Tight Lower Bounds for the Complexity of Multicoloring*†

Marthe Bonamy[1], Łukasz Kowalik[2], Michał Pilipczuk[3], Arkadiusz Socała[4], and Marcin Wrochna[5]

1 **CNRS, LaBRI, Bordeaux, France**
marthe.bonamy@labri.fr
2 **University of Warsaw, Warsaw, Poland**
kowalik@mimuw.edu.pl
3 **University of Warsaw, Warsaw, Poland**
michal.pilipczuk@mimuw.edu.pl
4 **University of Warsaw, Warsaw, Poland**
arkadiusz.socala@mimuw.edu.pl
5 **University of Warsaw, Warsaw, Poland**
m.wrochna@mimuw.edu.pl

Abstract

In ¢the *multicoloring* problem, also known as *(a:b)-coloring* or *b-fold coloring*, we are given a graph G and a set of a colors, and the task is to assign a subset of b colors to each vertex of G so that adjacent vertices receive disjoint color subsets. This natural generalization of the classic coloring problem (the $b = 1$ case) is equivalent to finding a homomorphism to the Kneser graph $KG_{a,b}$, and gives relaxations approaching the fractional chromatic number.

We study the complexity of determining whether a graph has an $(a{:}b)$-coloring. Our main result is that this problem does not admit an algorithm with running time $f(b) \cdot 2^{o(\log b) \cdot n}$, for any computable $f(b)$, unless the Exponential Time Hypothesis (ETH) fails. A $(b+1)^n \cdot \mathrm{poly}(n)$-time algorithm due to Nederlof [2008] shows that this is tight. A direct corollary of our result is that the graph homomorphism problem does not admit a $2^{\mathcal{O}(n+h)}$ algorithm unless ETH fails, even if the target graph is required to be a Kneser graph. This refines the understanding given by the recent lower bound of Cygan et al. [SODA 2016].

The crucial ingredient in our hardness reduction is the usage of *detecting matrices* of Lindström [Canad. Math. Bull., 1965], which is a combinatorial tool that, to the best of our knowledge, has not yet been used for proving complexity lower bounds. As a side result, we prove that the running time of the algorithms of Abasi et al. [MFCS 2014] and of Gabizon et al. [ESA 2015] for the r-monomial detection problem are optimal under ETH.

1998 ACM Subject Classification G.2.2 Graph Theory, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases multicoloring, Kneser graph homomorphism, ETH lower bound

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.18

* A full version of the paper is available at https://arxiv.org/abs/1607.03432.

† Work supported by the National Science Centre of Poland, grants number 2013/11/D/ST6/03073 (MP, MW) and 2015/17/N/ST6/01224 (AS). 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). Michał Pilipczuk is supported by the Foundation for Polish Science (FNP) via the START stipend programme.

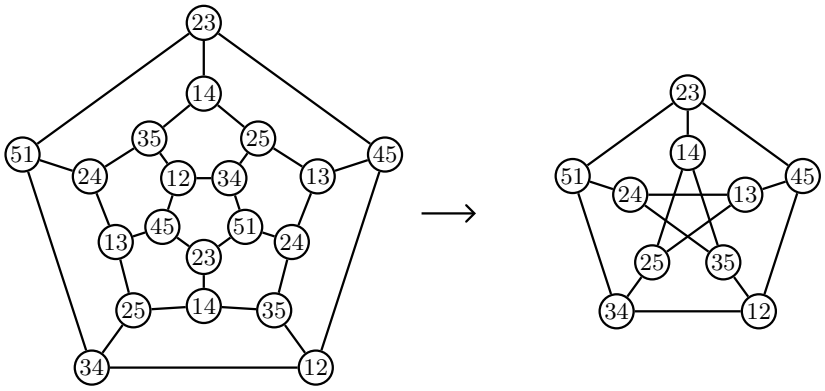

Figure 1 A (5:2)-coloring of the dodecahedron (left) which can be seen as a homomorphism to $KG_{5,2}$ (the Petersen graph, right). The homomorphism is given by identifying the pairs of opposite vertices in the corresponding regular solid.

1 Introduction

The complexity of determining the chromatic number of a graph is undoubtedly among the most intensively studied computational problems. Countless variants and generalizations of graph colorings have been introduced and investigated. Here, we focus on *multicolorings*, also known as $(a{:}b)$-colorings. In this setting, we are given a graph G, a palette of a colors, and a number $b \leq a$. An *$(a{:}b)$-coloring* of G is any assignment of b distinct colors to each vertex so that adjacent vertices receive disjoint subsets of colors. The $(a{:}b)$-COLORING problem asks whether G admits an $(a{:}b)$-coloring. For $b = 1$ we obtain the classic graph coloring problem. The smallest a for which an $(a{:}b)$-coloring exists is called the *b-fold chromatic number*, denoted by $\chi_b(G)$.

The motivation behind $(a{:}b)$-colorings can be perhaps best explained by showing the connection with the *fractional chromatic number*. For a graph G, it is denoted as $\chi_f(G)$ and defined as the optimum value of the natural LP relaxation of the problem of computing the chromatic number of G, expressed as finding a cover of the vertex set using the minimum possible number of independent sets. It can be easily seen that by relaxing the standard coloring problem by allowing b times more colors while requiring that every vertex receives b colors and adjacent vertices receive disjoint subsets, with increasing b we approximate $\chi_f(G)$ better and better. Consequently, $\lim_{b\to\infty} \chi_b(G)/b = \chi_f(G)$.

Another connection concerns *Kneser graphs*. Recall that for positive integers a, b with $b < a/2$, the Kneser graph $KG_{a,b}$ has all b-element subsets of $\{1, 2, \ldots, a\}$ as vertices, and two subsets are considered adjacent if and only if they are disjoint. For instance, $KG_{5,2}$ is the well-known Petersen graph (see Fig. 1, right). Thus, $(a{:}b)$-coloring of a graph G can be interpreted as a homomorphism from G to the Kneser graph $KG_{a,b}$ (see Fig. 1). Kneser graphs are well studied in the context of colorings, mostly due to the celebrated result of Lovász [28], who determined their chromatic number, initiating the field of topological combinatorics.

Multicolorings and $(a{:}b)$-colorings have been studied both from combinatorial [6, 12, 26] and algorithmic [5, 18, 19, 24, 25, 29, 30, 33] points of view. The main real-life motivation comes from the problem of assigning frequencies to nodes in a cellular network so that adjacent nodes receive disjoint sets of frequencies on which they can operate. This makes (near-)planar and distributed settings particularly interesting for practical applications. We refer to the survey of Halldórsson and Kortsarz [17] for a broader discussion.

In this paper we focus on the paradigm of exact exponential time algorithms: given a graph G on n vertices and numbers $a \geq b$, we would like to determine whether G is $(a{:}b)$-colorable as quickly as possible. Since the problem is already NP-hard for $a = 3$ and $b = 1$, we do not expect it to be solvable in polynomial time, and hence look for an efficient exponential-time algorithm. A straightforward dynamic programming approach yields an algorithm with running time[1] $\mathcal{O}^\star(2^n \cdot (b+1)^n)$ as follows. For each function $\eta\colon V(G) \to \{0, 1, \ldots, b\}$ and each $k = 0, 1, \ldots, a$, we create one boolean entry $D[\eta, k]$ denoting whether one can choose k independent sets in G so that every vertex $v \in V(G)$ is covered exactly $\eta(v)$ times. Then value $D[\eta, k]$ can be computed as a disjunction of values $D[\eta', k-1]$ over η' obtained from η by subtracting 1 on vertices from some independent set in G.

