Tight Lower Bounds for the Complexity of Multicoloring

MARTHE BONAMY, CNRS, LaBRI, France
ŁUKASZ KOWALIK, Michał PILIPCZUK, ARKADIUSZ SOCAŁA, and
MARcin WROCHNA, Institute of Informatics, University of Warsaw, Poland

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 runtime \(f(b) \cdot 2^{o(\log b) \cdot n}\) for any computable \(f(b)\) unless the Exponential Time Hypothesis (ETH) fails. A \((b + 1)^n \cdot \text{poly}(n)\)-time algorithm due to Nederlof [33] shows that this is tight. A direct corollary of our result is that the graph homomorphism problem does not admit a \(2^{O(n+h)}\) algorithm unless the 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. [9].

The crucial ingredient in our hardness reduction is the usage of detecting matrices of Lindström [28], 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 runtime of the algorithms of Abasi et al. [1] and of Gabizon et al. [14] for the \(r\)-monomial detection problem are optimal under the ETH.

CCS Concepts: • Mathematics of computing → Graph coloring; Graph algorithms; • Theory of computation → Parameterized complexity and exact algorithms; Problems, reductions and completeness;

Additional Key Words and Phrases: Kneser graph, homomorphism, Exponential Time Hypothesis, detecting matrix

ACM Reference format:
Marthe Bonamy, Łukasz Kowalik, Michał Pilipczuk, Arkadiusz Socała, and Marcin Wrochna. 2019. Tight Lower Bounds for the Complexity of Multicoloring. ACM Trans. Comput. Theory 11, 3, Article 13 (March 2019), 19 pages.
https://doi.org/10.1145/3313906

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 L. 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) FNP via the START stipend programme.

A preliminary version of this paper appeared in the proceedings of ESA 2017.

Authors’ addresses: M. Bonamy, CNRS, LaBRI, Bordeaux, France; Ł. Kowalik, M. Pilipczuk, A. Socała, and M. Wrochna, Institute of Informatics, University of Warsaw, Warsaw, Poland; emails: marthe.bonamy@labri.fr, {kowalik, michal.pilipczuk, a.socala, m.wrochna}@mimuw.edu.pl.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
1942-3454/2019/03-ART13 $15.00
https://doi.org/10.1145/3313906

ACM Transactions on Computation Theory, Vol. 11, No. 3, Article 13. Publication date: March 2019.
1 INTRODUCTION

The complexity of determining the chromatic number of a graph is undoubtedly among the most intensively studied computational problems. Countless variants, extensions, 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. Note that, 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 \(\chi_b(G)\).

A basic motivation behind \((a:b)\)-colorings can be perhaps best explained by showing the connection with the fractional chromatic number. The fractional chromatic number of a graph \(G\), denoted \(\chi_f(G)\), is 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 \(I(G)\) below denotes the family of all independent sets of \(G\):

\[
\begin{align*}
\text{minimize} & \quad \sum_{I \in I(G)} x_I \\
\text{subject to} & \quad \sum_{I \in I(G)} x_I \geq 1, \text{ for each } v \in V(G).
\end{align*}
\]

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 the fractional chromatic number better and better. Consequently, \(\lim_{b \to \infty} \chi_b(G) / b = \chi_f(G)\).

Another interesting 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 (Figure 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 Figure 1). Kneser graphs are well studied in the context of graph colorings mostly owing to the celebrated result of Lovász [29], who determined their chromatic number, initiating the field of topological combinatorics.

Multicolorings and \((a:b)\)-colorings have been studied both from combinatorial [7, 12, 27] and algorithmic [5, 19, 20, 25, 26, 30, 31, 34] 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 [18] for a broader discussion.

1.1 Previous Work

In this article, 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 runtime \(^1\) \(O^*(2^a \cdot (b + 1)^n)\) as follows. For each function \(\eta : V(G) \to \{0, 1, \ldots, b\}\) 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 \(v \in V(G)\)

\(^1\)The \(O^*(\cdot)\) notation hides factors polynomial in the input size.
is covered exactly $\eta(v)$ times. Then, the value $D[\eta, k]$ can be computed as a disjunction of values $D[\eta', k - 1]$ over $\eta'$ obtained from $\eta$ 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 [33, 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 $O^*((b + 1)^n)$-time exponential-space algorithm. Hua et al. [22] noted that the formulation of Nederlof [33] for Multi Set Cover can also be 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 [32], we obtain an algorithm for $(a:b)$-COLORING that runs in time $O^*\left(\frac{3n}{2}\right)$. It uses polynomial space. Note that by plugging $b = 1$ to the results above, we obtain algorithms for the standard coloring problem with runtime $O^*(2^n)$ and exponential space usage, and with runtime $O^*(2.8845^n)$ and polynomial space usage, which almost matches the fastest known procedures [2].

