


Tight Lower Bounds for List Edge Coloring

Łukasz Kowalik¹

Institute of Informatics, University of Warsaw, Poland

kowalik@mimuw.edu.pl

 <https://orcid.org/0000-0002-7546-2969>

Arkadiusz Socała²

Institute of Informatics, University of Warsaw, Poland

arkadiusz.socala@mimuw.edu.pl

Abstract

The fastest algorithms for edge coloring run in time $2^m n^{O(1)}$, where m and n are the number of edges and vertices of the input graph, respectively. For dense graphs, this bound becomes $2^{\Theta(n^2)}$. This is a somewhat unique situation, since most of the studied graph problems admit algorithms running in time $2^{O(n \log n)}$. It is a notorious open problem to either show an algorithm for edge coloring running in time $2^{o(n^2)}$ or to refute it, assuming the Exponential Time Hypothesis (ETH) or other well established assumptions.

We notice that the same question can be asked for list edge coloring, a well-studied generalization of edge coloring where every edge comes with a set (often called a *list*) of allowed colors. Our main result states that list edge coloring for simple graphs does not admit an algorithm running in time $2^{o(n^2)}$, unless ETH fails. Interestingly, the algorithm for edge coloring running in time $2^m n^{O(1)}$ generalizes to the list version without any asymptotic slow-down. Thus, our lower bound is essentially tight. This also means that in order to design an algorithm running in time $2^{o(n^2)}$ for edge coloring, one has to exploit its special features compared to the list version.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness, Theory of computation → Design and analysis of algorithms, Mathematics of computing → Graph coloring

Keywords and phrases list edge coloring, complexity, ETH lower bound

Digital Object Identifier 10.4230/LIPIcs.SWAT.2018.28

Acknowledgements We thank an anonymous reviewer for numerous useful remarks.

1 Introduction

An edge coloring of a graph $G = (V, E)$ is a function $c : E \rightarrow \mathbb{N}$ which has different values (called colors) on incident edges. This is one of the most basic graph concepts with plethora of results, including classical theorems of Vizing, Shannon and Kőnig. In the decision problem EDGE COLORING we are given a simple graph G and an integer k . The question is if G can be edge colored using only k colors. This is an NP-complete problem, as shown by Holyer [9], similarly to many other natural graph decision problems like CLIQUE, VERTEX COLORING, HAMILTONICITY or SUBGRAPH ISOMORPHISM. However, there is an intriguing difference between our understanding of EDGE COLORING and most of the studied graph problems,

¹ The work of Ł. Kowalik is a part of the project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677651).

² Supported by the National Science Centre of Poland, grant number 2015/17/N/ST6/01224.



© Łukasz Kowalik and Arkadiusz Socała;
licensed under Creative Commons License CC-BY

16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018).

Editor: David Eppstein; Article No. 28; pp. 28:1–28:12

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

including the four mentioned above. Namely, the latter ones admit algorithms running in time $2^{O(n \log n)}$, and often even $2^{O(n)}$ for an n -vertex input graph, while it is not known whether EDGE COLORING can be solved in time $2^{o(n^2)}$. Indeed, the fastest known algorithm for edge coloring is obtained by applying the vertex coloring algorithm of Björklund, Husfeldt and Koivisto [2] to the line graph of the input graph. As a result, we get an edge coloring algorithm which, for any graph with m edges and n vertices, runs in time $2^m n^{O(1)}$ and exponential space, which is $2^{\Theta(n^2)}$ for dense graphs. The only progress towards a tailor-made approach for edge coloring is the more recent algorithm of Björklund, Husfeldt, Kaski and Koivisto [1] which still runs in time $2^m n^{O(1)}$ but uses only polynomial space. In this context it is natural to ask for a lower bound. Clearly, any superpolynomial lower bound would imply $P \neq NP$. However, a more feasible goal is to prove a meaningful lower bound under the assumption of a well established conjecture, like Exponential Time Hypothesis (ETH, see Section 2 for a precise formulation). The reduction of Holyer, combined with standard tools (see Section 2) proves that EDGE COLORING does not admit an algorithm running in time $2^{o(m)}$ or $2^{o(n)}$, unless ETH fails. At the open problem session of Dagstuhl Seminar 08431 in 2008 [7] it was asked to exclude $2^{O(n)}$ algorithms, assuming ETH. Despite considerable progress in ETH-based lower bounds in recent years [4, 6, 13] this problem stays unsolved (see the report from Dagstuhl Seminar 16451 in 2017 [12]).

List edge coloring is a generalization of edge coloring. An *edge list assignment* $L : E(G) \rightarrow 2^{\mathbb{N}}$ is a function that assigns to each edge e of G a set (often called a *list*) $L(e)$ of allowed colors. A function $c : E(G) \rightarrow \mathbb{N}$ is a *list edge coloring* of (G, L) if $c(e) \in L(e)$ for every $e \in E(G)$, and $c(e) \neq c(f)$ for every pair of incident edges $e, f \in E(G)$. The notion of list edge coloring is also a frequent topic of research. For example, it is conjectured that if G can be edge colored in k colors for some k , then it can be list edge colored for any edge list assignment with all lists of size at least k . This conjecture has been proved in some classes of graphs like bipartite graphs [8] or planar graphs of maximum degree at least 12 [3].

In this work, we study the computational complexity of list edge coloring. The basic decision problem, LIST EDGE COLORING IN SIMPLE GRAPHS, asks if for a given simple graph G with edge list assignment L there is a list edge coloring of (G, L) . Its more general variant, called LIST EDGE COLORING IN MULTIGRAPHS asks the same question but the input graph does not need to be simple, i.e., it can contain parallel edges. Although the problem seems much more general than EDGE COLORING, the two best known algorithms [2, 1] that decide if a given graph admits an edge coloring in k colors solve LIST EDGE COLORING IN MULTIGRAPHS (and hence also LIST EDGE COLORING IN SIMPLE GRAPHS) within the same time bound, i.e., $2^m m^{O(1)} + O(L)$, where L is the total length of all lists, after only minor modifications (see Proposition 3 in [2]). Multigraphs do not admit any upper bound on the number of edges, hence this time complexity does not translate to a function on n . We show that this is not an accident, because satisfiability of any sufficiently sparse 3-CNF-SAT formula can be efficiently encoded as a list edge coloring instance with a bounded number of vertices. This gives the following result.

► **Theorem 1.** *If there is a function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that LIST EDGE COLORING IN MULTIGRAPHS can be solved in time $f(n) \cdot m^{O(1)}$ for any input graph on n vertices and m edges, then $P = NP$.*

For simple graphs $m = O(n^2)$ and hence LIST EDGE COLORING IN SIMPLE GRAPHS admits an algorithm running in time $2^{O(n^2)}$. Our main result states that this bound is essentially optimal, assuming ETH.