This simple algorithm can be improved by finding an appropriate algebraic formula for the number of $(a{:}b)$-colorings of the graph and using the inclusion-exclusion principle to compute it quickly, similarly as in the case of standard colorings [2]. Such an algebraic formula was given by Nederlof [32, Theorem 3.5] in the context of a more general Multi Set Cover problem. Nederlof also observed that in the case of $(a{:}b)$-coloring, a simple application of the inclusion-exclusion principle to compute the formula yields an $\mathcal{O}^\star((b+1)^n)$-time exponential-space algorithm. Hua et al. [21] noted that the formulation of Nederlof [32] for Multi Set Cover can be also used to obtain a polynomial-space algorithm for this problem. By taking all maximal independent sets to be the family in the Multi Set Cover problem, and applying the classic Moon-Moser upper bound on their number [31], we obtain an algorithm for $(a{:}b)$-coloring that runs in time $\mathcal{O}^\star(3^{n/3} \cdot (b+1)^n)$ and uses polynomial space. Note that by plugging $b = 1$ to the results above, we obtain algorithms for the standard coloring problem using $\mathcal{O}^\star(2^n)$ time and exponential space, or using $\mathcal{O}^\star(2.8845^n)$ time and polynomial space, which almost matches the fastest known procedures [2].

The complexity of $(a{:}b)$-coloring becomes particularly interesting in context of the Graph Homomorphism problem: given graphs G and H, with n and h vertices respectively, determine whether G admits a homomorphism to H. By the celebrated result of Hell and Nešetřil [20] the problem is in P if H is bipartite and NP-complete otherwise. For quite a while it was open whether there is an algorithm for Graph Homomorphism running in time $2^{\mathcal{O}(n+h)}$. It was recently answered in the negative by Cygan et al. [9]; more precisely, they proved that an algorithm with running time $2^{o(n \log h)}$ contradicts the Exponential Time Hypothesis (ETH) of Impagliazzo et al. [22]. However, Graph Homomorphism is a very general problem, hence researchers try to uncover a more fine-grained picture and identify families of graphs $\mathcal{H}$ such that the problem can be solved more efficiently whenever $H \in \mathcal{H}$. For example, Fomin, Heggernes and Kratsch [13] showed that when H is of treewidth at most t, then Graph Homomorphism can be solved in time $\mathcal{O}^\star((t+3)^n)$. It was later extended to graphs of cliquewidth bounded by t, with $\mathcal{O}^\star((2t+1)^{\max\{n,h\}})$ time bound by Wahlström [35]. On the other hand, H needs not be sparse to admit efficient homomorphism testing: the family of cliques admits the $\mathcal{O}^\star(2^n)$ running time as shown by Björklund et al. [2]. As noted above, this generalizes to Kneser graphs $KG_{a,b}$, by the $\mathcal{O}^\star((b+1)^n)$-time algorithm of Nederlof. In this context, the natural question is whether the appearance of b in the base of the exponent is necessary, or is there an algorithm running in time $\mathcal{O}^\star(c^n)$ for some universal constant c independent of b.

Our contribution. We show that the algorithms for $(a{:}b)$-coloring mentioned above are essentially optimal under the Exponential Time Hypothesis. Specifically, we prove the following results:

[1] The $\mathcal{O}^\star(\cdot)$ notation hides factors polynomial in the input size.

▶ **Theorem 1.** *If there is an algorithm for* $(a{:}b)$-COLORING *that runs in time* $f(b) \cdot 2^{o(\log b)\cdot n}$, *for some computable function* $f(b)$, *then ETH fails. This holds even if the algorithm is only required to work on instances where* $a = \Theta(b^2 \log b)$.

▶ **Corollary 2.** *If there is an algorithm for* GRAPH HOMOMORPHISM *that runs in time* $f(h) \cdot 2^{o(\log\log h)\cdot n}$, *for some computable* $f(h)$, *then ETH fails. This holds even if the algorithm is only required to work on instances where* H *is a Kneser graph* $KG_{a,b}$ *with* $a = \Theta(b^2 \log b)$.

The bound for $(a{:}b)$-COLORING is tight, as the straightforward $\mathcal{O}^\star(2^n \cdot (b{+}1)^n) = 2^{\mathcal{O}(\log b)\cdot n}$ dynamic programming algorithm already shows. At first glance, one might have suspected that $(a{:}b)$-COLORING, as an interpolation between classical coloring and fractional coloring, both solvable in $2^{\mathcal{O}(n)}$ time [16], should be just as easy; Theorem 1 refutes this suspicion.

Corollary 2 in particular excludes any algorithm for testing homomorphisms into Kneser graphs with running time $2^{\mathcal{O}(n+h)}$. It cannot give a tight lower bound matching the result of Cygan et al. [9] for general homomorphisms, because $h = |V(KG_{a,b})| = \binom{a}{b}$ is not polynomial in b. On the other hand, it exhibits the first explicit family of graphs H for which the complexity of GRAPH HOMOMORPHISM increases with h.

In our proof, we first show a lower bound for the list variant of the problem, where every vertex is given a list of colors that can be assigned to it (see Section 2 for formal definitions). The list version is reduced to the standard version by introducing a large Kneser graph $KG_{a+b,b}$; we need a and b to be really small so that the size of this Kneser graph does not dwarf the size of the rest of the construction. However, this is not necessary for the list version, where we obtain lower bounds for a much wider range of functions $b(n)$.

▶ **Theorem 3.** *If there is an algorithm for* LIST $(a{:}b)$-COLORING *that runs in time* $2^{o(\log b)\cdot n}$, *then ETH fails. This holds even if the algorithm is only required to work on instances where* $a = \Theta(b^2 \log b)$ *and* $b = \Theta(b(n))$ *for an arbitrarily chosen polynomial-time computable function* $b(n)$ *such that* $b(n) \in \omega(1)$ *and* $b(n) = \mathcal{O}(n/\log n)$.

The crucial ingredient in the proof of Theorem 3 is the usage of *d-detecting matrices* introduced by Lindström [27]. We choose to work with their combinatorial formulation, hence we shall talk about *d-detecting families*. Suppose we are given some universe U and there is an unknown function $f\colon U \to \{0, 1, \ldots, d-1\}$, for some fixed positive integer d. One may think of U as consisting of coins of unknown weights that are integers between 0 and $d-1$. We would like to learn f (the weight of every coin) by asking a small number of queries of the following form: for a subset $X \subseteq U$, what is $\sum_{e\in X} f(e)$ (the total weight of coins in X)? A set of queries sufficient for determining all the values of an arbitrary f is called a *d-detecting family*. Of course f can be learned by asking $|U|$ questions about single coins, but it turns out that significantly fewer questions are needed: there is a d-detecting family of size $\mathcal{O}(|U|/\log|U|)$, for every fixed d [27]. The logarithmic factor in the denominator will be crucial for deriving our lower bound.

