On the complexity of restoring corrupted colorings

Marzio De Biasi1 · Juho Lauri2,3

Published online: 30 August 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract
In the $r$-Fix problem, we are given a graph $G$, a (non-proper) vertex-coloring $c : V(G) \rightarrow [r]$, and a positive integer $k$. The goal is to decide whether a proper $r$-coloring $c'$ is obtainable from $c$ by recoloring at most $k$ vertices of $G$. Recently, Junosza-Szaniawski et al. (in: SOFSEM 2015: theory and practice of computer science, Springer, Berlin, 2015) asked whether the problem has a polynomial kernel parameterized by the number of recolorings $k$. In a full version of the manuscript, the authors together with Garnero and Montealegre, answered the question in the negative: for every $r \geq 3$, the problem $r$-Fix does not admit a polynomial kernel unless $\text{NP} \subseteq \text{coNP}/\text{poly}$. Independently of their work, we give an alternative proof of the theorem. Furthermore, we study the complexity of $r$-Swap, where the only difference from $r$-Fix is that instead of $k$ recolorings we have a budget of $k$ color swaps. We show that for every $r \geq 3$, the problem $r$-Swap is $\text{W}[1]$-hard whereas $r$-Fix is known to be FPT. Moreover, when $r$ is part of the input, we observe both Fix and Swap are $\text{W}[1]$-hard parameterized by the treewidth of the input graph. We also study promise variants of the problems, where we are guaranteed that a proper $r$-coloring $c'$ is indeed obtainable from $c$ by some finite number of swaps. For instance, we prove that for $r = 3$, the problems $r$-Fix-Promise and $r$-Swap-Promise are $\text{NP}$-hard for planar graphs. As a consequence of our reduction, the problems cannot be solved in $2^{o(\sqrt{n})}$ time unless the Exponential Time Hypothesis fails.

Keywords Graph coloring · Computational complexity · Parameterized complexity · Combinatorial reconfiguration · Local search

Work partially supported by the Emil Aaltonen Foundation (J.L).

✉ Juho Lauri
juho.lauri@tut.fi

1 Treviso, Italy
2 Tampere University of Technology, Tampere, Finland
3 Present Address: Nokia Bell Labs, Dublin, Ireland

Springer
1 Introduction

Computational models are sometimes too idealized and do not capture all information available or relevant to a problem. Moreover, in a dynamic world, constraints change over time due to more information becoming available. A problem arising frequently in practice in e.g., scheduling (Marx 2004) is graph coloring. An assignment of \( r \) colors to the vertices of a graph \( G = (V, E) \) is an \( r \)-coloring. Formally, an \( r \)-coloring \( c : V \to [r] \) is said to be proper if \( c(u) \neq c(v) \) for every \( uv \in E \). In the graph coloring problem, the goal is to find the smallest \( r \) for which a graph is \( r \)-colorable. This quantity is known as the chromatic number of \( G \), and is denoted by \( \chi(G) \). Graph coloring is one of the most central problems in discrete mathematics and optimization. For a general introduction to the topic, we refer the reader to the book (Jensen and Toft 2011).

Suppose we have a graph \( G \) for which we have computed a proper vertex-coloring using \( \chi(G) \) colors. Then due to constraints changing, a malicious adversary, or e.g., a system failure, the coloring is corrupted by redistributing the vertex colors over the graph \( G \). How hard is it to restore the corrupted coloring to some optimal proper coloring of \( G \)? Clark et al. (2006) introduced and investigated this problem under the name chromatic villainy. For restoring a corrupted coloring, they used the following operation called a swap. A swap between two distinct vertices \( u \) and \( v \) is the operation of assigning to \( u \) the color that appears on \( v \), and vice versa. For convenience, we might sometimes denote the swap between \( u \) and \( v \) as \( u \leftrightarrow v \). Formally, let \( c \) be a vertex-coloring of a graph \( G \). The villainy of \( c \), denoted by \( B(c) \), is the minimum number of swaps to be performed to transform \( c \) into some proper vertex-coloring of \( G \). The quantity \( B(c) \) can also be seen as the minimum number of recolorings with the constraint that each color used in the new coloring \( c' \) must be used the same number of times as in \( c \). In addition to the graph-theoretic viewpoint, there has been interest in the computational aspects of chromatic villainy (West 2013).

Recently, Junosza-Szaniawski et al. (2015) studied a problem they call \( r \)-Fix. In the problem, we are given a graph \( G = (V, E) \), a non-proper \( r \)-coloring \( c \) of \( V(G) \), and an integer \( k \). The task is to decide whether a proper vertex-coloring \( c' \) is obtainable from \( c \) using at most \( k \) vertex recolorings. The authors observe the problem is \( \text{NP} \)-complete, and give several positive complexity results as well. In particular, using the framework of Björklund et al. (2009) they obtain a \( O^*(2^n) \)-time and exponential space algorithm, where \( n \) is the number of vertices in the input graph. Furthermore, they show that for any fixed \( r \), the problem is fixed-parameter tractable (FPT) parameterized by the number of recolorings \( k \). Finally, in the same paper, the authors show that for graphs of treewidth \( t \), the problem can be solved in \( O(nrt^{t+2}) \) time. For a discussion on several related reoptimization and reconfiguration problems, we refer the reader to Junosza-Szaniawski et al. (2015). We also note that their results are considerably expanded in a full version of the manuscript (Garnero et al. 2018).

The main difference between \( r \)-Fix and chromatic villainy as defined by Clark et al. (2006) is the basic operation used. In \( r \)-Fix it is a recoloring, whereas in chromatic villainy it is a swap. For this reason, we shall refer to the computational problem arising from chromatic villainy as \( r \)-Swap. That is, the input to \( r \)-Swap is exactly the same as it is in \( r \)-Fix, but instead of \( k \) recolorings we have a budget of \( k \) swaps. Strictly speaking, there is another difference. In fact, in chromatic villainy, the corrupted vertex-coloring
is not an arbitrary one, but has an additional property that could possibly be exploited. Namely, the property is that by using some finite number of swaps, a proper vertex-coloring is indeed obtainable from the given one. Additional promises of properties of a problem are captured by the notion of a promise problem. Promise problems were introduced and studied by Even et al. (1984), and they have several applications (for a survey, see Goldreich 2006). In fact, it seems fair to argue that promise problems model real-world problems more accurately. Indeed, Goldreich (2006) writes: “... I contend that almost all readers refer to this notion when thinking about computational problems, although they may be often unaware of this fact”. Motivated by these facts, we also separate two additional problems, $r$-Fix-Promise and $r$-Swap-Promise, for which the input is guaranteed to satisfy additional properties (a precise definition is provided in Sect. 2).

In this context, it is natural to investigate whether a hard problem becomes easy when the set of instances is restricted. A priori, it is unknown how the addition of a promise affects the computational complexity of a problem. For example, it is hard to decide whether a graph contains a Hamiltonian cycle, even if we are promised it contains at most one such cycle (Johnson 1985). In the other direction, one can efficiently find a satisfying assignment for an $n$-variable SAT formula that promises to have at least $2^n/n$ satisfying assignments. However, as shown by Valiant and Vazirani (1986), it is hard to find an assignment even when the formula is promised to have exactly one solution.

**Motivation** The problems $r$-Fix and $r$-Swap are tightly related to local search which is a core technique in solving combinatorial optimization and operations research problems in practice. In local search, one aims to improve upon the current solution by replacing it with a better solution in its neighborhood. Specifically, the neighborhood is defined by the set of allowed operations that modify the current solution. Plausibly, the larger the neighborhood, the less likely is the local search to get stuck in a local optimum. On the other hand, the allowed operations should not be too demanding to compute. In fact, there has been significant interest in applying methods from parameterized complexity to analyzing local search procedures (see e.g., de Berg et al. 2016; Fellows et al. 2012; Khuller et al. 2003; Krokhin and Marx 2012; Szeider 2011).