The complexity of $(a:b)$-COLORING becomes particularly interesting in the 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 [21], the problem is in P if $H$ is bipartite and NP-complete otherwise. For quite a while, it was an open question as to whether there is an algorithm for GRAPH HOMOMORPHISM running in time $2^{O(n \log h)}$. It was recently answered in the negative by Cygan et al. [9]; more precisely, they proved that an algorithm with runtime $2^{O(n \log h)}$ would contradict the Exponential Time Hypothesis (ETH) of Impagliazzo and Paturi [23]. 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 et al. [13] showed that when $H$ is of treewidth at most $t$, then GRAPH HOMOMORPHISM can be solved in time $O^*((t + 3)^n)$. It was later extended to graphs of cliquewidth bounded by $t$, with an $O^*((2t + 1)^{\max\{n, h\}})$ time bound by Wahlström [36]. On the other hand, $H$ need not be sparse to admit efficient homomorphism testing: the family of cliques admits the $O^*(2^n)$ runtime as shown by...
Björklund et al. [2]. As noted above, this generalizes to Kneser graphs $KG_{a,b}$ by the $O^*((b+1)^n)$-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 whether there is an algorithm running in time $O^*(c^n)$ for some universal constant $c$ independent of $b$.

1.2 Our Contribution

We show that the algorithms for $(a:b)$-coloring mentioned above are essentially optimal under the ETH. Specifically, we prove the following results:

**Theorem 1.1.** If there is an algorithm for $(a:b)$-coloring that runs in time $f(b)\cdot 2^{o(\log b)}n$ for some computable function $f(b)$, then the ETH fails. This holds even if the algorithm is required to work only in instances in which $a = \Theta(b^2 \log b)$.

**Corollary 1.2.** If there is an algorithm for Graph Homomorphism that runs in time $f(h)\cdot 2^{o(\log \log h)}n$, for some computable function $f(h)$, then the ETH fails. This holds even if the algorithm is required to work only in 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 $O^*(2^n \cdot (b+1)^n) = 2^{O(\log b) n}$ 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 $2^{O(n) \text{ time}}$ [17], should be just as easy; Theorem 1 refutes this suspicion.

Corollary 1.2, in particular, excludes any algorithm for testing homomorphisms into Kneser graphs with runtime $2^{O(n+h)}$. 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 very 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 1.3.** If there is an algorithm for List $(a:b)$-coloring that runs in time $2^{o(\log b)}n$, then the ETH fails. This holds even if the algorithm is required to work only in instances in which $a = \Theta(b^2 \log b)$ and $b = \Theta(b(n))$ for an arbitrarily chosen polynomial-time computable function $b(n)$ such that $b(n) = \omega(1)$ and $b(n) = O(n/\log n)$.

The crucial ingredient in the proof of Theorem 1.3 is the use of $d$-detecting matrices introduced by Lindström [28]. We choose to work with their combinatorial formulation; hence, we shall talk about $d$-detecting families. Suppose that we are given some universe $U$ and there is an unknown function $f : U \to \{0, 1, \ldots, d-1\}$, 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 of 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 $O(|U|/\log |U|)$ for every fixed $d$ [28]. 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 1.3. Given an instance \( \varphi \) of 3-SAT with \( n \) variables and \( O(n) \) clauses, and a number \( b \leq n/\log n \), we will construct an instance \( G \) of \( \text{LIST} \ (a:b)\)-\text{COLORING} for some \( a \). This instance will have a positive answer if and only if \( \varphi \) is satisfiable, and the constructed graph \( G \) will have \( O(n/\log b) \) vertices. It can be easily seen that this will yield the promised lower bound.

Partition the clause set \( C \) of \( \varphi \) into groups \( C_1, C_2, \ldots, C_p \), each of size roughly \( b \); thus, \( p = O(n/b) \). Similarly, partition the variable set \( V \) of \( \varphi \) into groups \( V_1, V_2, \ldots, V_q \), each of size roughly \( \log_2 b \); thus, \( q = O(n/\log b) \). In the output instance, we create one vertex per each variable group—hence, we have \( O(n/\log b) \) 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. The plan for the rest of the construction is to verify whether these assignments make at least one literal per clause satisfied. In order to avoid some technicalities here, assume that we want to test whether exactly one literal per clause is satisfied.

