Approximately coloring graphs without long induced paths

Maria Chudnovsky · Oliver Schaudt · Sophie Spirkl · Maya Stein · Mingxian Zhong

Received: date / Accepted: date

Abstract It is an open problem whether the 3-coloring problem can be solved in polynomial time in the class of graphs that do not contain an induced path on \( t \) vertices, for fixed \( t \). We propose an algorithm that, given a 3-colorable graph without an induced path on \( t \) vertices, computes a coloring with max \( \{5, 2 \lceil \frac{t-1}{2} \rceil - 2\} \) many colors. If the input graph is triangle-free, we only need max \( \{4, \lceil \frac{t-1}{2} \rceil + 1\} \) many colors. The running time of our algorithm is \( O((3^t - 2 + t^2)m + n) \) if the input graph has \( n \) vertices and \( m \) edges.

Keywords graph coloring · forbidden induced paths · approximation algorithm

* An extended abstract of the paper has previously appeared in Bodlaender H., Woeginger G. (eds): Graph-Theoretic Concepts in Computer Science (WG) 2017. Lecture Notes in Computer Science, vol 10520. Springer, Cham.

The first author was supported by National Science Foundation grant DMS-1550991 and US Army Research Office Grant W911NF-16-1-0404. The fourth author was supported by Fondecyt grants 1140766 and 1180830, by CMM-Basal AFB 170001, and by Millennium Nucleus Information and Coordination in Networks.

M. Chudnovsky
Princeton University, Princeton, NJ 08544

O. Schaudt
Universität zu Köln, Köln, Germany
Current address: RWTH Aachen, Aachen, Germany

S. Spirkl
Princeton University, Princeton, NJ 08544
Current address: Rutgers University, Piscataway, NJ 08854
E-mail: sspirkl@math.princeton.edu

M. Stein
Universidad de Chile, Santiago, Chile

M. Zhong
Columbia University, New York, NY 10027
Current address: Lehman College, CUNY, Bronx, NY 10468
1 Introduction

A $k$-coloring of a graph $G$ is a function $c : V(G) \rightarrow \{1, \ldots, k\}$ such that $c(v) \neq c(u)$ for all $vu \in E(G)$. In the $k$-coloring problem, one has to decide whether a given graph admits a $k$-coloring or not; it is NP-complete for all $k \geq 3$, as Karp proved in his seminal paper [19].

Coloring $H$-free graphs. One way of dealing with this hardness is to restrict the structure of the instances. In this paper we study $H$-free graphs, that is, graphs that do not contain a fixed graph $H$ as an induced subgraph. It is known that for fixed $k \geq 3$, the $k$-coloring problem is NP-hard on $H$-free graphs if $H$ is any graph other than a subgraph of a chordless path [15,18,21,22]. Therefore, we further restrict our attention to $P_t$-free graphs, $P_t$ being the chordless path on $t$ vertices.

A substantial number of papers study the complexity of coloring $P_t$-free graphs, and most of the results are gathered in the survey paper of Golovach et al. [12]. Let us recall a few results that define the current state of the art regarding the complexity of $k$-coloring in $P_t$-free graphs.

Theorem 1 (Bonomo et al. [2]) The 3-coloring problem can be solved in polynomial time in the class of $P_7$-free graphs. This holds true even if each vertex comes with a subset of $\{1,2,3\}$ of feasible colors.

It is an intriguing open question whether the 3-coloring problem is solvable in polynomial time in the class of $P_t$-free graphs, whenever $t > 7$ is fixed.

Theorem 2 (Chudnovsky et al. [4,5]) The 4-coloring problem can be solved in polynomial time in the class of $P_6$-free graphs.

Going back to $P_5$-free graphs, an elegant algorithm of Hoàng et al. [14] shows that this class is structurally restricted enough to allow for a polynomial time algorithm solving the $k$-coloring problem.

Theorem 3 (Hoàng et al. [14]) The $k$-coloring problem can be solved in polynomial time in the class of $P_5$-free graphs, for each fixed $k$.

The above result is interesting also for the fact that if $k$ is part of the input, the $k$-coloring problem in $P_5$-free graphs becomes NP-hard again [21].

Regarding negative results, the following theorem of Huang is the best known so far.

Theorem 4 (Huang [16]) For all $k \geq 5$, the $k$-coloring problem is NP-complete in the class of $P_6$-free graphs. Moreover, the 4-coloring problem is NP-complete in the class of $P_7$-free graphs.
Together, these results resolve the questions of the complexity of \(k\)-coloring in the class of \(P_t\)-free graphs for all \(k\) and \(t\), except for the case when \(k = 3\) and \(t \geq 8\). For this case, the following is known.

**Theorem 5 (Groenland et al. [11])** For fixed \(t\), the number of 3-colorings of a \(P_t\)-free graph can be found in subexponential time.

Our contribution is an approximation algorithm for the latter case. This line of research was first suggested to us by Chuzhoy [6]. The approximation guarantee of our algorithm is better in the triangle-free case. Even in the triangle-free case, the complexity of 3-coloring \(P_t\)-free graphs is open for \(t \geq 8\), and 4-coloring is known to be \(NP\)-hard for large \(t\):

**Theorem 6 (Huang et al. [17])** The 4-coloring problem is \(NP\)-hard in the class of \(\{P_2, \text{triangle}\}\)-free graphs.

**Approximation.** The hardness of approximating the \(k\)-coloring problem has been in the focus of the research on approximation algorithms. Dinur, Mossel and Regev [7] proved that coloring a 3-colorable graph with \(C\) colors, where \(C\) is any constant, is \(NP\)-hard assuming a variant of the Unique Games Conjecture. More precisely, the assumption is that a certain label cover problem is \(NP\)-hard (where the label cover instances are what the authors call \(\alpha\)-shaped). Even without this assumption, Brakensiek and Guruswami [1] proved that for all \(k \geq 3\), \((2k - 2)\)-coloring a \(k\)-colorable graph is \(NP\)-hard.

On the upside, it is known how to color a 3-colorable graph with relatively few colors in polynomial time, and there has been a long line of subsequent improvements on the number of colors needed. The current state of the art, according to our knowledge, is the following result, which combines a semidefinite programming result by Chlamtac [3] with a combinatorial algorithm for the case of large minimum degree.