► **Theorem 2.** *If there is an algorithm for LIST EDGE COLORING IN SIMPLE GRAPHS that runs in time $2^{o(n^2)}$, then Exponential Time Hypothesis fails.*

Our results have twofold consequences for the EDGE COLORING problem. First, one may hope that our reductions can inspire a reduction for EDGE COLORING. However, it is possible that such a reduction does not exist and researchers may still try to get an algorithm for EDGE COLORING running in time $2^{o(n^2)}$. Then we offer a simple way of verifying if a new idea works: if it applies to the list version as well, there is no hope for it.

2 Preliminaries

For an integer k , we denote $[k] = \{0, \dots, k-1\}$. If I and J are instances of decision problems P and R , respectively, then we say that I and J are *equivalent* if either both I and J are YES-instances of the respective problems, or both are NO-instances. A clause in a CNF-formula is represented by the set of its literals. For two subsets of vertices A, B of a graph $G = (V, E)$ by $E(A, B)$ we denote the set of edges with one endpoint in A and the other in B .

Exponential-Time Hypothesis.

The Exponential Time Hypothesis (ETH) of Impagliazzo et al. [10] states that there exists a constant $c > 0$, such that there is no algorithm solving 3-SAT in time $O(2^{cn})$. During the recent years, ETH became the central conjecture used for proving tight bounds on the complexity of various problems. One of the most important results connected to ETH is the *Sparsification Lemma* [11], which essentially gives a (many-one) reduction from an arbitrary instance of k -SAT to an instance where the number of clauses is linear in the number of variables. The following well-known corollary can be derived by combining ETH with the Sparsification Lemma.

► **Theorem 3** (see e.g. Theorem 14.4 in [5]). *Unless ETH fails, there is no algorithm for 3-SAT that runs in time $2^{o(n+m)}$, where n, m denote the numbers of variables and clauses, respectively.*

We need the following regularization result of Tovey [14]. Following Tovey, by (3,4)-SAT we call the variant of 3-SAT where each clause of the input formula contains exactly 3 different variables, and each variable occurs in at most 4 clauses.

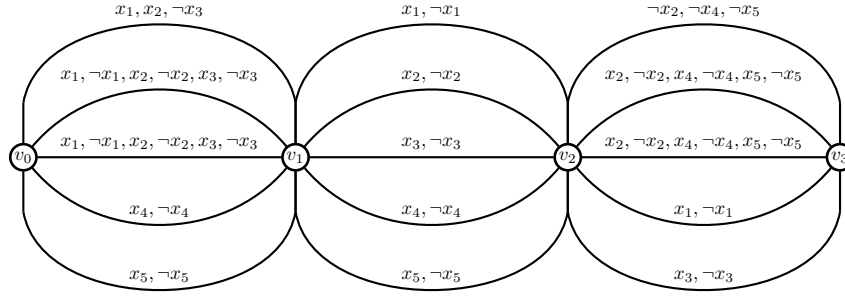
► **Lemma 4** ([14]). *Given a 3-SAT formula φ with n variables and m clauses one can transform it in polynomial time into an equivalent (3,4)-SAT instance φ' with $O(n+m)$ variables and clauses.*

Theorem 3 and Lemma 4 give the following corollary.

► **Corollary 5.** *Unless ETH fails, there is no algorithm for (3,4)-SAT that runs in time $2^{o(n)}$, where n denotes the number of variables of the input formula.*

3 Hardness of List Edge Coloring in Multigraphs

In order to prove Theorems 1 and 2 we show reductions from (3,4)-SAT to LIST EDGE COLORING with strong bounds on the number of vertices in the output instance. The basic idea of both our reductions is to use two colors, denoted by x_i and $\neg x_i$ for every variable x_i so that in every coloring of the output graph the edges colored in x_i or $\neg x_i$ form a single path with alternating colors. Then colors at the edges of this path of fixed parity can encode



■ **Figure 1** Edges related to clauses $(x_1 \vee x_2 \vee \neg x_3)$ and $(\neg x_2 \vee \neg x_4 \vee \neg x_5)$ assuming that the first of these clauses has color 0 and the second has color 1.

the value of x_i in a satisfying boolean assignment. Moreover, testing a clause $C = \ell_1 \vee \ell_2 \vee \ell_3$ can be done very easily: it suffices to add an edge with the list $\{\ell_1, \ell_2, \ell_3\}$. However this edge can belong to the alternating path of at most one of the three variables in C , and we add two more parallel edges which become elements of the two other alternating paths. Unfortunately, in order to get similar phenomenon in simple graphs, we need to introduce a complicated gadget.

► **Lemma 6.** *For any instance φ of (3,4)-SAT with n variables there is an equivalent instance (G, L) of LIST EDGE COLORING IN MULTIGRAPHS with 21 vertices and $O(n)$ edges. Moreover, the instance (G, L) can be constructed in polynomial time.*

Proof. Let $\text{vrb}(\varphi)$ and $\text{cls}(\varphi)$ be the sets of variables and clauses of φ , respectively. W.l.o.g. assume $\text{vrb}(\varphi) = \{x_0, \dots, x_{n-1}\}$.

We construct an auxiliary graph G_φ with $V(G_\varphi) = \text{cls}(\varphi)$ and such that two clauses $C_1, C_2 \in \text{cls}(\varphi)$ are adjacent in G_φ iff $C_1 \cap C_2 \neq \emptyset$. Since every clause has three variables and each variable can belong to at most three other clauses, it follows that the maximum degree of G_φ is at most 9. Let $g : \text{cls}(\varphi) \rightarrow [10]$ be the greedy vertex coloring of G_φ in 10 colors, which can be found in linear time in a standard way. For $i \in [10]$, let $\mathcal{C}_i = g^{-1}(i)$.

Let us describe the output instance (G, L) . We put $V(G) = \{v_0, \dots, v_{20}\}$. The edges of G join only vertices of consecutive indices. For every $r \in [10]$, for every clause $C \in \mathcal{C}_r$ we add three new edges with endpoints v_{2r} and v_{2r+1} . The first of these edges, denoted by e_C^1 , gets list C , i.e., the three literals of clause C . Let x_i, x_j and x_k be the three variables that appear in C . Then, the two remaining edges, e_C^2 and e_C^3 , get identical lists of $\{x_i, \neg x_i, x_j, \neg x_j, x_k, \neg x_k\}$. Moreover, for every $r \in [10]$ and for every variable x_i that does not appear in any of the clauses of \mathcal{C}_r , we add a new edge $v_{2r}v_{2r+1}$ with list $\{x_i, \neg x_i\}$. Finally, for every $r \in [10]$ and for every variable $x_i \in \text{vrb}(\varphi)$ we add a single new edge $v_{2r+1}v_{2r+2}$ with list $\{x_i, \neg x_i\}$. This finishes the description of the output instance. See Fig. 1 for an example.

In what follows, edges of the form $v_{2r}v_{2r+1}$ are called *positive* and edges of the form $v_{2r+1}v_{2r+2}$ are called *negative*.