The studied problems are also related to combinatorial reconfiguration problems, in particular $r$-coloring reconfiguration (Bonsma et al. 2014; Johnson et al. 2014; Wrochna 2018). In this problem, we are given two proper $r$-colorings for a graph, and asked whether one can be transformed into the other by changing one color at a time, maintaining a proper coloring throughout. We argue that in the context of local search, the property of maintaining a proper $r$-coloring at each step can be relaxed: we are only interested in eventually arriving at a solution. To further motivate the use of promise conditions, we remark that there are $r$-coloring problems in which we know the sizes of the color classes (if an $r$-coloring exists). These include well-known problems arising from coding theory, such as partitioning of the $n$-dimensional Hamming space into binary codes with certain properties (Östergård 2004).

**Our results** We continue the investigation of the complexity of restoring corrupted colorings. Specifically, we further study the complexity of $r$-Fix, under different basic operations and/or promise conditions.
For Sect. 3, our main result is that for any fixed \( r \geq 3 \), the problem \( r\text{-SWAP} \) is \( \mathsf{W}[1] \)-hard parameterized by the number of swaps \( k \). Moreover, the same is true for \( r\text{-SWAP- PROMISE} \). This should be contrasted with the positive FPT result of Junosza-Szaniawski et al. (2015) for \( r\text{-FIX} \). In addition, we observe both problems \( r\text{-FIX} \) and \( r\text{-SWAP} \) become \( \mathsf{W}[1] \)-hard parameterized by the treewidth of the input graph when the number of colors \( r \) is part of the input. The constructions exhibit gadget ideas we use for the sections to follow.

In Sect. 4, we prove that under plausible complexity assumptions, \( r\text{-FIX} \) has no polynomial kernel parameterized by the number of recolorings \( k \), for every \( r \geq 3 \). We stress that while mentioned as an open problem in Junosza-Szaniawski et al. (2015), the question was subsequently answered by Garnero, Junosza-Szaniawski, Liedloff, Montealegre, and Rzążewski in a full version Garnero et al. (2018) of Junosza-Szaniawski et al. (2015). Our result was obtained independently of their work, and uses slightly different ideas.

Finally, in Sect. 5, we consider the complexity of the promise variants of the problems (see Sect. 2 for precise definitions). We show that for \( r = 3 \), both \( r\text{-SWAP- PROMISE} \) and \( r\text{-FIX- PROMISE} \) are \( \mathsf{NP} \)-hard for planar graphs. Moreover, the problems cannot be solved in \( 2^{o(\sqrt{n})} \) time unless the Exponential Time Hypothesis fails. On the positive side, using known results, we derive an algorithm for the problem working in \( 2^{O(\sqrt{n})} \) time.

2 Preliminaries

All graphs in this paper are simple and undirected. For graph-theoretic notion not defined here, we refer the reader to Diestel (2010). We write \([n]\) to denote the set \([1, 2, \ldots, n]\).

2.1 Promises and problem statements

A promise problem is a generalization of a decision problem, where the input is guaranteed to belong to a restricted subset among all possible inputs (Goldreich 2008).

Definition 1 (Promise problem) A promise problem is a pair of disjoint sets of strings \((S_Y, S_N)\), and their union \(S_Y \cup S_N\) is called the promise set. An algorithm \(A\) decides a promise problem if for every \(x \in S_Y\), \(A(x) = 1\) and for every \(x \in S_N\), \(A(x) = 0\); for strings that do not belong to the promise set \(x \notin S_Y \cup S_N\) the algorithm \(A\) must halt, but can answer arbitrarily.

A promise problem is in \( \mathsf{PromiseNP} \), the promise extension of \( \mathsf{NP} \), if there exists a polynomial \(p\) and a polynomial-time verifier \(V\) such that for every \(x \in S_Y\) there exists \(y\) of length at most \(p(|x|)\) such that \(V(x, y) = 1\) and for every \(x \in S_N\) and every \(y\) it holds that \(V(x, y) = 0\). For a more comprehensive treatment, we refer the reader to Goldreich (2008).

We are then ready to formally define the problems studied in this work. For the promise variants, a coloring \(c'\) is said to be a permutation of a proper vertex-coloring
c if $c'$ can be obtained from $c$ by a finite number of swaps. In other words, the sizes of the color classes of $c'$ match those of an optimal proper coloring.

**r-Fix**

**Instance:** A graph $G = (V, E)$, an $r$-coloring $c : V \rightarrow [r]$, and a positive integer $k$.

**Question:** Can $c$ be made into a proper $r$-coloring of $G$ using at most $k$ recolorings?

**r-Fix- Promise**

**Instance:** A graph $G = (V, E)$, an $r$-coloring $c : V \rightarrow [r]$, and a positive integer $k$.

**Promise:** $\chi(G) = r$, and $c$ is a permutation of an optimal proper vertex-coloring of $G$.

**Question:** Can $c$ be made into a proper $r$-coloring of $G$ using at most $k$ recolorings?

Note that the number of recolorings needed is precisely the minimum Hamming distance between the given coloring $c$ and a valid coloring $c'$ (if existing).

Similarly, we also define $r$-SWAP and $r$-SWAP- PROMISE, where instead of at most $k$ recolorings we have a budget of at most $k$ swaps.

**r-Swap**

**Instance:** A graph $G = (V, E)$, an $r$-coloring $c : V \rightarrow [r]$, and a positive integer $k$.

**Question:** Can $c$ be made into a proper $r$-coloring of $G$ using at most $k$ swaps?

**r-Swap- Promise**

**Instance:** A graph $G = (V, E)$, an $r$-coloring $c : V \rightarrow [r]$, and a positive integer $k$.

**Promise:** $\chi(G) = r$, and $c$ is a permutation of an optimal proper vertex-coloring of $G$.

**Question:** Can $c$ be made into a proper $r$-coloring of $G$ using at most $k$ swaps?

At first glance, the promise conditions might seem to make the two problems $r$-Fix- Promise and $r$-Swap- Promise similar: one could think that two recolorings correspond to a swap because if we recolor a vertex, then by the promise, the color must be reinserted elsewhere. However, it is easy to build graphs in which this does not hold. For example, consider a graph $G$ constructed from a triangle with the vertices $v_1$, $v_2$, $v_3$ colored $c_1$, $c_2$, $c_3$, respectively. Also, connect vertices to $G$ such that $v_1$ has three pendants (neighbours) colored $c_1$ and three pendants colored $c_2$; $v_2$ has three pendants colored $c_2$ and three pendants colored $c_3$; and $v_3$ has three pendants colored $c_3$ and
three pendants colored $c_1$. For an illustration, see Fig. 1. Clearly, three recolorings are enough to get a proper coloring. Indeed, we color $v_1$ with $c_3$, $v_2$ with $c_1$, and $v_3$ with $c_2$, and are done. In contrast, in the swap variant, two swaps are needed (e.g., swap colors on $v_1$ and $v_3$, and then colors on $v_3$ and $v_2$). Similarly, if we add another triangle with the same coloring and attach it to the same pendants, then six recolorings are needed to fix the whole graph, but only four swaps suffice.

### 2.2 FPT-reductions and (kernelization) lower bounds

In this subsection, we briefly review the necessary basics of parameterized complexity.

**Definition 2** Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. A parameterized reduction from $A$ to $B$ is an algorithm such that given an instance $(x, k)$ of $A$, it outputs an instance $(x', k')$ of $B$ such that

1. $(x, k)$ is a YES-instance of $A$ iff $(x', k')$ is a YES-instance of $B$,
2. $k' \leq g(k)$ for some computable function $g$, and
3. the running time is $f(k) \cdot |x|^{O(1)}$ for some computable function $f$.

In the CLIQUE problem, we are given a graph $G$ and an integer $k$. The task is to decide whether $G$ contains a complete subgraph on $k$ vertices. The class of problems reducible to CLIQUE under parameterized reductions is denoted by $\text{W}[1]$. We define hardness and completeness analogously to classical complexity, but assume parameterized reductions. That is, a problem is said to be $\text{W}[1]$-hard if CLIQUE (and thus each problem in $\text{W}[1]$) can be reduced to it by a parameterized reduction. It is widely believed that $\text{FPT} \neq \text{W}[1]$.