**Theorem 7 (Kawarabayashi and Thorup [20])** There is a polynomial time algorithm to color a 3-colorable \(n\)-vertex graph with \(O(n^{0.1998})\) colors.

In this work we combine these two lines of research and strive to use the structure of \(P_t\)-free graphs to give an approximation algorithm for the 3-coloring problem. We are inspired by a result of Gyárfás, who proved the following.

**Theorem 8 (Gyárfás [13])** If \(G\) is a graph with no induced subgraph isomorphic to \(P_t\), then \(\chi(G) \leq (t - 1)^{\omega(G)} - 1\).

Thus, for a graph with no \(P_t\), we can check if it is \((t - 1)^2\)-colorable or not 3-colorable by checking whether it contains a \(K_4\). For a connected graph, Theorem 8 also holds if the requirement of being \(P_t\)-free is weakened to the assumption that there is a vertex \(v\) in \(G\) that does not start an induced \(P_t\) in \(G\). We use a technique similar to the proof of Theorem 8 in the proof of our key lemma, Lemma 1, taking advantage of the fact that the input graph is 3-colorable. This allows us to improve the bound of \((t - 1)^2\) on the number of colors given by Gyárfás’ theorem. We remark that our result is not an improvement of Theorem 8, but incomparable to it.
Our contribution. We prove the following.

**Theorem 9** Let $t \in \mathbb{N}$. There is an algorithm that computes for any 3-colorable $P_t$-free graph $G$

(a) a coloring of $G$ with at most $\max \{5, 2 \left\lceil \frac{t+1}{2} \right\rceil - 2\}$ colors, and a triangle of $G$, or

(b) a coloring of $G$ with at most $\max \{4, \left\lceil \frac{t+1}{2} \right\rceil + 1\}$ colors

with running time $O((3^{t-2} + t^2)|E(G)| + |V(G)|)$.

The proof of this theorem is given in Section 2. The outline of the proof is as follows. Starting at a vertex $v$, our goal is to split the graph into two parts, one that has a bounded-size dominating set $S$, and another that requires few colors. To begin, we let $S = \emptyset$ and proceed by fixing a vertex $v$ and considering components of its non-neighbors. No two such components can contain a path of length $\approx \frac{t}{2}$ ending in a vertex of $N(v)$, and hence there is a vertex $u$ in $N(v)$ such that every component of $G - N[v]$ not containing a neighbor of $u$ can be colored with a small number of colors (inductively); then we replace $v$ by $u$, $S$ by $S \cup \{v\}$, and consider the remaining components, and iterate. If this continues for more than $t$ steps, then the set $S$ of vertices we fixed induces a $P_t$ in $G$ (starting $v - u - \ldots$); thus we may assume we found a coloring of $G - (S \cup N(S))$ with a small number of colors and $|S| \leq t$. The 3-coloring problem in graphs with bounded domination number can be reduced to a constant number of 2SAT instances, which can be solved in linear time [8,9,23], and so we can compute a 3-coloring of $G[S \cup N(S)]$ and add it to the colorings of components; this yields the desired coloring.

There is a variant of this problem where we replace the requirement that $G$ is $P_t$-free with the weaker restriction that in each connected component $G$ has at least one vertex that is not a starting vertex of a $P_t$. We give an algorithm for this harder problem as well, with a worse approximation bound, see Lemmas 1 and 2 below. Additionally, we give a hardness result, Theorem 11, to show that Lemma 1 can probably not be improved.

We remark that our algorithm can easily be implemented such that it takes an arbitrary graph as its input. It then either refutes the graph by outputting that it contains a $P_t$ or that it is not 3-colorable or computes a coloring as promised by Theorem 9. In the case that the graph is refuted for not being 3-colorable, the algorithm can output a certificate that is easily checked in polynomial time. If the graph is refuted because it contains an induced $P_t$, our algorithm outputs the path.

2 Algorithm

Throughout this paper, we use $N(v)$ to denote the set of neighbors of a vertex $v$, and $N(X)$ to denote the set of neighbors of a set $X$. Moreover, we let $N[v] = N(v) \cup \{v\}$ and $N[X] = N(X) \cup X$.

We start with a lemma that uses ideas from the proof of Theorem 8 to color connected graphs in which some vertex does not start a $P_t$. The coloring
it returns, if any, is a 3-coloring as long as $t \leq 4$; in Section 3 we show that 3-coloring becomes NP-hard in this setting for $t \geq 5$, which means that our result is best possible in this sense. In Lemma 2, we bound the running time of the algorithm given in Lemma 1.

**Lemma 1** Let $G$ be connected, $v \in V(G)$, and $t \in \mathbb{N}$ with $t \geq 1$. There is a polynomial-time algorithm that outputs

(a) that $G$ is not 3-colorable, or
(b) an induced path $P_t$ starting with vertex $v$, or
(c) a max $\{2,t-2\}$-coloring of $G$, or
(d) a max $\{3,2t-5\}$-coloring of $G$ and a triangle in $G$.

**Proof** We prove this by induction on $t$ and $|V(G)|$. If $V(G) = \{v\}$, then (c) holds, and we return a 1-coloring of $G$. From now on, we may therefore assume that $|V(G)| > 1$.

We first prove the result for $t \leq 4$. Let $Z = V(G) \setminus N[v]$. Consider a component $C$ of $G[Z]$ (if one exists). By connectivity, there is a vertex $x \in V(C)$ such that $N(x) \cap N(v) \neq \emptyset$. If there is an induced $P_4$ in $G$ starting at $v$, we output it. Otherwise, each neighbor of $x$ in $C$ is adjacent to all of $N(x) \cap N(v)$. Thus $N(y) \cap N(v) = N(x) \cap N(v)$ for every $y \in V(C)$. In particular, if $|C| \geq 2$, we found a triangle.