► **Claim 1.** *For every list edge coloring c of (G, L) , for every $i \in [n]$, the edges in $c^{-1}(\{x_i, \neg x_i\})$ form a path v_0, v_1, \dots, v_{20} .*

Proof. For every $r \in [10]$, there is exactly one edge $v_{2r+1}v_{2r+2}$ with list containing x_i or $\neg x_i$, namely with list $\{x_i, \neg x_i\}$. It follows that these 10 edges belong to $c^{-1}(\{x_i, \neg x_i\})$. It suffices to prove that for every $r \in [10]$ there is also exactly one edge $v_{2r}v_{2r+1}$ in $c^{-1}(\{x_i, \neg x_i\})$. This

is clear when x_i does not appear in any of the clauses of \mathcal{C}_r , because then there is exactly one edge $v_{2r+1}v_{2r+2}$ with list containing x_i or $\neg x_i$, namely with list $\{x_i, \neg x_i\}$. Otherwise, let $C = \{\ell_i, \ell_j, \ell_k\}$ be the clause of \mathcal{C}_r where $\ell_i \in \{x_i, \neg x_i\}$. Let $\ell_j \in \{x_j, \neg x_j\}$, $\ell_k \in \{x_k, \neg x_k\}$. Then there are exactly three edges e_C^1, e_C^2, e_C^3 incident to v_{2r} and v_{2r+1} and with list containing one of literals in the set $\{x_i, \neg x_i, x_j, \neg x_j, x_k, \neg x_k\}$. Indeed, $L(e_C^1) = \{\ell_i, \ell_j, \ell_k\}$, and $L(e_C^2) = L(e_C^3) = \{x_i, \neg x_i, x_j, \neg x_j, x_k, \neg x_k\}$. However, we have already proved that for every $q \in \{i, j, k\}$, one of the edges with endpoints v_{2r+1} and v_{2r+2} is colored with x_q or $\neg x_q$. Hence, since every color class is a matching, for every $q \in \{i, j, k\}$, at most one of the edges in $\{e_C^1, e_C^2, e_C^3\}$ is colored with x_q or $\neg x_q$. However, the lists of e_C^1, e_C^2, e_C^3 contain only colors of the form x_q or $\neg x_q$ for $q \in \{i, j, k\}$. It follows that for every $q \in \{i, j, k\}$ exactly one of the edges in $\{e_C^1, e_C^2, e_C^3\}$ is colored with x_q or $\neg x_q$. In particular there is exactly one edge $v_{2r}v_{2r+1}$ in $c^{-1}(\{x_i, \neg x_i\})$. ◀

Since c is an edge coloring, the path from the claim above is colored in one of two ways, either by $x_i, \neg x_i, x_i, \neg x_i, \dots$, or by $\neg x_i, x_i, \neg x_i, x_i, \dots$. This implies the following claim.

► **Claim 2.** *For every list edge coloring c of (G, L) , for every $i \in [n]$, we have $|c^{-1}(x_i)| = |c^{-1}(\neg x_i)| = 10$ and either all edges in $c^{-1}(x_i)$ are positive and all edges in $c^{-1}(\neg x_i)$ are negative or all edges in $c^{-1}(x_i)$ are negative and all edges in $c^{-1}(\neg x_i)$ are positive.*

Now we are ready to prove that φ and (G, L) are equivalent.

Assume c is a list edge coloring of (G, L) . Define a boolean assignment $f : \text{vrb}(\varphi) \rightarrow \{T, F\}$ by setting x_i to T iff all edges in $c^{-1}(x_i)$ are positive. Now consider an arbitrary clause C . By construction, there is a positive edge e with $L(e) = C$. If $c(e) = x_q$ for some variable x_q then by Claim 2 all edges in $c^{-1}(x_q)$ are positive, and hence $f(x_q) = T$. Since $c(e) \in L(e)$ we have $x_q \in C$, so C is satisfied. If $c(e) = \neg x_q$ for some variable x_q then by Claim 2 all edges in $c^{-1}(x_q)$ are negative and hence $f(x_q) = F$. Again, since $c(e) \in L(e)$ we have $\neg x_q \in C$, so C is satisfied.

Assume φ is satisfiable and let $f : \text{vrb}(\varphi) \rightarrow \{T, F\}$ be a satisfying assignment. We define a list edge coloring c of (G, L) as follows. Recall that for every $r \in [10]$, and for every clause $C \in \mathcal{C}_r$ there is an edge e_C^1 with $L(e_C^1) = C$ and edges e_C^2, e_C^3 with $L(e_C^2) = L(e_C^3) = \{x_i, \neg x_i, x_j, \neg x_j, x_k, \neg x_k\}$, where x_i, x_j and x_k are the three variables that appear in C . We color e_C^1 with any of the satisfied literals of C . Without loss of generality, assume $c(e_C^1) \in \{x_i, \neg x_i\}$. Then we color e_C^2 with x_j if $f(x_j) = T$ and with $\neg x_j$ otherwise. Similarly, we color e_C^3 with x_k if $f(x_k) = T$ and with $\neg x_k$ otherwise. Each of the remaining positive edges e of G has its list equal $\{x_i, \neg x_i\}$ for some $x_i \in \text{vrb}(\varphi)$. We color e with x_i if $f(x_i) = T$ and with $\neg x_i$ otherwise. It follows that every positive edge is colored with a satisfied literal. Every negative edge \tilde{e} has its list equal to $\{x_i, \neg x_i\}$ for some $x_i \in \text{vrb}(\varphi)$. We color \tilde{e} with x_i when $f(x_i) = F$ and with $\neg x_i$ when $f(x_i) = T$. It follows that every negative edge is colored with an unsatisfied literal. Let us show that c does not color incident edges with the same color. Since the lists of parallel negative edges are disjoint, in our coloring there are no parallel negative edges of the same color. Assume there are two parallel positive edges of the form $v_{2r}v_{2r+1}$ of the same color ℓ , for some $r \in [10]$. Then the variable of ℓ belongs to a clause in \mathcal{C}_r , for otherwise there is exactly one edge with endpoints $v_{2r}v_{2r+1}$ and with list containing ℓ . However, since \mathcal{C}_r is independent in G_φ , there is exactly one such clause C in \mathcal{C}_r . It follows that the two parallel edges are among the three edges e_C^1, e_C^2, e_C^3 . However, these three edges got different colors, a contradiction. If two edges are incident but not parallel, one of them is positive and the other negative. The former is colored with a satisfied literal and the latter with an unsatisfied literal, so they are colored differently. Hence c is a proper list edge coloring, as required. This ends the proof of Lemma 6. ◀

Theorem 1 follows immediately from Lemmas 4 and 6 and the NP-hardness of 3-SAT.

4 Hardness of List Edge Coloring in Simple Graphs

This section is devoted to the proof of the following lemma.