Let us now sketch how d-detecting families are used in the proof of Theorem 3. Given an instance φ of 3-SAT with n variables and $\mathcal{O}(n)$ clauses, and a number $b \leq n/\log n$, we will construct an instance G of LIST $(a{:}b)$-COLORING for some a. This instance will have a positive answer if and only if φ is satisfiable, and the constructed graph G will have $\mathcal{O}(n/\log b)$ vertices. It can be easily seen that this will yield the promised lower bound.

Partition the clause set C of φ into groups $C_1, C_2, \ldots, C_p$, each of size roughly b; thus $p = \mathcal{O}(n/b)$. Similarly, partition the variable set V of φ into groups $V_1, \ldots, V_q$, each of size roughly $\log_2 b$; thus $q = \mathcal{O}(n/\log b)$. In the output instance we create one vertex per each variable group—hence we have $\mathcal{O}(n/\log b)$ such vertices—and one block of vertices per each

clause group, whose size will be determined in a moment. Our construction ensures that the set of colors assigned to a vertex created for a variable group misses one color from some subset of b colors. The choice of the missing color corresponds to one of $2^{\log_2 b} = b$ possible boolean assignments to the variables of the group.

Take any vertex u from a block of vertices created for some clause group C_j. We make it adjacent to vertices constructed for precisely those variable groups V_i, for which there is some variable in V_i that occurs in some clause of C_j. This way, u can only take a subset of the above missing colors corresponding to the chosen assignment on variables relevant to C_j. By carefully selecting the list of u, and some additional technical gadgeteering, we can express a constraint of the following form: the total number of satisfied literals in some subset of clauses of C_j is exactly some number. Thus, we could verify that every clause of C_j is satisfied by creating a block of $|C_j|$ vertices, each checking one clause. However, the whole graph output by the reduction would then have $\mathcal{O}(n)$ vertices, and we would not obtain any non-trivial lower bound. Instead, we create one vertex per each question in a d-detecting family on the universe $U = C_j$, which has size $\mathcal{O}(|C_j|/\log|C_j|) = \mathcal{O}(|C_j|/\log b)$. Then, the total number of vertices in the constructed graph will be $\mathcal{O}(n/\log b)$, as intended.

Finally, we observe that from our main result one can infer a lower bound for the complexity of the (r,k)-Monomial Testing problem. Recall that in this problem we are given an arithmetic circuit that evaluates a homogenous polynomial $P(x_1, x_2, \ldots, x_n)$ over some field $\mathbb{F}$; here, a polynomial is homogenous if all its monomials have the same total degree k. The task is to verify whether P has some monomial in which every variable has individual degree not larger than r, for a given parameter r. Abasi et al. [1] gave a randomized algorithm solving this problem in time $\mathcal{O}^\star(2^{\mathcal{O}(k\cdot\frac{\log r}{r})})$, where k is the degree of the polynomial, assuming that $\mathbb{F} = \mathrm{GF}(p)$ for a prime $p \leq 2r^2 + 2r$. This algorithm was later derandomized by Gabizon et al. [14] within the same running time, but under the assumption that the circuit is *non-cancelling*: it has only input, addition, and multiplication gates. Abasi et al. [1] and Gabizon et al. [14] gave a number of applications of low-degree monomial detection to concrete problems. For instance, r-Simple k-Path, the problem of finding a walk of length k that visits every vertex at most r times, can be solved in time $\mathcal{O}^\star(2^{\mathcal{O}(k\cdot\frac{\log r}{r})})$. However, for r-Simple k-Path, as well as other problems that can be tackled using this technique, the best known lower bounds under ETH exclude only algorithms with running time $\mathcal{O}^\star(2^{o(\frac{k}{r})})$. Whether the $\log r$ factor in the exponent is necessary was left open by Abasi et al. and Gabizon et al.

We observe that the List $(a{:}b)$-coloring problem can be reduced to (r,k)-Monomial Testing over the field $\mathrm{GF}(2)$ in such a way that an $\mathcal{O}^\star(2^{k\cdot o(\frac{\log r}{r})})$-time algorithm for the latter would imply a $2^{o(\log b)\cdot n}$-time algorithm for the former, which would contradict ETH. Thus, we show that the known algorithms for (r,k)-Monomial Testing most probably cannot be sped up in general; nevertheless, the question of lower bounds for specific applications remains open. However, going through List $(a{:}b)$-coloring to establish a lower bound for (r,k)-Monomial Testing is actually quite a detour, because the latter problem has a much larger expressive power. Therefore, we also give a more straightforward reduction that starts from a convenient form of Subset Sum; this reduction also proves the lower bound for a wider range of r, expressed as a function of k.

Outline. In Section 2 we set up the notation as well as recall definitions and well-known facts. We also discuss d-detecting families, the main combinatorial tool used in our reduction. In Section 3 we prove the lower bound for the list version of the problem, i.e., Theorem 3, and sketch the few steps needed for the standard version, thereby proving Theorem 1. Section 4 is

devoted to deriving lower bounds for low-degree monomial testing. Due to space constraints, proofs of statements marked with (♠) are deferred to the full version [3] of this paper.

2 Preliminaries

Notation. We use standard graph notation, see e.g. [10, 11]. All graphs we consider in this paper are simple and undirected. For an integer k, we denote $[k] = \{0, \ldots, k-1\}$. By $\uplus$ we denote the disjoint union, i.e., by $A \uplus B$ we mean $A \cup B$ with the indication that A and B are disjoint. 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, or both are NO-instances of the respective problems.

Exponential-Time Hypothesis. The Exponential Time Hypothesis (ETH) of Impagliazzo et al. [22] states that there exists a constant $c > 0$, such that there is no algorithm solving 3-SAT in time $\mathcal{O}^\star(2^{cn})$. 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* [23], which essentially gives a 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 4** (see e.g. Theorem 14.4 in [10]). *Unless ETH fails, there is no algorithm for* 3-SAT *that runs in time* $2^{o(n+m)}$*, on formulas with* n *variables and* m *clauses.*

We need the following regularization result of Tovey [34]. 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 5** ([34]). *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* $\mathcal{O}(n+m)$ *variables and clauses.*

▶ **Corollary 6.** *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.*

List and nonuniform list (a:b)-coloring. For integers a, b and a graph G with a function $L\colon V(G) \to 2^{[a]}$ (assigning a list of colors to every vertex), an *L-(a:b)-coloring* of G is an assignment of exactly b colors from $L(v)$ to each vertex $v \in V(G)$, such that adjacent vertices get disjoint color sets. The List (a:b)-coloring problem asks, given (G, L), whether an L-(a:b)-coloring of G exists.

As an intermediary step of our reduction, we use the following generalization of list colorings where the number of demanded colors varies with every vertex. For integers a, b, a graph G with a function $L\colon V(G) \to 2^{[a]}$ and a *demand function* $\beta\colon V(G) \to \{1, \ldots, b\}$, an *$L$-($a$:$\beta$)-coloring* of G is an assignment of exactly $\beta(v)$ colors from $L(v)$ to each vertex $v \in V(G)$, such that adjacent vertices get disjoint color sets. Nonuniform List (a:b)-coloring is then the problem in which given (G, L, β) we ask if an L-(a:β)-coloring of G exists.

