table 2. As was mentioned earlier, there as actually a very simple way to construct this gadget. However, we use a more involved construction in order to keep the treewidth of the reduction output graph low.

First, we ensure that the colors used to color the nodes in V_{in} are the same as the colors used to color the nodes in V_{out}. We do this by connecting the all nodes in the former set to the nodes in the latter set via a $2m \cdot 2n^6$ equality gadget.

We now introduce $2r$ new colorless vertices S, for $r := \lceil \frac{(1+\varepsilon)\log(m)}{2} \rceil$ (where ε was given at the start of the reduction). This choice will guarantee $\binom{2r}{r} > 2m$, as we will argue later. Each $0 \leq i \leq 2m$ gets assigned a distinct size r subset S_i of S. This is possible due to $\binom{2r}{r} > 2m$. Each vertex of X_i is now connected to all vertices in S_i, meaning X_i and S_i now form a biclique. Similarly, each vertex of Y_i is connected to all vertices in $S \setminus S_i$.

This concludes the construction of the gadget.

Lemma 4.9. *The gadget constructed above satisfies the specification table of the color class copy gadget, as shown in Table 2.*

Proof. **Output Guarantees:** Let a square q-coloring χ be given such that $\chi|_{V_{\text{in}}} \vdash C_{\text{in}}$.

The $2m \cdot 2n^6$ equality gadget forces all vertices in Y to use the colors from $\chi(X)$.

Now consider some fixed i. First, note that the colors of the vertices in Y_i must be pairwise distinct, since they are all adjacent to any arbitrary vertex in $S \setminus S_i$ and thus within distance two of each other.

Now consider any $j \neq i$. We know that S_j and $S \setminus S_i$ share a vertex. Hence the colors used to color X_j cannot be used in Y_i, since the nodes in these sets are within distance two of each other. Furthermore, we have already argued above that the vertices in Y, and hence in Y_i, can only be colored using the colors in $\chi(X)$.

Hence the only colors in $\chi(X)$ which remain to color Y_i with are those used to color X_i. Since there are $2n^6$ vertices in Y_i which must have pairwise distinct colors, they must therefore have exactly the colors used to color X_i.

Hence $(\chi|_X, \chi|_Y, ()) \vdash C_R$.

Existence Guarantees: Let $(\chi_{\text{in}}, \chi_{\text{out}}, ()) \in R$ and $S = ()$ be given. Obviously, the input and output color sets of the subset gadget connecting X and Y are the same. Hence we can use the existence guarantees for the subset gadget to derive a coloring for it. Combining this coloring with χ_{in} and χ_{out} gives a q-coloring χ.

It remains to show that this χ is a valid square q-coloring. It is easy to see that no color conflict arises with one the complement vertices of the subset gadget connecting X and Y. The only remaining possible conflicts are between vertices of X_i and Y_i for some i, since these are the only vertices that share colors. Consider some fixed i. Certainly, the vertices in Y_i share no common neighbours in S with vertices in X_i, since the vertices in X_i are connected to S_i while the vertices in Y_i are connected to $S \setminus S_i$. Hence they are not within distance two.

□

Finally, let us quickly mention why we chose r as $\lceil \frac{(1+\varepsilon)\log(m)}{2} \rceil$. As we mentioned above, we need $\binom{2r}{r} > 2m$. It is well-known that for any r, the inequality $\binom{2r}{r} \geq \frac{2^{2r}}{2r+1}$ holds, so we need only ensure that $\frac{2^{2r}}{2r+1} > 2m$. Substituting our choice $r = \lceil \frac{(1+\varepsilon)\log(m)}{2} \rceil$ into the inequality and bounding the rounding in both numerator and denominator, we get that is suffices to satisfy the inequality $\frac{m^{(1+\varepsilon)}}{(1+\varepsilon)\log(m)+3} > 2m$. By rewriting, it suffices to satisfy $m^{\varepsilon} > 2(1+\varepsilon)\log m + 6$. However, we chose m_ε such that any $m > m_\varepsilon$ fulfills this inequality.

Socket Gadget We now come to the series of subgadgets that are needed for the vector selection gadget. We start with the most low-level ones and use them to build more complex ones.

The first fundamental building block is the socket gadget. There will be two types of socket gadgets: constant sockets and switch gadgets. Both will have a control input which must receive either red or blue. Constant sockets will always expose a color from one color class I_i (for some i) to x, irregardless of the color of the control input (recall that exposing a color to x means that a neighbour of x is assigned that color). We call constant sockets that always expose a color from the coloring of I_1 constant sockets of type 1, and similarly constant sockets that always expose a color from the coloring of I_2 sockets of type 2. On the other hand, switch sockets will output either a color from I_1 or I_2 depending on the control input. We describe switch sockets by an abbreviation of their control-to-exposed-color mapping. Specifically, switch gadgets that show a color from I_1 for a red control input and a color from I_2 for a blue control input will be called switch sockets of type $r1b2$, while the other type where the outputs are switched will be called switch sockets of type $r2b1$.

We specify all the different types of socket gadgets in Table 4. We use the Kronecker delta δ_{ij}, defined as $\delta_{ij} := \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$.

Socket Gadget	
parameter(s)	$n \in \mathbb{Z}_{>0}$ and the type (constant / switch) and subtype (1 / 2 / r1b2 / r2b1) of the gadget
V_{in}	• Two groups of $2n^6$ vertices I_1 and I_2 (the color class inputs), • three "logic" input vertices r, g, b, and • a "control" input vertex s.
V_{out}	• Two groups of $2n^6$ vertices I'_1, I'_2, • three "logic" output vertices r', g', b', and • a "control" output vertex s'.
V_x	a single vertex z
C_{in}	• $\lvert\chi_{\text{in}}(I_1 \cup I_2 \cup \{r, g, b\})\rvert = \lvert I_1 \cup I_2 \cup \{r, g, b\}\rvert$, • $\chi_{\text{in}}(r) =$ red, $\chi_{\text{in}}(g) =$ green, $\chi_{\text{in}}(b) =$ blue, and • $\chi_{\text{in}}(s) \in \{\text{red, blue}\}$
C_R	We divide C_R into output and vector conditions: 1. • $\chi_{\text{out}}(r') = \chi_{\text{in}}(r)$, $\chi_{\text{out}}(g') = \chi_{\text{in}}(g)$, $\chi_{\text{out}}(b') = \chi_{\text{in}}(b)$, • $\chi_{\text{out}}(s') = \chi_{\text{in}}(s)$, as well as • $\forall i \in [6] : \chi_{\text{out}}(I'_i) = \chi_{\text{in}}(I_i)$. 2. **For constant sockets of type I ($I \in [2]$),** $v_x = (\delta_{I1}, \delta_{I2})$. **For switch sockets of type rIbJ ($I, J \in [2]$),** $v_x = \begin{cases} (\delta_{I1}, \delta_{I2}) & \text{if } \chi_{\text{in}}(s) = \text{red} \\ (\delta_{I2}, \delta_{I1}) & \text{if } \chi_{\text{in}}(s) = \text{blue} \end{cases}$

Table 4: The specification table of the socket gadget

The constant sockets are very easy to implement. They simply copy their inputs to the outputs via equality gadgets. Furthermore, one of the input color classes (the one from which the socket should expose a color to x) is connected to z, the vertex from V_x, via a $(2n^6, 1)$ subset gadget.

Now for the switch sockets, one of which is sketched in Figure 6. We describe an r2b1 switch socket. Switch sockets of type r1b2 are completely analogous, the two color class inputs are simply switched.

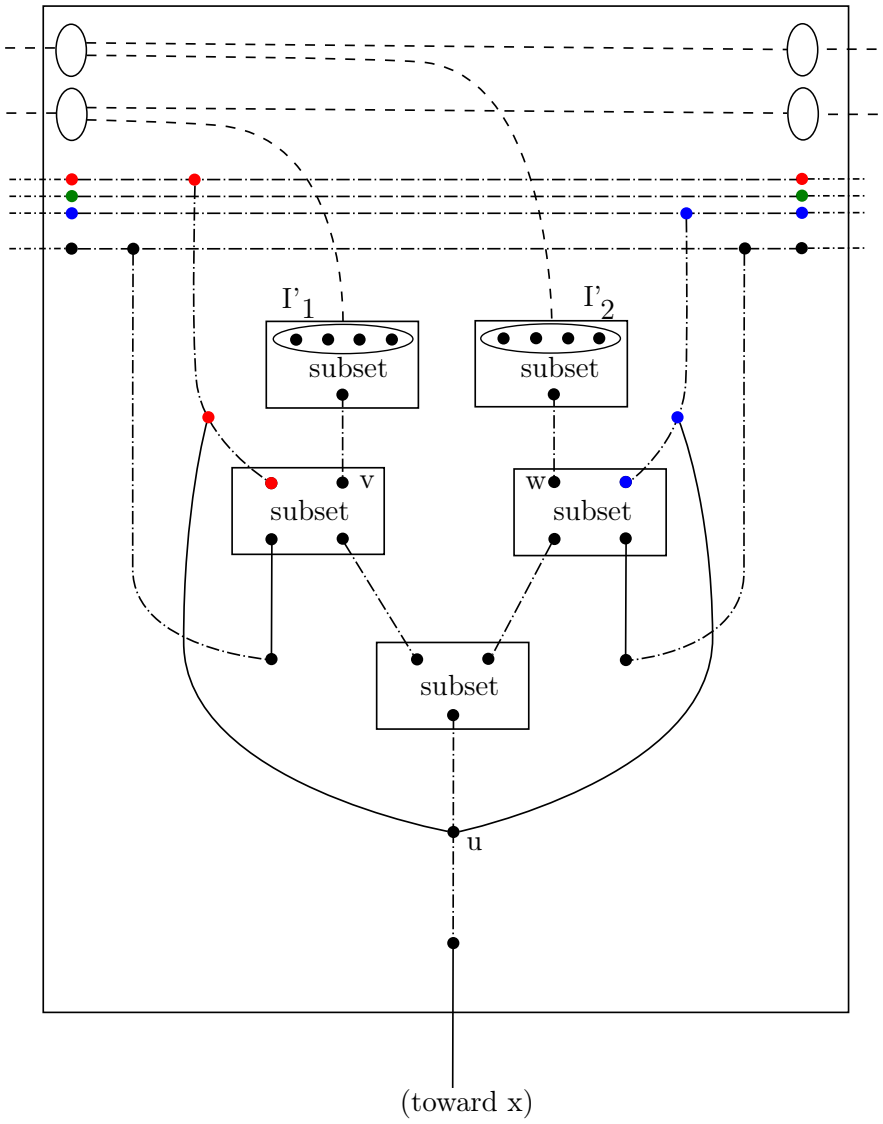


Figure 6: Illustration of a switch socket gadget. The lower color class input is I_1 in this figure, and hence this socket is of type $r2b1$.

First, we create a copy I_1' of the color class input I_1 and a copy I_2' of color class input I_2. Each is connected to the original color class input by an equality gadget. The vertices in the copy I_1' will serve as input to a subset gadget whose output is copied to a vertex v. The vertices in I_2' are the input to another subset gadget whose input is copied to a vertex w.

The vertex v, together with a red vertex, serves as input to a (2,2) equality gadget. We call this the left equality gadget. One of the outputs of the left equality gadget is connected by a normal edge to a vertex that copies the control input. The other is copied to a (2,1) subset gadget which we call the final subset gadget. On the other side, the vertex w, together with a blue vertex, serves as input to a (2,2) equality gadget. We call this the right equality gadget. One of the outputs of the right equality gadget is also connected to a vertex that copies the control input, while the other is copied to the second input of the final subset gadget.

The output of the final subset gadget is copied to a vertex u, on which we forbid the colors red and blue by connecting it to a red and a blue vertex. Finally, we connect u to z via an equality gadget.

Lemma 4.10. *The gadget constructed above satisfies the specification table of the socket gadget, as shown in Table 4.*

Proof. **Output Guarantees:** For constant socket gadgets, the output guarantees should be clear. We hence turn to switch socket gadgets, specifically to a switch socket gadget of type r2b1. Switch sockets of type r1b2 are analogous.

Let a square q-coloring χ be given such that $\chi|_{V_{\text{in}}} \vdash C_{\text{in}}$. It is easy to verify that χ must satisfy part 1 of C_R. Hence we focus on condition 2. If $\chi_{\text{in}}(s) =$ red, part 2 of C_R becomes $v_x = (0, 1)$ and if $\chi_{\text{in}}(s) =$ blue, it becomes $v_x = (1, 0)$. Hence if the control input is red, we must show that $\chi(z) \in \chi(I_2)$, and if it is blue, that $\chi(z) \in \chi(I_1)$.

Due to the output guarantees of the first two subset gadgets, v and w must get a color from $\chi(I_1)$ and $\chi(I_2)$, respectively. The left equality gadget gets as input v and a red vertex, and one of its outputs is within distance one of a vertex which copies the control input. Hence if the control input is red, that output cannot be red, so the other one must be. This other output is copied to the first input of the final subset gadget. Similarly, the right equality gadget gets w and a blue vertex as input, and one of its outputs is within distance one of a vertex which copies the control input.

Note that each of the equality gadgets has one output which is fed into the final subset gadget. Since the output of the final gadget is copied to u which cannot be red or blue, one of the inputs of the final subset gadget is required to be neither red nor blue. Recall that if the control input is red, the left equality gadget must copy red to the final subset gadget. For this red control input, the other equality gadget must therefore copy one of the colors from $\chi(I_2)$ to the final equality gadget. This must therefore be the output of the gadget. Similarly, if the control input is blue, the right equality gadget must copy blue to the final gadget. Hence the left equality gadget must copy one of the colors from $\chi(I_1)$ to the final equality gadget, and this must be the output of the gadget.

Hence $(\chi|_{V_{\text{in}}}, \chi|_{V_{\text{out}}}, p(\chi|_{V_x}))$ must also satisfy condition 2.

Existence Guarantees Let $(\chi_{\text{in}}, \chi_{\text{out}}, v_x) \in R$ and $S = (S_1, S_2)$ be given. We have $|S_1|+|S_2| = 1$.

We start with constant socket gadgets. Assume without loss of generality that the socket gadget is of type 1. Then we must have $S_1 =: \{\tau\} \subseteq \chi(I_1)$ and $S_2 = \emptyset$, and there must be a subset gadget connecting I_1 to z. First, we use the existence guarantees of the subset gadgets copying inputs to outputs to extend the input coloring to those subset gadgets. Then we extend the input coloring further by using the existence guarantees for the subset gadget connecting I_1 to z such that z is colored using τ. This obviously suffices.

Now for the switch socket gadgets. We describe the proof for sockets of type $r2b1$, sockets of type $r1b2$ are analogous. Furthermore, without loss of generality let $\chi_{\text{in}}(s) = \text{red}$, the case that it is blue is analogous. We must hence have $S_1 = \emptyset$ and $S_2 = \{\tau\} \subseteq \chi_{I_2}$. We will construct a coloring where τ is the output.

We start to extend χ_{in} by using the existence guarantees for any equality gadgets which already have a coloring on all their inputs, repeating this until no such equality gadget exists. We now have input colorings for the topmost pair of subset gadgets, which receive colors from $\chi_{\text{in}}(I_1)$ and $\chi_{\text{in}}(I_2)$, respectively. We then use the existence guarantees for those subset gadgets such that the left gadget outputs an arbitrary color σ from $\chi_{\text{in}}(I_1)$ and the right gadget outputs τ, and then use the existence guarantees for the equality gadgets copying those outputs to the left and right equality gadgets. These now have colored inputs – the left one gets red and the σ as input, while the right one gets blue and τ. On both gadgets, one of the outputs is already within distance one of a vertex with red coloring and is hence restricted in that it cannot also be colored red. In the right gadget, we must hence use the existence guarantees such that this output is colored using σ, while the unrestricted output is colored using red. In the right gadget, we use the existence guarantees such that the restricted output is colored using blue, while the unrestricted output is colored using τ. We can now use the existence guarantees for the equality gadgets copying the two unrestricted to the final subset gadget. That gadget now has an input coloring – one of its inputs being red, the other τ, and we can use its existence guarantees to find a coloring such that its output is colored using τ. We can then use the existence guarantees for equality gadgets two final times for the remaining two equality gadgets. Note, too, that the vertex above z – which is restricted to not use red or blue – is then also colored using τ, hence its restriction is satisfied.

Note finally that no color conflicts arise between inner vertices of different subset gadgets by Property 4.7. □

Edge Selection Gadget We now use the low-level socket gadget to build a slightly higher-level gadget, the edge selection gadgets. Intuitively, it uses subset gadgets and socket gadgets to simulate the selection of an edge between two groups of vertices. Its behaviour is specified in Table 5.

The construction we will use for this gadget is depicted in Figure 7. It will contain a long chain of $2n^2$ socket gadgets, which we denote as $G^{(1)}, \dots, G^{(2n^2)}$. Furthermore for each $j \in [3]$, we identify any of the elements (e.g. sets, vertices, conditions) of $G^{(j)}$ by superscripting them with (j). E.g. the input vertix set V_{in} of $G^{(1)}$ and the vertex z of $G^{(2)}$ become $V_{\text{in}}^{(1)}$ and $z^{(2)}$, respectively.

The wiring of the gadget is very simple: the inputs V_{in} of the edge selection gadget are copied via equality gadgets directly to the corresponding inputs in the set $V_{\text{in}}^{(1)}$. Then for each $j \in [2n^2 - 1]$, the outputs $V_{\text{out}}^{(j)}$ are then copied to the inputs $V_{\text{in}}^{(j+1)}$. The outputs of $G^{(2n^2)}$ are copied to the outputs V_{out} of the edge selection gadget.

We now describe the chain in more detail. The gadgets $G^{(1)}$ to $G^{(n_1)}$, where $n_1 = \min\{n^2, n^2+\alpha\}$, are constant sockets of type 1. After that, the gadgets $G^{(n_1+1)}$ to $G^{(n_1+n_2)}$, where $n_2 = \min\{n^2, n^2 - \alpha\}$, are constant sockets of type 2. Finally, the gadgets $G^{(n_1+n_2+1)}$ to $G^{(n_1+n_2+n_3)}$, where $n_3 = |\alpha|$, are switch sockets. If α is positive, they will be $r2b1$ switch sockets. If α is negative, they will be $r1b2$ switch sockets.

Finally, the vertices in V_x are not new vertices. Instead, we set $V_x = \bigcup_{j \in [2n^2]} V_x^{(j)}$. Recall that $V_x^{(j)}$ in a socket gadget is simply the set containing only the vertex $z^{(j)}$.

Lemma 4.11. *The gadget constructed above satisfies the specification table of the edge selection gadget, as shown in Table 5.*

Proof. First, let us note generally that the respective vector output $v_x^{(j)}$ of a socket gadget $G^{(j)}$ is

Edge Selection Gadget	
parameter(s)	integers $n \in \mathbb{Z}_{>0}$ and $\alpha \in [-n^2, n^2]$
V_{in}	• Two groups of $2n^6$ vertices I_1 and I_2 (the color class inputs), • three "logic" input vertices r, g, b, and • a "control" input vertex s
V_{out}	• Two groups of $2n^6$ vertices I_1', I_2', • three "logic" output vertices r', g', b', and • a "control" output vertex s'
V_x	a total of $2n^2$ vertices
C_{in}	• $\lvert\chi_{\text{in}}(I_1 \cup I_2 \cup \{r, g, b\})\rvert = \lvert I_1 \cup I_2 \cup \{r, g, b\}\rvert$ • $\chi_{\text{in}}(r) = \text{red}, \chi_{\text{in}}(g) = \text{green}, \chi_{\text{in}}(b) = \text{blue}$ • $\chi_{\text{in}}(s) \in \{\text{red, blue}\}$
C_R	We divide C_R into output and vector conditions: 1. • $\chi_{\text{out}}(r') = \chi_{\text{in}}(r)$, $\chi_{\text{out}}(g') = \chi_{\text{in}}(g)$ and $\chi_{\text{out}}(b') = \chi_{\text{in}}(b)$, • $\chi_{\text{out}}(s') = \chi_{\text{in}}(s)$, as well as • $\forall i \in [2] : \chi_{\text{out}}(I_i') = \chi_{\text{in}}(I_i)$. 2. • $v_x = (n^2, n^2)$ if $\chi_{\text{in}}(s) = \text{red}$, and • $v_x = (n^2 + \alpha, n^2 - \alpha)$ if $\chi_{\text{in}}(s) = \text{blue}$.