► **Lemma 7.** *For any instance φ of (3,4)-SAT with n variables there is an equivalent instance (G, L) of LIST EDGE COLORING IN SIMPLE GRAPHS with $O(\sqrt{n})$ vertices. Moreover, the instance (G, L) can be constructed in polynomial time.*

4.1 Intuition

The general idea is to follow the approach of Lemma 6 and replace the edges with multiplicity $O(n)$ with bipartite graphs with $O(\sqrt{n})$ vertices and $O(n)$ edges. In our construction, for every $r \in [10]$, we replace every two consecutive bundles of parallel edges between v_{2r} , v_{2r+1} , and v_{2r+2} from the construction in Lemma 6 by seven layers L_i , $i = 6r + 1, \dots, 6r + 7$, each of $O(\sqrt{n})$ vertices, with edges joining both consecutive and non-consecutive layers. The subgraph induced by $\bigcup_{i=6r+1}^{6r+7} L_i$ is called the r -th *clause verifying gadget* G_r . (Note that the layers L_i for $i \equiv 1 \pmod{6}$ are shared between consecutive gadgets.) Analogously as in Lemma 6, the role of G_r is to check whether all clauses in \mathcal{C}_r are satisfied. We add also two additional layers L_0 and L_{62} which make some of our arguments simpler.

4.2 Construction

It will be convenient to assume that $\sqrt{n} \in \mathbb{N}$. We do not lose on generality because otherwise we just add $n^+ = (\lceil \sqrt{n} \rceil + 1)^2 - n$ variables y_1, y_2, \dots, y_{n^+} and clauses

$$\{y_1, y_2, y_3\}, \{y_2, y_3, y_4\}, \dots, \{y_{n^+-2}, y_{n^+-1}, y_{n^+}\}.$$

Note that $n^+ \geq 3$, $n^+ \leq (\sqrt{n} + 2)^2 - n = 4\sqrt{n} + 4$ and $\sqrt{n + n^+} = \lceil \sqrt{n} \rceil + 1 \in \mathbb{N}$. Hence we added only $O(\sqrt{n})$ variables and clauses, and the resulting formula is still a (3,4)-SAT instance.

We begin as in Lemma 6, by building the graph G_φ , and finding its greedy coloring g which partitions the clause set into 10 color classes \mathcal{C}_r , $r \in [10]$. Let us build the instance (G, L) step by step.

Add two sets of vertices (called *layers*) $L_i = \{v_j^i \mid j \in [\sqrt{n}]\}$, $i = 0, 1$. Then add all possible n edges between L_0 and L_1 forming a complete bipartite graph. Map the n variables to the n edges in a 1 – 1 way. For every $i \in [n]$, set the list of the edge assigned to x_i to $\{x_i, \neg x_i\}$.

The vertex set $V(G)$ contains 60 more layers of vertices L_i , $i = \{2, \dots, 61\}$, where $L_i = \{v_j^i \mid j \in [6\sqrt{n} + 3]\}$. Finally, $L_{62} = \{v_j^{62} \mid j \in [\sqrt{n} + 1]\}$. Also denote $L_{-1} = L_{63} = \emptyset$. In what follows we add the remaining edges of G . Whenever we add edges between L_i and L_{i-1} , for every $j < i$ all the edges of the output graph between L_j and L_{j-1} are already added. We will make sure to keep the following invariants satisfied during the process of construction (note that they hold for the part constructed so far).

► **Invariant 1 (Uniqueness).** *For every $i \in [62]$, for every variable $x_j \in \text{vrb}(\varphi)$ there is at most one edge $uv \in E(L_i, L_{i+1})$ such that $\{x_j, \neg x_j\} \cap L(uv) \neq \emptyset$. Moreover, after finishing adding edges between L_i and L_{i+1} , there is exactly one such edge.*

Using the notation from Invariant 1, if the edge uv exists, we can denote $v_{i,j}^+ = u$ and $v_{i+1,j}^- = v$.

► **Invariant 2 (Flow).** For every $i \in \{1, \dots, 62\}$, for every variable $x_j \in \text{vrb}(\varphi)$ we have that $v_{i,j}^- = v_{i,j}^+$, unless $v_{i,j}^-$ or $v_{i,j}^+$ is undefined. Moreover, the equality holds after finishing adding edges between L_i and L_{i+1} .

Thanks to Invariant 2, after finishing adding edges between L_i and L_{i+1} , we can just define $v_{i,j} := v_{i,j}^- = v_{i,j}^+$ for $i \in \{1, \dots, 61\}$. We also put $v_{0,j} = v_{0,j}^+$ and $v_{62,j} = v_{62,j}^-$. In our construction we will use some additional colors apart from the literals. The following invariant holds.

► **Invariant 3 (Lists).** For every edge e of G , the list $L(e)$ contains at least one literal.

For every $i \in [62]$, for every vertex $v \in L_i$, let $\deg^-(v) = |E(L_{i-1}, \{v\})|$ and $\deg^+(v) = |E(L_{i+1}, \{v\})|$.

► **Invariant 4 (Indegrees).** For every $i \in [62]$ for vertex $v \in L_i$ we have $\deg^-(v) \leq \sqrt{n}$.

► **Invariant 5 (Jumping edges).** For every $i \in [62]$, for vertex $v \in L_i$ there are at most \sqrt{n} edges from v to layers L_j for $j > i + 1$.

By Invariant 2 and Invariant 3, for every vertex $v \in V(G)$ it holds that $\deg^+(v) \leq \deg^-(v)$. Hence Invariant 4 gives the claim below.

► **Claim 3 (Outdegrees).** For every $i \in [62]$, for every vertex $v \in L_i$, we have $\deg^+(v) \leq \sqrt{n}$.

Invariants 1 and 3 immediately imply the following.

► **Claim 4.** For every $i \in [62]$, we have $|E(L_i, L_{i+1})| \leq n$.

Let us fix $r \in [10]$. We add the edges of the r -th clause verifying gadget G_r . Although G is undirected, we will say that an edge uv between L_i and L_j for $i < j$ is from u to v and from L_i to L_j . Below we describe the edges in G_r in the order which is convenient for the exposition. However, the algorithm adds the edges between layers in the left-to-right order, i.e., for $i < j$, edges to L_i are added before edges to L_j .

1. Edges to L_ℓ for $\ell = 6r + 2, 6r + 4, 6r + 6$.

For every clause $C \in \mathcal{C}_r$ we do the following. Let $x_{i_1}, x_{i_2}, x_{i_3}$ be the three different variables that appear in the literals of C . Let $v_j = v_{\ell-1, i_j}^-$ for $j = 1, 2, 3$. Note that vertices v_1, v_2, v_3 need not be distinct. By Claim 3, $|N(v_j) \cap L_\ell| \leq \sqrt{n}$ for $j = 1, 2, 3$. Let $S = \{v \in L_\ell \mid \deg^-(v) = \sqrt{n}\}$. By Claim 4, $|S| \leq \sqrt{n}$. Hence, for $j = 1, 2, 3$ we have $|L_\ell \setminus (N(\{v_j\}) \cup S)| \geq 4\sqrt{n} + 3$ and we can pick a vertex $w_j \in L_\ell$ that has at most $\sqrt{n} - 1$ edges from $L_{\ell-1}$, is not adjacent to v_j , and is different than $w_{j'}$ for each $j' < j$. If $\ell = 6k + 6$ we additionally require that for every $j = 1, 2, 3$, the vertex w_j is not adjacent to v_{6r+2, i_j}^- or v_{6r+4, i_j}^- . By Invariant 5 this eliminates at most $2\sqrt{n}$ more candidates, so it is still possible to choose all the w_j 's. For each $j = 1, 2, 3$, we add an edge $v_j w_j$ with $L(v_j w_j) = \{x_{i_j}, \neg x_{i_j}\}$. Moreover, if $\ell = 6k + 6$, for every $j = 1, 2, 3$ we add an edge $v_{6r+2, i_j}^- w_j$ with list $\{x_{i_j}, \neg x_{i_j}, a_{i_j}\}$ and an edge $v_{6r+4, i_j}^- w_j$ with list $\{x_{i_j}, \neg x_{i_j}, b_{i_j}\}$. The conditions used to choose w_1, w_2 and w_3 guarantee that we do not introduce parallel edges.