***d*-detecting families.** In our reductions the following notion plays a crucial role.

▶ **Definition 7.** A *d-detecting family* for a finite set U is a family $\mathcal{F} \subseteq 2^U$ of subsets of U such that for every two functions $f, g : U \to \{0, \ldots, d-1\}$, $f \neq g$, there is a set S in the family such that $\sum_{x \in S} f(x) \neq \sum_{x \in S} g(x)$.

A deterministic construction of sublinear, d-detecting families was given by Lindström [27], together with a proof that even the constant factor 2 in the family size cannot be improved.

▶ **Theorem 8** ([27]). *For every constant $d \in \mathbb{N}$ and finite set U, there is a d-detecting family $\mathcal{F}$ on U of size $\frac{2|U|}{\log_d |U|} \cdot (1 + o(1))$. Furthermore, $\mathcal{F}$ can be constructed in* $\mathrm{poly}(|U|)$ *time.*

Other constructions, generalizations, and discussion of similar results can be found in Grebinski and Kucherov [15], and in Bshouty [4]. Note that the expression $\sum_{x \in S} f(x)$ is just the product of f as a vector in $[d]^{|U|}$ with the characteristic vector of S. Hence, instead of subset families, Lindström speaks of *detecting vectors*, while later works see them as *detecting matrices*, that is, $(0,1)$-matrices with these vectors as rows (which define an injection on $[d]^{|U|}$ despite having few rows). Similar definitions appear in the study of query complexity, e.g., as in the popular Mastermind game [7].

3 Hardness of List $(a{:}b)$-coloring

In this section we show our main technical contribution: an ETH-based lower bound for List $(a{:}b)$-coloring. We begin with key part: reducing an n-variable instance 3-SAT to an instance of Nonuniform List $(a{:}b)$-coloring with only $\mathcal{O}(\frac{n}{\log b})$ vertices. Next, it is rather easy to reduce Nonuniform List $(a{:}b)$-coloring to List $(a{:}b)$-coloring.

3.1 The nonuniform case

We prove the following theorem through the remaining part of this section.

▶ **Theorem 9.** *For any instance ϕ of* (3,4)-SAT *with n variables and any integer $2 \leq b \leq n/\log_2 n$, there is an equivalent instance (G, β, L) of* Nonuniform List $(a{:}2b)$-coloring *such that $a = \mathcal{O}(b^2 \log b)$, $|V(G)| = \mathcal{O}(\frac{n}{\log b})$ and G is 3-colorable. Moreover, the instance (G, β, L) and the 3-coloring of G can be constructed in* $\mathrm{poly}(n)$ *time.*

Consider an instance ϕ of 3-SAT where each variable appears in at most four clauses. Let V be the set of its variables and C be the set of its clauses. Note that $\frac{1}{3}|V| \leq |C| \leq \frac{4}{3}|V|$. Let $a = 12b^2 \cdot \lfloor \log_2 b \rfloor$. We shall construct, for some integers $n_V = \mathcal{O}(|V|/\log b)$ and $n_C = \mathcal{O}(|C|/b)$:

- a partition $V = V_1 \uplus \ldots \uplus V_{n_V}$ of variables into groups of size at most $\lfloor \log_2 b \rfloor$,
- a partition $C = C_1 \uplus \ldots \uplus C_{n_C}$ of clauses into groups of size at most b,
- a function $\sigma \colon \{1, \ldots, n_V\} \to [12 \cdot b \cdot \lfloor \log_2 b \rfloor]$,

such that the following condition holds:

> For any $j = 1, \ldots, n_C$, the variables occurring in clauses of C_j are all different and they all belong to pairwise different variable groups. Moreover, the indices of these groups are mapped to pairwise different values by σ. (✠)

In other words, any two literals of clauses in C_j have different variables, and if they belong to V_i and $V_{i'}$ respectively, then $\sigma(i) \neq \sigma(i')$.

▶ **Lemma 10** (♠). *Partitions $V = V_1 \uplus \ldots \uplus V_{n_V}$, $C = C_1 \uplus \ldots \uplus C_{n_C}$ and a function σ satisfying* (✠) *can be found in time $\mathcal{O}(n)$.*

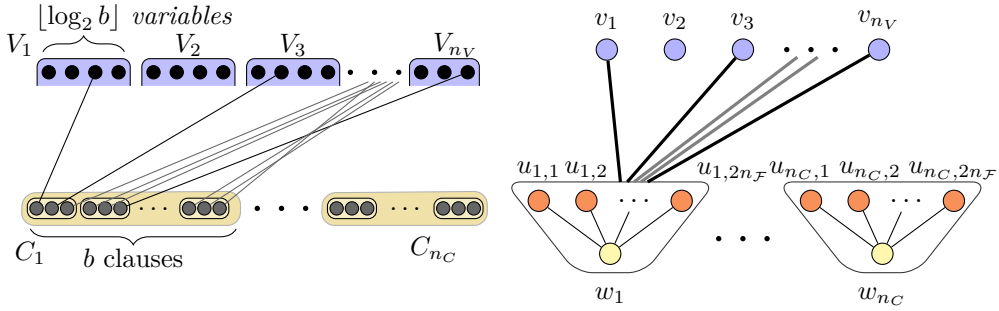

Figure 2 *(left)* The groups of variables and clauses of the formula; literals in C_1 are joined with their variables. Since no variable of V_2 occurs in C_1, we have $2 \notin I_1$ – this may allow us to make $\sigma(2)$ the same number as $\sigma(3)$, say, reducing the total number a of colors needed. *(right)* The constructed graph; thick lines represent edges to all vertices corresponding to C_1.

For every $1 \leq i \leq n_V$, the set V_i of variables admits $2^{|V_i|} \leq b$ different assignments. We will therefore say that each assignment on V_i is *given* by an integer $x \in [b]$, for example by interpreting the first $|V_i|$ bits of the binary representation of x as truth values for variables in V_i. Note that when $|V_i| < \log_2 b$, different integers from $[b]$ may give the same assignment on V_i.

For $1 \leq j \leq n_C$, let $I_j \subseteq \{1, \ldots, n_V\}$ be the set of indices of variable groups that contain some variable occurring in the clauses of C_j. Since every clause contains exactly three literals, property (✠) means that $|I_j| = 3|C_j|$ and that σ is injective over each I_j. See Fig. 2.

For $1 \leq j \leq n_C$, let $\{C_{j,1}, \ldots, C_{j,n_\mathcal{F}}\}$ be a 4-detecting family of subsets of C_j, for some $n_\mathcal{F} = \mathcal{O}(\frac{b}{\log b})$ (we can assume $n_\mathcal{F}$ does not depend on j by adding arbitrary sets when $|C_j| < b$). For every $1 \leq k \leq n_\mathcal{F}$, let $C_{j,n_\mathcal{F}+k} = C_j \setminus C_{j,k}$.

We are now ready to build the graph G, the demand function $\beta : V(G) \to \{1, \ldots, 2b\}$, and the list assignment L as follows.