Table 5: The specification table of the edge selection gadget

determined by the input coloring on its respective input vertices $V_{\text{in}}^{(j)}$. Suppose all socket gadgets have the same input coloring χ_{in}. Since $V_x = \bigcup_{j \in [2n^2]} V_x^{(j)}$, the vector output of the edge selection gadget is simply the sum $\sum_{j=1}^{2n^2} v_x^{(j)}$, and is hence also determined. Let us calculate it explicitly. There are four cases to consider:

1. $\chi_{\text{in}}(s) = \text{red}$ and $\alpha > 0$
2. $\chi_{\text{in}}(s) = \text{red}$ and $\alpha < 0$
3. $\chi_{\text{in}}(s) = \text{blue}$ and $\alpha > 0$
4. $\chi_{\text{in}}(s) = \text{blue}$ and $\alpha < 0$

We have

$$n_1 = \min\{n^2, n^2 + \alpha\} = \begin{cases} n^2 & \text{in case 1 or 3} \\ n^2 + \alpha & \text{in case 2 or 4} \end{cases}$$

$$n_2 = \min\{n^2, n^2 - \alpha\} = \begin{cases} n^2 & \text{in case 2 or 4} \\ n^2 - \alpha & \text{in case 1 or 3} \end{cases}$$

$$n_3 = |\alpha| = \begin{cases} \alpha & \text{in case 1 or 3} \\ -\alpha & \text{in case 2 or 4} \end{cases}$$

Certainly, we have $v_x^{(1)} = \ldots = v_x^{(n_1)} = (1, 0)$ and $v_x^{(n_1+1)} = \ldots = v_x^{(n_1+n_2)} = (0, 1)$ in all cases. For the last n_3 gadgets, we have that

$$v_x^{(n_1+n_2+1)} = \ldots = v_x^{(n_1+n_2+n_3)} = \begin{cases} (1, 0) & \text{in case 2 or 3} \\ (0, 1) & \text{in case 1 or 4} \end{cases}$$

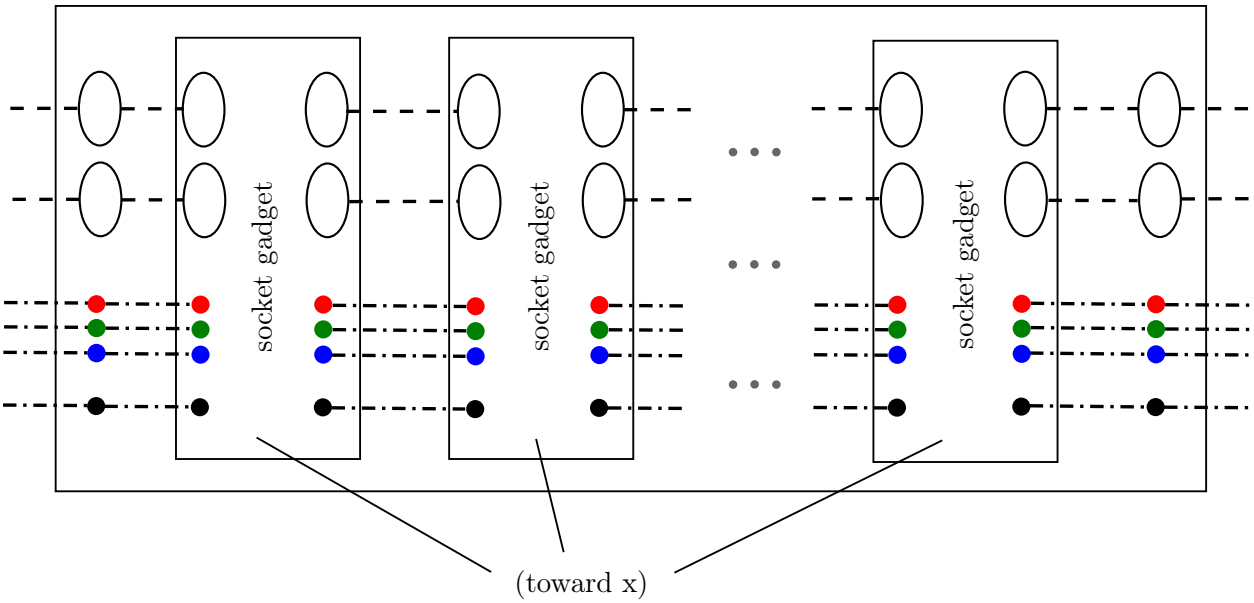


Figure 7: Illustration of the edge selection gadget.

Hence, the vector output of the edge selection gadget is given by

$$\sum_{j=1}^{2n^2} v_x^{(j)} = \begin{cases} n_1(1,0) + n_2(0,1) + n_3(0,1) = (n^2, n^2 - \alpha + \alpha) = (n^2, n^2) & \text{in case 1} \\ n_1(1,0) + n_2(0,1) + n_3(1,0) = (n^2 + \alpha - \alpha, n^2) = (n^2, n^2) & \text{in case 2} \\ n_1(1,0) + n_2(0,1) + n_3(1,0) = (n^2 + \alpha, n^2 - \alpha) & \text{in case 3} \\ n_1(1,0) + n_2(0,1) + n_3(0,1) = (n^2 + \alpha, n^2 - \alpha) & \text{in case 4} \end{cases}$$
$$= \begin{cases} (n^2, n^2) & \text{if } \chi_{\text{in}}(s) = \text{red} \\ (n^2 + \alpha, n^2 - \alpha) & \text{if } \chi_{\text{in}}(s) = \text{blue} \end{cases}$$

Output Guarantees: Let a square q-coloring χ be given such that $\chi|_{V_{\text{in}}} \vdash C_{\text{in}}$. If $\chi(s) = \text{red}$, part 2 of C_R becomes $v_x = (n^2, n^2)$, and if $\chi(s) = \text{blue}$, it becomes $v_x = (n^2 + \alpha, n^2 - \alpha)$.

First, note that the output guarantees of the equality gadgets as well as those of the socket gadgets mean that the input coloring $\chi|_{V_{\text{in}}}$ is copied to the corresponding inputs of each of the socket gadgets and finally to the corresponding outputs in V_{out}. Hence χ satisfies the first condition of the output guarantee of the edge selection gadget.

For the second part, by the fact that each of the socket gadgets receives the same input coloring mirrored from $\chi|_{V_{\text{in}}}$, the calculation above with $\chi_{\text{in}} = \chi|_{V_{\text{in}}}$ shows that $p(\chi|_{V_x}) = \sum_{j=1}^{2n^2} v_x^{(j)} = v_x$. Hence the second part of C_R is also satisfied, meaning $(\chi|_{V_{\text{in}}}, \chi|_{V_{\text{out}}}, p(\chi|_{V_x})) \vdash C_R$.

Existence Guarantees: Let $(\chi_{\text{in}}, \chi_{\text{out}}, v_x) \in R$ and $S = (S_1, S_2)$ with $S_1 \subseteq \chi_{\text{in}}(I_1), S_2 \subseteq \chi_{\text{in}}(I_2)$ and $(|S_1|, |S_2|) = v_x$ be given. We will construct the square q-coloring χ by extending χ_{in} and χ_{out}. First, for each of the socket gadgets $G^{(j)}$, we color both their inputs $V_{\text{in}}^{(j)}$ and their outputs $V_{\text{out}}^{(j)}$ using the corresponding colors of χ_{in}.

We can then obviously find colorings for the equality gadgets connecting these inputs and outputs using the existence guarantees for the equality gadgets.

Recall that the vector output $v_x^{(j)} \in \{(1,0), (0,1)\}$ of each socket gadget $G^{(j)}$ is determined by the coloring of its input, and we've already colored their inputs. Hence collect in I all $j \in [2n^2]$ such that $v_x^{(j)} = (1,0)$ and in J all $j \in [2n^2]$ such that $v_x^{(j)} = (0,1)$. Since we have colored all of the

socket gadget inputs using the same coloring, we can now use the calculation above to show that $(|I|, |J|) = v_x$.

Note that we also have $v_x = (|S_1|, |S_2|)$. Hence, we can do the following: For each $j \in I$, define a set $S_1^{(j)} = \tau$ for a unique $\tau \in S_1$. Furthermore, for each $j \in J$, define a set $S_2^{(j)} = \tau$ for a unique $\tau \in S_2$. It must be true that $\bigcup_{j \in I} S_1^{(j)} = S_1$ and $\bigcup_{j \in I} S_2^{(j)} = S_2$.

We can now, for each $j \in I$, use the existence guarantees of $G^{(j)}$ for $S = (S_1^{(j)}, \emptyset)$ and, for each $j \in J$, use the existence guarantees of $G^{(j)}$ for $S = (\emptyset, S_2^{(j)})$. We use the colorings obtained to complete χ.

Note finally that no conflict can arise between inner vertices of different subgadgets by Property 4.7. □

Vector State Gadget Now for the next-higher level of gadget, the vector state gadget. It simply combines three vector state gadgets that act on different color classes, but are controlled by the same control input. It is described by Table 6.

Vector State Gadget	
parameter(s)	an integer $n \in \mathbb{Z}_{>0}$ and a vector $y = (y_1, y_2, y_3) \in [-n^2, n^2]^3$
V_{in}	• Six groups of $2n^6$ vertices $I_1, I_2, I_3, I_4, I_5, I_6$ (the color class inputs), • three "logic" input vertices r, g, b, • and a "control" input vertex s.
V_{out}	• Six groups of $2n^6$ vertices $I'_1, I'_2, I'_3, I'_4, I'_5, I'_6$, • three "logic" output vertices r', g', b', and • a "control" output vertex s'.
V_x	a total of $6n^2$ vertices
C_{in}	• $\lvert\chi_{\text{in}}(I_1 \cup I_2 \cup I_3 \cup I_4 \cup I_5 \cup I_6 \cup \{r, g, b\})\rvert =$ $\lvert I_1 \cup I_2 \cup I_3 \cup I_4 \cup I_5 \cup I_6 \cup \{r, g, b\}\rvert$ • $\chi_{\text{in}}(r) = \text{red}, \chi_{\text{in}}(g) = \text{green}, \chi_{\text{in}}(b) = \text{blue}$ • $\chi_{\text{in}}(s) \in \{\text{red, blue}\}$
C_R	We divide C_R into output and vector conditions: 1. • $\chi_{\text{out}}(r') = \chi_{\text{in}}(r)$, $\chi_{\text{out}}(g') = \chi_{\text{in}}(g)$, $\chi_{\text{out}}(b') = \chi_{\text{in}}(b)$ • $\chi_{\text{out}}(s') = \chi_{\text{in}}(s)$ • $\forall i \in [6] : \chi_{\text{out}}(I'_i) = \chi_{\text{in}}(I_i)$ 2. • $v_x = (n^2, n^2, n^2, n^2, n^2, n^2)$ if $\chi_{\text{in}}(s) = \text{red}$ • $v_x = (n^2 + y_1, n^2 - y_1, n^2 + y_2, n^2 - y_2, n^2 + y_3, n^2 - y_3)$ if $\chi_{\text{in}}(s) = \text{blue}$

Table 6: The specification table of the vector state gadget

Our construction of the vector state gadget is depicted in Figure 8. Let us describe the construction in detail. As was said, the gadget consists of three edge selection gadgets, one for each pair of color classes. Hence we create three edge selection gadgets $G^{(1)}, G^{(2)}, G^{(3)}$. The parameter of $G^{(j)}$ is $\alpha_j := y_j$ for each $j \in [3]$. Furthermore for each $j \in [3]$, we identify any of the elements (e.g. sets, vertices, conditions) of $G^{(j)}$ by superscripting them with (j).

The overall wiring of the vector state gadget is simple. Via equality gadgets, the gadget $G^{(j)}$ ($j \in [3]$) will receive the color class input I_{2j-1} as its input I_1 and the color class input I_{2j} as its input I_2. Similarly, equality gadgets copy the output I'_1 of $G^{(j)}$ to the output I'_{2j-1} and the output

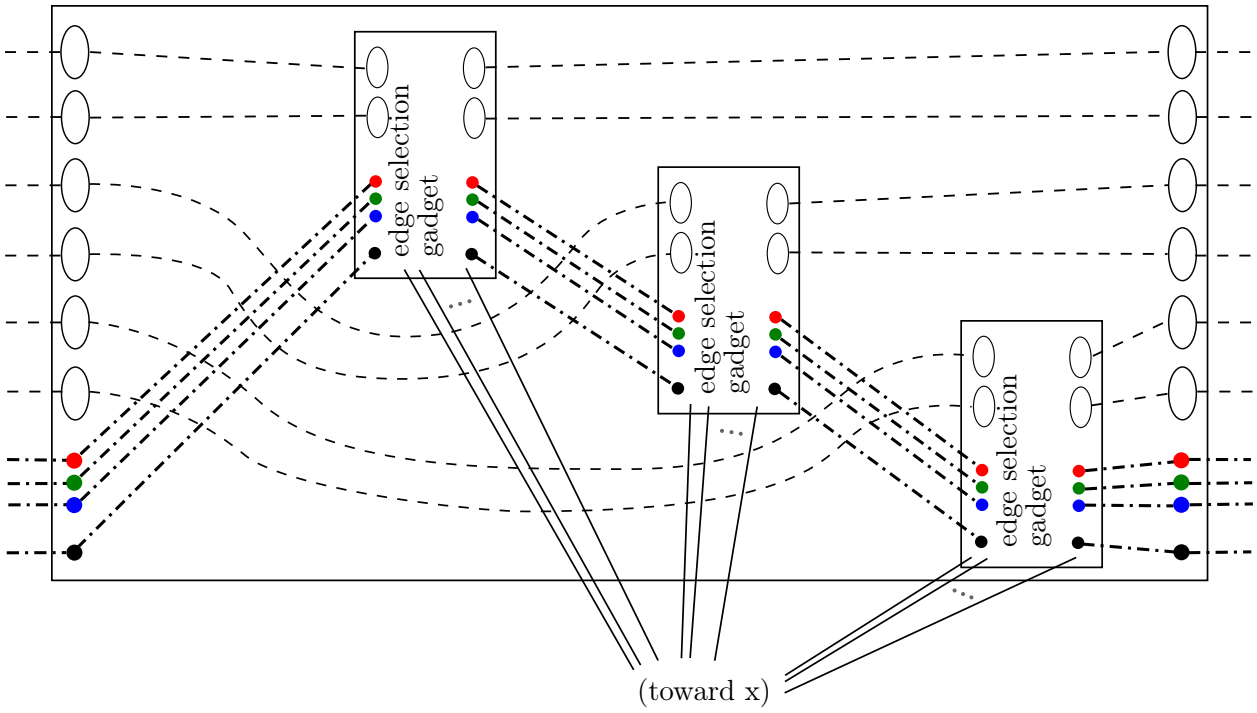


Figure 8: Illustration of a vector state gadget.

I'_2 to the output I'_{2j}. The three logic colors and the control input are copied from the vector state gadget inputs to the inputs of $G^{(1)}$, then from the outputs of $G^{(1)}$ to the inputs of $G^{(2)}$, from the outputs of $G^{(2)}$ to the inputs of $G^{(3)}$, and from the outputs of $G^{(3)}$ to the outputs of the vector state gadget.

Finally, the vertices in V_x are not new vertices. Instead, we set $V_x = \bigcup_{j\in[2n^2]} V_x^{(j)}$.

Lemma 4.12. *The gadget constructed above satisfies the specification table of the vector state gadget, as shown in Table 6.*

Proof. First, let us note generally that the respective vector output $v_x^{(j)}$ of an edge selection gadget $G^{(j)}$ is determined by the input coloring on its respective input vertices $V_{\text{in}}^{(j)}$.

Let an input coloring $\chi_{\text{in}} : V_{\text{in}} \to [q]$ be given. Suppose each of the inputs $V_{\text{in}}^{(j)}, j \in [3]$ is colored using a coloring $\chi_{\text{in}}^{(j)}$ such that $\chi_{\text{in}}^{(j)}(r^{(j)}) = \text{red}, \chi_{\text{in}}^{(j)}(b^{(j)}) = \text{blue}, \chi_{\text{in}}^{(j)}(g^{(j)}) = \text{green}, \chi_{\text{in}}^{(j)}(s^{(j)}) = \chi_{\text{in}}(s), \chi_{\text{in}}^{(j)}(I_1^{(j)}) = \chi_{\text{in}}(I_{2j-1})$ and $\chi_{\text{in}}^{(j)}(I_2^{(j)}) = \chi_{\text{in}}(I_{2j})$. Since $V_x = \bigcup_{j\in[3]} V_x^{(j)}$, the vector output of the vector state gadget is $\sum_{j=1}^{3} v_x^{(j)}$. Hence, we have that the vector output is simply $\sum_{j=1}^{3} v_x^{(j)} = (n^2, n^2, 0, 0, 0, 0) + (0, 0, n^2, n^2, 0, 0) + (0, 0, 0, 0, n^2, n^2) = (n^2, n^2, n^2, n^2, n^2, n^2)$ if $\chi_{\text{in}}(s) = \text{red}$, or $\sum_{j=1}^{3} v_x^{(j)} = (n^2 + y_1, n^2 - y_1, 0, 0, 0, 0) + (0, 0, n^2 + y_2, n^2 - y_2, 0, 0) + (0, 0, 0, 0, n^2 + y_3, n^2 - y_3) = (n^2 + y_1, n^2 - y_1, n^2 + y_2, n^2 - y_2, n^2 + y_3, n^2 - y_3)$ if $\chi_{\text{in}}(s) = \text{blue}$.

Output Guarantees: Let a square q-coloring χ be given such that $\chi|_{V_{\text{in}}} \vdash C_{\text{in}}$. If $\chi(s) = \text{red}$, part 2 of C_R becomes $v_x = (n^2, n^2, n^2, n^2, n^2, n^2)$, and if $\chi(s) = \text{blue}$, it becomes $v_x = (n^2 + y_1, n^2 - y_1, n^2 + y_2, n^2 - y_2, n^2 + y_3, n^2 - y_3)$.

First of all, note that the vertices of V_{out} are connected to the corresponding vertices of V_{in} by a chain of equality gadgets or by gadgets that copy their inputs to their outputs. Hence χ is forced to color the output vertices such that they mirror the input colors, satisfying part 1 of C_R.

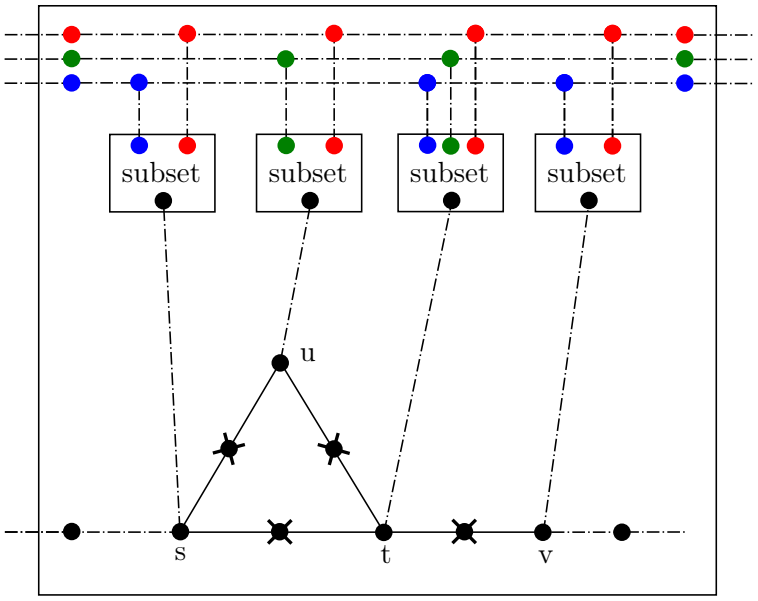


Figure 9: Illustration of a one-way switch gadget. Again, dash-dotted edges represent (1,1) equality gadgets. Crossed out vertices are colorless vertices.

Furthermore, the coloring χ mirrors the input $\chi|_{V_{\text{in}}}$ to all edge selection gadget inputs (either by subset gadgets or by the previous edge selection gadgets mirroring the control input to the control output) such that it fulfills the conditions for the calculation above. By this calculation, $p(\chi|_{V_x}) = \sum_{j=1}^{3} v_x^{(j)} = v_x$. Hence part 2 of C_R is also satisfied, meaning $(\chi|_{V_{\text{in}}}, \chi|_{V_{\text{out}}}, p(\chi|_{V_x})) \vdash C_R$.

Existence Guarantees: Let $(\chi_{\text{in}}, \chi_{\text{out}}, v_x) \in R$ and $S = (S_1, S_2, S_3, S_4, S_5, S_6)$ with $S_1 \subseteq \chi_{\text{in}}(I_1), S_2 \subseteq \chi_{\text{in}}(I_2), \ldots, S_6 \subseteq \chi_{\text{in}}(I_6)$ with $(|S_1|, |S_2|, |S_3|, |S_4|, |S_5|, |S_6|) = v_x$ be given. We will construct the square q-coloring χ by extending χ_{in} and χ_{out}.

For all $j \in [3]$, we color the inputs and outputs $V_{\text{in}}^{(j)}, V_{\text{out}}^{(j)}$ of $G^{(j)}$ in the obvious way. We can now use the existence guarantees of each of the equality gadgets contained in the vector state gadget to obtain a coloring for them, and we incorporate these colorings into χ. All that remains is to find a coloring for each of the edge selection gadgets. To do this, we use the existence guarantees of the gadget $G^{(j)}$ for $j \in [3]$ with $S^{(j)} = (S_{2j-1}, S_{2j})$ and use these colorings to complete χ. Hence, we have $\chi(V_x) = \chi(\bigcup_{j\in[3]} V_x^{(j)}) = \bigcup_{j\in[3]} \chi(V_x^{(j)}) = \bigcup_{k\in[6]} S_k$. Obviously, we thus have $p(\chi|_{V_x}) = v_x$ and $\forall k \in [6] : S_k \subseteq \chi(V_x)$.