Color $v$ with color 1, and give each vertex in a singleton component of $Z$ color 1. For each non-singleton component $C$ of $Z$, note that if $C$ is not bipartite, then $G$ is not 3-colorable (and we have outcome (a)). So we may assume $C$ is bipartite, and we color all vertices from one partition class with 1. This is valid since every vertex of $C$ has the same set of neighbors in $N(v)$, and no vertex in $N(v)$ is colored 1. We call $G'$ the subgraph of $G$ that contains all yet uncolored vertices. (So all remaining vertices of $Z$ form singleton components of $G' \setminus N(v)$.)

If $G'$ has no edges, we can color $V(G')$ with color 2 to obtain a valid 2-coloring of $G$, and are done with outcome (c). If $G'$ is bipartite and has an edge $xy$, then we can color $V(G')$ with colors 2 and 3 to obtain a valid 3-coloring of $G$. Observe that if $x,y \in N(v)$, then $G$ has a triangle, and that otherwise, we can assume $x \in N(v)$ and $y \in Z$. In $G$, vertex $y$ belongs to a non-trivial component of $G \setminus N(v)$; thus, as noted above, $G$ has a triangle containing $xy$. In either case, we have outcome (d).

Now suppose that $G'$ is not bipartite, that is, $G'$ has an odd cycle $C_{\ell}$, on vertices $c_1, \ldots, c_{\ell}$, say. Then, for each $c_i$ lying in $Z$, we know that in $G$, there is a vertex $c'_i$ (from the non-trivial component of $Z$ that $c_i$ belongs to) which is adjacent to all three of $c_{i-1}, c_i, c_{i+1}$ (mod $\ell$). So in any valid 3-coloring of $G$, vertices $c_{i-1}$ and $c_{i+1}$ have the same color. Since $\ell$ is odd, no such coloring exists; and we can output (a). This proves the result for $t \leq 4$.

Now let $t \geq 5$, and assume that the result is true for all smaller values of $t$. For every component $C$ of $Z = V(G) \setminus N[v]$, there is a vertex $w_C$ in $N(v)$ with neighbors in $C$. We apply the induction hypothesis (for $t-1$) to $G_C := G[V(C) \cup \{w_C\}]$ and $w_C$. If this subgraph is not 3-colorable, neither
is $G$ (and we have outcome (a)). If there is an induced $P_{t-1}$ starting at $w_C$, then we can add $v$ to this path and have found an induced $P_t$ in $G$ starting at $v$, giving outcome (b). If neither outcome (a) nor outcome (b) occurred in any component, then each component $C$ of $G[Z]$ (without $w_C$) can be colored with $2(t-1) - 5$ colors if the algorithm detected a triangle in $G_C$, and with $t-3$ colors otherwise.

If $N(v)$ is a stable set, and no triangle was detected, then we color each component of $G[Z]$ with $t-3$ colors (which can be repeated), and use one more color for $N(v)$, and repeat one of the colors from $Z$ for $v$ to obtain a $(t-2)$-coloring of $G$, obtaining outcome (c).

Therefore, we may assume that the algorithm detected a triangle in $G[N[v]]$ or some $G_C$, and we output this triangle. If $G[N(v)]$ is not bipartite, then $G$ is not 3-colorable. Otherwise, we color each component of $G[Z]$ with the same at most $2(t-1) - 5$ colors, color $N(v)$ with at most two new colors, and repeat a color from $Z$ for $v$. Then, this yields a coloring of $G$ with $2t - 5$ colors, and we found a triangle, which is outcome (d).

\begin{lemma}
The algorithm from Lemma 1 can be implemented with a running time of $O(t|E(G)|)$ for a connected input graph $G$.
\end{lemma}

\begin{proof}
For $t \leq 4$, we can compute $N(v)$ and $Z = V(G) \setminus N[v]$ in time $O(|E(G)|)$. The components of $Z$ can be found in linear time. By going through each vertex $w$ in $N(v)$, and for each such $w$, going through each component $C$ of $Z$, we can check that $w$ has exactly 0 or $|V(C)|$ neighbors in $C$; if this is not true for some component $C$, then we have found a $P_t$ starting at $v$, obtaining outcome (b).

Otherwise, color $v$ with color 1, as well as all components in $Z$ of size 1. If a component $C$ contains two or more vertices, then we check if it is bipartite (in linear time); if not, then since there is a neighbor $w$ of $C$ in $N(v)$ and $w$ is complete to $C$, we output that $G$ is not 3-colorable for outcome (a). If $C$ is bipartite, we choose one of the partition classes of the bipartition, and give all vertices in this class color 1.

Let $G'$ be the remaining graph after removing all vertices colored so far. We check if $G'$ has an edge; if not, then we can give a 2-coloring of $G'$ and output (c). If $G'$ has an edge $xy$, then check if $G'$ is bipartite. If so, we can get a valid 3-coloring of $G$. Moreover, $xy$ lies in a triangle (either because $x, y \in N(v)$ or because $x$ and $y$ have a common neighbor in $Z$), and we can output (d). So assume we found that $G'$ is not bipartite, that is, we found an odd cycle $C_t$ in $G'$, on vertices $c_1, \ldots, c_t$, say. For each $c_i \in Z \cap V(G')$, there is a vertex $c'_i \in Z \cap V(G)$ adjacent to all three of $c_{i-1}, c_i, c_{i+1}$ (mod $t$), hence we can output (a), as vertices $c'_i, V(C_t)$ and $v$ induce an obstruction to 3-coloring $G$.

Now let $t \geq 5$. We compute $N(v)$ in time $|d(v)|$, compute components of $G - N[v]$ in linear time, check if $N(v)$ is bipartite in linear time (if not, return that $G$ is not 3-colorable), check if $N(v)$ contains two adjacent vertices, and correspondingly 1 or 2-color $N(v)$. Then we go through $N(v)$ to find a neighbor $w_C$ for each component $C$ of $G - N[v]$ and run the algorithm with vertex $w_C$ and parameter $t-1$ on the graph $G[V(C) \cup \{w_C\}]$. 

\end{proof}
If the outcome in any component $C$ is an induced $P_{t-1}$ starting at $w_c$, we can add $v$ at the start of the path and get outcome (b). If some component is not 3-colorable, then neither is $G$, giving outcome (a). Otherwise, we find the necessary colorings (and possibly a triangle) to output (c) or (d).