1. For $1 \leq i \leq n_V$, create a vertex v_i with $\beta(v_i) = b - 1$ and $L(v_i) = \{b \cdot \sigma(i) + x \mid x \in [b]\}$.
2. For $1 \leq j \leq n_C$ and $1 \leq k \leq 2n_\mathcal{F}$, create a vertex $u_{j,k}$ adjacent to each v_i for $i \in I_j$. Let $\beta(u_{j,k}) = |C_{j,k}|$ and

$$L(u_{j,k}) = \{b \cdot \sigma(i) + x \;\mid\; 1 \leq i \leq n_V, x \in [2^{|V_i|}] \text{ such that } x \text{ gives an assignment of } V_i \text{ that satisfies some clause of } C_{j,k}\}.$$

3. For $1 \leq j \leq n_C$, create a vertex w_j, adjacent to each v_i for $i \in I_j$ and to each $u_{j,k}$ $(1 \leq k \leq 2n_\mathcal{F})$. Let $\beta(w_j) = 2|C_j|$ and $L(w_j) = \bigcup_{i \in I_j} \{b \cdot \sigma(i) + x \mid x \in [b]\}$.

Before giving a detailed proof of the correctness, let us describe the reduction in intuitive terms. Note that vertices of type v_i get all but one color from their list; this missing color, say $b \cdot \sigma(i) + x_i$, for some $x_i \in [b]$, defines an assignment on V_i. For every $j = 1, \ldots, n_C$ the goal of the gadget consisting of w_j and vertices $u_{j,k}$ is to express the constraint that every clause in C_j has a literal satisfied by this assignment. Since $w_j, u_{j,k}$ are adjacent to all vertices in $\{v_i \mid i \in I_j\}$, they may only use the missing colors (of the form $b \cdot \sigma(i) + x_i$, where $i \in I_j$). Since $|I_j| = 3|C_j|$, there are $3|C_j|$ such colors and $2|C_j|$ of them go to w_j. This leaves exactly $|C_j|$ colors for vertices of type $u_{j,k}$, corresponding to a choice of $|C_j|$ satisfied literals from the $3|C_j|$ literals in clauses of C_j. The lists and demands for vertices $u_{j,k}$ guarantee that

exactly $|C_{j,k}|$ chosen satisfied literals occur in clauses of $C_{j,k}$. The properties of 4-detecting families will ensure that every clause has exactly one chosen, satisfied literal, and hence at least one satisfied literal. We proceed with formal proofs.

▶ **Lemma 11.** *If ϕ is satisfiable then G is L-$(a{:}\beta)$-colorable.*

Proof. Consider a satisfying assignment η for ϕ. For $1 \leq i \leq n_V$, let $x_i \in [2^{|V_i|}]$ be an integer giving the same assignment on V_i as η. For every clause c of ϕ, choose one literal satisfied by η in it, and let i_c be index of the group V_{i_c} containing the literal's variable. Let $\alpha : V(G) \to \binom{[a]}{\leq 2b}$ be the L-$(a{:}\beta)$-coloring of G defined as follows, for $1 \leq i \leq n_V$, $1 \leq j \leq n_C$, $1 \leq k \leq 2n_{\mathcal{F}}$:

- $\alpha(v_i) = L(v_i) \setminus \{b \cdot \sigma(i) + x_i\}$
- $\alpha(u_{j,k}) = \{b \cdot \sigma(i_c) + x_{i_c} \mid c \in C_{j,k}\}$
- $\alpha(w_j) = \{b \cdot \sigma(i) + x_i \mid i \in I_j \setminus \{i_c \mid c \in C_j\}\}$.

Let us first check that every vertex v gets colors from its list $L(v)$ only. This is immediate for vertices v_i and w_j, while for $u_{j,k}$ it follows from the fact that x_{i_c} gives a partial assignment to V_i that satisfies some clause of $C_{j,k}$.

Now let us check that for every vertex v, the coloring α assigns exactly $\beta(v)$ colors to v. For $\alpha(v_i)$ this follows from the fact that $|L(v_i)| = b$ and $0 \leq x_i < 2^{|V_i|} \leq b$. Since by property (✠), σ is injective on I_j, and thus on $\{i_c \mid c \in C_{j,k}\} \subseteq I_j$, we have $|\alpha(u_{j,k})| = |C_{j,k}| = b(u_{j,k})$. Similarly, since σ is injective on I_j and $|I_j \setminus \{i_c \mid c \in C_j\}| = 3|C_j| - |C_j| = 2|C_j|$, we get $|\alpha(w_j)| = 2|C_j| = \beta(w_j)$.

It remains to argue that the sets assigned to any two adjacent vertices are disjoint. There are three types of edges in the graph, namely $v_i u_{j,k}$, $v_i w_j$, and $w_j u_{j,k}$. The disjointness of $\alpha(w_j)$ and $\alpha(u_{j,k})$ is immediate from the definition of α, since $C_{j,k} \subseteq C_j$. Fix $j = 1, \ldots, n_C$. Since σ is injective on I_j, for any two different $i, i' \in I_j$, we have $b \cdot \sigma(i) + x_i \notin L(v_{i'})$. Hence,

$$\bigcup_{i \in I_j} \alpha(v_i) = \{b \cdot \sigma(i) + x \mid i \in I_j \text{ and } x \in [b]\} \setminus \{b \cdot \sigma(i) + x_i \mid i \in I_j\}.$$

Since $\alpha(u_{j,k}), \alpha(w_j) \subseteq \{b \cdot \sigma(i) + x_i \mid i \in I_j\}$, it follows that edges of types $v_i u_{j,k}$ and $v_i w_j$ received disjoint sets of colors on their endpoints, concluding the proof. ◀

▶ **Lemma 12.** *If G is L-$(a{:}\beta)$-colorable then ϕ is satisfiable.*

Proof. Assume that G is L-$(a{:}\beta)$-colorable, and let α be the corresponding coloring.

For $1 \leq i \leq n_V$, we have $|L(v_i)| = b$ and $|\alpha(v_i)| = b - 1$, so v_i misses exactly one color from its list. Let $b \cdot \sigma(i) + x_i$, for some $x_i \in [b]$, be the missing color. We want to argue that the assignment x for ϕ given by x_i on each V_i satisfies ϕ.

Consider any clause group C_j, for $1 \leq j \leq n_C$. Every vertex in $\{w_j\} \cup \{u_{j,k} \mid 1 \leq k \leq 2n_{\mathcal{F}}\}$ contains $\{v_i \mid i \in I_j\}$ in its neighborhood. Therefore, the sets $\alpha(u_{j,k})$ and $\alpha(w_j)$ are disjoint from $\bigcup_{i \in I_j} \alpha(v_i)$. Since $L(u_{j,k}), L(w_j) \subseteq \{b \cdot \sigma(i) + x' \mid i \in I_j, x' \in [b]\}$, we get that $\alpha(u_{j,k})$ and $\alpha(w_j)$ are contained in the set of missing colors $\{b \cdot \sigma(i) + x_i \mid i \in I_j\}$ (corresponding to the chosen assignment). By property (✠), this set has exactly $|I_j| = 3|C_j|$ different colors. Of these, exactly $2|C_j|$ are contained in $\alpha(w_j)$. Let the remaining $|C_j|$ colors be $\{b \cdot \sigma(i) + x_i \mid i \in J_j\}$, for some subset $J_j \subseteq I_j$ of $|C_j|$ indices.