Note finally that no conflict can arise between inner vertices of different subgadgets due to Property 4.7.

□

One-Way Switch Gadget We now build another low-level gadget which will we will later combine with vector state gadgets to build the vector selection gadget. The one-way switch gadget is shown in Figure 9. It receives and outputs logic colors and a control color and allows the control color to either stay constant or to switch from blue to red, but not the other way around.

The gadget is specified in Table 7.

The construction for the one-way switch gadget is as follows. We create four new vertices s, t, u and v. We restrict s, t, u, v to only use subsets of the logic colors by connecting via an equality gadget to the output of subset gadgets copying colors from the logic color bus. Specifically, s and

One-Way Switch Gadget	
parameter(s)	none
V_{in}	• Three "logic" input vertices r, g, b, and • a "control" input vertex s.
V_{out}	• three "logic" output vertices r', g', b', and • a "control" output vertex s'
V_x	$\emptyset$
C_{in}	• $\chi_{\text{in}}(r) = \text{red}, \chi_{\text{in}}(g) = \text{green}, \chi_{\text{in}}(b) = \text{blue}$ • $\chi_{\text{in}}(s) \in \{\text{red, blue}\}$
C_R	We group the conditions of C_R into two parts: 1. $\chi_{\text{out}}(r') = \chi_{\text{in}}(r)$, $\chi_{\text{out}}(g') = \chi_{\text{in}}(g)$ and $\chi_{\text{out}}(b') = \chi_{\text{in}}(b)$ 2. The control output $\chi_{\text{out}}(s')$ must satisfy • $\chi_{\text{out}}(s') = \text{red}$ if $\chi_{\text{in}}(s) = \text{red}$, and • $\chi_{\text{out}}(s') \in \{\text{red, blue}\}$ if $\chi_{\text{in}}(s) = \text{blue}$.

Table 7: The specification table of the one-way-switch gadget

v will be restricted to red and blue, u will be restricted to green and red and t will be restricted to red, blue or green.

The vertex s will copy the control input vertex via an equality gadget, while the control output vertex copies v via an equality gadget.

We now connect these vertices via subdivided edges (i.e. edges that have a colorless vertex in the middle) as follows: s, t and v form a triangle, and t is connected to v. This concludes the construction.

Let us argue why this behaves in the desired way, i.e. that it satisfies the following lemma:

Lemma 4.13. *The gadget constructed above satisfies the specification table of the one-way switch gadget, as shown in Table 7.*

Proof. **Output Guarantees:** Let a square q-coloring χ be given such that $\chi|_{V_{\text{in}}} \vdash C_{\text{in}}$.

Part 1 of C_R is obviously satisfied due to the equality gadgets.

Now suppose $\chi(s) = \text{red}$. Since s is within distance two of t and t is restricted to red or blue, t must be blue. Since t is distance-2-adjacent to v, v (and hence the output) must then be red, which is what we wanted. Now suppose the input is blue. Then the fact that v is restricted to red or blue already guarantees that the output cannot have any other color. Hence part 2 of C_R is also satisfied.

Existence Guarantees: Let $(\chi_{\text{in}}, \chi_{\text{out}}, v_x) \in R$ and $S = ()$ be given. We construct the square q-coloring χ by extending χ_{in} and χ_{out}. Suppose $\chi_{\text{in}}(s) = \text{red}$. Then we extend χ_{in} by coloring u green, t blue and v red. We then use the existence guarantees of the subset and equality gadgets to find colorings for them. It is obvious that this is possible.

Now suppose $\chi_{\text{in}}(s) = \text{blue}$. We extend χ_{in} by coloring u red, t green and v with $\chi_{\text{out}}(s')$. We then use the existence guarantees of the subset and equality gadgets to find colorings for them. Again, it is obvious that this is possible.

As always, note that no conflict can arise between inner vertices of different subgadgets due to Property 4.7.

□

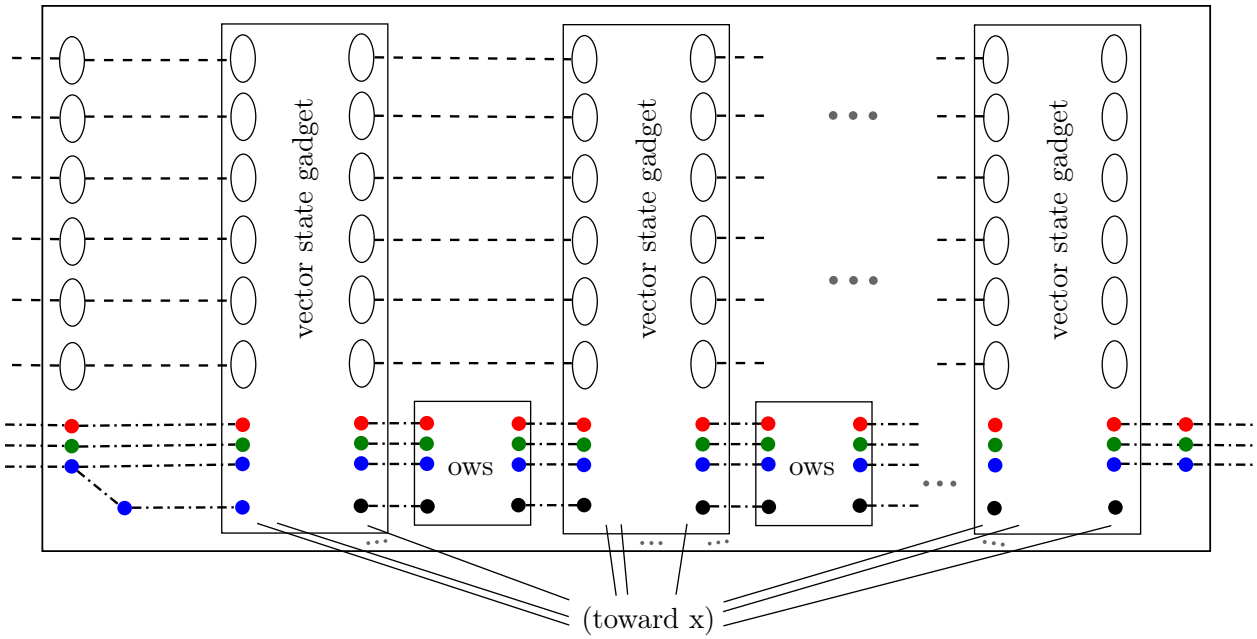


Figure 10: Illustration of the vector selection gadget. Dashed edges represent equality gadgets with more than one input, while dash-dotted edges represent (1,1) equality gadgets.

Vector Selection Gadget Finally, we can construct the actual vector selection gadget. Recall the definition of a vector-generated output and the specification of the vector selection gadget in Table 3.

Please see Figure 10 for a sketch of the gadget. The overall structure of the vector selection gadget is a long chain of vector state gadgets, with one-way switch gadgets in-between. Specifically, we create n^4 vector state gadgets $G^{(\text{vst},1)}, ..., G^{(\text{vst},n^4)}$ (with parameters that we will specify later) and $n^4 - 1$ one-way switch gadgets $G^{(\text{ows},1)}, ..., G^{(\text{ows},n^4-1)}$. For each j, we identify any of the elements (e.g. sets, vertices, conditions) of $G^{(\text{vst},j)}$ or $G^{(\text{ows},j)}$ by superscripting them with (vst, j) or (ows, j), respectively. E.g. the input vertex set V_{in} of $G^{(\text{vst},1)}$ and the vertex b' of $G^{(\text{ows},2)}$ become $V_{\text{in}}^{(\text{vst},1)}$ and $(b')^{(\text{ows},2)}$, respectively.

We create one additional vertex $\widetilde{b}$. Now for the wiring of the gadget. In the following, when we say we "connect" two vertices or two color class inputs/outputs, we mean create an equality gadget where one of the vertices or vertex groups is the input, and the other is the output.

We connect $\widetilde{b}$ separately to the input vertex b and to $s^{(\text{vst},1)}$. For all $j \in [6]$, we connect I_j to $I_j^{(\text{vst},1)}$. We connect r, g, b to $r^{(\text{vst},1)}, g^{(\text{vst},1)}, b^{(\text{vst},1)}$, respectively. Then, for each $i \in [n^4 - 1]$, we connect $I_j^{(\text{vst},i)}$ to $I_j^{(\text{vst},i+1)}$ for all $j \in [6]$. Furthermore we connect $(r')^{(\text{vst},i)}, (g')^{(\text{vst},i)}, (b')^{(\text{vst},i)}, (s')^{(\text{vst},i)}$ to $r^{(\text{ows},i)}, g^{(\text{ows},i)}, b^{(\text{ows},i)}, s^{(\text{ows},i)}$, respectively, and also $(r')^{(\text{ows},i)}, (g')^{(\text{ows},i)}, (b')^{(\text{ows},i)}, (s')^{(\text{ows},i)}$ to $r^{(\text{vst},i+1)}, g^{(\text{vst},i+1)}, b^{(\text{vst},i+1)}, s^{(\text{vst},i+1)}$, respectively. Finally, we connect $(r')^{(\text{vst},n^4)}, (g')^{(\text{vst},n^4)}, (b')^{(\text{vst},n^4)}$ to the logic color outputs r', g', b', respectively. The chain state output $(s')^{(\text{vst},n^4)}$ and the color class outputs $I_j^{(\text{vst},n^4)}$ for all $j \in [6]$ are left unconnected.

Note that the output guarantees of the one-way switch and vector state gadgets guarantee that each one-way switch gadget outputs either red or blue on its control output, and that the vector state gadgets output the same color on their control output as they receive on their control input. For any coloring χ of the gadget, let us call $\chi(s^{(\text{vst},p)})$ (for $p \in [n^4]$) the *chain state at position p*. Due to the behaviour of the one-way switch gadgets, it must be true that for any coloring of the

gadget, there exists some $t \in [n^4]$ such that for all $t' \leq t$, the chain state at position t' is blue, while for all $t' > t$, the chain state at position t' is red.

Indeed, there are n^4 possible positions t for the chain state switch from blue to red to occur in the chain. These represent the n^4 possible choices for the choice of $a_i \in A$, i.e. the position of this switch from blue to red will indicate what vector we selected from A. More specifically, if the chain state switch happens at position t, we want the output of the vector selection gadget to be $(n^6 + \mathrm{NZ}(a_t)_1, n^6 - \mathrm{NZ}(a_t)_1, n^6 + \mathrm{NZ}(a_t)_2, n^6 - \mathrm{NZ}(a_t)_2, n^6 + \mathrm{NZ}(a_t)_3, n^6 - \mathrm{NZ}(a_t)_3)$, i.e. we want it to be the vector-generated output generated by a_t.

To complete the construction, we must still specify what the parameters of the vector state gadgets are. First, for all $i \in [3]$, define

$$R_i = \begin{cases} [1, n^2] & \text{if } z_i \in D^+ \\ [-n^2, -1] & \text{if } z_i \in D^- \end{cases}$$

Obviously we have that $\forall j \in [n^4] : \mathrm{NZ}(a_j) \in R_1 \times R_2 \times R_3$. We now define the parameter y_j for the vector state gadget in the jth chain link gadget, for all $j \in [n^4]$. We do this inductively by setting $y_1 := \mathrm{NZ}(a_1)$ and $\forall j \in [n^4] \setminus \{1\} : y_j := \mathrm{NZ}(a_j) - \mathrm{NZ}(a_{j-1})$. Note that the parameter of the vector state gadget must fulfill $y_j \in [-n^2, n^2]^3$. However, it is easy to see that our choice of y_j adheres to this: Simply note that for all $i \in [3]$, we have $\mathrm{NZ}(a_j)_i, \mathrm{NZ}(a_{j-1})_i \in R_i$ and hence $\mathrm{NZ}(a_j)_i - \mathrm{NZ}(a_{j-1})_i \in [\min R_i - \max R_i, \max R_i - \min R_i] = [-n^2 + 1, n^2 - 1]$.

The reason for this choice of parameters is that now, $\forall j \in [n^4] : \sum_{k=1}^{j} y_k = \mathrm{NZ}(a_j)$. This will be a crucial part of our correctness proof.

Lemma 4.14. *The gadget constructed above satisfies the specification table of the vector selection gadget, as shown in Table 3.*

Proof. Define $V_{\mathrm{in}}^{(\mathrm{vst})} = \bigcup_{j=1}^{n^4} V_{\mathrm{in}}^{(\mathrm{vst},j)}$. We say that a coloring $\chi_{\mathrm{in}}^{\mathrm{vst}} : V_{\mathrm{in}}^{(\mathrm{vst})} \to [q]$ is *compliant* if it fulfills the following conditions:

1. $\forall j \in [n^4] : \chi_{\mathrm{in}}^{(\mathrm{vst})}(r^{(\mathrm{vst},j)}) = \mathrm{red}$, $\chi_{\mathrm{in}}^{(\mathrm{vst})}(b^{(\mathrm{vst},j)}) = \mathrm{blue}$, $\chi_{\mathrm{in}}^{(\mathrm{vst})}(g^{(\mathrm{vst},j)}) = \mathrm{green}$
2. $\forall j, j' \in [n^4] : \forall i \in [6] : \chi_{\mathrm{in}}^{(\mathrm{vst})}(I_i^{(\mathrm{vst},j)}) = \chi_{\mathrm{in}}^{(\mathrm{vst})}(I_i^{(\mathrm{vst},j')})$
3. There exists a $t \in [n^4]$ such that $\forall t' \leq t : \chi_{\mathrm{in}}^{(\mathrm{vst})}(s^{(\mathrm{vst},t')}) = \mathrm{blue}$ and $\forall t' > t : \chi_{\mathrm{in}}^{(\mathrm{vst})}(s^{(\mathrm{vst},t')}) = \mathrm{red}$.

Furthermore, we say that it *inherits* a coloring $\chi_{\mathrm{in}} : V_{\mathrm{in}} \to [q]$ with $\chi_{\mathrm{in}} \vdash C_{\mathrm{in}}$ if $\forall j \in [n^4] : \forall i \in [6] : \chi_{\mathrm{in}}^{(\mathrm{vst})}(I_i^{(\mathrm{vst},j)}) = \chi_{\mathrm{in}}(I_i)$.

Now, let us note generally that in the relation R of the vector state gadget, the third entry is uniquely determined by the first, i.e. the input coloring uniquely determines the vector output of the gadget. Hence, if we have a compliant coloring for $V_{\mathrm{in}}^{(\mathrm{vst})}$, we can uniquely determine the corresponding vector output $v_x^{(\mathrm{vst},j)}$ of the jth vector state gadget, for all $j \in [n^4]$. Since $V_x = \bigcup_{j \in [n^4]} V_x^{(\mathrm{vst},j)}$, the vector output of the vector selection gadget is then $\sum_{j=1}^{n^4} v_x^{(\mathrm{vst},j)}$.

More explicitly, let a compliant coloring with chain state switch at position t be given. Then the vector output v_x of the vector selection gadget is given by

$$v_x = \sum_{j=1}^{n^4} v_x^{(\mathrm{vst},j)}$$

$$
\begin{aligned}
&= \sum_{j=1}^{t} v_x^{(\mathrm{vst},j)} + \sum_{j=t+1}^{n^4} v_x^{(\mathrm{vst},j)} \\
&= \sum_{j=1}^{t} (n^2 + y_1^{(\mathrm{vst},j)}, n^2 - y_1^{(\mathrm{vst},j)}, n^2 + y_2^{(\mathrm{vst},j)}, n^2 - y_2^{(\mathrm{vst},j)}, n^2 + y_3^{(\mathrm{vst},j)}, n^2 - y_3^{(\mathrm{vst},j)}) \\
&\quad + \sum_{j=t+1}^{n^4} (n^2, n^2, n^2, n^2, n^2, n^2) \\
&= (\sum_{j=1}^{t} y_1^{(\mathrm{vst},j)}, -\sum_{j=1}^{t} y_1^{(\mathrm{vst},j)}, \sum_{j=1}^{t} y_2^{(\mathrm{vst},j)}, -\sum_{j=1}^{t} y_2^{(\mathrm{vst},j)}, \sum_{j=1}^{t} y_3^{(\mathrm{vst},j)}, -\sum_{j=1}^{t} y_3^{(\mathrm{vst},j)}) \\
&\quad + t(n^2, n^2, n^2, n^2, n^2, n^2) + (n^4 - t)(n^2, n^2, n^2, n^2, n^2, n^2) \\
&= (\mathrm{NZ}(a_t)_1, -\mathrm{NZ}(a_t)_1, \mathrm{NZ}(a_t)_2, -\mathrm{NZ}(a_t)_2, \mathrm{NZ}(a_t)_3, -\mathrm{NZ}(a_t)_3) + (n^6, n^6, n^6, n^6, n^6, n^6) \\
&= \big(n^6 + \mathrm{NZ}(a_t)_1, n^6 - \mathrm{NZ}(a_t)_1, n^6 + \mathrm{NZ}(a_t)_2, n^6 - \mathrm{NZ}(a_t)_2, n^6 + \mathrm{NZ}(a_t)_3, n^6 - \mathrm{NZ}(a_t)_3\big)
\end{aligned}
$$

Hence v_x is vector-generated. More specifically, it is generated by a_t.

Output Guarantees: Let a square q-coloring χ be given such that $\chi|_{V_{\mathrm{in}}} \vdash C_{\mathrm{in}}$.

Part 1 of C_R is obviously satisfied due to the equality gadgets.

For part 2, notice that the coloring of the logic color inputs and color class inputs, i.e. $\chi(V_{\mathrm{in}} \setminus \{s\})$, is propagated to the corresponding inputs of every vector state gadget, i.e. $V_{\mathrm{in}}^{(\mathrm{vst},j)} \setminus \{s^{(\mathrm{vst},j)}\}$. This happens either via equality gadgets or via other subgadgets' constraints guaranteeing that they copy inputs to outputs. Hence the first two conditions for $\chi|_{V_{\mathrm{in}}^{(\mathrm{vst})}}$ being a compliant coloring are already satisfied. Due to the the way one-way switch gadgets, it is also not hard to see that condition 3 must be satisfied. Let the chain state switch be at position t. Then can use the calculation above to show $p(\chi|_{V_x}) = \big(n^6 + \mathrm{NZ}(a_t)_1, n^6 - \mathrm{NZ}(a_t)_1, n^6 + \mathrm{NZ}(a_t)_2, n^6 - \mathrm{NZ}(a_t)_2, n^6 + \mathrm{NZ}(a_t)_3, n^6 - \mathrm{NZ}(a_t)_3\big)$. It also immediately shows that $p(\chi|_{V_x})$ is vector-generated, which proves part 2 of C_R. Hence we have that $(\chi|_{V_{\mathrm{in}}}, \chi|_{V_{\mathrm{out}}}, p(\chi|_{V_x})) \vdash C_R$.

Existence Guarantees: Let $(\chi_{\mathrm{in}}, \chi_{\mathrm{out}}, v_x) \in R$ and $S = (S_1, S_2, S_3, S_4, S_5, S_6)$ with $S_1 \subseteq \chi_{\mathrm{in}}(I_1), S_2 \subseteq \chi_{\mathrm{in}}(I_2), \ldots, S_6 \subseteq \chi_{\mathrm{in}}(I_6)$ with $(|S_1|, |S_2|, |S_3|, |S_4|, |S_5|, |S_6|) = v_x$ be given. Let v_x be generated by a_t, for some $t \in [n^4]$. We will construct the square q-coloring χ by extending χ_{in} and χ_{out}.

There is a unique coloring $\chi_{\mathrm{in}}^{(\mathrm{vst})}$ that is compliant, that has the chain state switch at position t and that inherits χ_{in}. We incorporate it into χ. We can then use the existence guarantees of all subgadgets except the vector state gadgets to find colorings for them that agree with the χ constructed so far. It is easy to see that this is always possible.

Now for the vector state gadgets. Let us denote by $v_x^{(\mathrm{vst},j)}$ the vector output of $G^{(\mathrm{vst},j)}$ that is already uniquely determined by $\chi|_{V_{\mathrm{in}}^{(\mathrm{vst})}}$. We wish to find colorings for the vector state gadgets resulting in this vector output.

We arbitrarily choose $S^{(j)}$ for all $j \in [n^4]$ such that the fulfill the conditions $\forall j \in n^4 : (|S_1^{(j)}|, |S_2^{(j)}|, |S_3^{(j)}|, |S_4^{(j)}|, |S_5^{(j)}|, |S_6^{(j)}|) = v_x^{(\mathrm{vst},j)}$ and $\forall i \in [6] : \bigcup_{j=1}^{n^4} S_i^{(j)} = S_i$. Now we can use the existence guarantees of the gadget $G^{(\mathrm{vst},j)}$ for each $j \in [n^4]$ with $S^{(j)}$ and use the resulting colorings to complete χ. We have that $\chi(V_x) = \chi(\bigcup_{j\in[n^4]} V_x^{(j)}) = \bigcup_{j\in[n^4]} \chi(V_x^{(j)}) = \bigcup_{k\in[6]} S_k$. By the calculation above, we thus have that $p(\chi|_{V_x})$ is the vector output generated by a_t and thus equal to v_x. Furthermore, $\forall k \in [6] : S_k \subseteq \chi(V_x)$.