Note that no edge occurs in two components, therefore we require $O(|E(G)|)$ processing time before using recursion and a total amortized running time of at most $O((t-1)|E(G)|)$ for recursive calls of the algorithm, which implies the overall running time.\hfill $\square$

In the following, we will use a slightly modified version of Lemma 1:

**Corollary 1** Let $G$ be connected, $v \in V(G)$, and $t \in \mathbb{N}$. Then, there is an algorithm that outputs

(a) that $G$ is not 3-colorable, or
(b) an induced path $P_t$ starting with vertex $v$, or
(c) a max $\{1, t-2\}$-coloring of $G-v$, or
(d) a max $\{2, 2t-5\}$-coloring of $G-v$ and a triangle in $G$

with running time $O(|tE(G)|)$.

**Proof** This is a direct consequence of Lemma 1 unless $t \leq 3$. If $t \leq 3$, then we can find an induced $P_t$ starting at $v$ unless $v$ is adjacent to every vertex in $G-v$. So assume $v$ is adjacent to every other vertex. If $G-v$ is not bipartite, then $G$ is not 3-colorable. Otherwise, $G-v$ is 2-colorable and the algorithm detects a triangle, or $G-v$ is 1-colorable.\hfill $\square$

For a set $S$ of vertices of $G$, we let $F(S)$ denote the smallest set such that $S \subseteq F(S)$ and no vertex in $G-F(S)$ has two adjacent neighbors in $F(S)$. Note that $F(S)$ can be computed by starting with $S$ and repeatedly adding vertices that have two adjacent neighbors in the current set; since $G$ is finite, this process terminates and the set at termination has the required properties. In a 3-coloring, the colors of the vertices in $S$ uniquely determine the colors of all vertices in $F(S)$.

Below, in Lemma 3, we prove the main technical result of this paper, and in Lemma 4 we analyze the running time of the algorithm from Lemma 3.

**Lemma 3** Let $G$ be connected, $v \in V(G)$ and $k, t \in \mathbb{N}$ with $k \geq 1$, $t \geq 2$. There is a polynomial-time algorithm that outputs

(a) that $G$ is not 3-colorable, or
(b) an induced path $P_t$ in $G$, or
(c) an induced path $P_k$ in $G$ starting in $v$, or
(d) a set $S$ of size at most max $\{1, k-2\}$ with $v \in S$, and a max $\{1, \left\lfloor \frac{t-1}{2} \right\rfloor - 2\}$-coloring of $G-N[F(S)]$, or
(e) a set $S$ of size at most max $\{1, k-2\}$ with $v \in S$, and a max $\{2, 2 \left\lfloor \frac{t-1}{2} \right\rfloor - 5\}$-coloring of $G-N[F(S)]$, and a triangle in $G$.\hfill $\square$
Proof We prove this by induction on $k$ and $|V(G)|$. If $V(G) = \{v\}$, then outcome (d) holds with $S = \{v\}$; therefore we may assume from now on that $|V(G)| > 1$. If $k \leq 2$, then outcome (c) holds. If $k = 3$, then the result follows since either $N[v] = V(G)$ or $G$ contains a $P_3$ starting in $v$. In the latter case, outcome (c) holds, and in the former case, we return $S = \{v\}$ and outcome (d) holds.

Now let $k > 3$. Note that we can assume $k \leq t$, because otherwise we can run the algorithm for $k$ set to $t$, and all outcomes except (c) will be valid for the original $k$ as well, and if we do get outcome (c), we can use it as outcome (b) instead. Furthermore, if $k \leq \left\lceil \frac{k-1}{2} \right\rceil$, then Corollary 1 with input $G$, $v$ and $k$ yields a coloring of $G - v$ with at most as many colors as claimed in the lemma. The result of the lemma follows by setting $S = \{v\}$.

So we may assume that $k > \left\lceil \frac{k-1}{2} \right\rceil$. Consider $Z = V(G) \setminus N[v]$. Let $C = \{C_1, \ldots, C_r\}$ be the list of components of $G[Z]$, and let $D = \{D_1, \ldots, D_l\}$ be the list of components of $G[N(v)]$. We now describe a procedure during which

at each step, we color one of the components of $C$, and then put it aside, and continue working with the remaining graph, until one component $D \in \mathcal{D}$ has neighbors in all remaining components of $C$.

The details are as follows. We may assume that for every component in $\mathcal{D}$, some component of $\mathcal{C}$ has no neighbors in $\mathcal{D}$. Now choose a component $D \in \mathcal{D}$ that has neighbors in as many components of $\mathcal{C}$ as possible. Then some component $C' \in \mathcal{C}$ has no neighbor in $D$, and therefore, there is a component $D' \in \mathcal{D}$ with a neighbor in $C'$ by the connectivity of $G$. But $C'$ has a neighbor in $D'$ and not $D$, and so by choice of $D$, $D'$ cannot have a neighbor in all components $C \in \mathcal{C}$ in which $D$ has a neighbor; now let $C$ be a component in $\mathcal{C}$ such that $D$ has a neighbor in $C$ and $D'$ does not. Let $x \in V(D)$ such that $x$ has a neighbor in $C$, and let $x' \in V(D')$ such that $x'$ has a neighbor in $C'$.

Then, we apply Corollary 1 to $G[\{x\} \cup V(C)]$ (with parameter $\left\lceil \frac{k-1}{2} \right\rceil$ and vertex $x$) and to $G[\{x'\} \cup V(C')]$ (with parameter $\left\lceil \frac{k-3}{2} \right\rceil$ and vertex $x'$). If either of these graphs is not 3-colorable, then $G$ is not 3-colorable. If, in both cases, there is an induced $P_{\left\lceil \frac{k-1}{2} \right\rceil}$ starting at $x$ and at $x'$, respectively, then, since $x$ has no neighbors in $V(C') \cup V(D')$ and $x'$ has no neighbors in $V(D) \cup V(C)$, we can combine them, using the path $xx'$, to obtain an induced $P_{\left\lceil \frac{k-1}{2} \right\rceil + 1}$ in $G$, which contains an induced $P_4$. Thus, we can assume that for at least one of the two components, we found a coloring instead. In particular, we found a coloring of $C'$ or of $C'$ with max $\{1, \left\lceil \frac{k-1}{2} \right\rceil - 2\}$ colors, or a triangle in $G$, and a coloring of $C$ or of $C'$ with max $\{2, 2 \left\lceil \frac{k-1}{2} \right\rceil - 5\}$ colors. We then remove the component with the coloring and continue.