For every variable x_i that is not present in any of the clauses of \mathcal{C}_r we find a vertex $w \in L_\ell$ that has at most $\sqrt{n} - 1$ edges from $L_{\ell-1}$ and is not adjacent to $v_{\ell-1, i}^-$. Again, this is possible because there are at most $2\sqrt{n}$ vertices in L_ℓ that violate any of these constraints. We add an edge $v_{\ell-1, i}^- w$ with $L(v_{\ell-1, i}^- w) = \{x_i, \neg x_i\}$.

Note that all invariants are satisfied: for Invariant 1 it follows from the fact that \mathcal{C}_r is independent in G_φ , while invariants 2, 3, 4 follow immediately from the construction.

Invariant 5 stays satisfied after adding $v_{6r+2,i_j}^- w_j$ because for every variable x_k such that $v_{6r+2,k}^- = v_{6r+2,i_j}^-$ we add at most one edge from v_{6r+2,i_j}^- to L_{6r+6} , and the number of such variables is equal to $\deg^-(v_{6r+2,i_j}^-)$, which is at most \sqrt{n} by Invariant 4 (analogous argument applies to adding the edge $v_{6r+4,i_j}^- w_j$).

2. Edges to L_ℓ for $\ell = 6r + 3, 6r + 5, 6r + 7$.

For every clause $C \in \mathcal{C}_r$ we do the following. Let $C = \{\ell_1, \ell_2, \ell_3\}$ and let x_{i_j} be the variable from the literal of ℓ_j , for $j = 1, 2, 3$. Let $w_j = v_{\ell-1,i_j}^-$ for $j = 1, 2, 3$. By Claim 3, $|N(\{w_1, w_2, w_3\} \cap L_\ell)| \leq 3\sqrt{n}$. Also, there are at most $\sqrt{n} + 2$ vertices in L_ℓ with at least $\sqrt{n} - 2$ edges from $L_{\ell-1}$. Indeed, otherwise $|E(L_{\ell-1}, L_\ell)| \geq n + \sqrt{n} - 6$ and either $n \leq 36$ (and the lemma is trivial) or there is a contradiction with Claim 4. Hence, we can find a vertex $z_{\ell,C} \in L_\ell$ that has at most $\sqrt{n} - 3$ edges to $L_{\ell-1}$ and is not adjacent to $\{w_1, w_2, w_3\}$. If $\ell = 6k + 7$ we additionally require that the vertex $z_{6k+7,C}$ is not adjacent to $z_{6k+3,C}$ or $z_{6k+5,C}$. By Invariant 5 this eliminates at most $2\sqrt{n}$ more candidates, so it is still possible to choose vertex $z_{6k+7,C}$. For each $j = 1, 2, 3$, we add an edge $w_j z_{\ell,C}$. We put $L(w_j z_{6r+3,C}) = \{x_{i_j}, \neg x_{i_j}, a_{i_j}\}$, $L(w_j z_{6r+5,C}) = \{x_{i_j}, \neg x_{i_j}, b_{i_j}\}$, and $L(w_j z_{6r+7,C}) = \{\ell_j, c_C, d_C\}$. (The colors $a_{i_j}, b_{i_j}, c_C, d_C$ are not literals — these are new auxiliary colors; each variable x_i has its own distinct auxiliary colors a_i, b_i , and each clause C has its own auxiliary colors c_C, d_C .) We add edges $z_{6r+3,C} z_{6r+7,C}$ and $z_{6r+5,C} z_{6r+7,C}$, both with lists $\{x_{i_1}, \neg x_{i_1}, x_{i_2}, \neg x_{i_2}, x_{i_3}, \neg x_{i_3}\}$.

For every variable x_i that is not present in any of the clauses of \mathcal{C}_r we proceed analogously as in Step 1.

The invariants hold for the similar reasons as before. In particular, Invariant 5 stays satisfied after adding $z_{6r+3,C} z_{6r+7,C}$ because for every clause C' such that $z_{6r+3,C'} = z_{6r+3,C}$ we add exactly one edge from $z_{6r+3,C}$ to L_{6r+7} , and the number of such clauses is bounded by $\deg^-(z_{6r+3,C})/3$, which is at most $\sqrt{n}/3$ by Invariant 4 (analogous argument applies to adding the edge $z_{6r+5,C} z_{6r+7,C}$).

Finally, we add edges between L_{61} and L_{62} . For every variable x_i we find a vertex $w \in L_{62}$ that is not adjacent to $v_{61,i}^-$, which is possible because $\deg^+(v_{61,i}^-) \leq \sqrt{n}$. We add an edge $v_{61,i}^- w$ with $L(v_{61,i}^- w) = \{x_i, \neg x_i\}$.

The following claims follow directly from the construction.

► **Claim 5.** For every $r \in [10]$, for every clause $C \in \mathcal{C}_r$ with variables $x_{i_1}, x_{i_2}, x_{i_3}$, and for each $\ell = 6r + 3, 6r + 5, 6r + 7$ we have $v_{\ell,i_1} = v_{\ell,i_2} = v_{\ell,i_3} = z_{\ell,C}$. Moreover, for each $\ell = 6r + 3, 6r + 5, 6r + 7$ and $j = 1, 2, 3$ we have $L(z_{\ell,C} v_{\ell+1,i_j}) = \{x_{i_j}, \neg x_{i_j}\}$.