Finally, as always, note that no conflict can arise between inner vertices of different subgadgets due to Property 4.7.

□

4.4.4 Removing Colorless Vertices

So far, the reduction relies on colorless vertices. We will now argue that they can be replaced by normal vertices.

Before we do this, we need to introduce one small change to the subset gadget. The end goal is that that all colorless nodes are replaced by normal vertices, which the current structure of the subset gadget does not allow.

To fix this, we reduce the number of complement vertices by two – meaning that the total number of complement vertices is now $q - \alpha - 2$ instead of $q - \alpha$ – and add an edge between the two colorless vertices.

Consider what happens if the replace the colorless vertices with normal vertices in the subset gadget. We argue that this modified gadget still encodes a subset constraint. All complement vertices, as well as both previously colorless nodes, are now within distance two of the α input vertices. They are also pairwise within distance two of each other. Hence these $q-\alpha-2+2$ vertices must be colored using exactly the $q - \alpha$ colors which do not appear on the input vertices. But now the output vertices are also within distance two of all complement vertices and the two previously colorless vertices, hence they cannot be colored using any of those colors. Hence their colors must be a subset of the colors used for the input vertices (and all their colors must be distinct). Hence the subset gadget still behaves the same.

We now prove that we can replace all colorless vertices in the entire graph by normal vertices. We argue that to convert a solution for the instance with colorless vertices to a solution for the instance without colorless vertices, one can simply assign each previously colorless vertex one of the neutral colors.

Hence the number of neutral colors $q_{\text{colorless}}$ is set to the number of colorless vertices which appear in the graph. Note that the construction of the graph does depend on the number of colors because of the number of complement colors in the subset gadget. However, this does not lead to a circular definition, since the complement vertices are not colorless. The number of colorless vertices remains the same, even for a varying number of neutral colors.

It might seem counterintuitive to be so wasteful with the number of colors allowed when the problem is to determine whether the graph can be colored using a certain number of colors q. After all, we are proposing to use one color for each colorless vertex. Multiple colorless vertices can have the same color without introducing distance-two coloring conflicts, so does this not mean that we can trivially get away with using less colors, invalidating the entire reduction? Here is why this is not true: Let us call all non-complement, non-colorless nodes "bound" nodes. The construction of the subset gadgets depends on the total number of colors q, and works no matter how large q grows. All bound nodes are connected by a series of subset gadgets to one of the color classes X_i or Y_i. All these subset gadgets have as input only colors from color classes or one or more of the three logic colors. They forbid the neutral colors by having an appropriate number of complement vertices. Hence all bound nodes cannot be colored using these neutral colors, and can instead only be colored using one of the counting colors, or one of the logic colors.

Let us argue why this instance is equivalent to the one with colorless vertices.

If the instance with colorless vertices has a solution, we can simply convert this to a solution for the instance without colorless vertices by coloring each colorless vertices using its unique neutral color. For subsets gadgets, we must remove the two neutral colors from the complement vertices which were used to color the two previously colorless nodes in the gadget. We must check that there are no new distance-two color conflicts between these newly colored, previously colorless

vertices and the bound nodes as well as the complement vertices. Since all bound nodes cannot be colored using neutral colors, this certainly does not introduce any distance-two color conflicts between a bound and a previously colorless vertex. Furthermore, no new distance-two conflicts arise between previously colorless vertices, since each neutral color is only used on one previously colorless node. There are no conflicts between complement vertices and previously colorless nodes by the construction of the subset gadget (and since a previously colorless vertex in a subset gadget cannot be distance-two adjacent to complement vertices from another subset gadget). Hence, the colors of the previously colorless nodes introduce no new conflicts. Furthermore, the additional edge in the subset gadget does not produce new conflicts, since it only changes distance-two relationships between nodes within the subset gadget, for which we have already proven correctness above.

Now let us prove the other direction. Suppose the instance without colorless vertices has a solution. We wish to convert this to a solution for the instance with colorless vertices. To do this, we simply remove the colors of the colorless vertices. In subset gadgets, we color the two additional complement vertices per subset gadgets by the colors which the two colorless vertices had. Certainly, this does not introduce any new distance-two coloring conflicts.

4.4.5 Properties of the Reduction

Number of Vertices and Running Time Finally, let us quickly discuss an upper bound on the number of vertices this newly constructed graph has. We will first bound the number of vertices in the graph. The result will depend on $q_{\text{colorless}}$, since each subset gadget has $q - \alpha - 2$ complement vertices. Recall that in our construction, $q_{\text{colorless}}$ is the number of colorless vertices in the construction. Hence by also bounding the number of colorless vertices, we can get a bound on the total number of vertices of the graph that does not depend on $q_{\text{colorless}}$.

Discounting its inputs and outputs, a subset gadget has $O(q)$ vertices total, of which 2 are colorless. The color class copy gadget, which consists of the vertices $V_{\text{in}}^{(\text{cpy})}$, $V_{\text{out}}^{(\text{cpy})}$, the subset gadget connecting the two as well as the vertex set S, has $O(m \cdot n^6) + O(m \cdot n^6) + O(q) + O(\log m) = O(m \cdot n^6 + q)$ vertices in total. Of those, $O(m \cdot n^6)$ vertices are colorless.

Now for the socket gadget. Both switch sockets and constant sockets simply consist of a constant number of subset gadgets where $\alpha + \beta$ is always bounded by $O(n^6)$. Hence, the total number of vertices is $O(q + n^6)$, and the total number of colorless vertices is $O(1)$.

The edge selection gadget consists of $2n^2$ socket gadgets and $O(n^2)$ subset gadgets connecting them. Hence the total number of vertices is $O(n^2(q+n^6))$, and the total number of colorless vertices is $O(n^2)$.

Similarly, the vector state gadget consists of three edge selection gadgets and a constant number of subset gadgets connecting them. Hence, again, the total number of vertices is $O(n^2(q+n^6))$, and the total number of colorless vertices is $O(n^2)$.

The one-way switch gadget has a constant number of subset gadgets with $\alpha + \beta = O(1)$ and a constant number of other vertices, including colorless vertices. Hence, it has $O(1)$ total and $O(1)$ colorless vertices.

The vector selection gadget consists of n^4 vector state gadgets and one-way switch gadgets, as well as $O(n^4)$ subset gadgets connecting them. Hence, we have $O(n^4 \cdot n^2(q + n^6)) = O(n^6(q + n^6))$ vertices total, of which $O(n^4 \cdot n^2) = O(n^6)$ are colorless.

Finally, the construction has k vector selection gadgets, one color class copy gadget, $O(k)$ subset gadgets connecting them, plus a constant number of additional vertices. Hence the total number of vertices of the reduction output is $k \cdot O(n^6(q + n^6)) + O(m \cdot n^6 + q) + O(k) \cdot O(q) + O(1) = O(kn^6(q + n^6) + mn^6)$, of which $k \cdot O(n^6) + O(m \cdot n^6) + O(k) \cdot O(1) + O(1) = O((k + m)n^6)$ are colorless.

Recall that by construction, we have that $q_{\text{colorless}}$ is set to the number of colorless vertices. Hence, $q = 2m \cdot 2n^6 + 3 + q_{\text{colorless}} = 2m \cdot 2n^6 + 3 + O(mn^6) = O((k+m)n^6)$. Hence, the total number of vertices is $O(kn^6((k+m)n^6 + n^6) + mn^6) = O(k(k+m)n^{12}))$.

Certainly, the construction of the graph can be done in time $\text{poly}(|V(G)|)$, where G is the output graph. Since, as shown above, there are $\text{poly}(k,m)\,\text{poly}(n)$ vertices, this running time is also upper-bounded by $\text{poly}(k,m)\,\text{poly}(n)$.

Treewidth We now show that the graph we constructed has low treewidth. Specifically, we show it has treewidth at most $4\log(m) + c$, for some constant c. We show this via the cops-and-robbers game from Theorem 2.2.

First, we place $2r+4$ cops on the central vertex x, on w_X, on the $2r$ vertices of S in $G^{(\text{cpy})}$, and on the two formerly colorless vertices in the $2m \cdot 2n^6$ equality gadget connecting $V_{\text{in}}^{(\text{cpy})}$ and $V_{\text{out}}^{(\text{cpy})}$. These cops will never move. If the robber is on one of the complement vertices of the $2m \cdot 2n^6$ equality gadget, we can catch them immediately with another cop.

We will now show that if the robber is not on one of the complement vertices, we can catch them using only a constant number of additional cops.

Note that each $X_i^{(\text{cpy})}$ or $Y_i^{(\text{cpy})}$ is only connected to one vector selection gadget. Suppose we deleted x, w_X, as well as S and the $2m \cdot 2n^6$ subset gadget in $G^{(\text{cpy})}$. Furthermore, suppose we delete, for each i, the equality gadgets connecting $(r')^{(\text{sel},i)}, (g')^{(\text{sel},i)}, (b')^{(\text{sel},i)}$ to $r^{(\text{sel},i+1)}, g^{(\text{sel},i+1)}, b^{(\text{sel},i+1)}$, respectively. Then for each i, the vertices of $G^{(\text{sel},i)}$ along with the six vertex groups $X_i^{(\text{cpy})}$ or $Y_i^{(\text{cpy})}$ that $G^{(\text{vst},i)}$ is connected to via equality gadgets form a connected component. Hence the robber must be in one of these connected components, or in one of the equality gadgets connecting them.

It suffices to use six cops to block the logic color inputs and outputs of a vector selection gadget. Hence, we will use twelve cops to slowly work through the chain of vector selection gadgets, always blocking two consecutive vector selection gadgets. We start by blocking the inputs and outputs of $G^{(\text{sel},1)}$ and $G^{(\text{sel},2)}$. Then for each $i \in [k-2]$, we remove the cops from $G^{(\text{sel},i)}$ and place them on the inputs and output of $G^{(\text{sel},i+2)}$. At some point during this process, the robber must be trapped within the equality gadgets connecting two vector selection gadgets, or within a vector selection gadget and its connected vertex groups $X_i^{(\text{cpy})}$ or $Y_i^{(\text{cpy})}$. In the former case, it is easy to catch them using a constant number of additional cops. In the latter case, the six cops blocking the inputs and outputs of that vector selection gadget will now remain there, and we must now catch the robber within this vector selection gadget or its connected $X_i^{(\text{cpy})}$ and $Y_i^{(\text{cpy})}$ using a constant number of additional cops.

The vector selection gadget has its six color class inputs (connected to the corresponding six X_i and Y_i) and its three logic color inputs and outputs. The main body of the gadget is made of the chain of vector state gadgets $G^{(\text{vst},i)}$ and one-way switch gadgets $G^{(\text{ows},i)}$.

We can use two cops to disconnect the inputs and outputs of a subset gadget by placing them on both of the formerly colorless vertices (note that one would suffice, but we use two to make the argument below cleaner). Hence, for any vector state gadget, we can block the equality gadgets that externally connect to its inputs and outputs using a constant number of cops.

We do this input-output blocking for the first vector selection gadget. Specifically, this blocks the color class inputs $I_j^{(\text{sel},i)}$ from the rest of the vector selection gadget. The robber is now either on one of the equality gadgets leading to an X_i or Y_i (which also includes the vertices of the X_i or Y_i), or they are within the vector selection gadget itself.

Assume they are on one of the equality gadgets going to the X_i or Y_i. Certainly, we can block the formerly colorless vertices using two cops, then catch the robber – who is either on one of the complement vertices or on vertices of the X_i or Y_i – using a third cop. Hence this case is trivial.

Now assume the robber is within the vector selection gadget. We can now slowly go through the chain of vector state and one-way switch gadgets, always blocking off the inputs and outputs of two consecutive vector state gadgets.

We start by blocking off the inputs and outputs of $G^{(\text{vst},1)}$ and $G^{(\text{vst},2)}$. Then for each $i \in [n^4-2]$, we remove the cops that are currently doing the input-output blocking for $G^{(\text{vst},i)}$ and use them to block off the inputs and outputs of $G^{(\text{vst},i+2)}$. At some point during this process, the robber must be trapped either within the complement vertices of one of the equality gadgets, within a vector state gadget, or within a one-way-switch gadget.

If they are on complement vertices or within a one-way-switch gadget, a constant number of additional cops obviously suffice to catch them. Hence, assume they are within a vector state gadget. We must now use a constant number of additional cops to watch them within that gadget.

In the vector state gadget, there are constantly many subset gadgets connecting the edge selection gadget, hence we can block off all of them with a constant number of cops. Without loss of generality we can assume the robber is now within one of the edge selection gadgets (they can also be in complement vertices of the subset gadgets, but that case is trivial).

The edge selection gadget consists of a chain of socket gadgets. Again, we can go through the chain by blocking off the input and output subset gadgets of a window of two consecutive socket gadgets, always using a constant number of cops. Without loss of generality the robber is within one of the socket gadgets.

Finally, both the constant and the switch socket gadget consist only of a constant number of subset gadgets. If we block the constantly many formerly colorless vertices of these gadgets with cops, the remaining vertices within the socket gadget form a collection of connected components that each have a constant number of vertices (in fact, most are simply a single vertex). Certainly, we can use a constant number of cops to go through these connected components one by one.

Hence, we have shown that after the initial $2r + 4$ stationary cops have been placed, the cops can catch the robber using a constant number of additional cops. Hence, the total number of cops needed – and hence the treewidth of the graph – is bounded by $2r + O(1)$. Using our definition $r = \lceil \frac{(1+\varepsilon)\log(m)}{2} \rceil$, we have shown that the graph has treewidth at most $(1 + \varepsilon)\log(m) + O(1)$.

This concludes the proof of Theorem 4.4.

5 Algorithms for Planar Graphs

We now turn to designing algorithms for Square Coloring on planar graphs. Let us write Planar Square Coloring (resp. Planar Square-q-Coloring) to refer to the variant of Square Coloring (resp. Square-q-Coloring) where the input graph is planar. The main result of this section is an algorithm that solves Planar Square Coloring in subexponential time $2^{O(n^{2/3}\log n)}$.

The algorithm consists of two subroutines, one of which covers the case that the number q of available colors is small, and the second subroutine is used for large numbers of colors. The first subroutine relies on the fact that we can bound the treewidth of G^2 by $O(\sqrt{nq})$ at which point we can rely on standard dynamic programming algorithms for q-Coloring. The second subroutine is more intricate and here, the idea is to construct a $(O(n/q), O(n/q), O(1))$-protrusion decomposition (see, e.g., [22, Chapter 15]) and then rely on dynamic programming ideas that are similar to those already used in Section 3. This results in an algorithm for Planar Square Coloring running in time $n^{O(n/q)}$.

Let us start with the first subroutine. Here, we use the fact that planar graphs of bounded diameter have bounded treewidth. Let G be a graph. A *spanning tree* of G is a tree T with vertex $V(G)$ and edge set $E(T) \subseteq E(G)$. Here, we consider rooted spanning trees where an arbitrary

vertex of T is declared to be the root of T. The *height* of T is the maximum distance between the root of T and any other vertex of T.

Lemma 5.1 (see, e.g., [21, Lemma 12.10])**.** *Let G be a planar graph that has a spanning tree of height ℓ. Then*

$$\mathrm{tw}(G) \leq 3\ell.$$

Moreover, given a planar graph G and a spanning tree of G of height ℓ, a tree decomposition of G of width at most 3ℓ can be computed in time $O(\ell \cdot n)$.

The next lemma is the key insight for the first subroutine.

Lemma 5.2. *Let G be a planar graph of maximum degree Δ. Then*

$$\mathrm{tw}(G^2) = O\left(\sqrt{n\Delta}\right).$$

Moreover, given a planar graph G, a tree decomposition of G^2 of width $O\left(\sqrt{n\Delta}\right)$ can be computed in polynomial time.

Proof. Since the treewidth of a graph G equals the maximum treewidth of its connected components we may assume without loss of generality that G is connected. Fix $r \in V(G)$ to be an arbitrary vertex of G. We define

$$D_i \coloneqq \left\{ v \in V(G) \;\middle|\; \left\lfloor \frac{\mathrm{dist}_G(v,r)}{2} \right\rfloor = i \right\}$$

for all $i \in \mathbb{Z}$. We define

$$M \coloneqq \left\lceil \frac{\sqrt{n\Delta}}{\Delta} \right\rceil$$

and let

$$L_j \coloneqq \bigcup_{i \equiv j \mod M} D_i$$

for all $j \in \{0, \ldots, M-1\}$. Now let $j^* \in \{0, \ldots, M-1\}$ such that $|L_{j^*}|$ is minimal. Clearly,

$$|L_{j^*}| \leq \frac{n}{M} \leq \frac{n}{\frac{\sqrt{n\Delta}}{\Delta}} = \sqrt{n\Delta}.$$

Claim 5.1. Let $i \in \mathbb{Z}$ such that $i \equiv j^* \mod M$. Then

$$\mathrm{tw}(G^2[D_{i+1} \cup \cdots \cup D_{i+M-1}]) \leq (\Delta+1)(4+6(M+1)) - 1.$$

Moreover, a tree decomposition of $G^2[D_{i+1} \cup \cdots \cup D_{i+M-1}]$ of width at most $(\Delta+1)(4+6(M+1))-1$ can be computed in polynomial time.

Proof. For ease of notation let us define $C \coloneqq D_{i+1} \cup \cdots \cup D_{i+M-1}$. Consider the sets $R' \coloneqq D_0 \cup \cdots \cup D_{i-1}$ and $C' \coloneqq D_i \cup \cdots \cup D_{i+M}$. Note that $G[R']$ is connected (if $R' \neq \emptyset$). Now let G' be the graph obtained from G by deleting all vertices outside of $R' \cup C'$ and contracting R' to a single vertex. Performing breath-first search on G' starting at R' results in a spanning tree of height at most $\ell \coloneqq 1 + 2(M+1)$.

So $\mathrm{tw}(G') \leq 3\ell$ by Lemma 5.1. Let (T, β') be a tree decomposition of G' of width at most 3ℓ. We construct a tree decomposition (T, β) of $G^2[C]$ as follows (observe that the tree T remains unchanged). For each $t \in V(T)$ we define

$$\beta(t) \coloneqq \left(\bigcup_{w \in \beta'(t) \cap C'} N_G[w] \right) \cap C.$$

Clearly, (T, β) can be computed in polynomial time using the algorithm from Lemma 5.1.

We show that (T, β) is indeed a tree decomposition of $G^2[C]$. First suppose that $uv \in E(G^2[C])$. Then there is some vertex $w \in V(G)$ such that $uw, wv \in E(G)$. Since $w \in N_G[C]$ and $N_G[C] \subseteq C'$ we conclude that $w \in C'$. In particular, there is some node $t \in V(T)$ such that $w \in \beta'(t)$. But then $u, v \in \beta(t)$ by definition.

So let $v \in C$ and let $T_v \coloneqq \{t \in V(T) \mid v \in \beta(t)\}$. We have that

$$T_v = \{t \in V(T) \mid N_G[v] \cap \beta'(t) \neq \emptyset\}.$$

Since $N_G[v]$ induces a connected subgraph in G' and (T, β') is a tree decomposition, we conclude that T_v induces a connected subtree of T (see Observation 2.1). So overall, (T, β) is a tree decomposition of $G^2[C]$.

To complete the proof, we bound the width of (T, β). Let $t \in V(T)$. We have that

$$|\beta(t)| \leq (\Delta + 1) \cdot |\beta'(t)| \leq (\Delta + 1)(3\ell + 1) = (\Delta + 1)(4 + 6(M + 1)).$$ ⌟

Now, we construct a tree decomposition (T, β) of G as follows. Let $i_1, \ldots, i_k$ be the list of all indices $i \in \mathbb{Z}$ such that $i \equiv j^* \mod M$ and $C_i \coloneqq D_{i+1} \cup \cdots \cup D_{i+M-1} \neq \emptyset$. For every $p \in [k]$ we construct a tree decomposition (T_p, β_p) of $G^2[C_{i_p}]$ via the last claim. Moreover, let $r_p \in V(T_p)$ denote an arbitrary node which is designated as the root of T_p. We define T to be the tree obtained from the disjoint union of all trees T_p and a fresh root node r that is connected to the root node r_p for every $p \in [k]$. Formally,

$$V(T) \coloneqq \{r\} \cup \bigcup_{p \in [k]} (\{p\} \times V(T_p))$$

and

$$E(T) \coloneqq \{r(p, r_p) \mid p \in [k]\} \cup \bigcup_{p \in [k]} \{(p, t)(p, t') \mid tt' \in E(T_p)\}.$$

We define

$$\beta(r) \coloneqq L_{j^*}$$

and

$$\beta(p, t) \coloneqq L_{j^*} \cup \beta_p(t)$$

for all $p \in [k]$ and $t \in V(T_p)$. Clearly, (T, β) can be computed in polynomial time.