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 take only 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 \( C_j \) has exactly one satisfied literal by creating a block of \(|C_j|\) vertices, each checking one clause. However, the whole graph output by the reduction would then have \( O(n) \) vertices, and we would not obtain any nontrivial lower bound. Instead, we create a \( 4 \)-detecting family on the universe \( U = C_j \). For every question \( Q \subseteq C_j \) in the family, we create one vertex \( u_Q \) and by setting its list appropriately we test whether the answer is \(|Q| \), i.e., whether the clauses in \( Q \) have exactly \(|Q|\) satisfied literals. The constant function \( f : C_j \to \{0, 1, 2, 3\} \), \( f \equiv 1 \) passes all of the tests and, by the definition of the detecting family, it is unique. This means that our tests, in fact, verify whether whether every clause has exactly one satisfied literal. Since the detecting family has size \( O(|C_j|/\log |C_j|) = O(|C_j|/\log b) \), the total number of vertices in the constructed graph will be \( O(n/\log b) \), as intended.

### 1.3 Low-Degree Monomial Testing

We observe that, from our main result, one can infer a lower bound for the complexity of the \((r,k)\)-\text{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 of its monomials have the same total degree \( k \). The task is to verify whether \( P \) has some monomial in which every variable has an individual degree not larger than \( r \), for a given parameter \( r \). Abasi et al. [1] provided a randomized algorithm solving this problem in time \( \mathcal{O}^*(2^{\mathcal{O}(k \cdot \log r)}) \), where \( k \) is the degree of the polynomial, assuming that \( \mathbb{F} = \text{GF}(p) \) for a prime \( p \leq 2r^2 + 2r \). This algorithm was later derandomized by Gabizon et al. [14] within the same runtime, but under the assumption that the circuit is noncancelling: 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, \( r \)-\text{SIMPLE} \( k \)-\text{PATH}, the problem of finding a walk of length \( k \) that visits every vertex at most \( r \) times, can be solved in time \( \mathcal{O}^*(2^{\mathcal{O}(k \cdot \log r)}) \). However, for \( r \)-\text{SIMPLE} \( k \)-\text{PATH}, as well as other problems that can be tackled using this technique, the best-known lower bounds under the ETH exclude only algorithms with...
runtime $O^*(2^{o(k)})$. 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 $\mathbb{GF}(2)$ in such a way that an $O^*(2^{k \cdot o(\log r)})$-time algorithm for the latter would imply a $2^{o(\log k) \cdot n}$-time algorithm for the former, which would contradict the 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 $k$, as stated below.

**Theorem 1.4.** Let $\sigma \in [0,1)$. Then, unless the ETH fails, there is no algorithm for $(r,k)$-Monomial Testing that solves instances with $r = \Theta(k^\sigma)$ in time $2^{o(k \cdot \frac{\log r}{r})} \cdot |C|^{O(1)}$.

1.4 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 1.3. In Section 4, we give a reduction from the list version to the standard version, thereby proving Theorem 1.1. Section 5 is devoted to deriving lower bounds for low-degree monomial testing.

2 PRELIMINARIES

2.1 Notation
We use standard graph notation; see, e.g., [10, 11]. All graphs that we consider in this article are simple and undirected. For an integer $k$, we denote $[k] = \{0, \ldots, k-1\}$. For a set $A$ and an integer $k$ by $\binom{A}{k}$, we denote the family of all $k$-element subsets of $A$; note that $|\binom{A}{k}| = \binom{|A|}{k}$. 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 of respective problems or both are NO instances.

2.2 Exponential Time Hypothesis
The ETH of Impagliazzo and Paturi [23] states that there exists a constant $c > 0$ such that there is no algorithm solving 3-SAT in time $O^*(2^{cn})$. Recently, ETH has become 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 [24], 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 2.1** (see, e.g., Theorem 14.4 in [10]). Unless the 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 [35]. 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 2.2 ([35]). Given a 3-SAT formula \( \varphi \) with \( n \) variables and \( m \) clauses, one can transform it in polynomial time into an equivalent \((3,4)\)-SAT instance \( \varphi' \) with \( O(n + m) \) variables and clauses.

Corollary 2.3. Unless the 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.

2.3 List and Nonuniform List \((a:b)\)-Coloring

For integers \( a, b \) and a graph \( G \) with a function \( L : V(G) \to 2^{|a|} \) (assigning a list of colors to every vertex), an \((a:b)\)-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 \((a:b)\)-COLORING problem is defined as follows.

| List \((a:b)\)-coloring |
|------------------------|
| **Input:** Graph \( G \), function \( L : V(G) \to 2^{|a|} \) |
| **Question:** Is there an \((a:b)\)-coloring of \( G \)? |

As an intermediary step of our reduction, we will use the following generalization of list colorings in which the number of demanded colors varies with every vertex. For integers \( a, b \), a graph \( G \) with a function \( L : V(G) \to 2^{|a|} \) and a demand function \( \beta : V(G) \to \{1, \ldots, b\} \), an \((a: \beta)\)-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 \((a:b)\)-COLORING is then defined as follows.

| Nonuniform List \((a:b)\)-coloring |
|-------------------------------|
| **Input:** Graph \( G \), a list function \( L : V(G) \to 2^{|a|} \) and a demand function \( \beta : V(G) \to \{1, \ldots, b\} \) |
| **Question:** Is there an \((a: \beta)\)-coloring of \( G \)? |

2.4 \(d\)-Detecting Families

In our reductions, the following notion plays a crucial role.