Finally, we arrive at a point where there is a component $D \in \mathcal{D}$ that has neighbors in all remaining components of $\mathcal{C}$. Note that if $S$ is a set that includes $v$ and any vertex $x \in D$, then $F(S) \supseteq V(D)$ and thus $N(V(D)) \subseteq N[F(S)]$. Therefore, we call a remaining component of $\mathcal{C}$ good if it is contained in $N(V(D))$, and bad otherwise. Our goal is to find a vertex $x \in V(D)$ with neighbors in all bad components.
Suppose that there is no vertex in $V(D)$ with neighbors in all bad components. Then, we can find two bad components $C, C'$ among the remaining components of $C$ such that $C$ has a neighbor $y$ in $D$, $C'$ has a neighbor $y'$ in $D$, $y$ has no neighbors in $C'$ and $y'$ has no neighbors in $C$. As before, we can find these components by choosing $y$ with neighbors in as many bad components as possible, and then letting $y'$ be a vertex with a neighbor in a bad component $C'$ in which $y$ does not have a neighbor. Consequently, $y'$ has no neighbor in at least one bad component $C$ in which $y$ does have a neighbor.

As $C$ and $C'$ are bad, there exist components $E$ and $E'$, of $C \setminus N(V(D))$, and of $V(C') \setminus N(V(D))$, respectively. Let $x$ be the second vertex on a shortest path $P$ from $E$ to $y$, and define $x'$ and $P'$ analogously; that is, $x$ is the unique vertex of $P$ such that $x \not\in V(E)$ and $x$ has a neighbor in $E$, and similarly, $x'$ is the unique vertex of $P'$ such that $x' \not\in V(E')$ and $x'$ has a neighbor in $E'$. Since $y, y' \in V(D)$, and $E$ is disjoint from $N(V(D))$, it follows that $x \neq y$ and $x' \neq y'$. Since $C$ and $C'$ are different components of $G - N[v]$, it follows that there are no edges between $V(P)$ and $V(P')$ except possibly $yy'$.

We apply Corollary 1 to $G[[x] \cup V(E)]$ (with parameter $\left\lceil \frac{t-2}{2} \right\rceil$ and vertex $x$) and to $G[[x'] \cup V(E')]$ (with parameter $\left\lceil \frac{t-2}{2} \right\rceil$ and vertex $x'$). If either of these two graphs is not 3-colorable, then $G$ is not 3-colorable. If, in both cases, there is an induced $P_t \left\lceil \frac{t-2}{2} \right\rceil$, say $Q$ starting at $x$ and $Q'$ starting at $x'$, respectively, then we can combine these paths to an induced path $R$ of length at least 1, and therefore $y, y' \not\in V(Q) \cup V(Q')$, it follows that $R$ has at least $2 \left\lceil \frac{t-2}{2} \right\rceil + 2 \geq t$ vertices.

Thus, we may assume that for at least one of $G[[x] \cup V(E)], G[[x'] \cup V(E')]$, we found a coloring instead. In particular, we found a coloring of $E$ or of $E'$ with $\{1, \left\lfloor \frac{t-2}{2} \right\rfloor - 2\}$ colors, or a triangle in $G$, and a coloring with $\{2, 2 \left\lfloor \frac{t-2}{2} \right\rfloor - 5\}$ colors. We then remove the component with the coloring and continue.

We are left with the case when there is a single vertex $v' \in V(D)$ that has neighbors in all remaining components of $C$, except possibly those contained in $N(V(D))$. Let $V'$ be the set of vertices in the remaining components of $C$ that are not contained in $N(V(D))$. Then we can apply the induction hypothesis with $k - 1, t, v'$ to $G' = G[V' \cup \{v'\}]$. If $G'$ is not 3-colorable, neither is $G$. If $G'$ contains an induced path $P_t$, so does $G$. If $G'$ contains an induced path $P_{k-1}$ starting in $v'$, then we can add $v$ to this path to obtain an induced path $P_k$ starting in $v$. If there is a set $S$ of size at most $k - 3$ with $v' \in S$, and a max $\{1, \left\lfloor \frac{t-2}{2} \right\rfloor - 2\}$-coloring of $G' - N[F(S)]$ or a max $\{2, 2 \left\lfloor \frac{t-2}{2} \right\rfloor - 5\}$-coloring of $G' - N[F(S)]$ and a triangle, then we proceed as follows. We add $v$ to $S$, and now $S$ has size at most $k - 2$. Moreover, since both $v$ and $v'$ are in $S$, we know that $V(D) \subseteq F(S)$, thus all vertices in $N[v] \cup N[D]$ are in $N[F(S)]$. We colored different components of $Z \setminus N(V(D))$ at different stages, but we can reuse the colors used on these components. Therefore, this leads to
outcome (d) or (e), depending on whether the algorithm detected a triangle at any stage or not.

Lemma 4 The algorithm from Lemma 3 can be implemented with running time $O\left(k\left\lceil \frac{1}{2} \right\rceil |E(G)|\right)$ for a connected input graph $G$.

Proof For $k < 3$, we can check in linear time if (b) or (d) holds. For $k = 3$, we check in linear time if $V(G) = N[v]$; in this case, the algorithm returns $S = \{v\}$, and outcome (d) holds. Otherwise, $G$ contains a vertex $w \notin N[v]$ with a neighbor $u$ in $N(v)$, and $uvw$ is a $P_3$; so outcome (b) holds, and we can find $u$ and $w$ in linear time. For $k > 3$, we compute $Z = V(G) \setminus N[v]$ and the components $C_1, \ldots, C_r$ of $G[Z]$ and the components $D_1, \ldots, D_r$ of $G[N(v)]$ as in the proof of Lemma 3; all this can be done in linear time.