We show that (T, β) is indeed a tree decomposition of G^2. Let $uv \in G^2$. We distinguish several cases. If $u, v \in L_{j^*}$ then $u, v \in \beta(r)$. Otherwise, by symmetry, we may assume that $u \in C_{i_p}$ for some $p \in [k]$. Since $\mathrm{dist}_G(u, v) \leq 2$ and $u \in D_{i_p+1} \cup \cdots \cup D_{i_p+M-1}$ it follows that $v \in D_{i_p} \cup \cdots \cup D_{i_p+M} \subseteq C_{i_p} \cup L_{j^*}$. If $v \in C_{i_p}$ then there is some $t \in V(T_p)$ such that $u, v \in \beta_p(t) \subseteq \beta(p, t)$. Otherwise $v \in L_{j^*}$ and $u, v \in \beta(p, t)$ for any $t \in V(T_p)$ such that $u \in \beta_p(t)$.

Also, it is easy to see that for every $v \in V(G)$ the set $T_v \coloneqq \{t \in V(T) \mid v \in \beta(t)\}$ induces a connected subtree of T. So overall, we conclude that (T, β) is a tree decomposition of G^2.

Finally, for $t \in V(T)$, we have that

$$|\beta(t)| \leq |L_{j^*}| + (\Delta + 1)(4 + 6(M + 1)) = O\left(\sqrt{n\Delta}\right).$$ □

Now, to obtain an algorithm for Square-q-Coloring, we can combine the last lemma with well-known algorithms for q-Coloring on graphs of bounded treewidth. The following theorem can for example be obtained from [14, Theorem 7.9 & 7.18]

Theorem 5.3. *For $q \geq 3$, there is an algorithm that solves q-Coloring in time $q^{O(\mathsf{tw})} \cdot n^{O(1)}$ on graphs of treewidth at most tw.*

Theorem 5.4. *There is an algorithm that solves Square-q-Coloring on planar graphs of maximum degree Δ in time $q^{O(\sqrt{n\Delta})}$.*

Proof. Let G denote the input graph. We compute the graph G^2 and apply the algorithm from Theorem 5.4. By Lemma 5.2, we have that $\mathrm{tw}(G^2) = O(\sqrt{n\Delta})$. So the algorithm from Theorem 5.4 runs in time $q^{O(\sqrt{n\Delta})}$. □

Corollary 5.5 (Lemma 1.6 restated)**.** *There is an algorithm that solves Planar Square-q-Coloring in time $q^{O(\sqrt{qn})}$.*

Proof. Let G denote the input graph. If G has maximum degree at least q then G^2 contains a clique of size at least $q+1$ and the we return NO. Otherwise, we run the algorithm from Theorem 5.4. □

This completes the first subroutine. Next, we turn to the second subroutine which covers the case that the number q of available colors is large. The basic strategy for this case is as follows. First, we apply some simple reduction rules to remove vertices whose second neighborhood is small (those vertices can always be colored in the end using a greedy strategy). Afterwards, there need to be many vertices of large degree which allows us to identify a distance-3 dominating set D of small size, i.e., every vertex of G is at distance at most 3 from some vertex in D. Using known techniques [8], we use this dominating set to construct a $(O(n/q), O(n/q), O(1))$-protrusion decomposition. Finally, we use dynamic programming approaches to design an algorithm for Square-q-Coloring given the protrusion decomposition.

We start by describing a simple reduction rule. We say that a graph G is *q-irreducible* if $V(G) = U \cup N_G(U)$ where $U := \{u \in V(G) \mid |N_{G^2}[u]| > q\}$. If G is not q-irreducible then we can identify a strict induced subgraph of G that is equivalent to G with respect to the problem Square-q-Coloring as the next lemma shows.

Lemma 5.6. *Let G be a graph and let $q \geq 1$ be an integer. Let $U := \{u \in V(G) \mid |N_{G^2}[u]| > q\}$ and $W := U \cup N_G(U)$. Then G has a square q-coloring if and only if $G[W]$ has a square q-coloring.*

Proof. The forward direction is trivial since any square q-coloring of G immediately restricts to a square q-coloring of $G[W]$.

So assume that $G[W]$ has a square q-coloring χ', i.e., χ' is a q-coloring of $(G[W])^2$. We define a coloring $\chi \colon V(G) \to [q]$ as follows. First, we set $\chi(u) := \chi'(u)$ for all $u \in U$. Since $N_G(U) \subseteq W$, every path of length at most 2 between two vertices $u, u' \in U$ in G also exists in $G[W]$. This implies that $\chi(u) \neq \chi(u')$ for all distinct $u, u' \in U$ such that $\mathrm{dist}_G(u, u') \leq 2$. Now, let $\{v_1, \ldots, v_\ell\} = V(G) \setminus U$ and pick $i \in [\ell]$. Since $v_i \notin U$ we have that $|N_{G^2}[v_i]| \leq q$ by definition. This means there is some color $c_i \in [q]$ that is not used so far by any vertex in $N_{G^2}[v_i]$. We set $\chi(v_i) := c_i$. Doing this for all $i \in [\ell]$, we obtain a square q-coloring χ of G. □

Now, as the next intermediate target, we construct a distance-3 dominating set of a planar, q-irreducible graph.

Lemma 5.7. *Let $\ell \in \mathbb{Z}_{>0}$ be a positive integer. Also let G be a graph and suppose $U \subseteq V(G)$ such that $|N_{G^2}[u]| \geq \ell$ for all $u \in U$. Then there is a set $D \subseteq U$ such that*

$$|D| \leq \frac{2 \cdot |E(G)|}{\ell}$$

and for every $u \in U$ there is some $w \in D$ such that $\mathrm{dist}_G(w, u) \leq 2$. Moreover, given the graph G and the set U, one can compute such a set D in polynomial time.

Proof. We compute the set D using a greedy algorithm. We initialize $D \coloneqq \emptyset$ and, as long as D does not a dominate every vertex from U in the graph G^2, we pick an arbitrary undominated vertex and add it to D. Clearly, this can be done in polynomial time. We need to prove the desired bound on the size of D. Suppose $D = \{v_1, \ldots, v_s\}$ where we list vertices in the order in which they are added to D by the greedy algorithm described above. Observe that $N_G[v_i] \cap N_G[v_j] = \emptyset$ for all $i < j \in [s]$ since otherwise v_j would be dominated by v_i and the greedy algorithm would not have added it to D. Then

$$s \cdot \ell \leq \sum_{i=1}^{s} \sum_{w \in N_G[v_i]} \deg_G(w) \leq \sum_{w \in V(G)} \deg_G(w) \leq 2 \cdot |E(G)|.$$

Rearranging the terms gives the desired bound. □

Corollary 5.8. *There is a polynomial-time algorithm that, given an integer $q \geq 1$ and a q-irreducible graph G, computes a set $D \subseteq V(G)$ such that*

1. *for every $v \in V(G)$ there is some $w \in D$ such that $\mathrm{dist}_G(v, w) \leq 3$, and*
2. *$|D| \leq \frac{2 \cdot |E(G)|}{q}$.*

Proof. Let $U \coloneqq \{u \in V(G) \mid |N_{G^2}[u]| > q\}$. By Lemma 5.7 there is some set $D \subseteq U$ such that

$$|D| \leq \frac{2 \cdot |E(G)|}{q}$$

and for every $u \in U$ there is some $w \in D$ such that $\mathrm{dist}_G(w, u) \leq 2$. Also, for every $v \in V(G)$ there is some $u \in U$ such that $\mathrm{dist}_G(v, u) \leq 1$. In combination, gives the desired properties of the set D. Finally, the set U can clearly be computed in polynomial time and afterwards, D is computed using the algorithm from Lemma 5.7. □

Now, we use the dominating set constructed in the corollary to obtain a $(O(n/q), O(n/q), O(1))$-protrusion decomposition. Let us start by formally defining protrusion decompositions.

Definition 5.9. Let G be a graph and let $\alpha, \delta, k \geq 2$ be integers. An *(α, δ, k)-protrusion decomposition of G* is a rooted tree decomposition (T, β) of G such that

1. $|\beta(r)| \leq \alpha$ where r is the root of T,
2. $|\beta(t)| \leq k$ for every other node $t \in V(T) \setminus \{r\}$, and
3. $\deg_T(r) \leq \delta$.

Lemma 5.10 ([8, Lemma 6.2]). *Let $r \geq 1$ be a fixed integer. Let G be a planar graph and suppose that there is a set $D \subseteq V(G)$ such that for every $v \in V(G)$ there is some $w \in D$ such that $\mathrm{dist}_G(v, w) \leq r$. Then there is a $(c|D|, c|D|, c)$-protrusion decomposition of G where c is some constant that only depends on r.*

Moreover, given G and D, a $(c|D|, c|D|, c)$-protrusion decomposition of G can be computed in polynomial time.

We remark at this point that our definition of a protrusion decomposition, which follows [22], is slightly different from the definition from [8]. However, up to constant factors in the parameters, the precise definition does not matter for the existence of a protrusion decomposition (see also [22, Chapter 15]). Let us also point out that [8, Lemma 6.2] does not directly state the existence of a polynomial-time algorithm that computes the protrusion decomposition. However, the algorithm is imminent from the proof. Alternatively, an algorithm can also be obtained via methods described in [22, Chapter 15].

By combining Corollary 5.8 and Lemma 5.10 we obtain the following result.

Corollary 5.11. *There is a polynomial-time algorithm that, given an integer $q \geq 1$ and a q-irreducible, planar graph G, computes a $(\frac{cn}{q}, \frac{cn}{q}, c)$-protrusion decomposition of G for some absolute constant c.*

The next lemma implements the last step of the second subroutine.

Lemma 5.12. *There is an algorithm that, given a graph G, an integer q and an (α, δ, k)-protrusion decomposition (T, β) of G, decides whether G has a square q-coloring in time $|V(T)| \cdot n^{O(\alpha+\delta\cdot 2^k)}$.*

Proof. Let $X \coloneqq \beta(r)$ where r denotes the root of T. Also, let $t_1, \ldots, t_\delta$ denote the children of r in T, and define $Y_i \coloneqq \beta(r) \cap \beta(t_i)$ for all $i \in [\delta]$. Moreover, let V_i denote the set of vertices contained in bags below t_i (including t_i itself). Finally, we define

$$\mathcal{Y} \coloneqq \bigcup_{i\in\delta}(\{i\} \times Y_i)$$

and

$$\mathcal{Z} \coloneqq \{(i, A) \mid i \in [\delta], A \subseteq Y_i\}.$$

Observe that $|\mathcal{Y}| \leq \delta \cdot k$ and $|\mathcal{Z}| \leq \delta \cdot 2^k$.

The algorithm iterates over all triples of functions $\chi\colon X \to [q]$, $\xi\colon \mathcal{Y} \to 2^X$ such that $\xi(i, v) \subseteq Y_i$ for all $i \in [\delta]$ and $v \in Y_i$, and $\rho\colon \mathcal{Z} \to [q]_0$. Observe that the number of such triples is upper bounded by

$$q^{|X|} \cdot (2^k)^{\delta k} \cdot (q + 1)^{\delta 2^k} = q^{O(\alpha+\delta\cdot 2^k)}. \tag{5}$$

So let us fix such a triple (χ, ξ, ρ). We say that (χ, ξ, ρ) is *locally valid* if

(PL.1) $\chi(u) \neq \chi(v)$ for all distinct $u, v \in X$ such that $\mathrm{dist}_{G[X]}(u, v) \leq 2$,

(PL.2) there are no vertices $u, w \in Y_i$ and $v \in X$ such that $uv \in E(G)$, $u \in \xi(i, w)$ and $\chi(v) = \chi(w)$, and

(PL.3) there are no distinct $i, i' \in [\delta]$ and $v \in Y_i$, $w \in Y_{i'}$ such that $\chi(v) = \chi(w)$ and $\xi(i, v) \cap \xi(i', w) \neq \emptyset$.

Note that these conditions extend the conditions from Definition 3.4.

We also say that (χ, ξ, ρ) is *extendable* if, for every $i \in [\delta]$, there exists a square q-coloring $\bar{\chi}_i$ of $G[V_i]$ such that

(PD.1) $\chi(v) = \bar{\chi}_i(v)$ for all $v \in Y_i$,

(PD.2) $\xi(i, v) = A_i(\bar{\chi}_i, \chi(v))$ where

$$A_i(\bar{\chi}_i, c) \coloneqq \{v \in Y_i \mid \exists w \in V_i \setminus Y_i\colon \bar{\chi}_i(w) = c \wedge vw \in E(G)\},$$

and

(PD.3)
$$\rho(i,A) = |\{c \in [q] \mid A_i(\bar{\chi}_i, c) = A\} \setminus \{\chi(v) \mid v \in Y_i\}|$$
for all $A \in 2^{Y_i}$.

Observe that these conditions precisely reflect the conditions from Definition 3.2 and hence, it can be checked in time $|V(T)| \cdot n^{O(2^k)}$ whether (χ, ξ, ρ) is extendable by Lemma 3.3. If (χ, ξ, ρ) is not locally valid and extendable, then the algorithm rejects the current triple (χ, ξ, ρ) and moves to the next iteration.

Otherwise, similar to join operation in the proof Lemma 3.3, we need to verify that the partial square q-colorings are compatible to one another and can indeed be combined into a global square q-coloring. Let us say a color $c \in [q]$ *i-free* (with respect to (χ, ξ, ρ)) if $c \notin \{\chi(v) \mid v \in Y_i\}$. Here, the remaining task is to assign, to every $(i, A) \in \mathcal{Z}$, a set of $\rho(i, A)$ of i-free colors in a consistent way. We check this by implementing another dynamic programming algorithm.

For a set $C \subseteq [q]$ of colors and a function $\varrho\colon \mathcal{Z} \to [q]_0$ we define $\gamma(C, \varrho)$ to be true if there is a mapping $\eta\colon \mathcal{Z} \to 2^C$ satisfying the following conditions:

(i) $|\eta(i, A)| = \varrho(i, A)$,

(ii) $\eta(i, A) \cap \{\chi(v) \mid v \in Y_i\} = \emptyset$,

(iii) $\eta(i, A) \cap \eta(i, A') = \emptyset$ for all distinct $A, A' \subseteq Y_i$,

(iv) $\eta(i, A) \cap \eta(i', A') = \emptyset$ for all distinct $(i, A), (i', A') \in \mathcal{Z}$ such that $A \cap A' = \emptyset$, and

(v) there are no $(i, A) \in \mathcal{Z}$, $u \in A$ and $v \in X$ such that $uv \in E(G)$ and $\chi(v) \in \eta(i, A)$.

(vi) there are no distinct $i, i' \in [\delta]$, $A \subseteq Y_i$, $u \in Y_i \cap Y_{i'}$, $v \in Y_{i'}$ such that $u \in \xi(i', v)$, $u \in A$ and $\chi(v) \in \eta(i, A)$.

More intuitively, $\gamma(C, \varrho)$ is set to true if we can assign $\varrho(i, A)$ many free colors to every $(i, A) \in \mathcal{Z}$ in a consistent way by only using colors from the set C. In particular, our goal is to determine whether $\gamma([q], \rho)$ is true. We achieve this by computing a suitable subset of the entries $\gamma(C, \varrho)$ using a dynamic programming approach. More precisely, let $\mathcal{C}_{\mathsf{sing}} := \{\{c\} \mid c \in [q]\}$ be the set of all singleton subsets of $[q]$. Also, let $\mathcal{C}_{\mathsf{seg}} := \{[q'] \mid q' \in [q]\}$ be the set of all initial segments of $[q]$. We compute $\gamma(C, \varrho)$ for all $C \in \mathcal{C}_{\mathsf{sing}} \cup \mathcal{C}_{\mathsf{seg}}$ and all possible mappings ϱ using a dynamic programming approach. The next claim shows how to compute the entries $\gamma(C, \varrho)$ for $C \in \mathcal{C}_{\mathsf{sing}}$.

Claim 5.2. There is an algorithm that, given $c \in [q]$ and $\varrho\colon \mathcal{Z} \to [q]_0$, determines whether $\gamma(\{c\}, \varrho)$ is true and runs in time polynomial in n and $|\mathcal{Z}|$.

Proof. Since $|\{c\}| = 1$ there is at most one candidate function $\eta\colon \mathcal{Z} \to 2^C$ which satisfies Condition (i) (i.e., $\eta(i, A) = \emptyset$ if $\varrho(i, A) = 0$, and $\eta(i, A) = \{c\}$ if $\varrho(i, A) = 1$). Now, it can be easily checked in time polynomial in n and $|\mathcal{Z}|$ whether all other Conditions (ii) - (vi) are satisfied. ⌟

Now, we argue how to compute the remaining entries in a dynamic programming fashion.

Claim 5.3. Let $C \subseteq [q]$ and $\varrho\colon \mathcal{Z} \to [q]_0$ be some mapping. Also suppose $C = C_1 \uplus C_2$. Then $\gamma(C, \varrho)$ is true if and only if there are functions $\varrho_1, \varrho_2\colon \mathcal{Z} \to [q]_0$ such that

(A) $\gamma(C_1, \varrho_1)$ and $\gamma(C_2, \varrho_2)$ are true, and

(B) $\varrho(i, A) = \varrho_1(i, A) + \varrho_2(i, A)$ for all $(i, A) \in \mathcal{Z}$.

Proof. For the forward direction, first suppose that $\gamma(C, \varrho)$ is true, i.e., there is a function $\eta\colon \mathcal{Z} \to 2^C$ satisfying Conditions (i) - (vi). For $j \in \{1, 2\}$, we define $\eta_j\colon \mathcal{Z} \to 2^{C_j}$ via $\eta_j(i, A) \coloneqq \eta(i, A) \cap C_j$. Moreover, we define $\varrho_j(i, A) \coloneqq |\eta_j(i, A)|$. We have

$$\varrho(i, A) = |\eta(i, A)| = |\eta(i, A) \cap C_1| + |\eta(i, A) \cap C_2| = |\eta_1(i, A)| + |\eta_2(i, A)| = |\varrho_1(i, A)| + |\varrho_2(i, A)|$$

which shows Property (B). Also, it is easy to verify that Property (A) holds via the witnessing mappings η_1 and η_2. Indeed, for $j \in \{1, 2\}$, the function η_j satisfies Condition (i) by definition of ϱ_j, and Conditions (ii) - (vi) for η_j follow directly from the corresponding properties of η.

For the backward direction let $\varrho_1, \varrho_2\colon \mathcal{Z} \to [q]_0$ be functions satisfying (A) and (B). By Property (A), there are functions $\eta_j\colon \mathcal{Z} \to 2^{C_j}$, $j \in \{1, 2\}$, satisfying Conditions (i) - (vi) with respect to (C_j, ϱ_j). We define $\eta\colon \mathcal{Z} \to 2^C$ via $\eta(i, A) \coloneqq \eta_1(i, A) \cup \eta_2(i, A)$. It is again easy to check that η witnesses that $\gamma(C, \varrho)$ is true. Indeed, Condition (i) follows directly from Property (B) and the definition of η. Also, as before, Conditions (ii) - (vi) for η follow directly from the corresponding properties of η_1 and η_2. ⌟

Now, we can iteratively compute all entries $\gamma(C, \varrho)$ where $C \in \mathcal{C}_{\mathsf{sing}} \cup \mathcal{C}_{\mathsf{seg}}$ as follows. First, we iterate over all $C \in \mathcal{C}_{\mathsf{sing}}$ and all choices of ϱ and compute $\gamma(C, \varrho)$ using Claim 5.2. Afterwards, we iterate over all values $q' \in \{2, \dots, q\}$ (starting at $q' = 2$) as well as over all $\varrho\colon \mathcal{Z} \to [q]_0$. To determine whether $\gamma([q'], \varrho)$ is true, we set $C_1 \coloneqq \{1, \dots, q' - 1\}$ and $C_2 \coloneqq \{q'\}$ and iterate over all function pairs $\varrho_1, \varrho_2\colon \mathcal{Z} \to [q]_0$ such that $\varrho(i, A) = \varrho_1(i, A) + \varrho_2(i, A)$ for all $(i, A) \in \mathcal{Z}$. If there is such a pair such that $\gamma(C_1, \varrho_1)$ and $\gamma(C_2, \varrho_2)$ are true, then we set $\gamma(C, \varrho)$ to true. This correctly computes $\gamma(C, \varrho)$ by Claim 5.3.

Finally, the main algorithm returns YES (i.e., the algorithm concludes that G has a square q-coloring) if $\gamma([q], \rho)$ is true. Otherwise, the algorithm moves to the next iteration of the outer loop (i.e., the algorithm chooses the next triple (χ, ξ, ρ)).

If the algorithm has checked all triples (χ, ξ, ρ) and never returned YES, then it returns NO.

(**Correctness.**) We argue that the algorithm described above correctly decides whether G has a square q-coloring. If G is not q-irreducible, this follows directly from Lemma 5.6. So suppose that G is q-irreducible.