Definition 2.4. 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 [28] together with a proof that even the constant factor \( 2 \) in the family size cannot be improved.

Theorem 2.5 ([28]). 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 time polynomial in \( |U| \).

Other constructions, generalizations, and discussion of similar results can be found in Grebinski and Kucherov [16] and in Bshouty [3]. Note that the expression \( \sum_{x \in S} f(x) \) is just the product of \( f \) as a vector in \([d]^{\frac{|U|}{|U|}}\) 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 \([d]^{\frac{|U|}{|U|}}\) despite having few rows). Similar definitions appear in the study of query complexity, e.g., as in the popular Mastermind game [6].

While known polynomial deterministic constructions of detecting families involve some number theory or Fourier analysis, their existence can be argued with an elementary probabilistic argument. Intuitively, a random subset \( S \subseteq U \) will distinguish two distinct functions \( f, g : U \to \{0, \ldots, d - 1\} \) (meaning \( \sum_{x \in S} f(x) \neq \sum_{x \in S} g(x) \)) with probability at least \( \frac{1}{2} \). This is because any \( x \) where \( f \) and \( g \) disagree is taken or not taken into \( S \) with probability \( \frac{1}{2} \), while sums over \( S \) cannot
agree in both cases simultaneously, as they differ by \( f(x) \) and \( g(x) \), respectively. There are \( d^n \cdot d^n \) function pairs to be distinguished. In any subset of pairs, at least half of the pairs are distinguished by a random set in expectation; thus, at least one such set exists. Naively, by repeatedly finding such a set for undistinguished pairs, we get that \( |\log_2(d^n \cdot d^n)| = O(n \log d) \) sets that distinguish all functions.

This naïve argument can be improved by observing that distinguishing \( f \) and \( g \) is equivalent to distinguishing \( f - h \) and \( g - h \), where \( h(x) = \min(f(x), g(x)) \). Hence, it suffices to distinguish functions with disjoint supports \( \text{supp}(f) := \{x \mid f(x) \neq 0\} \). Let us assume that \( d = 2 \) for simplicity and add the whole of \( U \) as a single set so that we can assume that \( |\text{supp}(f)| = \sum_{x \in U} f(x) = \sum_{x \in U} g(x) = |\text{supp}(g)| \). Denote \( s := |\text{supp}(f)| = |\text{supp}(g)| \). If we focus only on pairs of functions that are close, i.e., \( s \leq \sqrt{n} \), then the naïve method gives a family \( \mathcal{F}_1 \) of \( |\log_2 \left( \binom{n}{s} \cdot \binom{n}{s} \right)| = O \left( \frac{n}{\log n} \right) \) sets that distinguishes all of them. Now, we can assume that \( s > \sqrt{n} \), i.e., we focus on distant pairs. Intuitively, such pairs should be much easier to distinguish using a random set. When \( d = 2 \), we have that \( \sum_{x \in S} f(x) = \sum_{x \in S} g(x) \) if and only if \( S \) contains the same number of elements from \( \text{supp}(f) \) and \( \text{supp}(g) \). It follows that the probability \( p \) of failing to distinguish \( f \) and \( g \) by a random set \( S \subseteq U \) can be bounded as follows.

\[
p = \Pr \left[ \sum_{x \in S} f(x) = \sum_{x \in S} g(x) \right] = \sum_{i=0}^{s} \left( \binom{s}{i}/2^s \right) \cdot \left( \binom{s}{i}/2^s \right) \leq \max_{i=0}^{s} \binom{s}{i}/2^s = \binom{s}{\lfloor s/2 \rfloor}/2^s = O \left( \frac{1}{\sqrt{s}} \right).
\]

Hence, by using \( p \) in the naïve argument, there is a family \( \mathcal{F}_2 \) of size \( \log_{\log \sqrt{n}}(2^n \cdot 2^n) = \log_{\log \sqrt{n}} \log_{\log \sqrt{n}} 2^{2n} = \tilde{O} \left( \frac{n}{\log n} \right) \) that distinguishes all distant pairs. Clearly, \( \mathcal{F}_1 \cup \mathcal{F}_2 \) is a \( 2 \)-detecting family of size \( \tilde{O} \left( \frac{n}{\log n} \right) \).