Since $\alpha(u_{j,k})$ is disjoint from $\alpha(w_j)$, we have $\alpha(u_{j,k}) \subseteq \{b \cdot \sigma(i) + x_i \mid i \in J_j\}$ for all k. By definition of I_j, for every $i \in J_j \subseteq I_j$ there is a variable in V_i that appears in some clause of C_j. By property (✠), it can only occur in one such clause, so let l_i be the literal in the clause of C_j where it appears. For every color $b \cdot \sigma(i) + x_i \in \alpha(u_{j,k})$, by definition of the lists

for $u_{j,k}$ we know that x_i gives a partial assignment to V_i that satisfies some clause of $C_{j,k}$. This means x_i makes the literal l_i true and l_i occurs in a clause of $C_{j,k}$. Therefore, for each k, at least $|\alpha(u_{j,k})| = |C_{j,k}|$ literals from the set $\{l_i \mid i \in J_j\}$ occur in clauses of $C_{j,k}$ and are made true by the assignment x.

Let $f : C_j \to \{0,1,2,3\}$ be the function assigning to each clause $c \in C_j$ the number of literals of c in $\{l_i \mid i \in J_j\}$. By the above, $\sum_{c \in C_{j,k}} f(c) \geq |C_{j,k}|$ for $1 \leq k \leq 2n_{\mathcal{F}}$. Since each literal in $\{l_i \mid i \in J_j\}$ belongs to some clause of C_j, we have $\sum_{c \in C_j} f(c) = |J_j| = |C_j|$. Then,

$$\sum_{c \in C_{j,k}} f(c) = \sum_{c \in C_j} f(c) - \sum_{c \in C_{j,n_{\mathcal{F}}+k}} f(c) \leq |C_j| - |C_{j,n_{\mathcal{F}}+k}| = |C_{j,k}|.$$

Hence $\sum_{c \in C_{j,k}} f(c) = |C_{j,k}|$ for $1 \leq k \leq 2n_{\mathcal{F}}$. Let $g : C_j \to \{0,1,2,3\}$ be the constant function $g \equiv 1$. Note that

$$\sum_{c \in C_{j,k}} g(c) = |C_{j,k}| = \sum_{c \in C_{j,k}} f(c).$$

Since $\{C_{j,1}, \ldots, C_{j,n_{\mathcal{F}}}\}$ is a 4-detecting family, this implies that $f \equiv 1$. Thus, for every clause c of C_j we have $f(c) = 1$, meaning that there is a literal from the set $\{l_i \mid i \in J_j\}$ in this clause. All these literals are made positive by the assignment η, therefore all clauses of C_j are satisfied. Since $j = 1, \ldots, n_C$ was arbitrary, this concludes the proof that η is a satisfying assignment for ϕ. ◀

The construction can clearly be made in polynomial time and the total number of vertices is $n_V + n_C \cdot \mathcal{O}(\frac{b}{\log b}) + n_C = \mathcal{O}(\frac{n}{\log b})$. Moreover, we get a proper 3-coloring of G, by coloring vertices of the type v_i by color 1, vertices of the type $u_{j,k}$ by color 2, and vertices of the type w_j by color 3. By Lemmas 11 and 12, this concludes the proof of Theorem 9.

3.2 The uniform case

In this section we reduce the nonuniform case to the uniform one, and state the resulting lower bound on the complexity of List $(a{:}b)$-coloring.

▶ **Lemma 13.** *For any instance $I = (G, \beta, L)$ of* Nonuniform List $(a{:}b)$-coloring *where the graph G is t-colorable, there is an equivalent instance (G, L') of* List $((a+tb){:}b)$-coloring. *Moreover, given a t-coloring of G the instance (G, L') can be constructed in time polynomial in $|I| + b$.*

Proof. Let $c\colon V(G) \to [t]$ be a t-coloring of G. For every vertex v, define a set of filling colors $F(v) = \{a + c(v)b + i \,:\, i = 0, \ldots, b - |\beta(v)| - 1\}$ and put $L'(v) = L(v) \cup F(v)$.

Let $\alpha\colon V(G) \to 2^{[a]}$ be an L-$(a{:}\beta)$-coloring of G. We define a coloring $\alpha'\colon V(G) \to 2^{[a+tb]}$ by setting $\alpha'(v) = \alpha(v) \cup F(v)$ for every vertex $v \in V(G)$. Observe that $\alpha'(v) \subseteq L'(v)$ and $|\alpha'(v)| = |\alpha(v)| + (b - |\beta(v)|) = b$. Since α was a proper L-$(a{:}\beta)$-coloring, adjacent vertices can only share the filling colors. However, the lists of adjacent vertices have disjoint subsets of filling colors, since these vertices are colored differently by c. It follows that α' is an L'-$(a{:}b)$-coloring of G.

Conversely, let $\alpha'\colon V(G) \to 2^{[a+tb]}$ be an L'-$(a{:}b)$-coloring of G. For every vertex v, we have $|\alpha'(v) \cap [a]| = b - |\alpha'(v) \cap F(v)| \geq b - (b - |\beta(v)|) = |\beta(v)|$. Define $\alpha(v)$ to be any cardinality $\beta(v)$ subset of $\alpha'(v) \cap [a]$. It is immediate that α is an L-$(a{:}\beta)$-coloring of G. ◀

We are now ready to prove one of our main results.

▶ **Theorem 3.** *If there is an algorithm for* List $(a{:}b)$-coloring *that runs in time* $2^{o(\log b)\cdot n}$, *then ETH fails. This holds even if the algorithm is only required to work on instances where* $a = \Theta(b^2 \log b)$ *and* $b = \Theta(b(n))$ *for an arbitrarily chosen polynomial-time computable function* $b(n)$ *such that* $b(n) \in \omega(1)$ *and* $b(n) = \mathcal{O}(n/\log n)$.

Proof. Let $b(n)$ be a function as in the statement. We can assume w.l.o.g. that $2 \leq b(n) \leq n/\log_2 n$ (otherwise, replace $b(n)$ with $b'(n) = 2 + \lfloor b(n)/c \rfloor$ in the reasoning below, for c a big enough constant; clearly $b'(n) = \Theta(b(n))$). Fix a function $f(b) = o(\log b)$ and assume there is an algorithm $\mathcal{A}$ for List $(a{:}b)$-coloring that runs in time $2^{f(b)\cdot n}$, whenever $b = \Theta(b(n))$. Consider an instance of (3,4)-SAT with n variables. Let $b = b(n)$. By Theorem 9 in poly(n) time we get an equivalent instance (G, β, L) of Nonuniform List $(a{:}(2b))$-coloring such that $a = \Theta(b^2 \log b)$, $|V(G)| = \mathcal{O}(\frac{n}{\log b})$, and a 3-coloring of G. Next, by Lemma 13 in poly(n) time we get an equivalent instance (G, L') of List $((a + 6b){:}(2b))$-coloring. Finally, we solve the instance (G, L') using algorithm $\mathcal{A}$. Since $b(n) = \omega(1)$, we have $f(b(n)) = o(\log(b(n)))$, and $\mathcal{A}$ runs in time $2^{o(\log b(n))\cdot|V(G)|}$. Thus, we solved (3,4)-SAT in time $2^{o(\log b(n))\cdot|V(G)|} = 2^{o(\log b(n))\cdot\frac{n}{\log b(n)}} = 2^{o(n)}$. By Corollary 6, this contradicts ETH. ◀

Finally, we reduce List $(a{:}b)$-coloring to $(a{:}b)$-coloring. This is done by increasing the number of colors by b, adding a Kneser graph $KG_{a+b,b}$ (which can be colored essentially only by assigning each b-set of colors to its corresponding vertex), and replacing the lists by edges to appropriate vertices of the Kneser graph.