First suppose there is some square q-coloring $\bar{\chi}$ of G. We define $\chi \coloneqq \bar{\chi}|_X$ and $\bar{\chi}_i \coloneqq \bar{\chi}|_{V_i}$ for all $i \in [\delta]$. Moreover, we define

$$\xi(i, v) = A_i(\bar{\chi}_i, \chi(v))$$

for all $(i, v) \in \mathcal{Y}$, and

$$\rho(i, A) = |\{c \in [q] \mid A_i(\bar{\chi}_i, c) = A\} \setminus \{\chi(v) \mid v \in Y_i\}|$$

for all $(i, A) \in \mathcal{Z}$. We claim the algorithm return YES in the iteration (χ, ξ, ρ), i.e., we need to show that (χ, ξ, ρ) is locally valid, extendable and $\gamma([q], \rho)$ is true.

Condition (PL.1) is clearly satisfied since $\bar{\chi}$ is a square coloring of G. For Condition (PL.2) suppose $uv \in E(G)$, $\chi(v) = \chi(w)$, and $u \in \xi(i, w) = A_i(\bar{\chi}_i, \chi(w))$. By definition, there is some $v' \in V_i \setminus Y_i$ such that $\chi(v') = \chi(w) = \chi(v)$ and $uv' \in E(G)$. Note that $v \neq v'$ since $v \in X$. This contradicts $\bar{\chi}$ being a square coloring of G. Similarly, for Condition (PL.3), suppose there are distinct $i, i' \in [\delta]$ and $v \in Y_i$, $w \in Y_{i'}$ such that $\chi(v) = \chi(w)$ and $\xi(i, v) \cap \xi(i, w)$. Let $u \in \xi(i, v) \cap \xi(i, w)$. By definition, there are $v' \in V_i \setminus Y_i$ such that $\chi(v') = \chi(v)$ and $uv' \in E(G)$, and $w' \in V_{i'} \setminus Y_{i'}$ such that $\chi(w') = \chi(w)$ and $uv' \in E(G)$. Overall, we obtain distinct vertices $v', u, w' \in V(G)$ such that $uv', uw' \in E(G)$ and $\chi(v') = \chi(w')$. Again, this contradicts $\bar{\chi}$ being a square coloring of G. So (χ, ξ, ρ) is locally valid.

Also, (χ, ξ, ρ) is extendable since Conditions (PD.1) - (PD.3) are satisfied by definition. So it remains to argue that there is some witness $\eta\colon \mathcal{Z} \to 2^{[q]}$ showing that $\gamma([q], \rho)$ is true. We define

$$\eta(i, A) \coloneqq \{c \in [q] \mid A_i(\bar{\chi}_i, c) = A\} \setminus \{\chi(v) \mid v \in Y_i\}.$$

We have that Conditions (i) - (iii) are trivially satisfied. Also, Conditions (iv) and (vi) can be verified using the same arguments as for Condition (PL.3) (note that, using Condition (iii), we only need to consider the case $i \neq i'$). Similarly, Condition (v) follows by the same arguments as Condition (PL.2).

Finally, observe that the algorithm correctly computes $\gamma([q], \rho)$ using Claims 5.2 and 5.3. So overall, the algorithm returns YES.

In the other direction, suppose that the algorithm outputs YES, i.e., there is a triple (χ, ξ, ρ) that is locally valid, extendable and $\gamma([q], \rho)$ is true. Since (χ, ξ, ρ) is extendable, for every $i \in [\delta]$, there is square q-coloring $\bar{\chi}_i$ of $G[V_i]$ satisfying (PD.1) - (PD.3). Also, $\gamma([q], \rho)$ is true meaning that there is a mapping $\eta\colon \mathcal{Z} \to 2^{[q]}$ satisfying (i) - (iv) (with respect to the tuple $([q], \rho)$). Since

$$|\eta(i, A)| = \rho(i, A) = |\{c \in [q] \mid A_i(\bar{\chi}_i, c) = A\} \setminus \{\chi(v) \mid v \in Y_i\}|$$

by Conditions (PD.3) and (i), we can rename the colors in the coloring $\bar{\chi}_i$ (using Conditions (ii) and (iii)) so that

$$\eta(i, A) = \{c \in [q] \mid A_i(\bar{\chi}_i, c) = A\} \setminus \{\chi(v) \mid v \in Y_i\} \tag{6}$$

for all $(i, A) \in \mathcal{Z}$ while preserving Conditions (PD.1) - (PD.3) (since we only rename free colors for every $i \in [\delta]$).

Now, we define the coloring $\bar{\chi}\colon V(G) \to [q]$ via

$$\bar{\chi}(v) \coloneqq \begin{cases} \chi(v) & \text{if } v \in X \\ \bar{\chi}_i(v) & \text{if } v \in V_i \text{ for some } i \in [\delta] \end{cases}$$

Note that $\bar{\chi}$ is well-defined by Condition (PD.1). We claim that $\bar{\chi}$ is a square coloring of G. Let $v, w \in V(G)$ be distinct vertices such that $\operatorname{dist}(v, w) \leq 2$. We need to argue that $\bar{\chi}(v) \neq \bar{\chi}(w)$.

First suppose that $v \in X$. If $vw \in E(G)$ then either $w \in X$ and $\bar{\chi}(v) \neq \bar{\chi}(w)$ by (PL.1), or $w \in V_i \setminus X$ for some $i \in [\delta]$ which implies that $v \in V_i$ and $\bar{\chi}(v) \neq \bar{\chi}(w)$ since $\bar{\chi}_i$ is a square coloring of $G[V_i]$. So suppose that $\operatorname{dist}(v, w) = 2$ and let $u \in N_G(v) \cap N_G(w)$. If $u, w \in X$ then $\bar{\chi}(v) \neq \bar{\chi}(w)$ by (PL.1). If $w \in X$, but $u \in V_i \setminus X$ for some $i \in [\delta]$ then $v, w \in V_i$, and $\bar{\chi}(v) \neq \bar{\chi}(w)$ follows as before. So suppose that $w \notin X$, i.e., $w \in V_i \setminus X$ for some $i \in [\delta]$. This means that $u \in V_i$. If $v \in V_i$ then again $\bar{\chi}(v) \neq \bar{\chi}(w)$ follows as before. So suppose that $v \in X \setminus V_i$ which means that $u \in Y_i$. If $\bar{\chi}(w)$ is not i-free then there is some $w' \in Y_i$ such that $\bar{\chi}(w) = \bar{\chi}(w') = \chi(w')$, and $u \in \xi(i, w')$ by (PD.2). So $\chi(w') \neq \chi(v)$ by (PL.2) which implies that $\bar{\chi}(v) \neq \bar{\chi}(w)$. In the other subcase $\bar{\chi}(w)$ is i-free which means that $\bar{\chi}(w) \in \eta(i, A)$ where $A = A_i(\bar{\chi}_i, \bar{\chi}(w))$ by Equation (6). Observe that $u \in A$ by definition. So $\bar{\chi}(v) = \chi(v) \notin \eta(i, A)$ by Condition (v). It follows that $\bar{\chi}(v) \neq \bar{\chi}(w)$. This completes the case that $v \in X$.

By symmetry, we also obtain that $\bar{\chi}(v) \neq \bar{\chi}(w)$ if $w \in X$. So suppose that $v, w \notin X$. So there are $i, i' \in [\delta]$ such that $v \in V_i \setminus Y_i$ and $w \in V_{i'} \setminus Y_{i'}$. If $i = i'$ then $\bar{\chi}(v) \neq \bar{\chi}(w)$ since $\bar{\chi}_i$ is a square coloring of $G[V_i]$. So suppose that $i \neq i'$. Observe that $vw \notin E(G)$. Since $\operatorname{dist}(v, w) \leq 2$ there is some $u \in N_G(v) \cap N_G(w)$. We have that $u \in Y_i \cap Y_{i'}$.

If $\chi(v)$ is i-free then $\bar{\chi}(v) \in \eta(i, A_v)$ where $A_v = A_i(\bar{\chi}_i, \bar{\chi}(v))$ by Equation (6), and $u \in A_v$. Similarly, if $\chi(w)$ is i'-free then $\bar{\chi}(w) \in \eta(i', A_w)$ where $A_w = A_{i'}(\bar{\chi}_{i'}, \bar{\chi}(v))$, and $u \in A_w$.

So if $\chi(v)$ is i-free and $\chi(w)$ is i'-free, then $A_v \cap A_w \neq \emptyset$ which implies that $\eta(i, A_v) \cap \eta(i', A_w) = \emptyset$ by Condition (iv). It follows that $\bar{\chi}(v) \neq \bar{\chi}(w)$.

Next, suppose that $\chi(v)$ is i-free and $\chi(w)$ is not i'-free. Then there is some $w' \in Y_{i'}$ such that $\bar{\chi}(w) = \bar{\chi}(w')$. Also, $u \in \xi(i', w')$ by (PD.2). So $\chi(w') \notin \eta(i, A_v)$ by Condition (vi). Since $\bar{\chi}(v) \in \eta(i, A_v)$, we conclude that $\bar{\chi}(v) \neq \bar{\chi}(w)$.

The case where $\chi(v)$ is not i-free and $\chi(w)$ is i'-free can be handled symmetrically.

So suppose $\chi(v)$ is not i-free and $\chi(w)$ is not i'-free. Then there is some $v' \in Y_i$ such that $\bar{\chi}(v) = \bar{\chi}(v')$, and there is some $w' \in Y_{i'}$ such that $\bar{\chi}(w) = \bar{\chi}(w')$. Also, $u \in \xi(i, v') \cap \xi(i', w')$ by (PD.2). So $\chi(v') \neq \chi(w')$ by (PL.3). It follows that $\bar{\chi}(v) \neq \bar{\chi}(w)$.

So overall, $\bar{\chi}$ is a square coloring of G which completes the correctness proof of the algorithm.

(**Running Time.**) By Equation 5, the outer loop of the algorithm considers at most $q^{O(\alpha+\delta\cdot 2^k)}$ many triples (χ, ξ, ρ). So let us fix such a triple. Clearly, it can be checked in polynomial time whether (χ, ξ, ρ) is locally valid. By turning the subtree rooted at t_i into a nice tree decomposition of $G[V_i]$ for all $i \in [\delta]$, we can use Lemma 3.3 to decide whether (χ, ξ, ρ) is extendable in time $|V(T)| \cdot n^{O(2^k)}$. So it only remains to analyze the time required to compute the entries $\gamma(C, \varrho)$ for all $C \in \mathcal{C}_{\mathsf{sing}} \cup \mathcal{C}_{\mathsf{seg}}$ and all possible mappings ϱ. First observe that the number of such pairs (C, ϱ) is bounded by

$$|\mathcal{C}_{\mathsf{sing}} \cup \mathcal{C}_{\mathsf{seg}}| \cdot (q+1)^{|\mathcal{Z}|} \leq 2q \cdot (q+1)^{\delta 2^k} = q^{O(\delta \cdot 2^k)}.$$

If $|C| = 1$ then we can compute $\gamma(C, \varrho)$ in time polynomial in n and $|\mathcal{Z}| \leq \delta \cdot 2^k$ by Claim 5.2. Otherwise, we use Claim 5.3 and computing an entry $\gamma(C, \varrho)$ takes time

$$\left((q+1)^{\delta 2^k}\right)^2 (n + |\mathcal{Z}|)^{O(1)}.$$

So overall, computing all entries takes time

$$q^{O(\delta \cdot 2^k)} \cdot \left((q+1)^{\delta 2^k}\right)^2 (n + |\mathcal{Z}|)^{O(1)} = q^{O(\alpha+\delta\cdot 2^k)} \cdot (n + |\mathcal{Z}|)^{O(1)}.$$

So overall, the algorithm takes time

$$q^{O(\alpha+\delta\cdot 2^k)} \cdot \left(|V(T)| \cdot n^{O(2^k)} + q^{O(\delta \cdot 2^k)} \cdot (n + |\mathcal{Z}|)^{O(1)}\right) = |V(T)| \cdot n^{O(\alpha+\delta\cdot 2^k)}.$$

as desired. □

Now, we can combine all the parts to obtain the following theorem.

Theorem 5.13 (Lemma 1.7 restated)**.** *There is an algorithm that solves* Planar Square-q-Coloring *in time* $n^{O(n/q)}$.

Proof. Let G denote the input graph. The algorithm computes $U \coloneqq \{u \in V(G) \mid |N_{G^2}[u]| > q\}$ and $W \coloneqq U \cup N_G(U)$. If $W \subsetneq V(G)$ the algorithm deletes all vertices from $V(G) \setminus W$ and recursively decides whether $G[W]$ has a square q-coloring. Observe that this is correct by Lemma 5.6.

Otherwise, G is q-irreducible. Using Corollary 5.11, we compute a (α, δ, c)-protrusion decomposition (T, β) of G where $\alpha, \delta = O(\frac{n}{q})$ and c is some absolute constant. Afterwards, we decide whether G has a square q-coloring using Lemma 5.12.

For the running time, observe that the algorithm always arrives at a q-irreducible graph after polynomially many steps. Also, given a q-irreducible graph, the protrusion decomposition (T, β) is computed in polynomial time by Corollary 5.11. Finally, the application of the algorithm from Lemma 5.12 takes time $n^{O(n/q)}$. □

Combining Corollary 5.5 and Theorem 5.13, we obtain a subexponential algorithm for Planar Square Coloring.

Corollary 5.14 (Theorem 1.4 restated)**.** *There is an algorithm that solves* Planar Square Coloring *in time* $2^{O(n^{2/3}\log n)}$.

Proof. Let G be the input graph and q the number of colors. If

$$q \leq n^{1/3}$$

then we apply Corollary 5.5 giving a running time of

$$q^{O(\sqrt{qn})} = n^{O(\sqrt{n^{4/3}})} = 2^{O(n^{2/3}\log n)}.$$

Otherwise, we apply Theorem 5.13 resulting in a running time

$$n^{O(\sqrt{n/q})} = n^{O(n^{2/3})} = 2^{O(n^{2/3}\log n)}.$$ □

6 Lower Bounds for Planar Graphs

In this section, we provide hardness results for Planar Square Coloring. More precisely, we prove the following theorem which implies Theorems 1.3 and 1.8.

Theorem 6.1. *For every fixed $q \geq 4$, the problem* Planar Square-q-Coloring *is* NP*-hard. Moreover, assuming* ETH*, it cannot be solved in time* $2^{o(\sqrt{n})}$.

Let us start with two basic remarks. First observe that the bound on q is optimal since Square-q-Coloring is polynomial-time solvable (on general graphs) for all $q \leq 3$. Indeed, if $q \leq 3$, then every graph G of maximum degree at least 3 is a trivial NO-instance since G^2 contains a clique of size at least 4. So it suffices to consider graphs of maximum degree at most 2 for which there is an easy algorithm.

Also note that the theorem immediately implies the same hardness results for Square Coloring.

Corollary 6.2. Planar Square Coloring *is* NP*-hard. Moreover, assuming* ETH*, it cannot be solved in time* $2^{o(\sqrt{n})}$.

To prove Theorem 6.1 we give a reduction from Planar 3-Coloring problem to Planar Square-q-Coloring for all $q \geq 4$ and then exploit known hardness results for Planar 3-Coloring.

Theorem 6.3. Planar 3-Coloring *is* NP*-hard. Moreover, assuming* ETH*, it cannot be solved in time* $2^{o(\sqrt{n})}$.

The NP-hardness of Planar 3-Coloring was first proved in [43]. For the second part of theorem we refer to [14, Theorem 14.9].

To describe the reduction, we split it into two steps. We first consider a variant where we also allow special *equality edges* where endpoints have to be assigned the same color. Then, in a second step, we replace the equality edges by a simple gadget to complete the intended reduction. We further split the first step depending on whether $q = 4$ or $q \geq 5$ as different constructions are needed for these two cases.

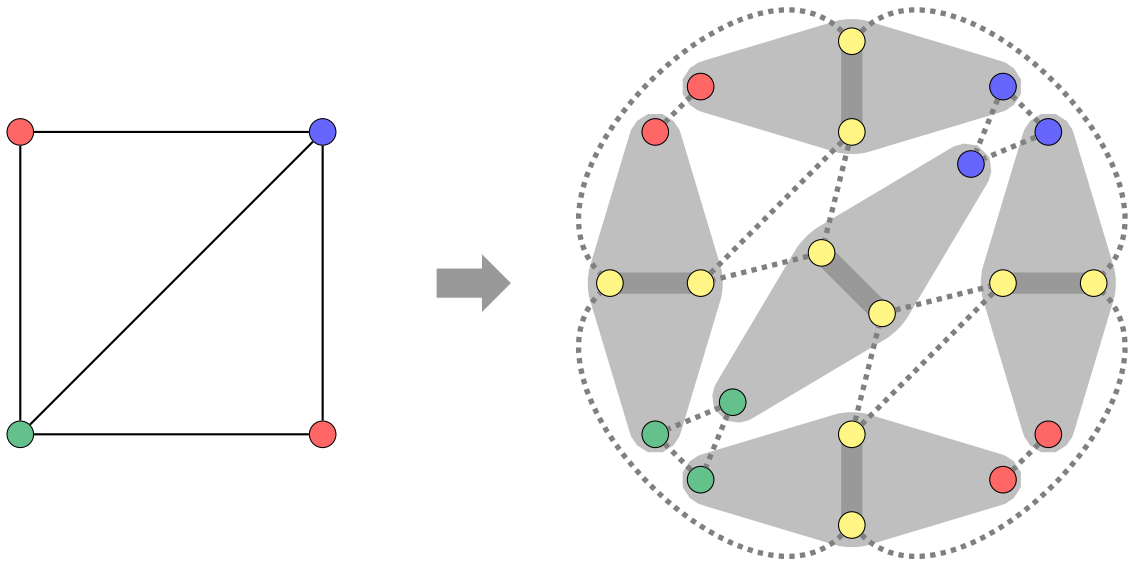


Figure 11: A graph G is shown on the left and the graph constructed by Lemma 6.4 is displayed on the right. All dashed edges represent equality edges contained in EQ. Each gray region is replaced by the gadget from Figure 12 which ensures that all colors of outgoing vertices are pairwise distinct except those connected by a thick edge which are forced to receive the same color.

6.1 Four Colors

We start by implementing the first step for $q = 4$. This is achieved by the next lemma.

Lemma 6.4. *Let G be a connected planar graph. Then there is a graph H and a set $\mathsf{EQ} \subseteq \binom{V(H)}{2}$ such that the following conditions are satisfied:*

1. $|V(H)| = 14|E(G)|$,
2. $H^{+\mathsf{EQ}} \coloneqq (V(H), E(H) \cup \mathsf{EQ})$ *is planar and has maximum degree* 3*, and*
3. G *is* 3*-colorable if and only if there is coloring* $\chi \colon V(H) \to \{1, 2, 3, 4\}$ *such that*
   (a) $\chi(u) = \chi(v)$ *for all* $uv \in \mathsf{EQ}$*, and*
   (b) $\chi(u) \neq \chi(v)$ *for all distinct* $u, v \in V(H)$ *such that* $\operatorname{dist}_H(u, v) \leq 2$.

Moreover, given the graph G, the pair (H, EQ) can be computed in polynomial time.

Proof. A visualization of the construction is given in Figure 11. Fix a planar embedding of G and let F denote the set of faces. We set

$$\begin{aligned} V(H) \coloneqq \; & \{(v, e) \mid v \in V(G), e \in E(G), v \in e\} \\ & \cup \{(f, e) \mid f \in F, e \in E(G), e \text{ is incident to } f\} \\ & \cup E(G) \times \{0, \dots, 9\}. \end{aligned}$$

Since every edge is incident to two vertices as well as two faces we get that $|V(H)| = 14|E(G)|$.

Let $v \in V(G)$ and let $e_1, \dots, e_d$ denote the indicent edges of v ordered cyclically according to the fixed embedding of G. We define

$$\mathsf{EQ}(v) \coloneqq \{(v, e_i)(v, e_{i+1}) \mid i \in [d]\} \cup \{(v, e_d)(v, e_1)\}. \tag{7}$$

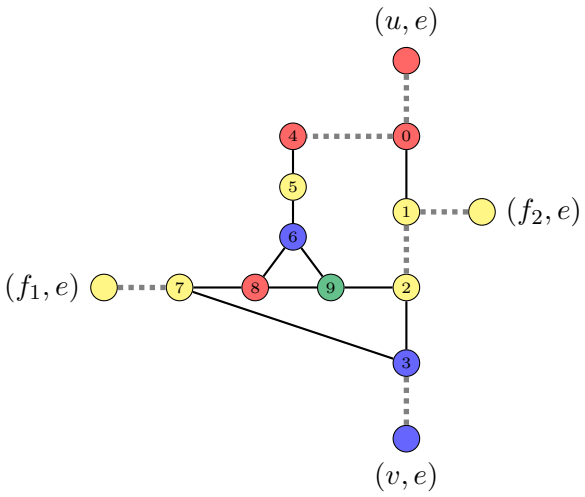


Figure 12: The gadget used in Lemma 6.4 where equality edges are represented by dashed edges. It has four outgoing vertices (u,e), (v,e), (f_1,e) and (f_2,e). The gadget enforces that (f_1,e) and (f_2,e) receive the same color, and (u,e), (v,e) and (f_1,e) receive pairwise distinct colors.