The same method allows one to show that, in fact, \( \tilde{O} \left( \frac{n}{\log^2 n} \right) \) random sets are enough to form a \( d \)-detecting family with positive probability; see Section 2.1 in [16] for full details.

3 HARDNESS OF LIST \((a:b)\)-COLORING

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

3.1 The Nonuniform Case

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

**Theorem 3.1.** For any instance \( \phi \) of \((3,4)\)-\textsc{SAT} with \( n \) variables and any integer \( 2 \leq b \leq n/\log_2 n \), there is an equivalent instance \((G, \beta, L)\) of \textsc{Nonuniform \((a:b)\)-coloring} such that \( a = \tilde{O}(b^2 \log b), |V(G)| = \tilde{O}(\frac{n}{\log n}), \) and \( G \) is 3-colorable. Moreover, the instance \((G, \beta, L)\) and the 3-coloring of \( G \) can be constructed in \( \text{poly}(n) \) time.

Consider an instance \( \phi \) of \(3\)-\textsc{SAT} in which 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 \( \frac{1}{3} |V| \leq |C| \leq \frac{1}{2} |V| \). Let \( a = 12b^2 \cdot \lfloor \log_2 b \rfloor \). We shall construct, for some integers \( n_V = \tilde{O}(|V|/\log b) \) and \( n_C = \tilde{O}(|C|/b) \):

- 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 \( b \), and
- a function \( \sigma : \{1, \ldots, n_V\} \to [12 \cdot b \cdot \lfloor \log_2 b \rfloor] \).
such that all literals occurring in \( C_j \) are distinct in the following strong sense:

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 \( \sigma \).  

(\*)

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 3.2.** Partitions \( V = V_1 \cup \ldots \cup V_{n_V}, C = C_1 \cup \ldots \cup C_{n_C} \) and a function \( \sigma \) satisfying (\*) can be found in time \( \text{poly}(n) \).

**Proof.** We first group variables in a way such that the following holds: (P1) the variables occurring in any clause are different and belong to different variable groups. To this end, consider the graph \( G_1 \) with variables as vertices and edges between any two variables that occur in a common clause (i.e., the primal graph of \( \phi \)). Since no clause contains repeated variables, \( G_1 \) has no loops. Since every variable of \( \phi \) occurs in at most four clauses and since those clauses contain at most two variables, the maximum degrees of \( G_1 \) is at most 8. Hence, \( G_1 \) can be greedily colored with 9 colors. Then, we refine the partition given by colors to make every group have size at most \( \lceil \log_2 b \rceil \), producing in total at most \( n_V := \lceil |V|/\lceil \log_2 b \rceil \rceil + 9 \) groups \( V_1, \ldots, V_{n_V} \). (P1) holds because any two variables occurring in a common clause are adjacent in \( G_1 \) and, thus, get different colors. Thus, they are assigned to different groups.

Next, we group clauses in a way such that (P2) the variables occurring in clauses of a group \( C_j \) are all different and belong to different variable groups. For this, consider the graph \( G_2 \) with clauses as vertices and with an edge between clauses if they contain two different variables from the same variable group. By (P1), \( G_2 \) has no loops. Since every clause contains exactly 3 variables, each variable is in a group with at most \( \lceil \log_2 b \rceil - 1 \) others, and every such variable occurs in at most 4 clauses, the maximum degree of \( G_2 \) is at most 12(\( \lceil \log_2 b \rceil - 1 \)). We can therefore color \( G_2 \) greedily with \( 12 \lceil \log_2 b \rceil \) colors. Similarly as before, we partition clauses into \( n_C := \lceil |C|/b \rceil + 12 \lceil \log_2 b \rceil \) monochromatic groups \( C_1, \ldots, C_{n_C} \) of size at most \( b \) each. Then, (P2) holds by construction of the coloring.

Finally, consider a graph \( G_3 \) with variable groups as vertices and with an edge between two variable groups if they contain two different variables occurring in clauses from a common clause group. More precisely, \( V_i \) and \( V_{i'} \) are adjacent if there are two different variables \( x \in V_i \) and \( x' \in V_{i'} \), and a clause group \( C_j \) with clauses \( c \) and \( c' \) (possibly \( c = c' \)), such that \( x \) occurs in \( c \) and \( x' \) occurs in \( c' \). By (P2), \( G_3 \) has no loops. Since each clause has at most \( \lceil \log_2 b \rceil - 1 \) other variables in its group, each of these variables occur in at most 4 clauses, each of these clauses has at most \( b - 1 \) other clauses in its group, and each of these contains exactly 3 variables, the maximum degree of \( G_3 \) is at most \( 4 \cdot (\lceil \log_2 b \rceil - 1) \cdot (b - 1) \cdot 3 \). We can therefore color it greedily into \( 12b \lceil \log_2 b \rceil \) colors. Let \( \sigma \) be the resulting coloring. By (P2) and the construction of this coloring, (\*) holds.