▶ **Lemma 14 (♠).** *Given an instance of* List $(a{:}b)$-coloring *with* n *vertices, an equivalent instance of* $(a + b : b)$-coloring *with* $n + \binom{a+b}{b}$ *vertices can be computed in* poly$(n, \binom{a+b}{b})$*-time.*

For $b \in o(\frac{\log n}{\log\log n})$, $a = \mathcal{O}(b^2 \log b)$, we show that $\binom{a+b}{b} = o(n)$, allowing us to compose our reductions with Lemma 14. An analysis similar to the one in Theorem 3 then concludes the proof of Theorem 1 and Corollary 2; we refer to the full version [3] for details.

4 Low-degree testing

In this section we derive lower bounds for (r, k)-Monomial Testing. In this problem, we are given an arithmetic circuit C over some field $\mathbb{F}$ (with input, constant, addition, and multiplication gates). One gate is designated to be the output gate, and it computes some polynomial P of the variables $x_1, x_2, \ldots, x_n$ that appear in the input gates. We assume that P is a homogenous polynomial of degree k, i.e., all its monomials have total degree k. The task is to verify whether P contains an r-monomial, i.e., a monomial in which every variable has its individual degree bounded by r, for a given $r \leq k$. Abasi et al. [1] gave a very fast randomized algorithm for (r, k)-Monomial Testing.

▶ **Theorem 15** (Abasi et al. [1]). *Fix integers* r, k *with* $2 \leq r \leq k$. *Let* $p \leq 2r^2 + 2r$ *be a prime, and let* $g \in \mathrm{GF}(p)[x_1, \ldots, x_n]$ *be a homogenous polynomial of degree* k, *computable by a circuit* C. *There is a randomized algorithm running in time* $\mathcal{O}(r^{2k/r}|C|(rn)^{\mathcal{O}(1)})$ *which:*

- *with probability at least* $1/2$ *answers YES when* g *contains an* r*-monomial,*
- *always answers NO when* g *contains no* r*-monomial.*

This result was later derandomized by Gabizon et al. [14] under the assumption that the circuit is *non-cancelling*, that is, it contains only input, addition, and multiplication gates. Many concrete problems like r-Simple k-Path can be reduced to (r, k)-Monomial

Testing by encoding the set of candidate objects as monomials of some large polynomial, so that "good" objects correspond to monomials with low individual degrees.

As we show in the full version [3] of this paper, this is also the case for List $(a{:}b)$-coloring. Namely, for an instance (G, L) of List $(a{:}b)$-coloring we can construct a homogeneous polynomial p_G of degree $k = 2bn$ with $n(a+1)$ variables, together with a circuit of size $2^n \operatorname{poly}(a, n)$ evaluating it, such that G admits a list $(a{:}b)$-coloring iff p_G contains a b-monomial. Using Theorem 15 with $r = b$, we get yet another polynomial-space algorithm for List $(a{:}b)$-coloring, running in time $\mathcal{O}(b^{\mathcal{O}(n)} \operatorname{poly}(n))$. Similarly, if the running time in Theorem 15 was improved to $2^{o(\log r/r)\cdot k} \cdot |C| \operatorname{poly}(r, n)$, then we would get an algorithm for List $(a{:}b)$-coloring in time $\mathcal{O}(2^{o(\log b)n} \operatorname{poly}(n))$, which contradicts ETH by Theorem 3. However, a careful examination shows that this chain of reductions would only yield instances of (r, k)-Monomial Testing with $r = \mathcal{O}(\sqrt{k/\log k})$. Hence, this does not exclude the existence of a fast algorithm that works only for large r. Below we show a more direct reduction, which excludes fast algorithms for a wider spectrum of pairs (r, k).

In the Carry-Less Subset Sum problem, we are given $n+1$ numbers $s, a_1, \ldots, a_n$, each represented as n decimal digits. For any number x, the j-th decimal digit of x is denoted by $x^{(j)}$. It is assumed that $\sum_{i=1}^{n} a_i^{(j)} < 10$, for every $j = 1, \ldots, n$. The goal is to verify whether there is a sequence of indices $1 \leq i_1 < \ldots < i_k \leq n$ such that $\sum_{q=1}^{k} a_{i_q} = s$. Note that by the small sum assumption, this is equivalent to the statement that $\sum_{q=1}^{k} a_{i_q}^{(j)} = s^{(j)}$, for every $j = 1, \ldots, n$. The standard NP-hardness reduction from 3-SAT to Subset Sum (see e.g. [8]) in fact gives instances of Carry-Less Subset Sum of linear size, yielding the following.

▶ **Lemma 16 (♠).** *Unless ETH fails, the* Carry-Less Subset Sum *problem cannot be solved in* $2^{o(n)}$ *time.*

We proceed with a sketch of the reduction from Carry-Less Subset Sum to (r, k)-Monomial Testing; details can be found in the full version [3] of this paper. Let us choose a parameter $t \in \{1, \ldots, n\}$. Assume w.l.o.g. that t divides n (otherwise, add zeroes at the end of every input number). Let $q = n/t$. For an n-digit decimal number x, for every $j = 1, \ldots t$, let $x^{[j]}$ denote the q-digit number given by the j-th block of q digits in x, i.e., $x^{[j]} = (x^{(jq-1)} \cdots x^{((j-1)q)})_{10}$.

Let $r = 10^q - 1$. Define the following polynomial over GF(2):

$$q_S = \prod_{i=1}^{n} \left(y_i + z_i \cdot \prod_{j=1}^{t} x_j^{a_i^{[j]}} \right) \cdot \prod_{j=1}^{t} x_j^{r - s^{[j]}} = \sum_{S \subseteq \{1,\ldots,n\}} \prod_{j=1}^{t} x_j^{\sum_{i \in S} a_i^{[j]} + r - s^{[j]}} \prod_{i \notin S} y_i \prod_{i \in S} z_i.$$

Let p_S denote the polynomial obtained from q_S by filtering out all the monomials of degree different than $k = tr + n$. The first expression defining q_S gives a circuit of size $\mathcal{O}(nt)$, and thus with a standard construction, we show p_S can be evaluated by a circuit of size $\mathcal{O}(nt^2 r + n^2 t)$. It is relatively easy to see that by construction, $(s, a_1, \ldots, a_n)$ is a YES-instance of Carry-Less Subset Sum iff q_S contains the monomial $\prod_{j=1}^{t} x_j^r \prod_{i \notin S} y_i \prod_{i \in S} z_i$, for some $S \subseteq \{1, \ldots, n\}$ (the variables y_i and z_i guaranteeing that no pair of monomials cancels out). This in turn holds iff p_S contains an r-monomial (with exactly n variables y_i and z_i, and hence degree exactly r for each x_i variable).