Next, let $f \in F$ be a face of G and let $e_1, \dots, e_k$ denote the indicent edges of f ordered cyclically according to the fixed embedding of G. We define

$$\mathsf{EQ}(f) \coloneqq \{(f,e_i)(f,e_{i+1}) \mid i \in [k]\} \cup \{(f,e_k)(f,e_1)\}. \tag{8}$$

Finally, let $e \in E(G)$ and suppose that $e = uv$. Also let $f_1, f_2 \in F$ denote the two faces incident to e. We set

$$\mathsf{EQ}(e) \coloneqq \{(e,0)(u,e),\ (e,0)(e,4),\ (e,1)(e,2),\ (e,1)(f_2,e),\ (e,3)(v,e),\ (e,7)(f_1,e)\}. \tag{9}$$

We also set

$$M(e) \coloneqq \{01, 23, 29, 37, 45, 56, 68, 69, 78, 89\}. \tag{10}$$

A visualization is given in Figure 12. Overall, we now define

$$E(H) \coloneqq \bigcup_{e \in E(G)} \{(e,i)(e,j) \mid ij \in M(e)\}$$

and

$$\mathsf{EQ} \coloneqq \bigcup_{v \in V(H)} \mathsf{EQ}(v) \cup \bigcup_{f \in F} \mathsf{EQ}(f) \cup \bigcup_{e \in E(G)} \mathsf{EQ}(e).$$

Clearly, $H^{+\mathsf{EQ}}$ is planar and has maximum degree 3. Also, it is easy to see that the pair (H, EQ) can be computed in polynomial time.

Suppose that G is 3-colorable via a coloring $\mu\colon V(G) \to \{1,2,3\}$. We define a coloring $\chi\colon V(H) \to \{1,2,3,4\}$ as follows. Let $e \in E(G)$ and suppose that $e = uv$. Also let $f_1, f_2 \in F$ denote the two faces incident to e. Suppose that $a \coloneqq \mu(u)$, $b \coloneqq \mu(v)$ and $\{a,b,c\} = \{1,2,3\}$. We set

- $\chi(u,e) = \chi(e,0) = \chi(e,4) = \chi(e,8) \coloneqq a$,
- $\chi(v,e) = \chi(e,3) = \chi(e,6) \coloneqq b$,
- $\chi(e,9) \coloneqq c$, and

- $\chi(f_1, e) = \chi(f_2, e) = \chi(e, 1) = \chi(e, 2) = \chi(e, 5) = \chi(e, 7) \coloneqq 4$.

It is easy to verify that the coloring χ satisfies the desired properties (see also Figures 11 and 12).

In the other direction, suppose that $\chi\colon V(H) \to \{1, 2, 3, 4\}$ is a coloring with the desired properties.

Claim 6.1. Let $e \in E(G)$ and let f_1, f_2 denote the two incident faces. Then $\chi(f_1, e) = \chi(f_2, e)$.

Proof. By Equation (9) we get that that $\chi(f_1, e) = \chi(e, 7)$ and $\chi(f_2, e) = \chi(e, 1) = \chi(e, 2)$. Also, $\mathrm{dist}_H((e, i), (e, j)) \leq 2$ for all $i, j \in \{6, 7, 8, 9\}$. This means that

$$\{\chi(e, 6), \chi(e, 7), \chi(e, 8), \chi(e, 9)\} = \{1, 2, 3, 4\}.$$

By the same argument

$$\{\chi(e, 2), \chi(e, 6), \chi(e, 8), \chi(e, 9)\} = \{1, 2, 3, 4\}.$$

Combining both equations implies that $\chi(e, 7) = \chi(e, 2)$. Overall, we get that $\chi(f_1, e) = \chi(f_2, e)$. ⌟

Combining the last claim and Equation (8) we obtain that all vertices from set $\{(f, e) \mid f \in F, e \in E(G), e \text{ is incident to } f\}$ receive the same color under χ. Without loss of generality suppose that $\chi(f, e) = 4$ for all $e \in E(G)$ and incident faces f.

Using Equation (7), for every $v \in V(G)$, there is some color $\mu(v) \in \{1, 2, 3, 4\}$ such that $\chi(v, e) = \mu(v)$ for all incident edges $e \in E(G)$. Also $\chi(v, e) \neq 4$ since $\chi(e, 1) = 4$ by Equation (9). So $\mu(v) \in \{1, 2, 3\}$ for every $v \in V(G)$. It remains to show that adjacent vertices receive different colors. So suppose that $e = uv \in E(G)$. We have that $\chi(e, 3) = \mu(v)$ using Equation (9). Also, $\mathrm{dist}_H((e, i), (e, j)) \leq 2$ for all $i, j \in \{6, 7, 8, 9\}$. This means that

$$\{\chi(e, 6), \chi(e, 7), \chi(e, 8), \chi(e, 9)\} = \{1, 2, 3, 4\}.$$

Since $\mathrm{dist}_H((e, i), (e, 3)) \leq 2$ for all $i \in \{7, 8, 9\}$, we conclude that $\chi(e, 6) = \chi(e, 3) = \mu(v)$. Finally $\mu(u) = \chi(e, 4)$ using Equation (9). Since $\mathrm{dist}_H((e, 3), (e, 4)) \leq 2$ it follows that $\mu(u) \neq \mu(v)$. □

6.2 More Than Four Colors

Next, we implement the first step for $q \geq 5$, i.e., we prove a varaint of Lemma 6.4 for $q \geq 5$. Before getting to the actual reduction, let us briefly explain why a different construction is required. Intuitively speaking, the main idea for all reductions is to enforce that the "original" vertices of the input graph can only be colored with one of three "candidate" colors. Hence, the information about which three colors are those candidates needs to distributed over the entire graph. Alternatively, it suffices to distribute the information which $q - 3$ colors are the "auxiliary" colors. For $q = 4$ this means that we only need to distribute one auxiliary color which turns out to be fairly easy. However, for larger numbers of colors, we require a more intricate distribution strategy where the idea is to forward the set of "auxiliary" colors along a cycle. In order to ensure that the set of "auxiliary" colors is available at every edge of the input graph, we start by showing the following lemma. Intuitively speaking, given a planar graph G together with an embedding of G, it constructs a cycle in the plane that crosses every edge of G exactly twice (see also Figure 13f).

Lemma 6.5. *Let G be a connected planar graph of minimum degree 3. Let G' be the graph defined via $V(G') \coloneqq V(G) \cup \vec{E}(G)$, where $\vec{E}(G) \coloneqq \{(u, v) \mid uv \in E(G)\}$, and*

$$E(G') \coloneqq \{u(u, v),\ (u, v)(v, u),\ (v, u)v \mid uv \in E(G)\}.$$

Then there is a set $E^* \subseteq \binom{\vec{E}(G)}{2}$ such that

1. *the graph* $C = (\vec{E}(G), E^*)$ *is a cycle, and*
2. *the multigraph* $G^+ = (V(G'), E(G') \cup E^*)$ *is planar via an embedding where, for every* $(u, v) \in \vec{E}(G)$*, its four incident edges* $e_1, \ldots, e_4$*, listed in the cyclic order of the embedding, are alternately contained in the sets* $E(G')$ *and* E^**.*

Moreover, given the graph G*, the set* E^* *together with the desired embedding of* G^+ *can be computed in polynomial time.*

Before diving into the proof, let us briefly clarify the last part of lemma. Consider the multigraph $G^+ = (V(G'), E(G') \cup E^*)$. Every vertex $(u, v) \in \vec{E}(G)$ in this graph has four incident edges, two of which are contained in the set $E(G')$ and the other two are being contained in E^*. The lemma guarantees that there is a planar embedding of the multigraph G^+ such that these two types of edges always alternate when looking at the cyclic order $e_1, \ldots, e_4$ induced by the embedding (see also Figure 13f).

Proof. Fix a planar embedding of G and let F denote the set of faces. Consider the graph G'' that is obtained from G by subdividing every edge once, i.e., $V(G'') \coloneqq V(G) \cup E(G)$ and

$$E(G'') \coloneqq \{ve \mid e \in E(G), v \in e\}.$$

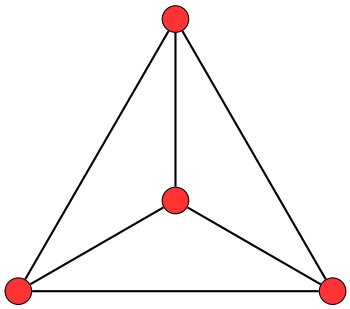
We define the set $\widetilde{E} \subseteq \binom{E(G)}{2}$ as follows. For every face $f \in F$ let $e_1, \ldots, e_k$ denote the incident edges of f ordered cyclically according to the fixed embedding of G. We add pairs $e_i e_{i+1}$, $i \in [k]$, and $e_1 e_k$ to the set $\widetilde{E}$. Since G has minimum degree 3 the graph $\widetilde{G} = (E(G), \widetilde{E})$ is 4-regular (see also Figure 13b). Hence, $\widetilde{G}$ has an Euler tour $\widetilde{e}_1, \ldots, \widetilde{e}_m$ where $\widetilde{e}_i \in \widetilde{E}$ for every $i \in [m]$.

We can naturally extend the embedding of G'' (obtained from the fixed embedding of G) to an embedding of $\widetilde{G}$. Now, consider again the Euler tour $\widetilde{e}_1, \ldots, \widetilde{e}_m$. Since $\widetilde{G}$ is 4-regular, each vertex $v \in V(\widetilde{G})$ is visited twice by the Euler, say via edges $\widetilde{e}_i, \widetilde{e}_{i+1}$ and $\widetilde{e}_j, \widetilde{e}_{j+1}$. We say that the Euler tour $\widetilde{e}_1, \ldots, \widetilde{e}_m$ is *crossing* at v if $\widetilde{e}_i, \widetilde{e}_{i+1}$ are not adjacent in the cyclic order of edges incident to v in $\widetilde{G}$ with respect to the fixed embedding of $\widetilde{G}$ (see Figure 13b). By reoredering the edges of the Euler tour, we may assume without loss of generality that $\widetilde{e}_1, \ldots, \widetilde{e}_m$ is not crossing at any vertex $v \in V(\widetilde{G})$ (see Figure 13c).

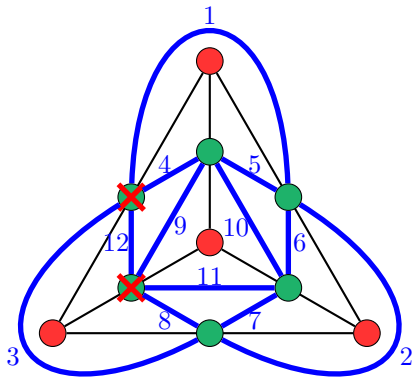
Now, since each element of $uv \in E(G)$ corresponds to two elements $(u, v), (v, u) \in \vec{E}(G)$, we can transform the Euler tour $\widetilde{e}_1, \ldots, \widetilde{e}_m$ into a cycle $\widehat{E} \subseteq \binom{\vec{E}(G)}{2}$ on $\vec{E}(G)$ in the natural way which provides a planar embedding of $\widehat{G} = (V(G'), E(G') \cup \widehat{E})$ (see Figure 13d). Note that every vertex $\vec{e} \in \vec{E}(G) \subseteq V(\widehat{G})$ has degree 4 in $\widehat{G}$ with two incident edges coming from $E(G')$ and the other two coming from $\widehat{E}$.

To complete the proof, it only remains to ensure that these edges appear alternately (see Figure 13d). This can be achieved as follows. Consider a pair $(u, v), (v, u) \in \vec{E}(G)$. Either edges from $E(G')$ and $\widehat{E}$ already appear alternately along the cyclic order associated with the embedding for both vertices (u, v) and (v, u), or this condition is violated for both of them. In the latter case, we locally modify the set $\widehat{E}$ as follows. First, we omit (u, v) from the cycle and connect its two adjacent vertices via a new edge that is added to $\widehat{E}$. Afterwards, we "pull the cycle defined by $\widehat{E}$ over the edge $uv \in E(G)$", i.e., we replace the occurrence of (v, u) by the pair $(u, v), (v, u)$. In particular, this adds $(u, v)(v, u)$ to the set $\widehat{E}$, creating a multiedge in the graph $\widehat{G}$. This multiedge can be placed on either side of the same edge $(u, v)(v, u) \in E(G')$ which means that there is always an embedding such that incident edges of (u, v) alternately come from the sets $E(G')$ and $\widehat{E}$ as desired (see Figure 13e).

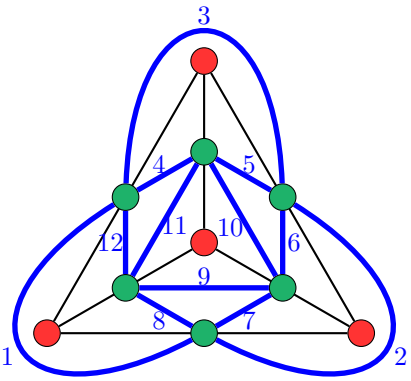
We perform this modification for all pairs $(u, v), (v, u) \in \vec{E}(G)$ to obtain the desired final outcome E^* together with an embedding of G^+ (see Figure 13f).



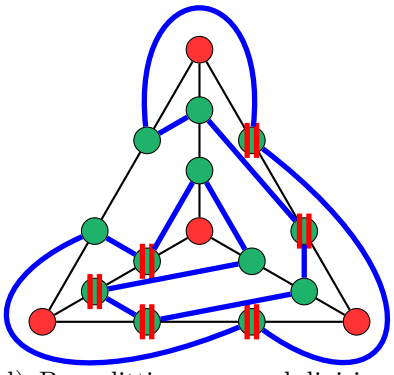
(a) The input graph G.



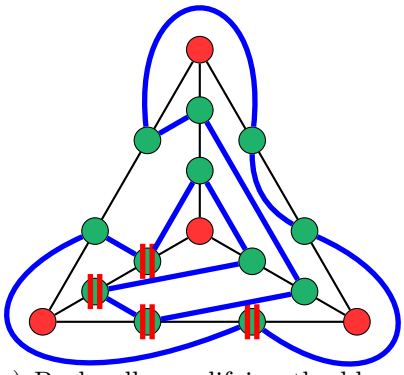
(b) We obtain G'' by subdividing every edge. The set $\widetilde{E}$ is makred in blue where the numbers $1, \ldots, 12$ describe an Euler tour of $\widetilde{G}$. The Euler tour has two "crossings" at the marked vertices.



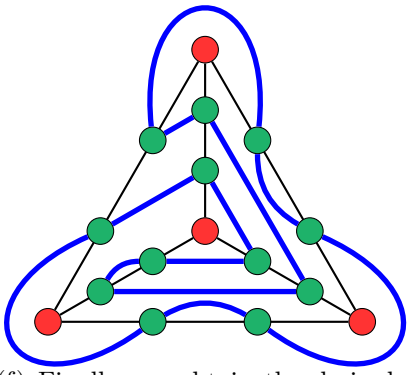
(c) By reoredering the edges on the Euler tour, we can always obtain an Euler tour without any "crossings".



(d) By splitting very subdivision vertex (the green vertices), we obtain the graph G' and a cycle $\widehat{E}$ (the blue edges) on $\vec{E}(G)$. For the marked vertices, the incident edges are not alternating between $\widehat{E}$ and $E(G')$.



(e) By locally modifying the blue edges, the constructed cycle is fixed step by step to obtain the desired outcome.



(f) Finally, we obtain the desired multigraph G^+. The edges from the set E^* are blue.

Figure 13: Visualization of the steps involved in the proof of Lemma 6.5. Given a planar graph of minimum degree 3, we eventually obtain a cycle (shown by the blue edges) that crosses every edge of the input graph exactly twice.

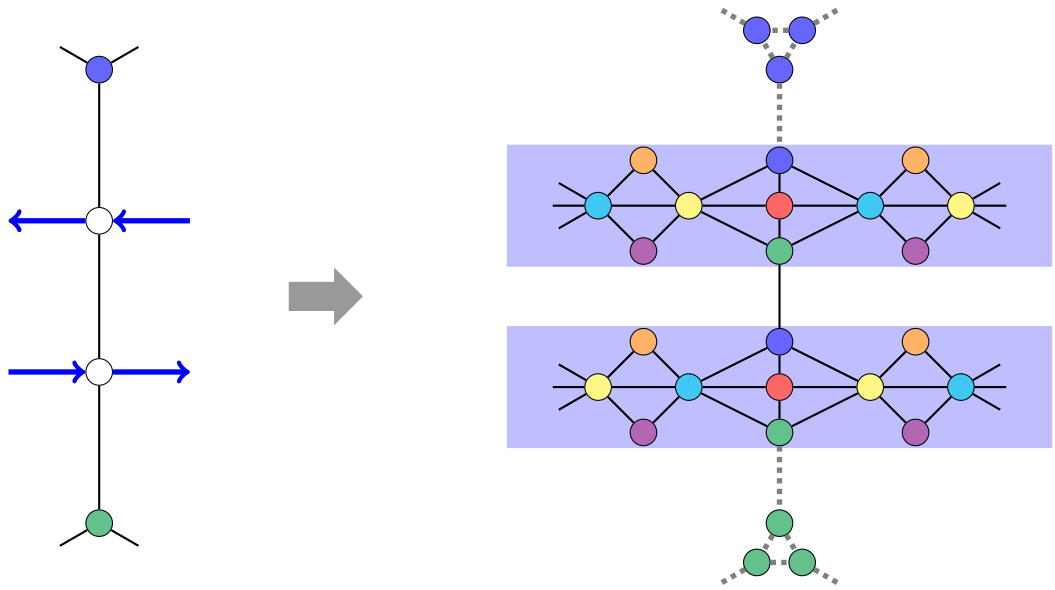


Figure 14: Visualization of the construction from Lemma 6.6 on a single edge of G. The directed cycle given by E^* is shown in blue and crosses the edge exactly twice.

We complete the proof by observing that all steps can be performed in polynomial time. □

Lemma 6.6. *Let $q \geq 5$. Let G be a connected planar graph of minimum degree 3. Then there is a graph H and a set $\mathsf{EQ} \subseteq \binom{V(H)}{2}$ such that the following conditions are satisfied:*

1. $|V(H)| = O(q \cdot |V(G)|)$,
2. $|\mathsf{EQ}| = O(|V(G)|)$,
3. $H^{+\mathsf{EQ}} \coloneqq (V(H), E(H) \cup \mathsf{EQ})$ *is planar and has maximum degree $q-1$, and*
4. *G is 3-colorable if and only if there is coloring $\chi \colon V(H) \to [q]$ such that*
   (a) *$\chi(u) = \chi(v)$ for all $uv \in \mathsf{EQ}$, and*
   (b) *$\chi(u) \neq \chi(v)$ for all distinct $u, v \in V(H)$ such that $\mathrm{dist}_H(u,v) \leq 2$.*

Moreover, given the graph G, the pair (H, EQ) can be computed in polynomial time.

Proof. Let G' be the graph defined via $V(G') \coloneqq V(G) \cup \vec{E}(G)$, where $\vec{E}(G) \coloneqq \{(u,v) \mid uv \in E(G)\}$, and

$$E(G') \coloneqq \{u(u,v),\ (u,v)(v,u),\ (v,u)v \mid uv \in E(G)\}.$$

By Lemma 6.5 there is a set $E^* \subseteq \binom{\vec{E}(G)}{2}$ such that

1. the graph $C = (\vec{E}(G), E^*)$ is a cycle, and
2. the multigraph $G^+ = (V(G'), E(G') \cup E^*)$ is planar via an embedding where, for every $(u,v) \in \vec{E}(G)$, its four incident edges $e_1, \ldots, e_4$, listed in the cyclic order of the embedding, are alternately contained in the sets $E(G')$ and E^*.

Actually, for the remainder of the proof, it is more convenient to assume that $C = (\vec{E}(G), E^*)$ is a directed cycle, i.e., let us arbitrarily direct an edge of E^* and then direct all other edges accordingly following the cycle defined by E^*.

We construct the graph H in two steps (see also Figure 14). First, we define

$$A \coloneqq \vec{E}(G) \times \{1, 2, 3\}$$

and

$$B \coloneqq E^* \times \{4, \dots, q\}.$$

Also, we define

$$\begin{aligned} E_{A,B} \coloneqq \; & \{(\vec{e}, 1)(\vec{e}, 2), (\vec{e}, 2)(\vec{e}, 3) \mid \vec{e} \in \vec{E}(G)\} \\ & \cup \{((u, v), 3)((v, u), 3) \mid uv \in E(G)\} \\ & \cup \{(\vec{e}_1, i)((\vec{e}_1, \vec{e}_2), 4), (\vec{e}_2, i)((\vec{e}_1, \vec{e}_2), 5) \mid (\vec{e}_1, \vec{e}_2) \in E^*, i \in \{1, 2, 3\}\} \\ & \cup \{((\vec{e}_1, \vec{e}_2), 4)((\vec{e}_1, \vec{e}_2), 5) \mid ((\vec{e}_1, \vec{e}_2)) \in E^*\} \\ & \cup \{((\vec{e}_1, \vec{e}_2), 4)((\vec{e}_1, \vec{e}_2), i), ((\vec{e}_1, \vec{e}_2), 5)((\vec{e}_1, \vec{e}_2), i) \mid (\vec{e}_1, \vec{e}_2) \in E^*, i \in \{6, \dots, q\}\} \end{aligned}$$

Now consider the graph $H' = (A \cup B, E_{A,B})$.