Let us recall the well-known Exponential Time Hypothesis (ETH), which is often the assumption used for excluding the existence of algorithms that are considerably faster than e.g., brute-force.

**Conjecture 3** (Exponential Time Hypothesis (Impagliazzo and Paturi 2001)) There exists a constant $c > 0$, such that there is no algorithm solving 3-SAT in time $O^*(2^{cn})$, where $n$ is the number of variables.

Suppose $\varphi$ is an instance of 3-SAT with $n$ variables and $m$ clauses. It holds that if there is a linear reduction from 3-SAT to, say, a graph problem $X$, then the problem
X cannot be solved in time $2^{o(n'+m')}$, where $n'$ and $m'$ denote the number of vertices and edges, respectively. Similar reasoning can be applied for $\mathbb{W}[1]$-hard problems. For instance, it is known that there is no $f(k)n^{o(k)}$-time algorithm for $\text{INDEPENDENT SET}$ for any computable function $f$, unless ETH fails. Then the existence of a parameterized reduction with a linear parameter dependence from $\text{INDEPENDENT SET}$ to a problem $X'$ implies a lower bound for $X'$ under ETH. For more examples and discussion, we refer the reader to Cygan et al. (2015), Lokshtanov et al. (2011).

Finally, let us then mention the machinery we use to obtain kernelization lower bounds later on (for more details, see Bodlaender et al. 2014; Cygan et al. 2015).

**Definition 4** An equivalence relation $\mathcal{R}$ on $\Sigma^*$ is a polynomial equivalence relation if (1) there is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether $\mathcal{R}(x, y)$ in $(|x| + |y|)^{O(1)}$ time; and (2) for any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of $S$ into at most $(\max_{x \in S} |x|)^{O(1)}$ classes.

**Definition 5** (Cross-composition) Let $L \subseteq \Sigma^*$ and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say $L$ cross-composes into $Q$ if there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm which, given $t$ strings $x_1, x_2, \ldots, x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^t |x_i|$ such that (1) $(x^*, k^*) \in Q$ iff $x_i \in L$ for some $i \in [t]$; and (2) $k'$ is bounded polynomially in $\max_{i=1}^t |x_i| + \log t$.

**Theorem 6** (Bodlaender et al. 2014) Assume that an $\mathbb{NP}$-hard language $L$ cross-composes into a parameterized language $Q$. Then $Q$ does not admit a polynomial kernel, unless $\mathbb{NP} \subseteq \mathbb{coNP}/\mathbb{poly}$ and the polynomial hierarchy collapses.

### 3 Parameterized aspects of restoring corrupted colorings

Junosza-Szaniawski et al. (2015) focused on the $r$-$\text{Fix}$ problem, that is, the number of colors in the coloring is fixed to be $r$. Among other results, they showed the problem is FPT parameterized by the treewidth of the input graph. In other words, the $\text{Fix}$ problem (same as $r$-$\text{Fix}$ but $r$ is part of the input) is FPT for the combined parameter $(r + t)$, where $t$ is the treewidth of the input graph. In contrast, we observe the problem is $\mathbb{W}[1]$-hard when the parameter is only treewidth.

In the $\text{PreCOLORING EXTENSION}$ problem ($\text{PrExt}$), we are given a graph $G = (V, E)$, a set $W \subseteq V$ of precolored vertices, and a precoloring $c : W \rightarrow [r]$ of the vertices in $W$. The goal is to decide whether there is a proper $r$-coloring $c'$ of $G$ extending the coloring $c$ (i.e., $c'(v) = c(v)$ for every $v \in W$). When $r$ is fixed, we call the problem $r$-$\text{PreCOLORING EXTENSION}$ ($r$-$\text{PrExt}$). Let us then proceed with the following observation.

**Lemma 7** There exists a polynomial time algorithm which given an instance $I = (G, W, c, r)$ of $\text{PreCOLORING EXTENSION}$ constructs an instance $I' = (G', r', c', k')$ of $\text{Fix}$, such that $I$ is a YES-instance of $\text{PreCOLORING EXTENSION}$ iff $I'$ is a YES-instance of $\text{Fix}$.
Proof Let $I = (G, W, c, r)$ be an instance of PRECOLORING EXTENSION. To obtain, in polynomial time, an instance $I' = (G', r', c', k')$ of Fix, we proceed as follows. First, let $G' = G, r' = r$, and set the number of recolorings $k' = |V \setminus W|$. Then to each precolored vertex $w \in W$, we attach $(r' - 1) \cdot (k' + 1)$ pendant vertices, called $P_w$. Build $c'$ from $c$ as follows. We color vertices in $P_w$ such that there are precisely $k' + 1$ vertices colored in every color $c \in ([r'] \setminus \{c(w)\})$. We retain the colors on the vertices in $W$, and color each uncolored vertex with color 1. Observe that $k'$ recolorings will not suffice to change the color of $w \in W$ as it has $k' + 1$ pendants colored in each color distinct from $c(w)$. Thus, it is easy to see that $I = (G, W, c, r)$ is a YES-instance of PRECOLORING EXTENSION iff $I' = (G', r', c', k')$ is a YES-instance of Fix. \hfill \qed

To make the result hold for SWAP, we add $r \cdot k'$ isolated vertices and color them so that we provide a choice of one of the $r$ colors for each of the $k'$ non-precolored vertices. Finally, it is well-known the addition of vertices of degree at most 1 does not increase the treewidth of a graph. As PRECOLORING EXTENSION is $\text{NP}$-hard parameterized by treewidth (Fellows et al. 2011), we obtain the following.

Corollary 8 Both problems Fix and Swap are $\text{NP}$-complete when restricted to distance-hereditary graphs (Bonomo et al. 2008) (and thus for e.g., chordal graphs), we immediately observe the following.

Corollary 9 Both problems Fix and Swap are $\text{NP}$-complete when restricted to the class of distance-hereditary graphs.

The above also implies hardness for bounded cliquewidth graphs.

Junosza-Szaniawski et al. (2015) proved that for every fixed $r$, the problem $r$-Fix is FPT parameterized by the number of recolorings. However, when the basic operation is a swap instead of a recoloring, the problem becomes hard. This is established by the following lemma. For the result, we give a parameterized reduction from the well-known INDEPENDENT SET problem. In this problem we are given a graph $G = (V, E)$, and an integer $k$. The goal is to decide whether $G$ contains a set of $k$ pairwise non-adjacent vertices. The problem is well-known to be $\text{NP}$-hard parameterized by $k$.

Lemma 10 There exists a polynomial time algorithm which given an instance $I = (G, k)$ of INDEPENDENT SET constructs an instance $I' = (G', r, c, k' = 2k)$ of $r$-Swap for $r = 3$ such that $I$ is a YES-instance of INDEPENDENT SET iff $I'$ is a YES-instance of $r$-Swap.

Proof Let $V(G) = \{v_1, v_2, \ldots, v_n\}$. To construct the graph $G'$, begin with $V(G') = V(G)$ and $k' = 2k$. For clarity, we use different symbols such that vertex $v_i$ of $G$ will correspond to $u_i$ in $G'$. Add $k$ disjoint triangles $\{a_1, b_1, c_1\}, \ldots, \{a_k, b_k, c_k\}$, and color them so that $c(a_j) = 3$ and $c(b_j) = c(c_j) = 2$, for $j \in [k]$. For each $i \in [n]$, add a disjoint triangle $C_i = \{u_{i,a}, u_{i,b}, u_{i,c}\}$. In each, color $c(u_{i,a}) = 1$, $c(u_{i,b}) = 2$, and $c(u_{i,c}) = 3$. Attach $2(k + 1)$ pendant vertices to $u_{i,a}$ and color $k + 1$ of them with color 2 and $k + 1$ of them with color 3. For each $i$, add the edge $u_{i,a}u_{j,b}$. For every
Fig. 2 An instance \((G, k)\) of INDEPENDENT SET with \(k = 2\) (highlighted in gray) transformed to an instance of \(r\)-SWAP. Thick edges at the bottom correspond to \(k + 1\) pendant vertices. A gadget has been expanded for \(u_1\): the three dots hide a similar gadget for each of \(u_2, u_3,\) and \(u_4\). The colors are “● = 1”, “○ = 2”, and “□ = 3”.