With this reduction, we obtain our main lower bound for (r, k)-Monomial Testing. We state it in the most general, but technical form, and derive an exemplary corollary below.

▶ **Theorem 17 (♠).** *If there is an algorithm solving* (r, k)-Monomial Testing *in time* $2^{o(k \log r/r)} |C|^{\mathcal{O}(1)}$*, then ETH fails. The statement remains true even if the algorithm works*

only for instances where $r = 2^{\Theta(n/t(n))}$ *and* $k = t(n)2^{\Theta(n/t(n))}$*, for an arbitrarily chosen function* $t : \mathbb{N} \to \mathbb{N}$ *computable in* $2^{o(n)}$ *time, such that* $t(n) = \omega(1)$ *and* $t(n) \leq n$ *for every* n*.*

▶ **Theorem 18 (♠).** *Let* $\sigma \in [0, 1)$*. Then, unless ETH fails, there is no algorithm for* (r, k)-Monomial Testing *that solves instances with* $r = \Theta(k^\sigma)$ *in time* $2^{o(k \cdot \frac{\log r}{r})} \cdot |C|^{\mathcal{O}(1)}$*.*

In particular, no algorithm solves (r, k)-Monomial Testing in time $2^{o(\frac{\log r}{r}) \cdot k} \cdot |C|^{\mathcal{O}(1)}$ for all input values r, unless ETH fails.

Acknowledgements. The authors thank Andreas Björklund and Matthias Mnich for sharing the problem considered in this paper.

References

1 Hasan Abasi, Nader H. Bshouty, Ariel Gabizon, and Elad Haramaty. On r-simple k-path. In *MFCS 2015*, volume 8635 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2014.

2 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009.

3 Marthe Bonamy, Lukasz Kowalik, Michal Pilipczuk, Arkadiusz Socala, and Marcin Wrochna. Tight lower bounds for the complexity of multicoloring. *CoRR*, abs/1607.03432, 2016. URL: `http://arxiv.org/abs/1607.03432`.

4 Nader H. Bshouty. Optimal algorithms for the coin weighing problem with a spring scale. In *COLT 2009*, 2009.

5 Marie G. Christ, Lene M. Favrholdt, and Kim S. Larsen. Online multi-coloring with advice. In *WAOA 2014*, volume 8952 of *Lecture Notes in Computer Science*, pages 83–94. Springer, 2014.

6 V. Chvátal, M. R. Garey, and D. S. Johnson. Two results concerning multicoloring. In *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Math.*, pages 151–154. Elsevier, 1978.

7 Vasek Chvátal. Mastermind. *Combinatorica*, 3(3):325–329, 1983. `doi:10.1007/BF02579188`.

8 T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2009.

9 Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub Pachocki, and Arkadiusz Socała. Tight bounds for Graph Homomorphism and Subgraph Isomorphism. In *SODA 2016*, pages 1643–1649, 2016.

10 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

11 Reinhard Diestel. *Graph Theory*. Springer-Verlag Heidelberg, 2010.

12 David C. Fisher. Fractional colorings with large denominators. *J. Graph Theory*, 20(4):403–409, 1995.

13 Fedor V. Fomin, Pinar Heggernes, and Dieter Kratsch. Exact algorithms for graph homomorphisms. *Theory Comput. Syst.*, 41(2):381–393, 2007. `doi:10.1007/s00224-007-2007-x`.

14 Ariel Gabizon, Daniel Lokshtanov, and Michal Pilipczuk. Fast algorithms for parameterized problems with relaxed disjointness constraints. In *ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 545–556. Springer, 2015.

15 Vladimir Grebinski and Gregory Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28(1):104–124, 2000. URL: `https://hal.inria.fr/inria-00073517`, `doi:10.1007/s004530010033`.

16 Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. Corrigendum available at: http://dx.doi.org/10.1007/BF02579139. doi:10.1007/BF02579273.

17 Magnús M. Halldórsson and Guy Kortsarz. Multicoloring: Problems and techniques. In *MFCS 2013*, volume 3153 of *Lecture Notes in Computer Science*, pages 25–41. Springer, 2004.

18 Magnús M. Halldórsson, Guy Kortsarz, Andrzej Proskurowski, Ravit Salman, Hadas Shachnai, and Jan Arne Telle. Multicoloring trees. *Inf. Comput.*, 180(2):113–129, 2003.

19 Frédéric Havet. Channel assignment and multicolouring of the induced subgraphs of the triangular lattice. *Discrete Math.*, 233(1-3):219–231, 2001.

20 Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990. doi:10.1016/0095-8956(90)90132-J.

21 Qiang-Sheng Hua, Yuexuan Wang, Dongxiao Yu, and Francis C. M. Lau. Dynamic programming based algorithms for set multicover and multiset multicover problems. *Theor. Comput. Sci.*, 411(26-28):2467–2474, 2010.

22 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.

23 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.

24 Mustapha Kchikech and Olivier Togni. Approximation algorithms for multicoloring planar graphs and powers of square and triangular meshes. *Discrete Math. Theor. Comput. Sci.*, 8(1):159–172, 2006.

25 Fabian Kuhn. Local multicoloring algorithms: Computing a nearly-optimal TDMA schedule in constant time. In *STACS 2009*, volume 3 of *LIPIcs*, pages 613–624. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany, 2009. doi:10.4230/LIPIcs.STACS.2009.1852.

26 Wensong Lin. Multicoloring and Mycielski construction. *Discrete Math.*, 308(16):3565–3573, 2008.

27 Bernt Lindström. On a combinatorial problem in number theory. *Canad. Math. Bull*, 8(4):477–490, 1965. doi:10.4153/CMB-1965-034-2.

28 László Lovász. Kneser's conjecture, chromatic number, and homotopy. *J. Comb. Theory, Ser. A*, 25(3):319–324, 1978.

29 Dániel Marx. The complexity of tree multicolorings. In *MFSC 2002*, volume 2420 of *Lecture Notes in Computer Science*, pages 532–542. Springer, 2002.

30 Colin McDiarmid and Bruce A. Reed. Channel assignment and weighted coloring. *Networks*, 36(2):114–117, 2000.

31 J. W. Moon and L. Moser. On cliques in graphs. *Israel J. Math.*, 3(1):23–28, 1965.

32 Jesper Nederlof. Inclusion exclusion for hard problems. Master's thesis, Department of Information and Computer Science, Utrecht University, 2008. Available at http://www.win.tue.nl/~jnederlo/MScThesis.pdf.

33 K. S. Sudeep and Sundar Vishwanathan. A technique for multicoloring triangle-free hexagonal graphs. *Discrete Math.*, 300(1-3):256–259, 2005.

34 Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Appl. Math.*, 8(1):85–89, 1984. doi:10.1016/0166-218X(84)90081-7.

35 Magnus Wahlström. New plain-exponential time classes for graph homomorphism. *Theory Comput. Syst.*, 49(2):273–282, 2011. doi:10.1007/s00224-010-9261-z.