Claim 6.2. Let χ' be a q-coloring of $(H')^2$. Then

$$\{\chi'(\vec{e}_1, i) \mid i \in \{1, 2, 3\}\} = \{\chi'(\vec{e}_2, i) \mid i \in \{1, 2, 3\}\}$$

for all $\vec{e}_1, \vec{e}_2 \in \vec{E}(G)$. Moreover,

$$\chi'((u, v), 1) \neq \chi'((v, u), 1)$$

for all $uv \in E(G)$.

Proof. Suppose that $e^* = (\vec{e}_1, \vec{e}_2) \in E^*$. Then the set

$$(\{\vec{e}_1\} \times \{1, 2, 3\}) \cup \{e^*\} \times \{4, \dots, q\}$$

forms a clique in $(H')^2$ and all q colors need to be used under χ'. The same statement holds for the

$$(\{\vec{e}_2\} \times \{1, 2, 3\}) \cup \{e^*\} \times \{4, \dots, q\}.$$

It follows that

$$\{\chi'(\vec{e}_1, i) \mid i \in \{1, 2, 3\}\} = \{\chi'(\vec{e}_2, i) \mid i \in \{1, 2, 3\}\}$$

Since $C = (\vec{E}(G), E^*)$ forms a cycle the first statement follows.

For the second statement suppose that $uv \in E(G)$. Then $\chi'((u, v), 3) \neq \chi'((v, u), 3)$ and $\chi'((u, v), 2) \neq \chi'((v, u), 3)$. In combination with the first statement it follows that

$$\chi'((u, v), 3) = \chi'((v, u), 1)$$

and hence,

$$\chi'((u, v), 1) \neq \chi'((v, u), 1).$$

⌟

We say that a q-coloring χ' of $(H')^2$ is *good* if $\chi'((u, v), 1) = \chi'((u, w), 1)$ for all $uv, uw \in E(G)$.

Claim 6.3. Let χ' be a good q-coloring of $(H')^2$. Then G is 3-colorable.

Proof. Let $\mu\colon V(G) \to \{1,2,3\}$ be define in such a way that $\mu(u) = \chi'((u,v),1)$ for all $uv \in E(G)$. Observe that this is well-defined since χ' is good. Now, $\mu(u) \neq \mu(v)$ for all $uv \in E(G)$ by Claim 6.2. ⌟

Claim 6.4. Let $\mu\colon V(G) \to \{1,2,3\}$ be a 3-coloring of G. Then there is a q-coloring χ' of $(H')^2$ such that

$$\chi'((u,v),1) = \mu(u)$$

for all $(u,v) \in \vec{E}(G)$.

Proof. We define $\chi'(e^*,i) \coloneqq i$ for all $(e^*,i) \in B$. Let $uv \in E(G)$. Then $\mu(u) \neq \mu(v)$. Let $c \in \{1,2,3\}$ be the unique third color that is distinct from $\mu(u)$ and $\mu(v)$. We set

- $\chi'((u,v),1) \coloneqq \mu(u)$,
- $\chi'((u,v),2) \coloneqq c$, and
- $\chi'((u,v),3) \coloneqq \mu(v)$.

It can be easily verified that χ' is a q-coloring of $(H')^2$. ⌟

Observe that the coloring defined in the last claim is a good q-coloring of $(H')^2$. Hence, G is 3-colorable if and only if $(H')^2$ has a good q-coloring. So to complete the proof, we use equality edges to ensure that every valid coloring of H' is indeed good. We define the graph H via

$$V(H) \coloneqq V(H') \cup \vec{E}(G)$$

and

$$E(H) \coloneqq E(H').$$

For $u \in V(G)$ let $v_1, \dots, v_d$ denote its neighbors ordered cyclically according to the embedding of G (which is inherited from the embedding of G^+ in the natural way). We define

$$\mathsf{EQ}(u) \coloneqq \{(u,v_i)(u,v_{i+1}) \mid i \in [d-1]\} \cup \{(u,v_1)(u,v_d)\}$$

and

$$\mathsf{EQ} \coloneqq \{(u,v)((u,v),1) \mid (u,v) \in \vec{E}(G)\} \cup \bigcup_{u \in V(G)} \mathsf{EQ}(u).$$

This completes the construction. We have

$$|V(H)| = |\vec{E}(G)| + |A| + |B| = |\vec{E}(G)| + 3|\vec{E}(G)| + (q-3)|\vec{E}(G)| = (q+1)|\vec{E}(G)| = O(q|V(G)|)$$

since G is planar. Similarly,

$$|\mathsf{EQ}| = 2|\vec{E}(G)| = O(|V(G)|).$$

We have $\deg_{H^{+\mathsf{EQ}}}(\vec{e}) = 3$ and $\deg_{H^{+\mathsf{EQ}}}(\vec{e},i) = 4$ for all $i \in \{1,2,3\}$ and $\vec{e} \in \vec{E}(G)$, and $\deg_{H^{+\mathsf{EQ}}}(e^*,i) = q-1$ for all $e^* \in E^*$ and $i \in \{4,\dots,q\}$. Since $q \geq 5$ it follows that $H^{+\mathsf{EQ}}$ has maximum degree $q-1$. Also, $H^{+\mathsf{EQ}}$ is planar using the properties guaranteed by Lemma 6.5. Finally, the last condition follows from Claim 6.4 and 6.3. Observe that the equality edges ensure that any valid coloring of $(H')^2$ has to be good.

Finally, it is easy to see that (H,EQ) can be computed in polynomial time using that the set E^* can be computed in polynomial time by Lemma 6.5. □

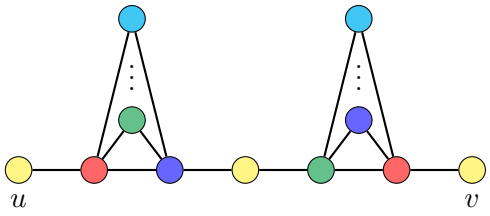


Figure 15: The equality gadget ensures that u and v need to be assigned the same color in any square q-coloring. Moreover, as long as u (resp. v) has at most $q-2$ further outside neighbors, every valid coloring of the outside vertices can be extended into the gadget.

6.3 Removing Equality Edges

Having completed the first step of the reduction chain, it only remains to replace the equality edges with suitable gadgets. Here, we can use ideas that already appeared in Section 4. However, let us point out that we can not use exactly the same gadgets as in Section 4 since we are restricted in the number of colors that are available. Indeed, suppose $uv \in \mathsf{EQ}$ is an equality edge. The vertex u may be adjacent (via normal edges) to $q-2$ further vertices and we need to ensure that the equality gadget does not interfere with coloring those vertices. In Section 4, we resolved this issue by introducing additional "neutral" colors which is not possible for the present reduction. Instead, we use a slightly more complicated gadget (see Figure 15) which, in essence, chains together two equality gadgets (using the construction from Section 4), to ensure that every valid coloring of the original vertices can be extended to the gadget vertices.

Lemma 6.7. *Let $q \geq 4$. There is a polynomial-time algorithm that, given a connected planar graph G, constructs a planar graph H such that*

1. *$|V(H)| = O(q \cdot |V(G)|)$, and*
2. *G is 3-colorable if and only if H^2 is q-colorable.*

Proof. By repeatedly removing vertices of degree at most 2 we may assume without loss of generality that G has minimum degree 3 (if G is 2-degenerate we return a trivial YES-instance).

By Lemmas 6.4 and 6.6 there is a graph H' and a set $\mathsf{EQ} \subseteq \binom{V(H')}{2}$ such that

1. $|V(H')| = O(q \cdot |V(G)|)$,
2. $|\mathsf{EQ}| = O(|V(G)|)$,
3. $(H')^{+\mathsf{EQ}} \coloneqq (V(H'), E(H') \cup \mathsf{EQ})$ is planar and has maximum degree $q-1$, and
4. G is 3-colorable if and only if there is coloring $\chi' \colon V(H') \to \{1, \ldots, q\}$ such that
   (a) $\chi'(u) = \chi'(v)$ for all $uv \in \mathsf{EQ}$, and
   (b) $\chi'(u) \neq \chi'(v)$ for all distinct $u, v \in V(H')$ such that $\mathrm{dist}_{H'}(u, v) \leq 2$.

We obtain H from H' by inserting an equality gadget between all pairs $uv \in \mathsf{EQ}$ as follows (see also Figure 15). Formally, we define

$$V(H) \coloneqq V(H') \cup (\mathsf{EQ} \times \{1, \ldots, 2q-1\})$$

and

$$E(H) \coloneqq E(H') \cup \{u(uv,1),\ v(uv,q+2) \mid uv \in \mathsf{EQ}\} \cup \{(uv,i)(uv,j) \mid ij \in M, uv \in \mathsf{EQ}\}$$

where

$$\begin{aligned} M = \ & \{\{1,q-1\},\{q-1,q\},\{q,q+1\},\{q+1,2q-1\}\} \\ & \cup \{\{i,1\},\{i,q-1\},\{q+i,q+1\},\{q+i,2q-1\} \mid i \in \{2,\dots,q-2\}\} \end{aligned}$$

Clearly, H can be computed in polynomial time and H is planar since $(H')^{+\mathsf{EQ}}$ is planar. Also,

$$|V(H)| = |V(H')| + (2q-1) \cdot |\mathsf{EQ}| = O(q \cdot |V(G)|).$$

So it remains to prove that G is 3-colorable if and only if H^2 is q-colorable. We start with two basic observations.

Claim 6.5. Let $u, v \in V(H')$. Then $\mathrm{dist}_{H'}(u,v) \leq 2$ if and only if $\mathrm{dist}_H(u,v) \leq 2$.

Proof. This is immediately clear by construction since the inserted gadgets do not allow for any shortcuts for distances at most 2. ⌟

Claim 6.6. Let χ be a q-coloring of H^2 and let $uv \in \mathsf{EQ}$. Then $\chi(u) = \chi(v)$.

Proof. Looking at the inserted gadget it is easy to see that $\chi(u) = \chi(uv,q) = \chi(v)$. ⌟

Now, first suppose that H^2 is q-colorable via a coloring $\chi\colon V(H) \to \{1,\dots,q\}$. Let $\chi' \coloneqq \chi|_{V(H')}$ be the restriction to $V(H')$. Then $\chi'(u) = \chi'(v)$ for all $uv \in \mathsf{EQ}$ by Claim 6.6. Also, for $u, v \in V(H')$ such that $\mathrm{dist}_{H'}(u,v) \leq 2$, we get that $\mathrm{dist}_H(u,v) \leq 2$ by Claim 6.5 which implies that $\chi'(u) \neq \chi'(v)$. So G is 3-colorable by Lemmas 6.4 and 6.6.

In the other direction, suppose that G is 3-colorable. By Lemmas 6.4 and 6.6 there is a coloring $\chi'\colon V(H') \to \{1,\dots,q\}$ such that

(a) $\chi'(u) = \chi'(v)$ for all $uv \in \mathsf{EQ}$, and

(b) $\chi'(u) \neq \chi'(v)$ for all distinct $u, v \in V(H')$ such that $\mathrm{dist}_{H'}(u,v) \leq 2$.

We claim that we can extend χ' to a q-coloring χ of H^2. Formally, we set $\chi(v) \coloneqq \chi'(v)$ for all $v \in V(H')$. Then $\chi(u) \neq \chi(v)$ for all $u, v \in V(H')$ such that $\mathrm{dist}_H(u,v) \leq 2$ by Claim 6.5. Now consider some $uv \in \mathsf{EQ}$. Let $a \coloneqq \chi'(u) = \chi'(v)$. We can color the vertices of the inserted gadget as follows. First, we set $\chi(uv,q) \coloneqq a$. Since H' has maximum degree $q-1$ we get that $|N_H[u]| \leq q$. This means there is some color $b \in \{1,\dots,q\}$ that is not used so far in $N_H[u]$ (we have $(uv,1) \in N_H[u]$ and this vertex is not colored yet). We set $\chi(uv,1) \coloneqq b$. Similarly, there is some color $c \in \{1,\dots,q\}$ that is not used so far in $N_H[v]$, and we set $\chi(uv,2q-1) \coloneqq c$. The remaining vertices of the gadget are not connected to vertices outside the gadget in H^2 and we can easily complete the coloring within the gadget. Indeed, there are $q-2$ colors remaining for each of the two sets $\{(uv,i) \mid i \in \{2,\dots,q-1\}\}$ and $\{(uv,i) \mid i \in \{q+1,\dots,2q-2\}\}$ and we only need to ensure that $\chi(uv,q-1) \neq \chi(uv,q+1)$. It is easy to see that this is always possible. □

Finally, Theorem 6.1 directly follows from Lemma 6.7 and Theorem 6.3.

References

[1] Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *European Symposium on Algorithms*, pages 1–12. Springer, 2014.

[2] Geir Agnarsson and Magnús M. Halldórsson. Coloring powers of planar graphs. *SIAM Journal on Discrete Mathematics*, 16(4):651–662, 2003.

[3] Noga Alon and Bojan Mohar. The chromatic number of graph powers. *Combin. Probab. Comput.*, 11(1):1–10, 2002.

[4] Julien Bensmail, Marthe Bonamy, and Hervé Hocquard. Strong edge coloring sparse graphs. *Electron. Notes Discret. Math.*, 49:773–778, 2015.

[5] Jean R. S. Blair and Fredrik Manne. An efficient self-stabilizing distance-2 coloring algorithm. *Theor. Comput. Sci.*, 444:28–39, 2012.

[6] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[7] Hans L. Bodlaender. Treewidth: Structure and algorithms. In Giuseppe Prencipe and Shmuel Zaks, editors, *Structural Information and Communication Complexity, 14th International Colloquium, SIROCCO 2007, Castiglioncello, Italy, June 5-8, 2007, Proceedings*, volume 4474 of *Lecture Notes in Computer Science*, pages 11–25. Springer, 2007.

[8] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (meta) kernelization. *J. ACM*, 63(5):44:1–44:69, 2016.

[9] Nicolas Bousquet, Lucas de Meyer, Quentin Deschamps, and Théo Pierron. Square coloring planar graphs with automatic discharging, 2022.

[10] Yuehua Bu and Xubo Zhu. An optimal square coloring of planar graphs. *J. Comb. Optim.*, 24(4):580–592, 2012.

[11] Gerard J. Chang and David Kuo. The l(2, 1)-labeling problem on graphs. *SIAM J. Discret. Math.*, 9(2):309–316, 1996.

[12] Lily Chen, Kecai Deng, Gexin Yu, and Xiangqian Zhou. Strong edge-coloring for planar graphs with large girth. *Discrete Math.*, 342(2):339–343, 2019.

[13] Joanna Chybowska-Sokół, Konstanty Junosza-Szaniawski, and Paweł Rzążewski. L(2,1)-labeling of disk intersection graphs. *Discrete Applied Mathematics*, 277:71–81, 2020.

[14] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

[15] Zdeněk Dvořák, Daniel Král', Pavel Nejedlý, and Riste Škrekovski. Coloring squares of planar graphs with girth six. *European Journal of Combinatorics*, 29(4):838–849, 2008. Homomorphisms: Structure and Highlights.

[16] Jeff Erickson, Shripad Thite, and David P. Bunde. Distance-2 edge coloring is np-complete, 2005.

[17] R. J. Faudree, R. H. Schelp, A. Gyárfás, and Zs. Tuza. The strong chromatic index of graphs. volume 29, pages 205–211. 1990. Twelfth British Combinatorial Conference (Norwich, 1989).

[18] Jirí Fiala, Tomas Gavenciak, Dusan Knop, Martin Koutecký, and Jan Kratochvíl. Parameterized complexity of distance labeling and uniform channel assignment problems. *Discret. Appl. Math.*, 248:46–55, 2018.

[19] Jirí Fiala, Petr A. Golovach, and Jan Kratochvíl. Distance constrained labelings of graphs of bounded treewidth. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 360–372. Springer, 2005.

[20] Jirí Fiala, Petr A. Golovach, and Jan Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theor. Comput. Sci.*, 412(23):2513–2523, 2011.

[21] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[22] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.

[23] Pierre Fraigniaud, Magnús M. Halldórsson, and Alexandre Nolin. Distributed testing of distance-k colorings. In Andrea Werneck Richa and Christian Scheideler, editors, *Structural Information and Communication Complexity - 27th International Colloquium, SIROCCO 2020, Paderborn, Germany, June 29 - July 1, 2020, Proceedings*, volume 12156 of *Lecture Notes in Computer Science*, pages 275–290. Springer, 2020.

[24] Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 662–673. IEEE Computer Society, 2018.

[25] Petr A. Golovach, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Cliquewidth III: the odd case of graph coloring parameterized by cliquewidth. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 262–273. SIAM, 2018.

[26] Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *Journal of Combinatorial Theory, Series B*, 99(1):218–228, 2009.

[27] G. Halász and V. T. Sós, editors. *Irregularities of partitions*, volume 8 of *Algorithms and Combinatorics: Study and Research Texts*. Springer-Verlag, Berlin, 1989. Papers from the meeting held in Fertőd, July 7–11, 1986.

[28] Magnús M. Halldórsson, Fabian Kuhn, and Yannic Maus. Distance-2 coloring in the CONGEST model. In Yuval Emek and Christian Cachin, editors, *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, pages 233–242. ACM, 2020.

[29] Dávid Hudák, Borut Lužar, Roman Soták, and Riste Škrekovski. Strong edge-coloring of planar graphs. *Discrete Mathematics*, 324:41–49, 2014.

[30] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 653–662. IEEE, 1998.

[31] Daniel Král and Riste Škrekovski. A theorem about the channel assignment problem. *SIAM J. Discrete Math.*, 16(3):426–437, 2003.

[32] Chia-Lin Lee and Tzong-Jye Liu. A Self-Stabilizing Distance-2 Edge Coloring Algorithm. *The Computer Journal*, 57(11):1639–1648, 07 2013.

[33] Bingkai Lin. Constant approximating k-clique is w[1]-hard. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1749–1756. ACM, 2021.

[34] Errol L. Lloyd and Subramanian Ramanathan. On the complexity of distance-2 coloring. In Waldemar W. Koczkodaj, Peter E. Lauer, and Anestis A. Toptsis, editors, *Computing and Information - ICCI'92, Fourth International Conference on Computing and Information, Toronto, Ontario, Canada, May 28-30, 1992, Proceedings*, pages 71–74. IEEE Computer Society, 1992.

[35] Dániel Marx. Can you beat treewidth? *Theory Comput.*, 6(1):85–112, 2010.

[36] S. Thomas McCormick. Optimal approximation of sparse hessians and its equivalence to a graph coloring problem. *Math. Program.*, 26(2):153–171, 1983.

[37] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins series in the mathematical sciences. Johns Hopkins University Press, 2001.

[38] Michael Molloy and Bruce Reed. A bound on the strong chromatic index of a graph. *J. Combin. Theory Ser. B*, 69(2):103–109, 1997.

[39] Michael Molloy and Mohammad R. Salavatipour. A bound on the chromatic number of the square of a planar graph. *Journal of Combinatorial Theory, Series B*, 94(2):189–213, 2005.

[40] Satyabrata Paul, Madhumangal Pal, and Anita Pal. A linear time algorithm to compute square of interval graphs and their colouring. *AKCE International Journal of Graphs and Combinatorics*, 13(1):54–64, 2016.

[41] Ian C. Ross and Frank Harary. The square of a tree. *Bell System Technical Journal*, 39(3):641–647, 1960.

[42] Paul D Seymour and Robin Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33, 1993.

[43] Larry J. Stockmeyer. Planar 3-colorability is polynomial complete. *SIGACT News*, 5(3):19–25, 1973.

[44] Carsten Thomassen. The square of a planar cubic graph is 7-colorable. *Journal of Combinatorial Theory, Series B*, 128:192–218, 2018.

[45] J. van den Heuvel, R. A. Leese, and M. A. Shepherd. Graph labeling and radio channel assignment. *J. Graph Theory*, 29(4):263–283, 1998.

[46] Jan van den Heuvel and Sean McGuinness. Coloring the square of a planar graph. *J. Graph Theory*, 42(2):110–124, 2003.

[47] Wei-Fan Wang and Ko-Wei Lih. Labeling planar graphs with conditions on girth and distance two. *SIAM J. Discrete Math.*, 17(2):264–275, 2003.

[48] Gerd Wegener. Graphs with given diameter and a coloring problem. Technical report, University of Dortmund, 1977.

[49] X. Zhou, Y. Kanari, and T. Nishizeki. Generalized vertex-colorings of partial k-trees. *IEICE Transactions*, E83–A(4):671–678, 2000.