\(i, j \in [n]\), if \(i < j\) and \((v_i, v_j) \in E(G)\), add the edge \(u^i_c u_j\). Finally, for each \(i \in [n]\), add \(k + 1\) disjoint triangles \(\{t^i_t, t^i_b, t^i_c\}, \ldots, \{t^i_{a,k+1}, t^i_{b,k+1}, t^i_{c,k+1}\}\). These \(k + 1\) disjoint triangles are colored such that \(c(t^i_t) = 3\), \(c(t^i_b) = 2\), and \(c(t^i_c) = 1\), where \(j \in [k + 1]\). Add the edge \(u_i t^i_{a,j}\), for \(i \in [n]\) and \(j \in [k + 1]\). This completes the construction of \(G'\). An example is shown in Fig. 2. Here, note that the triangles \(\{a_i, b_i, c_i\}\) are left disconnected for clarity, but we could make \(G'\) connected by adding an edge between \(a_i\) colored 2 and \(u^1_a\) colored 1. Then, by construction \(G'\) cannot be recolored in less than \(k\) swaps. Let us then prove that \(I = (G, k)\) is a YES-instance of INDEPENDENT SET iff \(I' = (G', r, c, k')\) is a YES-instance of \(r\)-SWAP.

First, we prove that if there is an independent set of size \(k\) in \(G\), then we can obtain a proper coloring for \(G'\) using at most \(2k\) swaps. Let \(S = \{v_{s_1}, \ldots, v_{s_k}\} \subseteq V(G)\) be such an independent set. By construction, there are exactly \(k\) conflicts in \(G'\) between \(b_j\) and \(c_j\), for \(j \in [k]\). In order to resolve these conflicts, we swap each \(c_j\) with \(u_{s_j}\), spending a total of \(k\) swaps. By doing so, we fix \(k\) conflicts but also introduce \(k\) new conflicts. In particular, the new conflicts are between \(u_{s_j}\) and \(u^i_{b_j}\), as they are both colored 2. We swap \(u^i_{b_j}\) with \(u^i_{c_j}\) (colored 3), and claim that \(G'\) is properly colored. By construction, \(u^i_{c_j}\) (colored 2) is adjacent to \(u_j \in V(G')\) only if \(v_{s_i}\) and \(v_j\) are adjacent in \(V(G)\). As \(S\) is an independent set, each \(u_j \in V(G)\) adjacent to \(u^i_{c_j}\) is still colored 1 because it was not affected by the initial \(c_j \leftrightarrow u_{s_j}\) swaps. Thus, \(c\) is a proper coloring for \(G'\), and we are done.

For the other direction, suppose \(k' = 2k\) swaps suffice to obtain a proper vertex coloring from \(c\). Again, each of the \(k\) conflicts between \(b_j\) and \(c_j\), for \(j \in [k]\), must be fixed. To resolve the conflicts, we must swap either \(b_j\) or \(c_j\) with a vertex colored 1. Without loss of generality, suppose we choose \(c_j\) over \(b_j\). Observe that for every \(i \in [n]\), we cannot swap \(c_j\) with \(u^i_a\) as it has \(k + 1\) pendant vertices colored 2, and \(k + 1\) pendant vertices colored 3. If the swap occurs between \(c_j\) and \(t^i_{b,j}\) we only “shift” the conflict making a useless move, so we must then swap \(t^i_{c,j}\) with \(u_i\) for some \(i \in [n]\) in order to reintroduce a vertex colored 1 in the triangle \(t^i_{a,j}, t^i_{b,j}, t^i_{c,j}\). Thus, the only
Fig. 3 The reduction described in Corollary 12 applied to the graph $G'$ of Fig. 2 (not fully displayed), where the source graph $G'$ has two conflicts. In order to satisfy the promise we have added two triangle stars $S^1$ and $S^2$ (to reduce clutter, we omit $k' + 1$ subscript in the labels). Now, we can use the central vertices of $S^1$ and $S^2$ colored 1 to solve the conflicts, but then we need at least $k' + 1$ more swaps to fix the conflict between $s$ and its pendants. The colors are “$\bullet = 1$”, “$\bigcirc = 2$”, and “$\diamondsuit = 3$” remaining possibility, as $k' = 2k$ swaps suffice, is a swap between each of the vertices $c_j$ with some $u_i$, and then, in order to fix the conflict between $u_i$ and $u'_j$, both colored 2, a swap of $u'_j$ with $u_i$. Clearly, two vertices adjacent $u_i$ and $u_j$ for $i, j \in [n], i < j$, cannot be used to fix the conflicts, for otherwise we would have a conflict between the vertices $u'_i$ and $u_j$, both colored 2 because of the sequence of swaps $c_{\ell'} \leftrightarrow u_i, u'_j \leftrightarrow u_i$, and $c_{\ell''} \leftrightarrow u_j$. We conclude that the vertices $v_i$ of $G$ corresponding to vertices $u_i$ of $G'$ that are swapped with a vertex $c_j$ form an independent set. □

By adding a properly colored $r$-clique to the constructed graph, we can extend the lemma to cover every fixed value of $r$. Thus, we have the following.

**Theorem 11** For every $r \geq 3$, the problem $r$-Swap is $\mathsf{W}[1]$-hard parameterized by the number of swaps $k$. Furthermore, there is no $f(k)n^{o(k)}$-time algorithm for the problem unless ETH fails, where $f$ is a computable function.

In addition, it is straightforward to extend the construction of Lemma 10 to show the promise variant, namely $r$-Swap-Promise, is $\mathsf{W}[1]$-hard parameterized by the number of swaps $k$.

**Corollary 12** For every $r \geq 3$, the problem $r$-Swap-Promise is $\mathsf{W}[1]$-hard parameterized by the number of swaps $k$. Furthermore, there is no $f(k)n^{o(k)}$-time algorithm for the problem unless ETH fails, where $f$ is a computable function.

**Proof** Assume the construction of an instance $I = (G', r, c, k')$ of $r$-Swap of Lemma 10. To prove the claim, it suffices to augment the construction to ensure the promise holds, i.e., that with a finite number of swaps we have that $G'$ is properly $r$-colored and $\chi(G') = r$. Informally, our plan is to add “spare” vertices colored 1 that can be used to fix all the initial conflicts of $G'$. However, in order to use even one of them we must exceed our budget of $k'$ swaps.

A triangle star (of order $m$), denoted by $S_m$, is constructed by starting from a disjoint union of $2m$ properly 3-colored triangles and an isolated vertex $s$ colored 1.
The vertex $s$ is connected to one of the vertices of each triangle by adding an edge between $s$ and the vertices colored 2 of the first $m$ triangles, and an edge between $s$ and the vertices colored 3 of the last $m$ triangles. To enforce the promise, we add $n$ disjoint triangle stars $S_{k'+1}'$ for $i \in [n]$ to $G'$ (see Fig. 3 for an illustration). Observe that after swapping the central vertex $s$ (colored 1) of a $S_{k'+1}'$ with (say) $b_i$ (colored 2), we require $k' + 1$ more swaps to fix the conflicts residing at that particular $S_{k'+1}':$ we need to rotate the colors of $k' + 1$ triangles (those with the neighbours of $s$ colored 2), and the same problem occurs if we swap $s$ (now colored 2) with a pendant colored 3 picked from another part of the graph. Nevertheless, given enough swaps—$(k' + 1)^2$ to be precise—we can properly $r$-color $G'$. Finally, to guarantee $\chi(G') = r$, it suffices to add a disjoint properly $r$-colored clique. \qed

4 No polynomial kernel for $r$-Fix

Junosza-Szaniawski et al. (2015) showed that for any fixed $r$, the problem $r$-Fix is FPT parameterized by the number of recolorings $k$. In particular, their result implies a kernel of exponential size for the problem. Thus, they asked whether or not there is a kernel of polynomial size. The question was answered in the negative in a full version of Junosza-Szaniawski et al. (2015) by Garnero et al. (2018). Independently of their work, in what is to follow, we give an alternative proof of the theorem.