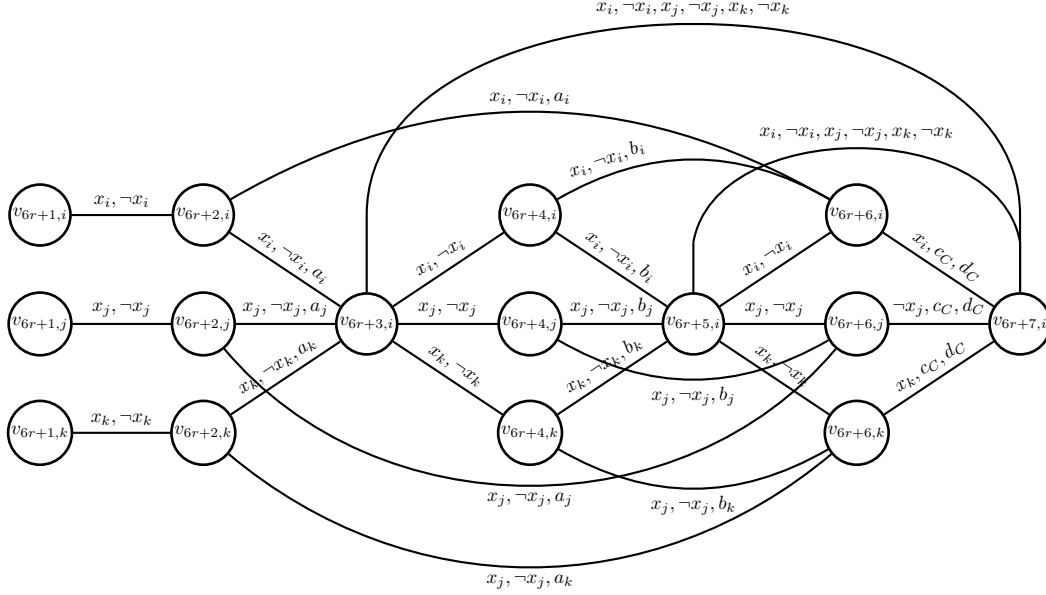
► **Claim 6.** For every edge $uv \in E(G)$, where $u \in L_j, v \in L_k$, if $\{x_i, \neg x_i\} \cap L(uv) \neq \emptyset$, then $u = v_{j,i}$ and $v = v_{k,i}$.

This finishes the description of the output instance. Since G contains $O(1)$ layers, each with $O(\sqrt{n})$ vertices, it follows that $|V(G)| = O(\sqrt{n})$, as required. See Fig 2 for an illustration of edges representing a single clause within a clause verifying gadget.

4.3 Structure of coloring

Similarly to multigraphs the crux of the equivalence between instances is the following claim.

► **Claim 7.** For every list edge coloring c of (G, L) , for every $i \in [n]$, the edges in $c^{-1}(\{x_i, \neg x_i\})$ form a path P_i from L_0 to L_{62} . Moreover, if P_i contains an edge $v_{6r+6,i} v_{6r+7,i}$ for some $r \in [10]$, then this edge is preceded by an even number of edges on P_i .



■ **Figure 2** Edges in the gadget G_r related to a clause $(x_i \vee \neg x_j \vee x_k)$ from \mathcal{C}_r .

Proof. Fix $i \in [n]$. For convenience, denote $E_i = c^{-1}(\{x_i, \neg x_i\})$. By Invariant 1 there is exactly one edge between L_0 and L_1 that has x_i or $\neg x_i$ on its list, namely $v_{0,i}v_{1,i}$. Similarly, there is exactly one edge between L_{61} and L_{62} that has x_i or $\neg x_i$ on its list, namely $v_{61,i}v_{62,i}$. Since $L(v_{0,i}v_{1,i}) = L(v_{61,i}v_{62,i}) = \{x_i, \neg x_i\}$, we know that $v_{0,i}v_{1,i}, v_{61,i}v_{62,i} \in E_i$, and these are the only edges of E_i in $E(L_0, L_1) \cup E(L_{61}, L_{62})$. Observe that edges between non-consecutive layers never leave the clause verifying gadgets. Hence, for the first part of the claim, it suffices to show that for every $r \in [10]$, the edges in $E_i \cap E(G_r)$ form a path between $v_{6r+1,i}$ and $v_{6r+7,i}$. In fact, by Claim 6 it suffices to show that $E_i \cap E(G_r)$ contains a path between $v_{6r+1,i}$ and $v_{6r+7,i}$ that visits all the vertices $\{v_{6r+j,i} \mid j = 1, \dots, 7\}$. To this end, fix $r \in [10]$.

First assume that x_i does not appear in any clause of \mathcal{C}_r . Then G_r contains the path $v_{6r+1,i}, v_{6r+2,i}, \dots, v_{6r+7,i}$, where each edge has the list $\{x_i, \neg x_i\}$. It immediately implies that all edges of this path are in $E_i \cap E(G_r)$, as required.

Now let us assume that x_i appears in a clause $C \in \mathcal{C}_r$. Let $C = \{\ell_i, \ell_j, \ell_k\}$ and assume that the literal ℓ_i contains x_i , the literal ℓ_j contains a variable x_j , and the literal ℓ_k contains a variable x_k . Observe that for $j = 1, 3, 5$ we have $v_{6r+j,i}v_{6r+j+1,i} \in E_i$ because these edges have their lists equal to $\{x_i, \neg x_i\}$. Note also that $\Delta(E_i) \leq 2$ because E_i is a union of two matchings (colors). We consider three subcases.

1. Assume $v_{6r+3,i}v_{6r+7,i} \in E_i$. Since $\Delta(E_i) \leq 2$ and $v_{6r+3,i}v_{6r+4,i} \in E_i$ we know that $v_{6r+2,i}v_{6r+3,i} \notin E_i$, and as a consequence, $c(v_{6r+2,i}v_{6r+3,i}) = a_i$. Hence $c(v_{6r+2,i}v_{6r+6,i}) \neq a_i$, which implies that $v_{6r+2,i}v_{6r+6,i} \in E_i$. Then, since $\Delta(E_i) \leq 2$ and $v_{6r+5,i}v_{6r+6,i} \in E_i$ we know that $v_{6r+4,i}v_{6r+6,i} \notin E_i$, and as a consequence, $c(v_{6r+4,i}v_{6r+6,i}) = b_i$. Hence $c(v_{6r+4,i}v_{6r+5,i}) \neq b_i$, which implies that $v_{6r+4,i}v_{6r+5,i} \in E_i$. Thus, we have shown that E_i contains the path $v_{6r+1,i}, v_{6r+2,i}, v_{6r+6,i}, v_{6r+5,i}, v_{6r+4,i}, v_{6r+3,i}, v_{6r+7,i}$, as required.
2. Assume $v_{6r+5,i}v_{6r+7,i} \in E_i$. Since $\Delta(E_i) \leq 2$ and $v_{6r+5,i}v_{6r+6,i} \in E_i$ we know that $v_{6r+4,i}v_{6r+5,i} \notin E_i$, and as a consequence, $c(v_{6r+4,i}v_{6r+5,i}) = b_i$. Hence $c(v_{6r+4,i}v_{6r+6,i}) \neq b_i$, which implies that $v_{6r+4,i}v_{6r+6,i} \in E_i$. Then, since $\Delta(E_i) \leq 2$ and $v_{6r+5,i}v_{6r+6,i} \in E_i$

we know that $v_{6r+2,i}v_{6r+6,i} \notin E_i$, and as a consequence, $c(v_{6r+2,i}v_{6r+6,i}) = a_i$. Hence $c(v_{6r+2,i}v_{6r+3,i}) \neq a_i$, which implies that $v_{6r+2,i}v_{6r+3,i} \in E_i$. Thus, we have shown that E_i contains the path $v_{6r+1,i}, v_{6r+2,i}, v_{6r+3,i}, v_{6r+4,i}, v_{6r+5,i}, v_{6r+6,i}, v_{6r+7,i}$, as required.