The colorings can be found in linear time in the sizes of the relevant graphs using standard techniques. Since \( G_1 \) has size \( O(n) \), \( G_2 \) has size \( O(n \log b) \), \( G_3 \) has size \( O(nb \log b) \), and \( b \leq n \log b n \), the total time is polynomial in \( n \). Note that we have that \( n_V = \lceil |V|/\lceil \log_2 b \rceil \rceil + 9 = O(|V|/\log b) \). Moreover, since \( b \leq n/\log_2 n \), we get that \( \log_2 b \leq \log_2 n \leq \frac{n}{b} = \Theta(|C|/b) \); hence, \( n_C = \lceil |C|/b \rceil + 12 \lceil \log_2 b \rceil = O(|C|/b) \).

For every \( 1 \leq i \leq n_V \), the set \( V_i \) of variables admits \( 2^{|V_i|} \leq b \) different assignments. Therefore, we will 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 $(\star)$ means that $|I_j| = 3|C_j|$ and that $\sigma$ is injective over each $I_j$ (see Figure 2).

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

For every $1 \leq k \leq n_F$, let $C_{j,n_F+k} = C_j \setminus C_{j,k}$.

We are now ready to build the graph $G_r$, 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 $1 \leq k \leq 2n_F$, 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
   \[
   L(u_{j,k}) = \{b \cdot \sigma(i) + x \mid i \in I_j, x \in [2^{|V_i|}] \text{ such that } x \text{ gives an assignment of } V_i \text{ that satisfies some clause of } C_{j,k}\}.
   \]
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 $u_{j,k}$ $(1 \leq k \leq 2n_F)$. 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 $|C_j|$ 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 3.3.** If $\phi$ is satisfiable, then $G$ is $L$-$(a; \beta)$-colorable.

**Proof.** Consider a satisfying assignment $\eta$ for $\phi$. 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 $\eta$. For every clause $c$ of $\phi$, choose one literal satisfied by $\eta$ in
it, and let $i_c$ be the index of the group $V_{i_c}$ containing the literal’s variable. Let $\alpha : V(\mathcal{G}) \rightarrow \{\frac{[a]}{2^{b}\leq}\}$ be the $L$-$(\alpha;\beta)$-coloring of $\mathcal{G}$ defined as follows, for $1 \leq i \leq n_V, 1 \leq j \leq n_C, 1 \leq k \leq 2n_F$:

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

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 $\alpha$ 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 ($\otimes$), $\sigma$ is injective on $I_j$ and, thus, on $\{i_c \mid c \in C_{j,k}\} \subseteq I_j$, we have that $|\alpha(u_{j,k})| = |C_{j,k}| = b(u_{j,k})$. Similarly, since $\sigma$ is injective on $I_j$ and $|I_j \setminus \{i_c \mid c \in C_j\}| = 3|C_j| - |C| = 2|C_j|$, we get that $|\alpha(w_j)| = 2|C_j| = \beta(w_j)$.

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

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

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 $v_iu_{j,k}$ and $v_iw_j$ received disjoint sets of colors on their endpoints, concluding the proof.

**Lemma 3.4.** If $\mathcal{G}$ is $L$-$(\alpha;\beta)$-colorable, then $\phi$ is satisfiable.

**Proof.** Assume that $\mathcal{G}$ is $L$-$(\alpha;\beta)$-colorable and let $\alpha$ be the corresponding coloring.

For $1 \leq i \leq n_V$, we have that $|L(v_i)| = b$ and $|\alpha(v_i)| = b - 1$; thus, $v_i$ 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 $x \phi$, for $\phi$ chosen by $x_i$ on each $V_i$, satisfies $\phi$.

Consider any clause group $C_j$ for $1 \leq j \leq n_C$. Every vertex in $\{w_j\} \cup \{u_{j,k} \mid 1 \leq k \leq 2n_F\}$ 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 $L(u_{j,k}), L(w_j) \subseteq \{b \cdot \sigma(i) + x_i' \mid i \in I_j, x_i' \in \{b\}\}$, 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 ($\otimes$), this set has exactly $|I_j| = 3|C_j|$ different colors. Of these, exactly $2|C_j|$ are contained in $\alpha(w_j)$. Let $I_j$ be the subset of $I_j$ such that the remaining $|C_j|$ colors form the set $\{b \cdot \sigma(i) + x_i \mid i \in I_j\}$.