Lemma 13 For $r = 3$, the problem $r$-Fix parameterized by the number of recolorings $k$ does not admit a polynomial kernel unless $\text{NP} \subseteq \text{coNP/poly}$.

Proof We show that 3-SAT cross-composes into $r$-Fix parameterized by the number of recolorings $k$. By choosing an appropriate polynomial equivalence relation $R$, we can assume we are given a sequence $\varphi_1, \varphi_2, \ldots, \varphi_t$ of 3-SAT instances with an equal number of variables, denoted by $n$, and an equal number of clauses, denoted by $m$.

Let us then proceed with a cross-composition algorithm that composes $t$ input instances $\varphi_1, \varphi_2, \ldots, \varphi_t$ which are equivalent under $R$ into a single instance of $r$-Fix parameterized by the number of recolorings. Specifically, we construct an instance $(G, k)$ of $r$-Fix, where $G$ is a vertex-colored graph, and $k$ the number of recolorings. Our plan is to convert each 3-SAT instance $\varphi_1, \varphi_2, \ldots, \varphi_t$ to an instance $\varphi_1', \varphi_2', \ldots, \varphi_t'$ of 4-SAT. For each resulting instance of 4-SAT, we apply the standard reduction from 4-SAT to 3-COLORING (see e.g., Garey et al. 1976). Finally, the resulting graphs are connected by a spread gadget, which acts as an instance selector. Let us first describe the gadgets, and then the construction of the whole graph $G$. At the same time, we describe a 3-coloring $c : V(G) \to [3]$. Without loss of generality, we assume that $t$ is a power of two (if necessary, we duplicate some of the formulas $\varphi_i$.) We set $k = 2 \log_2(t) + 2n + 15m$. Our construction depends crucially on $k$, and its choice will become apparent later on.

3-SAT to 4-SAT For each 3-SAT formula $\varphi_h$, where $h \in [t]$, introduce a new variable $u_h$, and add it to each clause of $\varphi_h$. We call the resulting 4-SAT formula $\varphi_h'$. Observe that by setting $u_h$ to true we satisfy $\varphi_h'$.
The variable vertices Let the \( n \) variables of \( \varphi_h \), where \( h \in [t] \), be \( x_{1,h}, x_{2,h}, \ldots, x_{n,h} \). We introduce \( n \) disjoint 2-cliques labelled \( \{x_{1,h}, \neg x_{1,h}\}, \ldots, \{x_{n,h}, \neg x_{n,h}\} \) and add an isolated vertex \( u_h \). We refer to each of the \( 2n + 1 \) vertices as variable vertices. We add an isolated vertex \( z \) colored 3 with \( k + 1 \) pendants colored 1 and \( k + 1 \) pendants colored 2. We link \( z \) to all \( x_{i,h}, \) to \( \neg x_{i,h}, \) and to \( u_h \). In this way we force all \( x_{i,h}, \) \( \neg x_{i,h}, \) and \( u_h \) to be colored 1 or 2 (they cannot be colored 3 for otherwise more than \( k \) recolorings would be needed). Intuitively, the colors correspond to truth values such that 1 represents false and 2 represents true. So when \( x_{i,h} \) is colored 2 and \( \neg x_{i,h} \) is colored 1, we represent the truth assignment \( x_{i,h} = T \); when \( x_{i,h} \) is colored 1 and \( \neg x_{i,h} \) is colored 2, we represent the truth assignment \( x_{i,h} = F \). We initially set \( c(x_{i,h}) = 2 \) and \( c(\neg x_{i,h}) = 1 \) for \( i \in [n] \), and \( c(u_h) = 2 \) (i.e., we set each variable true).

The clause gadget Denote by \( C_{h,j} \) the \( j \)th clause of \( \varphi_h \). For each clause \( C_{h,j} \), where \( h \in [t] \) and \( j \in [m] \), construct the following clause gadget \( H_{h,j} \). Take three disjoint triangles \( \{a_j, b_j, y_{1,j}\}, \{c_j, d_j, y_{2,j}\}, \{y_{3,j}, y_{4,j}, r_j\} \), and add the edges \( (y_{1,j}, y_{4,j}), (y_{2,j}, y_{3,j}), \) and \( (y_{5,j}, r_j) \). We color these vertices such that \( y_{1,j}, y_{3,j}, d_j \) are colored 1, \( y_{2,j}, a_j, r_j \) are colored 2, and \( y_{4,j}, b_j, c_j \) are colored 3. We add \( k + 1 \) pendant vertices adjacent to \( r_j \) and color them with color 1. This guarantees \( k \) recolorings cannot give \( r_j \) color 1. We extend \( a_j \) with a path of two vertices \( a'_j, a''_j \) colored 1 and 3, respectively, and we link \( a'_j \) and \( a''_j \) to \( k + 1 \) pendants colored 2. Informally, this

---

Fig. 4 a The vertex-coloring of a clause gadget for the clause \( \{x_1, \neg x_2, x_3, u_h\} \). The variable vertices are inside the dashed box at the bottom (the clause gadget outside of it) with their initial truth values \( x_1 = x_2 = x_3 = u_h = T \). Each clause gadget has the same initial vertex-coloring. All variable literals (colored 1 or 2) are linked to their corresponding \( a'_j, b'_j, \) and \( c'_j \) colored 3 of the clauses containing them. Here, \( u_h \) is colored 2 is linked to all the vertices \( d_j \) of the clauses in \( \varphi_h \). b The spread gadget, where fixing the conflict at the root \( r \) corresponds to choosing an instance \( \ell_h \) for \( h \in [t] \). In both figures, thick edges correspond to \( k + 1 \) pendant vertices. The colors are “\( \bullet = 1 \)”, “\( \circ = 2 \)” and “\( \Box = 3 \)”
sub-gadget is a “2-to-3 color converter”: if \( a_j \) is colored 2 then \( a_j'' \) can be colored 3, but if \( a_j \) is colored 1 then \( a_j'' \) must be colored 1 as well. We make the same extension for vertices \( b_j \) and \( c_j \). Vertices \( a_j'', b_j'', c_j'' \) and \( d_j \) correspond to the 4 literals each clause has. Thus, we connect them to the corresponding variable vertices. That is, when \( w \in \{ a_j'', b_j'', c_j'', d_j'' \} \) corresponds to the variable \( x_{i,h} \), \( i \in [n] \), we add the edge \((w, x_{i,h})\) (and similarly when it is negated). Note that initially all variable vertices of \( \varphi_h \) are linked to vertices colored 3 (vertices \( a_j'', b_j'', c_j'' \) of the clauses in which the variables appear) and \( u_h \) colored 2 is linked to the vertex \( d_j \) colored 1 of all the clauses of \( \varphi_h \). Figure 4a shows the structure and the coloring of a clause gadget.

The following property holds for a clause gadget \( H_{h,j} \).

\((P_1)\) If all four variable vertices of \( H_{h,j} \) have color 1 (i.e., they are all false), then \( r_j \) must have color 1 costing \( k + 1 \) recolorings to properly color the gadget. Alternatively, we could also recolor one of the \( a_j', b_j', c_j' \) (or \( a_j'', b_j'', c_j'' \)), but by construction this would cost more than \( k \) recolorings to fix their pendant neighbours.