Now, subsequently, for $j = 1, \ldots, r$, we consider $D_j$, and some of the $C_i$ adjacent to it, and after possibly coloring and deleting these components $C_i$, we might also delete $D_j$. More precisely, for $j = 1, \ldots, r$, we consider those $C_i$ that only have neighbors in $D_j$. For each such $C_i$, choose a neighbor $x_{ij}$ in $V(D_j)$ and apply Corollary 1 to $G[V(C_i) \cup \{x_{ij}\}]$. If this graph is not 3-colorable, then neither is $G$. If the algorithm returns a coloring (and possibly a triangle), then this is the coloring we will use in $C_i$, as explained in the proof of Lemma 3, so we can delete $C_i$. (But, if the algorithm found a triangle, we shall remember this triangle for a possible output, at least if it is the first one to be found.) Otherwise, the algorithm returns a path of length $\left\lceil \frac{1}{2} \right\rceil$, which we keep. After going through all $C_i$ with neighbors only in $D_j$, we delete $D_j$ if the algorithm always returned a coloring (or if there were no $C_i$ to consider), since we guarantee that $v \in S$ and therefore $V(D_j) \subseteq N(F(S))$, and in this case we have also already processed all components $C_i$ with a neighbor in $D_j$. That is, we keep $D_j$ if and only if for some $i$ we found a path starting from $x_{ij}$ with interior in $V(C_i)$.

The amortized time it takes to process all $D_j$ is $O\left(\left\lceil \frac{1}{2} \right\rceil |E(G)|\right)$. This is so because every component $C_i$ is used for the algorithm from Corollary 1 at most once.

In the end, if there are $D_j, D_{j'}$ with $j \neq j'$ that we did not delete, then there is a component $C_i$ that only has neighbors in $D_j$, and another component $C_{i'}$ that only has neighbors in $D_{j'}$, and both $C_i$ and $C_{i'}$ contain a $P_{\left\lceil \frac{1}{2} \right\rceil}$ in their interior, starting at $x_{ij}$ and $x_{i'j'}$ respectively. By connecting them using the middle segment $x_{ij}x_{i'j'}$, we find a $P_{3}$ in $G$ that we can output as outcome (b). Otherwise, there is only one $D_j$ left at the end. Since whenever we deleted a component $D_{j'}$, we ensured that each remaining $C_i$ has a neighbor in some $D_j$ with $j \neq j'$, this means that $D_j$ has neighbors in all $C_i$ that we did not color yet.

Let $Z''$ be the set of vertices of remaining components $C_i$, and let $Z''' = Z'' \setminus N(V(D_j))$. Each component of $G[Z''']$ is contained in some component $C_i$ and thus it has a neighbor in $N(V(D_j))$. Therefore, we can apply the same argument as before to components of $G[Z''']$ and components of $N(V(D_j))$ with neighbors in them. Whenever the algorithm for Corollary 1 outputs a
coloring we keep it (and we also keep the possibly found triangle, if it is the first triangle to be found), and if it outputs that a component is not 3-colorable, then $G$ is not 3-colorable, and if there is a path $P_{\lceil \frac{t-2}{2} \rceil}$, we keep track of it. (Note that components of $N(V(D_j))$ that are not adjacent to $Z''$ get deleted automatically.) When this terminates, if there are still two components of $N(V(D_j))$, then there are two paths we can combine to a $P_t$ as in Lemma 3. Otherwise, a single component $D^*$ of $N(V(D_j))$ has a neighbor in all remaining components $C_1', \ldots, C_n'$, so there is a vertex $v' \in V(D_j)$ such that $G'[\{v'\} \cup D^* \cup C_1' \cup \cdots \cup C_n']$ is connected, and $v'$ is the only neighbor of $v$ in that subgraph. Next, we apply induction for $k-1$ with root vertex $v$ on that set. If this finds a set $S$ and a coloring, we add $v$ to $S$. If it finds a path $P_{k-1}$, we add $v$ to the path. If it finds a $P_t$, we output it. If it is not 3-colorable, then neither is $G$.

The total running time of the recursive application of the algorithm is $O((k-1) \left\lceil \frac{t-2}{2} \right\rceil |E(G)|)$, and all preprocessing steps leading there can be implemented with a running time of $O \left( \left\lceil \frac{t-1}{2} \right\rceil |E(G)| \right)$, which implies the result.

$\square$

We can now give the proof of our main result, which we restate:

**Theorem 10** Let $t \in \mathbb{N}$. There is an algorithm that computes for any 3-colorable $P_t$-free graph $G$

(a) a coloring of $G$ with at most $\max \{5, 2 \left\lceil \frac{t-1}{2} \right\rceil - 2\}$ colors, and a triangle of $G$, or

(b) a coloring of $G$ with at most $\max \{4, \left\lceil \frac{t-1}{2} \right\rceil + 1\}$ colors

with running time $O((3^{t-2} + t^2)|E(G)| + |V(G)|)$.

**Proof of Theorem 10.** We use the algorithm from Lemma 3 with $k = t$. By Lemma 4, the running time of this algorithm is $O(t^6|E(G)|)$. Since only outcomes (d) and (e) can occur in this setting, it is sufficient to show that if $G$ is 3-colorable, we can find a 3-coloring of $N(F(S))$ in time $O(3^{t-2}|E(G)|)$, as follows: For each vertex in the set $S$, we try each possible color for a total of at most $3^{t-2}$ possibilities. From the definition of $F(S)$, it follows that this determines the color of every vertex in $F(S)$, and hence for every vertex in $N(F(S))$, there are at most two possible colors. Thus, we reduced our problem to a 2-list-coloring problem, which can be solved in linear time $O(|E(G)| + |N(F(S))|) = O(|E(G)|)$ (since $|N(F(S))| \leq |E(G)|$) by reduction to 2SAT [8, 9, 23]. It follows that, if $G$ is 3-colorable, we can 3-color $N(F(S))$ in time $O(3^{t-2}|E(G)|)$, and add these three new colors to the coloring from Lemma 3.