3. Assume $v_{6r+3,i}v_{6r+7,i}, v_{6r+5,i}v_{6r+7,i} \notin E_i$. Since $L(v_{6r+3,i}v_{6r+7,i}) = L(v_{6r+5,i}v_{6r+7,i}) = \{x_i, \neg x_i, x_j, \neg x_j, x_k, \neg x_k\}$ we infer that $v_{6r+3,i}v_{6r+7,i}, v_{6r+5,i}v_{6r+7,i} \in E_j \cup E_k$. By Claim 5 we know that $v_{6r+7,i} = v_{6r+7,j} = v_{6r+7,k}, v_{6r+7,i}v_{6r+8,j} \in E_j$ and $v_{6r+7,i}v_{6r+8,k} \in E_k$. Since $\Delta(E_j) \leq 2$ and $\Delta(E_k) \leq 2$, we get that $v_{6r+3,i}v_{6r+7,i} \in E_j$ and $v_{6r+5,i}v_{6r+7,i} \in E_k$ or vice versa. In any case, $v_{6k+6,j}, v_{6k+7,i} \notin E_j$, and $v_{6k+6,k}, v_{6k+7,i} \notin E_k$. Recall that $L(v_{6k+6,j}, v_{6k+7,i}) = \{\ell_j, c_C, d_C\}$ and $L(v_{6k+6,k}, v_{6k+7,i}) = \{\ell_k, c_C, d_C\}$. It follows that $c(\{v_{6k+6,j}v_{6k+7,i}, v_{6k+6,k}v_{6k+7,i}\}) = \{c_C, d_C\}$. Then $c(v_{6k+6,i}, v_{6k+7,i}) \notin \{c_C, d_C\}$. Since $L(v_{6k+6,i}, v_{6k+7,i}) = \{\ell_i, c_C, d_C\}$, we get that $v_{6k+6,i}, v_{6k+7,i} \in E_i$. Then, since $\Delta(E_i) \leq 2$ and $v_{6r+5,i}v_{6r+6,i} \in E_i$ we know that $v_{6r+2,i}v_{6r+6,i}, v_{6r+4,i}v_{6r+6,i} \notin E_i$, and as a consequence, $c(v_{6r+2,i}v_{6r+6,i}) = a_i$ and $c(v_{6r+4,i}v_{6r+6,i}) = b_i$. Hence $c(v_{6r+2,i}v_{6r+3,i}) \neq a_i$, and $c(v_{6r+4,i}v_{6r+5,i}) \neq b_i$ which implies that $v_{6r+2,i}v_{6r+3,i}, v_{6r+4,i}v_{6r+5,i} \in E_i$. Thus, E_i contains the path $v_{6r+1,i}, v_{6r+2,i}, v_{6r+3,i}, v_{6r+4,i}, v_{6r+5,i}, v_{6r+6,i}, v_{6r+7,i}$, as required.

For the second part of the claim recall that P_i decomposes into an edge from L_0 to L_1 , 10 paths of length 6 inside the gadgets and an edge from L_{61} to L_{62} . Moreover, if P_i contains an edge $v_{6r+6,i}v_{6r+7,i}$ for some $r \in [10]$, then this edge is the last edge of one of the 10 paths of length 6. It follows that it is preceded by $1 + 6r + 5$ edges, which is an even number. \blacktriangleleft

4.4 Equivalence

Assume c is a list edge coloring of (G, L) . Define a boolean assignment $f : \text{vrb}(\varphi) \rightarrow \{T, F\}$ by setting x_i to T iff the first edge of the path P_i from Claim 7 is colored by x_i . Note that P_i is colored alternately with x_i and $\neg x_i$ and every odd edge on P_i (i.e., preceded by an even number of edges) is colored with a satisfied literal. Now consider an arbitrary clause C . Let $r = g(C)$. Let $C = \{\ell_1, \ell_2, \ell_3\}$ and let x_{i_j} be the variable from the literal of ℓ_j , for $j = 1, 2, 3$. By construction, there are three edges $v_{6r+6,i_j}z_{6r+7,C}$, for $j = 1, 2, 3$ with $L(v_{6r+6,i_j}z_{6r+7,C}) = \{\ell_j, c_C, d_C\}$. At most two of these edges are colored with c_C or d_C , so there is $j = 1, 2, 3$ such that $c(v_{6r+6,i_j}z_{6r+7,C}) = \ell_j$. In particular, $v_{6r+6,i_j}z_{6r+7,C} \in c^{-1}(\{x_{i_j}, \neg x_{i_j}\})$ and hence, by Claim 7 we know that $v_{6r+6,i_j}z_{6r+7,C} \in P_{i_j}$. However, by the second part of Claim 7 this edge is preceded by an even number of edges on P_{i_j} . It follows that ℓ_j is satisfied.

Assume φ is satisfiable and let $f : \text{vrb}(\varphi) \rightarrow \{T, F\}$ be a satisfying assignment. We define a list edge coloring c of (G, L) as follows. Consider any edge $e \in E(L_0, L_1)$. Then $L(e) = \{x_i, \neg x_i\}$. We color e with x_i when $f(x_i) = T$ and with $\neg x_i$ otherwise. Now consider any edge $e \in E(L_{61}, L_{62})$. Again $L(e) = \{x_i, \neg x_i\}$. We color e with x_i when $f(x_i) = F$ and with $\neg x_i$ otherwise. By Invariant 1 incident edges get different colors in the partial coloring described so far. In what follows we describe $c|_{E(G_r)}$ for every $r \in [10]$ separately. Fix $r \in [10]$.

Consider an arbitrary clause $C \in \mathcal{C}_r$. Let $C = \{\ell_1, \ell_2, \ell_3\}$ and let x_{i_j} be the variable from the literal of ℓ_j , for $j = 1, 2, 3$. Since φ is satisfied by f , at least one literal of C is satisfied by f , by symmetry we can assume it is ℓ_1 . Consider the three edge disjoint paths

$$R_1 = v_{6r+1,i_1}, v_{6r+2,i_1}, v_{6r+3,i_1}, v_{6r+4,i_1}, v_{6r+5,i_1}, v_{6r+6,i_1}, v_{6r+7,i_1},$$

$$R_2 = v_{6r+1,i_2}, v_{6r+2,i_2}, v_{6r+6,i_2}, v_{6r+5,i_2}, v_{6r+4,i_2}, v_{6r+3,i_2}, v_{6r+7,i_2},$$

$$R_3 = v_{6r+1,i_3}, v_{6r+2,i_3}, v_{6r+3,i_3}, v_{6r+4,i_3}, v_{6r+6,i_3}, v_{6r+5,i_3}, v_{6r+7,i_3}.$$