Equivalently, \((P_1)\) states that the gadget \( H_{h,j} \) can be properly 3-colored in no more than \( k \) swaps if at least one of the attached variable literals (including \( u_h \)) have color 2 (i.e., at least one literal of \( \varphi_h \) must be true).

Figure 5 shows the valid colorings with respect to \( u_h \). If \( u_h \) is colored 2 (true), then all the other literals can be false. On the other hand, if \( u_h \) is colored 1 (false), then at least one literal must be true.

The spread gadget The spread gadget is constructed by starting from a complete binary tree on \( t \) leaves \( \ell_1, \ell_2, \ldots, \ell_t \) (note that \( t \) is a power of two) with the root \( r \). We replace each internal vertex with a triangle, and attach \( k + 1 \) pendant vertices to \( r \). Thus, the distance from \( r \) to any leaf is \( 2 \log_2(t) \). We color root \( r \), its pendant vertices, and each leaf \( \ell_1, \ell_2, \ldots, \ell_t \) with color 1. In a triangle, the top vertex receives color 1, the right vertex color 2, and the left vertex color 3 (see Fig. 4b). This finishes the construction of the spread gadget.

To complete the construction of the graph \( G \), each leaf \( \ell_h \) in the spread gadget is made adjacent to \( u_h \). Recall \( k = 2 \log_2(t) + 2n + 15m \), and output the instance \((G, k)\).
of $r$-Fix. Let us then prove that $(G, k)$ is a YES-instance of $r$-Fix iff one of $\varphi_h$ is satisfiable for $h \in [\tau]$.

**Correctness** Suppose $(G, k)$ is a YES-instance of $r$-Fix. By construction, the root $r$ and its $k + 1$ pendant vertices are colored with color 1 in the spread gadget. As we only have a budget of $k$ recolorings, we must recolor $r$. By doing so, we introduce a conflict into the triangle containing $r$. When this conflict is fixed, we move it to one of the two succeeding triangles. Further continuing to fix the conflict, we propagate it down to one of the leaves $\ell_h$, for some $h \in [\tau]$. Intuitively, the propagation to $\ell_h$ means we have chosen to solve the instance $\varphi_h$. By moving the conflict from $r$ to $\ell_h$ and its parent, we used precisely $2 \log_2(\tau)$ swaps. By construction, $\ell_h$ (which is now colored 1, but must be recolored to solve the conflict) forms a triangle with $u_h$ (colored 2) and $z$ (colored 3). As $z$ has $k + 1$ pendants colored 1 and $k + 1$ pendants colored 2, in a proper coloring $\ell_h$ must be colored 2 and therefore $u_h$ must be colored 1, i.e., we must set the literal corresponding to $u_h$ to false. By $(P_1)$ this implies that in each clause of $\varphi_h$ at least one another literal must be true, i.e., the original $\varphi_h$ must be satisfiable. Therefore, if $(G, k)$ is a YES-instance of $r$-Fix, then $\varphi_h$ is satisfiable.

For the other direction, suppose $\varphi_h$ is satisfiable for some $h \in [\tau]$. Recall that the initial vertex-coloring corresponds to a truth assignment $\tau = \{z_1, z_2, \ldots, z_n\} \in \{T\}^n$. All formulas $\varphi'_i$ are made true by the initial setting in which $u_i$ is true and the only conflict in the graph $G$ is at the top of the spread gadget. Using at most $2n + 15m$ recolorings, we turn the initial vertex-coloring to a vertex-coloring corresponding to $\tau' = \{z'_1, z'_2, \ldots, z'_n\} \in \{F, T\}^n$ such that $\varphi_h$ is satisfied under $\tau'$. Indeed, observe that as $\varphi_h$ is satisfiable, $(P_1)$ is not violated. Moreover, observe that $\tau'$ satisfies $\varphi_h$ regardless of the truth value of $u_h$. Thus, we can freely let $c(u_h) = 1$ (i.e., set it to false). We notice that changing the truth values of the variables in $\varphi_h$ does not affect the initial valid coloring of the other formulas $\varphi_i$ for $i \neq h$ because all literals are linked to vertices colored 3. This allows us to “push down” the only initial conflict of $G$ at the top of the spread gadget down to leaf $\ell_h$ which can now be colored 2 without
any conflict with $u_i$ using precisely $2 \log_2(t)$ recolorings, and proceeding in a way similar to the first direction of the proof (see Fig. 6 for an example). In particular, 15 recolorings will suffice to reverse the truth values of the literals in a clause gadget. So at most $k = 2 \log_2(t) + 2n + 15m$ recolorings have been used, and no conflicts remain in $G$. Thus, if at least one of $\varphi_1, \ldots, \varphi_i$ is a YES-instance of 3- SAT, then $(G, k)$ is a YES-instance of $r$-Fix. This concludes the proof. 

In order to extend the above result to hold for every $r \geq 4$, we attach $(r - 3) \cdot (k + 1)$ pendant vertices to each vertex of the construction. These pendant vertices are colored in the obvious way such that each “original” vertex must receive exactly one of the colors 1, 2, or 3.

**Theorem 14** (Garnero et al. 2018) *For every $r \geq 3$, the problem $r$-Fix parameterized by the number of recolorings $k$ does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.*

Note that in the light of Theorem 11, the existence of a kernel of any size depending only on $k$ and $r$ for $r$-Swap is highly unlikely.

## 5 Chromatic villainy: $r$-SWAP- PROMISE is hard

As the main result of this section, we prove that 3-SWAP- PROMISE is $NP$-hard when restricted to the class of planar graphs. In other words, even with the additional information that some proper vertex-coloring is always obtainable after a finite number of swaps (and no other proper vertex-coloring with fewer than 3 colors exists), the problem remains hard.

Our reduction will be from the $r$-PrExt problem, shown to be $NP$-complete for $r = 3$ when restricted to bipartite planar graphs by Kratochvíl (1993). In fact, although not explicitly stated, the following slightly stronger result is obtained from Kratochvíl (1993).

**Theorem 15** (Kratochvíl 1993) *The $r$-PrExt problem is $NP$-complete for $r = 3$ when restricted to the class of bipartite planar graphs, and each precolored vertex has degree 1, that is, $\deg(w) = 1$ for every $w \in W$.*

The reader should be aware that in the following, we use the color set $\{0, 1, 2\}$ instead of $[3]$. This will make it more convenient to describe the coloring through modular arithmetic. We are then ready to proceed with the main result of the section.

**Theorem 16** *3-SWAP- PROMISE is $NP$-hard when restricted to the class of planar graphs. Moreover, the same is true even when every swap must be between adjacent vertices.*

**Proof** Let $(G = (V, E), W, c)$ be an instance of $r$-PrExt, where $G$ is an $n$-vertex bipartite planar graph, $W \subseteq V$ a set of precolored vertices, and $c : W \to \{0, 1, 2\}$ a precoloring of the vertices in $W$. By Theorem 15, we may assume without loss of generality that each (precolored) vertex in $W$ has degree 1. Our construction crucially depends on this fact. Let $r = 3$ be a fixed color bound, and let $h = |V \setminus W|$. We will
Informally, the vertices $s$ can be extended to a valid $r$-coloring of $G$ if at most $h$ swaps are needed to transform $c_H$ to an optimal proper vertex-coloring of $H$, i.e., $B(c_H) \leq h$. To enforce the promise of the problem, it shall hold for $c_H$ that (i) it uses precisely $\chi(H)$ colors, and that (ii) by using a finite number of swaps $c_H$ can be transformed into a proper coloring of $H$.

**Construction** Let $X = V \setminus W$ be the set of uncolored vertices, let $A$ and $B$ be the bipartition of $G$, that is, $V = A \cup B$, and let us name the set of $r$ colors $C = \{0, 1, 2\}$. The graph $H$ and its vertex-coloring $c_H$ are constructed from $G$ and its precoloring $c$ as follows.