The total running time follows from the running time of the algorithm in Lemma 4 in addition to an algorithm determining the connected components of $G$. $\square$

By combining Theorem 9 with Lemma 1, we obtain the algorithms for coloring 3-colorable $P_t$-free graphs with the number of colors shown in Table 1 (if there is a triangle) and Table 2 (if there is no triangle). For $t$ larger than shown
in the table, Theorem 9 uses a smaller number of colors. In both tables, “Best option” shows the number of colors used by the better algorithm (assuming a 3-colorable \(P_t\)-free input graph), which is the algorithm from Lemma 1 for small values of \(t\), and the algorithm from Theorem 9 for large values of \(t\).

| \(t\) | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | > 11 |
|-------|---|---|---|---|---|---|---|----|----|-----|
| \(\max\{3, 2t - 5\}\) | 3 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 |
| \(\max\{5, 2\left\lceil \frac{t-1}{2} \right\rceil - 2\}\) | 5 | 5 | 5 | 5 | 6 | 6 | 8 | 8 | 8 |
| Best option | 3 | 3 | 3 | 5 | 5 | 6 | 6 | 8 | 8 | 2 \(\left\lceil \frac{t-1}{2} \right\rceil - 2\) |

Table 1 Number of colors we use for a 3-colorable \(P_t\)-free graph if there is a triangle

| \(t\) | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | > 11 |
|-------|---|---|---|---|---|---|---|----|----|-----|
| \(\max\{2, t - 2\}\) | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| \(\max\{4, \left\lceil \frac{t-1}{2} \right\rceil + 1\}\) | 4 | 4 | 4 | 4 | 4 | 5 | 5 | 6 | 6 | \(\left\lceil \frac{t-1}{2} \right\rceil + 1\) |
| Best option | 2 | 2 | 3 | 4 | 4 | 5 | 5 | 6 | 6 | \(\left\lceil \frac{t-1}{2} \right\rceil + 1\) |

Table 2 Number of colors we use for a 3-colorable \(P_t\)-free graph if there is no triangle

3 Hardness result

In this section, we show that improving Lemma 1 is hard. More precisely:

**Theorem 11** Let \(G\) be a connected graph and \(v \in V(G)\) such that there is no induced \(P_t\) in \(G\) starting at \(v\). Then, deciding \(k\)-colorability on this class of graphs is NP-hard if \(k \geq 4\) and \(t \geq 3\) or if \(k = 3\) and \(t \geq 5\). It can be solved in polynomial time if \(t \leq 2\) or if \(k = 3\) and \(t \leq 4\).

**Proof** For the polynomial time solvability, observe that if \(t \leq 2\), then \(|V(G)| \leq 1\). If \(k = 3, t \leq 4\), then the result follows from Lemma 1.

For the hardness, first consider the case \(k \geq 4, t \geq 3\). In this case, we can reduce the 3-coloring problem to this problem by taking any instance \(G\) and adding a clique of size \(k - 3\) complete to \(G\). Then, no vertex in this clique starts a \(P_3\), but the resulting graph is \(k\)-colorable if and only if \(G\) is 3-colorable.

It remains to consider the case \(k = 3, t \geq 5\). We show a reduction from the NP-complete problem NAE-3Sat [10]. An instance of NAE-3Sat is a boolean formula with variables \(x_1, \ldots, x_n\) and clauses \(C_1, \ldots, C_m\), where each

\[1 \text{ If } t = 5, \text{ we can improve the number of colors required if there is a triangle to 3, because it cannot happen that there are two components } C, C' \text{ of } G - N[v] \text{ and components } D, D' \text{ of } G[N(v)] \text{ such that } C \text{ has a neighbor in } D \text{ but not } D', \text{ and } C' \text{ has a neighbor in } D' \text{ but not } D', \text{ because this already yields an induced } P_5. \text{ Thus, by induction, all vertices will be in } N[F(S)], \text{ where we can test for 3-colorability as described in Theorem 9.} \]
clause contains exactly three literals (variables or their negations). It is a Yes-instance if and only if there is an assignment of the variables as true or false such that for every clause, not all three literals in the clause are true, and not all three are false.

We construct a graph \( G \) as follows: \( G \) contains a vertex \( v \), vertices labeled \( x_i \) and \( \overline{x}_i \) for \( 1 \leq i \leq n \), and a triangle \( T_j \) for each clause \( C_j \). The vertex \( v \) is adjacent to all vertices \( x_i \) and \( \overline{x}_i \), but not to any of the triangles. For each \( i \), \( x_i \) is adjacent to \( \overline{x}_i \). For each clause \( C_j \), we assign each literal a vertex of the triangle \( T_j \), and connect this vertex to the literal (the vertex labeled \( x_i \) or \( \overline{x}_i \)). There are no other edges in \( G \).

Then, there is no \( P_5 \) starting at \( v \) in \( G \), because such a path would have to contain exactly one of the vertices labeled \( x_i \) and \( \overline{x}_i \), and this would be the second vertex of the path. As there are no edges between the triangles \( T_j \), all remaining vertices of the \( P_5 \) would have to be in one triangle \( T_j \). But no triangle can contain a \( P_3 \). Therefore, \( G \) is a valid instance.

It remains to show that \( G \) is 3-colorable if and only if the instance of NAE-3Sat is a Yes-instance. If \( G \) has a 3-coloring, then the neighbors of \( v \) are 2-colored (say with colors 1 and 2) and \( x_i \) never receives the same color as \( \overline{x}_i \). Assign the variables so that literals colored 1 are true, and those colored 2 are false. Then, if there is a clause \( C_j \) such that all of its literals are true, this means that each vertex of \( T_j \) has a neighbor colored 1, so \( T_j \) uses only colors 2 and 3, which is impossible in a valid coloring of a triangle. For the same reason, there cannot be a clause such that all of its literals are false. Thus, \( G \) was constructed from a Yes-instance.

Conversely, if the instance we started with is a Yes-instance, we color \( v \) with color 3, true literals with color 1, and false literals with color 2. For each triangle \( T_j \), one of the vertices adjacent to a true literal is colored 2, one of the vertices adjacent to a false literal is colored 1, and the remaining vertex is colored 3. This is a valid 3-coloring of \( G \). \( \square \)

4 Conclusion

In this paper we showed how to color a given 3-colorable \( P_t \)-free graph with a number of colors that is \( t \), roughly. The running time of our algorithm is of the form \( O(f(t) \cdot n^{O(1)}) \), when the input graph has \( n \) vertices, and thus FPT in the parameter \( t \). (The class FPT contains the fixed parameter tractable problems, which are those that can be solved in time \( f(k) \cdot |x|^{O(1)} \) for some computable function \( f(k) \).