For each $j = 1, 2, 3$ the path R_j is colored by x_{i_j} and $\neg x_{i_j}$ alternately, beginning with $\neg x_{i_j}$ if $f(x_{i_j}) = T$ and with x_{i_j} if $f(x_{i_j}) = F$. Note that edges of R_1, R_2 and R_3 are colored by

colors from their lists. Indeed, this is obvious for every edge apart from $v_{6r+6,i_1}, v_{6r+7,i_1}$, because their lists contain $\{x_{i_j}, \neg x_{i_j}\}$. Edge $v_{6r+6,i_1}, v_{6r+7,i_1}$ is colored with x_{i_j} if $f(x_{i_j}) = T$ and with $\neg x_{i_j}$ if $f(x_{i_j}) = F$. It follows that $v_{6r+6,i_1}, v_{6r+7,i_1}$ is colored with the literal from $\{x_{i_1}, \neg x_{i_1}\}$ which is satisfied by f , hence it is colored by ℓ_1 , and $\ell_1 \in L(v_{6r+6,i_1}, v_{6r+7,i_1})$, as required. Finally, we put

$$\begin{aligned} c(v_{6r+2,i_1} v_{6r+6,i_1}) &= a_{i_1}, \\ c(v_{6r+4,i_1} v_{6r+6,i_1}) &= b_{i_1}, \\ c(v_{6r+2,i_2} v_{6r+3,i_2}) &= a_{i_2}, \\ c(v_{6r+4,i_2} v_{6r+6,i_2}) &= b_{i_2}, \\ c(v_{6r+2,i_3} v_{6r+6,i_3}) &= a_{i_3}, \\ c(v_{6r+4,i_3} v_{6r+5,i_3}) &= b_{i_3}, \\ c(v_{6r+6,i_2} v_{6r+7,i_2}) &= c_C, \\ c(v_{6r+6,i_3} v_{6r+7,i_3}) &= d_C. \end{aligned}$$

Thus we have colored all edges of G_r which have lists containing a variable from C .

Now consider any variable x_i that does not appear in any clause of \mathcal{C}_r . Consider the path $v_{6r+1,i}, v_{6r+2,i}, \dots, v_{6r+7,i}$. If $f(x_i) = T$, color the path with the sequence of colors $\neg x_i, x_i, \neg x_i, \dots, x_i$, and otherwise with the sequence of colors $x_i, \neg x_i, x_i, \dots, \neg x_i$.

Thus we have colored all the edges of G_r . It is straightforward to check that for every $r \in [10]$ the subgraph G_r is colored properly. It remains to show that vertices in the layers L_i for $i \equiv 1 \pmod{6}$ are not incident to two edges of the same color. Clearly, this cannot happen for colors a_j or b_j for any $j \in [n]$, because they are not present on lists of edges incident to L_i for $i \equiv 1 \pmod{6}$. Also, it cannot happen for colors c_C or d_C for any clause C , because edges with these colors on their list only join L_{i-1} with L_i for $i \equiv 1 \pmod{6}$, so two incident edges colored with c_C or d_C cannot belong to different gadgets. Finally, consider colors $\{x_i, \neg x_i\}$ for a fixed $i \in [n]$. The edges with these colors form a path of length 62, starting with $v_{0,i}v_{1,i}$, and continued as follows. The edge $v_{0,i}v_{1,i}$ is followed by 10 paths of length 6. For every $r \in [10]$, the r -th path of length 10 begins in $v_{6r+1,i}$ and ends in $v_{6r+7,i} = v_{6(r+1)+1,i}$. Finally, the 62-path ends with edge $v_{61,i}v_{62,i}$. Note that $v_{0,i}v_{1,i}$ is colored with the satisfied literal. Next, for every $r \in [10]$, the first edge of the r -th 10-path is colored with the non-satisfied literal and its last edge is colored by the satisfied literal. Finally, $v_{61,i}v_{62,i}$ is colored with the non-satisfied literal. It follows that the 62-path of all edges with colors from $\{x_i, \neg x_i\}$ is colored alternately in x_i and $\neg x_i$, as required. This finishes the proof that c is a list edge coloring of (G, L) , and the proof of Lemma 7.

4.5 Proof of Theorem 2

Theorem 2 follows immediately from Lemma 7 and Corollary 5. Indeed, if there is an algorithm A which solves LIST EDGE COLORING IN SIMPLE GRAPHS in time $2^{o(|V(G)|^2)}$, then by Lemma 7 an n -variable instance of (3,4)-SAT can be transformed to a $O(\sqrt{n})$ -vertex instance of LIST EDGE COLORING IN SIMPLE GRAPHS in polynomial time and next solved in time $2^{o(n)}$ using A , which contradicts ETH by Corollary 5.

5 Conclusions and further research

In this work we have shown that LIST EDGE COLORING IN SIMPLE GRAPHS does not admit an algorithm that runs in time $2^{o(n^2)}$, unless ETH fails. This has consequences for designing algorithms for EDGE COLORING: in order to break the barrier $2^{O(n^2)}$ one has to use methods

that exploit symmetries between colors, and in particular do not apply to the list version. On the other hand, one may hope that our reductions can inspire a reduction to EDGE COLORING which would exclude at least a $2^{O(n)}$ -time algorithm. However it seems that EDGE COLORING requires a significantly different approach. In our reductions we were able to encode information (namely, the boolean value of a variable in a satisfying assignment) in a *color* of an edge. In the case of EDGE COLORING this is not possible, because one can recolor any edge e by choosing an arbitrary different color c' and swapping c' and the color c of e on the maximal path/cycle that contains e and has edges colored with c and c' only.

References

- 1 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.
- 2 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009.
- 3 O. V. Borodin, A. V. Kostochka, and D. R. Woodall. List edge and list total colourings of multigraphs. *J. of Comb. Theory, Ser. B*, 71:184–204, 1997.
- 4 Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub Pachocki, and Arkadiusz Socala. Tight lower bounds on graph embedding problems. *J. ACM*, 64(3):18:1–18:22, 2017. doi:10.1145/3051094.
- 5 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshantov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
- 6 Marek Cygan, Marcin Pilipczuk, and Michał Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM J. Comput.*, 45(1):67–83, 2016. doi:10.1137/130947076.
- 7 Fedor V. Fomin, Kazuo Iwama, Dieter Kratsch, Petteri Kaski, Mikko Koivisto, Łukasz Kowalik, Yoshio Okamoto, Johan van Rooij, and Ryan Williams. 08431 open problems – moderately exponential time algorithms. In Fedor V. Fomin, Kazuo Iwama, and Dieter Kratsch, editors, *Moderately Exponential Time Algorithms*, number 08431 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2008. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany. URL: <http://drops.dagstuhl.de/opus/volltexte/2008/1798>.
- 8 Fred Galvin. The list chromatic index of a bipartite multigraph. *J. Comb. Theory, Ser. B*, 63(1):153–158, 1995. doi:10.1006/jctb.1995.1011.
- 9 Ian Holyer. The np-completeness of some edge-partition problems. *SIAM J. Comput.*, 10(4):713–717, 1981. doi:10.1137/0210054.
- 10 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
- 11 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.
- 12 Moshe Lewenstein, Seth Pettie, and Virginia Vassilevska Williams. Structure and hardness in P (dagstuhl seminar 16451). *Dagstuhl Reports*, 6(11):1–34, 2016. doi:10.4230/DagRep.6.11.1.
- 13 Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010. doi:10.4086/toc.2010.v006a005.
- 14 Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Appl. Math.*, 8(1):85–89, 1984.