- We retain the coloring on the vertices of $W$, that is, $c_H(w) = c(w)$, for every $w \in W$.
- If $x \in X$ has one or more neighbors colored with color $i$, then we set $c_H(x) = i$. Without loss of generality, we can assume that $x$ does not have distinctly colored neighbours, for otherwise we could pre-color it with the remaining third color, or immediately conclude that the $r$-PreExt has no solution and output a dummy NO-instance for our problem. If all neighbors of $x$ are uncolored, we set $c_H(x) = 0$ if $x \in A$. Otherwise, $x \in B$, so we set $c_H(x) = 1$. For each $x$, we also add two vertices $x_1$ and $x_2$ along with the edges $xx_1$ and $xx_2$. We color $c_H(x_1) = (i + 1) \mod 3$ and $c_H(x_2) = (i + 2) \mod 3$.
- For each precolored vertex $w \in W$ we add $2(h + 1)$ new vertices $s_{w,1}, \ldots, s_{w,h+1}$ and $t_{w,1}, \ldots, t_{w,h+1}$. These will be made pendant vertices of $w$ by adding the altogether $2(h + 1)$ edges $ws_{w,\ell}$ and $wt_{w,\ell}$ where $\ell \in [h + 1]$. They receive a color as follows, where $c(w)$ denotes the color of $w$:
  - if $w \in A$ and $c(w) \neq 0$, then $c_H(s_{w,\ell}) = 0$ and $c_H(t_{w,\ell}) = f$, where $f \in C \setminus \{0, c(w)\}$;
  - if $w \in B$ and $c(w) \neq 1$, then $c_H(s_{w,\ell}) = 1$ and $c_H(t_{w,\ell}) = g$, where $g \in C \setminus \{1, c(w)\}$; and
  - in all other cases $c_H(s_{w,\ell}) = (c(w) + 1) \mod 3$ and $c_H(t_{w,\ell}) = (c(w) + 2) \mod 3$.
- For every precolored vertex $w \in A$ and $c(w) \neq 0$, we add $h$ new vertices $s'_{w,1}, \ldots, s'_{w,h}$ with the edges $s_{w,j}s'_{w,j}$, and set $c_H(s'_{w,j}) = c(w)$, where $j \in [h]$.
- For every precolored vertex $w \in B$ and $c(w) \neq 1$, we add $h$ new vertices $s'_{w,1}, \ldots, s'_{w,h}$ with the edges $s_{w,j}s'_{w,j}$, and set $c_H(s'_{w,j}) = c(w)$, where $j \in [h]$.
- Finally, consider an arbitrary precolored vertex $w \in W$, and its set of $h + 1$ pendant vertices $t_{w,j}$ (of order 1). We choose an arbitrary vertex among the $t_{w,j}$ vertices, and call it $v$. Then, we add two vertices $r$ and $r'$ along with the edges $vr$, $vr'$, and $rr'$. These vertices are colored such that $c_H(r) = (c_H(v) + 1) \mod 3$ and $c_H(r') = (c_H(v) + 2) \mod 3$. The purpose of this triangle is to force the chromatic number of $H$ to be at least 3.

Informally, the vertices $s_{w,i}, s'_{w,i},$ and $t_{w,i}$ guarantee that given enough swaps (more than the $h$ allowed), we can color all the vertices of $A$ with color 0 and all the vertices of $B$ with color 1. This gives us a proper coloring and thus satisfies the promise condition. This finishes the construction of the graph $H$ along with its vertex-coloring $c_H$. It is straightforward to verify $G$ is planar, but not bipartite because of the triangle.
on the vertices $v$, $r$, and $r'$. An example is shown in Fig. 7. We will then prove $(G = (V, E), W, c)$ is YES-instance of PreExt iff $B(c_H) \leq h$, that is, if $h$ swaps suffice to transform $c_H$ into a proper coloring of $H$.

**Correctness** Suppose $c$ can be extended to a proper vertex-coloring $c'$ of $G$. We will show $h$ swaps suffice to transform $c_H$ into a proper coloring of $H$. For each $x \in X$, we perform a swap between $x$ and either $x_1$ or $x_2$. Then for every $x$ it holds that either $c_H(x) = c'(x)$ (and there is no need to swap it), or one of the two described swaps can change the color of $x$ to $c'(x)$. Now, let $c_H(x)$ be the color of $x$ before the swap, and $c'_H(x) = c'(x)$ its color after the swap. Let $u_1, \ldots, u_m$ be the neighbors of $x$. If $u_i \in W$ was a precolored vertex, then $c_H(u_i) = c_H(x) \neq c'_H(x)$ by construction; if $u_i \in X$ was an uncolored vertex then the valid coloring $c'$ guarantees that $c'_H(x) \neq c'_H(u_i)$. Thus, the claim follows.

For the other direction, suppose $B(c_H) \leq h$. Consider a precolored vertex $w \in W$ and let $c_H(w) = i$. We claim that for any valid extension $c'$ of $c$, it holds that $c'(w) = c_H(w) = c(w)$. More precisely, we will show that if the color of $w$ was changed, then it is impossible for $c'$ to be an extension of $c$. By construction, the vertex $w$ has $h + 1$ neighbors $s_{w, \ell}$ each colored $p$, and $h + 1$ neighbors $t_{w, \ell}$ each colored $q$ with $i$, $p$, and $q$ all pairwise distinct. Thus, if one swap was used to change the color on $w$, then after $h - 1$ swaps there would be at least one edge incident to $w$ with its endpoints having the same color. So we have that $c'(w) = c_H(w) = c(w)$. Moreover, $c'$ is completed to an extension of $c$ by picking the colors $c'_H(x)$ assigned to the uncolored vertices $x \in X$ after the $h$ swaps. This completes the proof of correctness for our reduction.

**Promise** Let us then show that the promise holds as well. That is, we show that $c_H$ can be transformed into a proper vertex-coloring $c'_H$ of $H$ with a finite number of swaps even if the original precoloring of $G$ cannot be extended to a proper coloring (but in this case, more than $h$ swaps are needed).

---

**Fig. 7** (a) A partially precolored input graph $G$ of $r$-PrExt. (b) To reduce clutter, the reduction of Theorem 16 expanded for only two vertices $x \in X$ and $w \in W$. Here, the vertex $t_{w, h+1}$ is shown with label $v$. The colors are “0”, “1”, and “2” on the vertices $v$, $r$, and $r'$.
First, we show that a finite number of swaps gives us a 2-coloring for $A \cup B$ such that every vertex in $A$ receives color 0, and every vertex in $B$ color 1. Afterwards, we will adjust the remaining pendant vertices $s_{w, \ell}, t_{w, \ell}$, and $s'_{w, j}$ so that no conflict remains.

If $x \in X \cap A$ and $c_H(x) \neq 1$, then we swap it with one of its neighbors $x_1$ or $x_2$ and get $c_H''(x) = 0$. Similarly, if $x \in X \cap B$ and $c_H(x) \neq 2$, a swap with either $x_1$ or $x_2$ gives us $c_H''(x) = 1$.

Let us then consider the precolored vertices. If $w \in W \cap A$ and $c_H(w) \neq 0$, we swap $w$ with $s_{w, h+1}$ which is colored 0. This causes a conflict $c_H''(w) = c_H(s_{w, j}) = 0$, which is fixed by swapping $s_{w, j}$ with $s'_{w, j}$ that are colored $c_H(s'_{w, j}) = c_H(w) \neq 0$. Similarly, if $w \in W \cap B$ and $c_H(w) \neq 1$, we swap $w$ with $s_{w, h+1}$ which is colored 1. This causes $c_H''(w) = c_H(s_{w, j}) = 1$, where $j \in [h]$. To fix them, we swap $s_{w, j}$ with $s'_{w, j}$ that are colored $c_H(s'_{w, j}) = c_H(w) \neq 1$. Thus, we have that all vertices in $A$ are colored 0, and all vertices in $B$ are colored 1. Moreover, for all $w \in W$ we have $c_H''(w) \neq c_H''(s_{w, \ell})$ and $c_H''(w) \neq c_H''(t_{w, \ell})$, where $\ell \in [h + 1]$. Also, $c''_H(s_{w, j}) \neq c''_H(s_{w, j})$, where $j \in [h]$. Thus, $c''_H$ is a valid coloring of $H$ and the triangle $v, r, r'$ guarantees that $\chi(H) = 3$. Thus, the claim follows.