Since $\alpha(u_{j,k})$ is disjoint from $\alpha(w_j)$, we have that $\alpha(u_{j,k}) \subseteq \{b \cdot \sigma(i) + x_i \mid i \in I_j\}$ for all $k$. By definition of $I_j$, for every $i \in I_j \subseteq I_j$ there is a variable in $V_i$ that appears in some clause of $C_j$. By property ($\otimes$), it can only occur in one such clause; thus, 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 that $x_i$ 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 $\{l_i \mid i \in I_j\}$ occur in clauses of $C_{j,k}$ and are made true by the assignment $x$.

Let $f : C_j \rightarrow \{0, 1, 2, 3\}$ be the function assigning to each clause $c \in C_j$ the number of literals of $c$ in $\{l_i \mid i \in I_j\}$. By the above, $\sum_{c \in C_{j,k}} f(c) \geq |C_{j,k}|$ for $1 \leq k \leq 2n_F$. Since each literal in $\{l_i \mid i \in I_j\}$
belongs to some clause of \( C_j \), we have that \( \sum_{c \in C_j} f(c) = |J_j| = |C_j| \). Then,
\[
\sum_{c \in C_j, k} f(c) = \sum_{c \in C_j} f(c) - \sum_{c \in C_{j, n_F + k}} f(c) \leq |C_j| - |C_{j, n_F + k}| = |C_{j, k}|.
\]
Hence, \( \sum_{c \in C_{j, k}} f(c) = |C_{j, k}| \) for \( 1 \leq k \leq 2n_F \). Let \( g : C_j \rightarrow \{0, 1, 2, 3\} \) 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 \( \{C_{j, 1}, \ldots, C_{j, n_F}\} \) is a 4-detecting family, this implies that \( f = g \). Thus, for every clause \( c \) of \( C_j \), we have that \( f(c) = 1 \), meaning that there is a literal from the set \( \{l_i \mid i \in J_j\} \) in this clause. All of these literals are made positive by the assignment \( \eta \); therefore, all clauses of \( C_j \) are satisfied. Since \( j = 1, \ldots, n_C \) was arbitrary, this concludes the proof that \( \eta \) is a satisfying assignment for \( \phi \). \( \square \)

The construction can clearly be made in polynomial time and the total number of vertices is
\[
n_V + n_C \cdot O\left(\frac{b}{\log b}\right) + n_C = O\left(\frac{n}{\log b}\right).
\]
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 3.3 and 3.4, this concludes the proof of Theorem 3.1.

### 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 3.5.** For any instance \( I = (G, \beta, L) \) of NONUNIFORM LIST \((a:b)\)-coloring in which 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 \( c : V(G) \rightarrow [t] \) be a \( t \)-coloring of \( G \). For every vertex \( v \), define a set of filling colors \( F(v) = \{a + c(v)b + i \mid i = 0, \ldots, b - |\beta(v)| - 1\} \) and put
\[
L'(v) = L(v) \cup F(v).
\]
Let \( \alpha : V(G) \rightarrow 2^{[a]} \) be an \( L-(a:b) \)-coloring of \( G \). We define a coloring \( \alpha' : V(G) \rightarrow 2^{[a + tb]} \) 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 \( \alpha \) was a proper \( L-(a:b) \)-coloring, adjacent vertices can share only 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 \( \alpha' \) is an \( L'-(a:b) \)-coloring of \( G \).

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

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

**Theorem 1.3.** If there is an algorithm for LIST \((a:b)\)-coloring that runs in time \( 2^{o(\log b) - n} \), then the ETH fails. This holds even if the algorithm is required to work only in instances in which \( a = \Theta(b^2 \log b) \) and \( b = \Theta(b(n)) \) for an arbitrarily chosen polynomial-time computable function \( b(n) \) such that \( b(n) = \omega(1) \) and \( b(n) = O(n/\log n) \).

**Proof.** Let \( b(n) \) be a function as in the statement. We can assume without loss of generality that \( 2 \leq b(n) \leq n/\log_2 n \); otherwise, we can replace \( b(n) \) with a function \( b'(n) = 2 + \lfloor b(n)/c \rfloor \) in the reasoning below, where \( c \) is a big enough constant. Note that \( b'(n) = \Theta(b(n)) \). Fix a function \( g(b) = o(\log b) \) and assume that there is an algorithm \( \mathcal{A} \) for LIST \((a:b)\)-coloring that runs in time \( 2^{g(b) - n} \) whenever \( b = \Theta(b(n)) \). Consider an instance of \((3,4)\)-SAT with \( n \) variables. Let
4 FROM LIST \((a:b)\)-COLORING TO LIST \((a:b)\)-COLORING

In this section, we reduce LIST \((a:b)\)-COLORING to \((a:b)\)-COLORING. This is done by adding a Kneser graph and replacing the lists by edges to appropriate vertices of the Kneser graph. We will need the following well-known property of Kneser graphs (see, e.g., Theorem 7.9.1 in the textbook [15]).