In view of this, it seems to be an intriguing question whether the 3-coloring problem is fixed-parameter tractable when parameterized by the length of the longest induced path. That is, whether there is an algorithm with running time \( O(f(t) \cdot n^{O(1)}) \) that decides 3-colorability in \( P_t \)-free graphs. So far, however, it is not even known whether there is an XP algorithm to decide 3-colorability in \( P_t \)-free graphs. (XP is the class of parameterized problems that can be solved in time \( O(n^{f(k)}) \) for some computable function \( f(k) \).) If such an XP-algorithm
existed, this would show that the problem is in P whenever $t$ is fixed. Therefore, attempting to prove W[1]-hardness seems to be more reasonable than trying to prove that the problem is in FPT.

Another question we addressed is $k$-coloring connected graphs such that some vertex is not the end vertex of an induced $P_t$. We showed that coloring in this case is NP-hard whenever $k = 3$ and $t \geq 5$, or $k \geq 4$ and $t \geq 3$. Lemma 1 gives a simple algorithm for an $f(t)$-approximate coloring for $k = 3$ and any $t$, and it would be interesting to have a complementing result proving hardness of approximation. On the other hand, any improvement of Lemma 1 would immediately yield an improvement of our main result.

Acknowledgements We are thankful to Paul Seymour for many helpful discussions. We thank Stefan Hougardy for pointing out [20] to us.

This material is based upon work supported in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under grant number W911NF-16-1-0404.

References

1. Brakensiek, Joshua, and Venkatesan Guruswami. New hardness results for graph and hypergraph colorings. In: LIPIcs-Leibniz International Proceedings in Informatics, vol. 50. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
2. Bonomo, Flavia, Maria Chudnovsky, Peter Maceli, Oliver Schaudt, Maya Stein, and Mingxian Zhong. Three-coloring and list three-coloring graphs without induced paths on seven vertices. appeared on-line in Combinatorica (2017)
3. E. Chlamtac. Approximation algorithms using hierarchies of semidefinite programming relaxations. In: 48th Annual IEEE Symposium on Foundations of Computer Science (2007)
4. Chudnovsky, Maria, Sophie Spirkl, and Mingxian Zhong. Four-coloring $P_6$-free graphs. I. Extending an excellent precoloring. arXiv preprint arXiv:1802.02282 (2018)
5. Chudnovsky, Maria, Sophie Spirkl, and Mingxian Zhong. Four-coloring $P_6$-free graphs. II. Finding an excellent precoloring. arXiv preprint arXiv:1802.02283 (2018)
6. Chuzhoy, Julia. Private communication.
7. Dinur, Irit, Elchanan Mossel, and Oded Regev. Conditional Hardness for Approximate Coloring. SIAM Journal on Computing 39, pp. 843–873 (2009)
8. Edwards, Keith. The complexity of colouring problems on dense graphs. Theoretical Computer Science 43, pp. 337–345 (1986)
9. Erdős, Paul, Arthur L. Rubin, and Herbert Taylor. Choosability in graphs. Congressus Numerantium 26, pp. 125–157 (1979)
10. Garey, Michael R., and David S. Johnson. A Guide to the Theory of NP-Completeness. WH Freemann, New York (1979)
11. Groenland, Carla, Karolina Okrassa, Pawel Rzążewski, Alex Scott, Paul Seymour, and Sophie Spirkl. H-colouring $P_t$-free graphs in subexponential time. arXiv preprint arXiv:1803.05396 (2018)
12. Golovach, Petr A., Matthew Johnson, Daniel Paulusma, and Jian Song. A Survey on the Computational Complexity of Colouring Graphs with Forbidden Subgraphs. Journal of Graph Theory 84.4, pp. 331–363 (2017)
13. Gyárfás, András. Problems from the world surrounding perfect graphs. Applicationes Mathematicae 19.3–4, pp. 413–441 (1987)
14. Hoàng, Chính T., Marcin Kamiński, Vadim Lozin, Joe Sawada, and Xiao Shu. Deciding $k$-colorability of $P_t$-free graphs in polynomial time. Algorithmica 57.1, pp. 74–81 (2010)
15. Holyer, Ian. The NP-completeness of edge-coloring. SIAM Journal on Computing 10, pp. 718–720 (1981)
16. Huang, Shenwei. Improved complexity results on $k$-coloring $P_t$-free graphs. European Journal of Combinatorics 51, pp. 336–346 (2016)
17. Huang, Shenwei, Matthew Johnson, and Daniël Paulusma. *Narrowing the complexity gap for colouring $(C_s, P_t)$-free graphs.* The Computer Journal 58.11, pp. 3074–3088 (2015)
18. Kamiński, Marcin, and Vadim Lozin. *Coloring edges and vertices of graphs without short or long cycles.* Contributions to Discrete Mathematics 2, pp. 61–66 (2007)
19. Karp, Richard. *Reducibility among combinatorial problems.* In R. Miller and J. Thatcher, editors, Complexity of Computer Computations, pages 85–103. Plenum Press, New York (1972)
20. Kawarabayashi, Ken-ichi, and Mikkel Thorup. *Coloring 3-colorable graphs with $o(n^{1/5})$ colors.* LIPIcs-Leibniz International Proceedings in Informatics. Vol. 25. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2014)
21. Král, Daniel, Jan Kratochvíl, Zsolt Tuza, and Gerhard J. Woeginger. *Complexity of coloring graphs without forbidden induced subgraphs.* Proceedings of WG 2001: 254–262, (2001)
22. Leven, Daniel, and Zvi Galil. *NP-completeness of finding the chromatic index of regular graphs.* Journal of Algorithms 4, pp. 35–44 (1983)
23. Vizing, Vadim G. *Coloring the vertices of a graph in prescribed colors.* Diskret. Analiz 29.3 (1976)