By removing the promise condition, we can modify our reduction to obtain the following.

**Corollary 17** For every $r \geq 3$, the problem $r$-SWAP is $\text{NP}$-complete for bipartite planar graphs.

Another corollary follows by a chain of reductions. First, Lichtenstein (1982) gives a reduction from 3-SAT to PLANAR 3-SAT showing PLANAR 3-SAT cannot be solved in time $2^{o(\sqrt{n+m})}$, unless ETH fails. Continuing to compose reductions, Mansfield (1983) gives a linear reduction from PLANAR 3-SAT to PLANAR 1-IN-3-SAT, which is then similarly reduced by Kratochvíl (1993) to $r$-PRExt (the result of Theorem 15). Finally, it can be verified the construction of Theorem 16 has linear size, giving us the following.

**Corollary 18** There is no algorithm which solves PLANAR 3-SWAP−Promise in $2^{o(\sqrt{n})}$ time unless ETH fails.

However, on a positive side, we claim that for any fixed $r$, the problem PLANAR $r$-FIX (and its promise variant) can be solved in $2^{O(\sqrt{n})}$ time. To see this, we recall that Junosza-Szaniawski et al. (2015) showed that for any fixed $r$, the optimization variant of $r$-FIX is solvable in $O(nr^{r+2})$ time on graphs of treewidth $r$. To leverage this result, it is enough to recall the treewidth of a planar graph is $O(\sqrt{n})$. This implies a $2^{O(\sqrt{n})}$-time algorithm for PLANAR $r$-FIX.

Finally, let us mention that by modifying the construction of Theorem 16 slightly, one can show a similar result for $r$-FIX−Promise, and its non-promise variant as well.

**Theorem 19** 3-FIX−Promise is $\text{NP}$-hard when restricted to the class of planar graphs.

**Proof** Consider the construction in Theorem 16. For each $x \in X$, remove the edges $xx_1$ and $xx_2$, and add the edge $x_1x_2$. 

\[ \text{ Springer} \]
Observe that it still holds that in $h$ recolorings, it is impossible to recolor a precolored vertex $w \in W$. Thus, the correctness of the reduction holds. To see that the promise is enforced, note that it is still true that $\chi'(G) = 3 = r$. For each $x \in X$, there is a corresponding 2-clique, from which a color distinct from $c'(x)$ can be swapped for $x$. Thus, the promise is also enforced. We have a valid reduction, and thus conclude the proof.

By relaxing the requirement on the promise condition, we establish again the following. We also note that using different ideas, the same conclusion was reached in Garnero et al. (2018).

**Corollary 20** (Garnero et al. 2018) For every $r \geq 3$, the problem $r$-Fix is $\text{NP}$-complete for bipartite planar graphs.

As also remarked in Garnero et al. (2018), it is interesting to contrast the above with the results of Junosza-Szaniawski et al. (2015) where it was shown that when $r = 2$, the problem $r$-Fix is solvable in polynomial time. In other words, if we have a bipartite graph that is not colored optimally (i.e., more than two colors are used), fixing the coloring is hard.

### 6 Conclusions

We further investigated the complexity of restoring corrupted colorings, especially from a parameterized perspective. Interestingly, we showed that $r$-Swap is $\text{W}[1]$-hard parameterized by the number of swaps, while $r$-Fix is known to be FPT parameterized by the number of recolorings. We believe the problems behave similarly for treewidth. Indeed, we conjecture that $r$-Swap is $\text{W}[1]$-hard parameterized by the treewidth of the input graph, for every $r \geq 3$. One could also consider other natural basic operations, such as swaps between adjacent vertices.

Finally, it might be interesting to perform a similar study for edge-colored graphs. In particular, how does the complexity of edge recoloring compare to vertex recoloring?

### References

Björklund A, Husfeldt T, Koivisto M (2009) Set partitioning via inclusion–exclusion. SIAM J Comput 39(2):546–563

Bodlaender HL, Jansen BMP, Kratsch S (2014) Kernelization lower bounds by cross-composition. SIAM J Discrete Math 28(1):277–305

Bonomo F, Durán G, Marenco J (2008) Exploring the complexity boundary between coloring and list-coloring. Ann Oper Res 169(1):3–16

Bonsma PS, Mouawad AE, Nishimura N, Raman V (2014) The complexity of bounded length graph recoloring and CSP reconfiguration. In: Proceedings of the 9th international symposium on parameterized and exact computation, IPEC 2014, Wroclaw, 10-12 Sept, pp 110–121

Clark SA, Holliday JE, Holliday SH, Johnson P, Trimm JE, Rubalcba RR, Walsh M (2006) Chromatic villainy in graphs. Congressus Numerantium 182:171

Cygan M, Fomin FV, Kowalik L, Lokshtheast D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) Parameterized algorithms. Springer, Berlin
de Berg M, Buchin K, Jansen BMP, Woeginger GJ (2016) Fine-grained complexity analysis of two classic TSP variants. In: Proceedings of the 43rd international colloquium on automata, languages, and programming, ICALP 2016, Rome, 11–15 July, pp 5:1–5:14
Diestel R (2010) Graph theory. Springer, Heidelberg
Even S, Selman AL, Yacobi Y (1984) The complexity of promise problems with applications to public-key cryptography. Inf Control 61(2):159–173
Fellows MR, Fomin FV, Lokshtanov D, Rosamond F, Saurabh S, Szeider S, Thomassen C (2011) On the complexity of some colorful problems parameterized by treewidth. Inf Comput 209(2):143–153
Fellows MR, Fomin FV, Lokshtanov D, Rosamond F, Saurabh S, Villanger Y (2012) Local search: Is brute-force avoidable? J Comput Syst Sci 78(3):707–719
Garey MR, Johnson DS, Stockmeyer L (1976) Some simplified NP-complete graph problems. Theor Comput Sci 1(3):237–267
Garnero V, Junosza-Szaniawski K, Liedloff M, Montealegre P, Rzążewski P (2018) Fixing improper colorings of graphs. Theor Comput Sci 711:66–78
Goldreich O (2008) Computational complexity: a conceptual perspective. Cambridge University Press, Cambridge
Impagliazzo R, Paturi R (2001) On the complexity of k-SAT. J Comput Syst Sci 62(2):367–375
Jensen TR, Toft B (2011) Graph coloring problems. Wiley, New York
Johnson DS (1985) The NP-completeness column: an ongoing guide. J Algorithms 6(3):434–451
Johnson M, Kratsch D, Kratsch S, Patel V, Paulusma D (2014) Finding shortest paths between graph colourings. In: Proceedings of the 9th international symposium on parameterized and exact computation, IPEC 2014, Wroclaw, 10–12 Sept, pp. 221–233
Junosza-Szaniawski K, Liedloff M, Rzążewski P (2015) Fixing improper colorings of graphs. In: SOFSEM 2015: theory and practice of computer science. Springer, Berlin, pp 254–290
Kratochvíl J (1993) Precoloring extension with fixed color bound. Acta Math Univ Comen 62:139–153
Lichtenstein D (1982) Planar formulae and their uses. SIAM J Comput 11(2):329–343
Lokshtanov D, Marx D, Saurabh S (2011) Lower bounds based on the exponential time hypothesis. Bull EATCS 105:41–72
West D (2013) Chromatic villainy of graphs. http://www.math.illinois.edu/~dwest/regs/chromvil.html. Accessed 3 Aug 2015
Wrochna M (2018) Reconfiguration in bounded bandwidth and tree-depth. J Comput Syst Sci 93:1–10