**Theorem 4.1.** If \(p > 2q\), then every homomorphism from \(KG_{p,q}\) to \(KG_{p,q}\) is bijective.

We proceed with the reduction.

**Lemma 4.2.** For any given instance of LIST \((a:b)\)-COLORING with \(n\) vertices, there exists an equivalent instance of \(((a + b):b)\)-COLORING with \(n + (a+b)\) vertices. Moreover, it can be computed in \(\text{poly}(n, (a+b))\)-time.

**Proof.** Let \((G, L)\) be an instance of LIST \((a:b)\)-COLORING where \(G\) is a graph and \(L: V(G) \rightarrow 2^\{a\}\) describes the lists of allowed colors. Define a graph \(K\) with \(V(K) = \{ [a+b] \}_{b}^{a}\) and \(E(K) = \{ XY : X, Y \in V(K) \text{ and } X \cap Y = \emptyset \}\).

That is, \(K\) is isomorphic to the Kneser graph \(KG_{a+b,b}\). Then, let \(V' = V(G) \cup V(K)\) and \(E' = E(G) \cup E(K) \cup \{ vX : v \in V(G) \text{ and } X \in V(K) \text{ and } L(v) \cap X = \emptyset \}\).

The graph \(G' = (V', E')\) has \(n + (a+b)\) vertices, and the construction can be done in polynomial time in \(n + (a+b)\). Let \(G'\) be our output instance of \(((a + b):b)\)-COLORING. We will show that it is equivalent to the instance \((G, L)\) of LIST \((a:b)\)-COLORING.

Let us assume that \(\alpha : V(G) \rightarrow \{ [a] \}_{b}^{a}\) is an \(L\)-(a:b)-coloring of \(G\). Consider \(\alpha' : V(G') \rightarrow \{ [a+b] \}_{b}^{a}\) such that

\[
\alpha'(v) = \begin{cases} 
\alpha(v) & \text{for } v \in V(G) \\
\nu & \text{for } v \in V(K) = \{ [a+b] \}_{b}^{a}\end{cases}
\]

We claim that \(\alpha'\) is an \(((a + b):b)\)-coloring of \(G'\). Indeed, for every edge \(uv \in E(G)\), we have that \(\alpha'(u) \cap \alpha'(v) = \alpha(u) \cap \alpha(v) = \emptyset\) because \(\alpha\) is an \(L\)-(a:b)-coloring of \(G\). For every edge \(XY \in E(K)\), we have that \(\alpha'(X) \cap \alpha'(Y) = X \cap Y = \emptyset\). For every edge \(vX\) such that \(v \in V(G)\) and \(X \in V(K)\), we have that \(\alpha'(v) \cap \alpha'(X) = \alpha(v) \cap X \subseteq L(v) \cap X = \emptyset\).

Now, let us assume that \(\alpha' : V(G') \rightarrow \{ [a+b] \}_{b}^{a}\) is an \(((a + b):b)\)-coloring of \(G'\). Recall that \(\alpha'\) is a homomorphism of \(G'\) to \(KG_{a+b,b}\). Denote \(\phi = \alpha'|_{V(G)}\). By Theorem 4.1, \(\phi\) is bijective. Define \(\alpha'' = \phi^{-1} \circ \alpha'\). Then, \(\alpha''\) is an \(((a + b):b)\)-coloring of \(G'\) with the property that \(\alpha''(X) = X\) for every \(X \in V(K)\). We claim that \(\alpha''|_{V(G)}\) is an \(L\)-(a:b)-coloring of \(G\). Since \(\alpha''\) is a \(((a + b):b)\)-coloring of \(G'\), it suffices to show that \(\alpha''(v) \subseteq L(v)\) for every vertex \(v \in V(G)\). Pick a color \(\gamma \notin L(v)\). Let \(X_{\gamma}\) be the \(b\)-element set consisting of \(\gamma\) and arbitrary \(b - 1\) elements from \([a+b]\) \(\setminus ([a] \cup \{\gamma\})\). Then, \(L(v) \cap X_{\gamma} = \emptyset\); hence, \(\nu X_{\gamma} \in E(G')\). It follows that \(X_{\gamma} \cap \alpha''(v) = \alpha''(X_{\gamma}) \cap \alpha''(v) = \emptyset\) and, in particular, \(\gamma \notin \alpha''(v)\). Thus, \(\alpha''(v) \subseteq L(v)\) as required. \(\square\)

We now prove our main result.
Theorem 1.1. 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 the ETH fails. This holds even if the algorithm is required to work only in instances in which \(a = \Theta(b^2 \log b)\).

Proof. Fix a computable function \(f(b)\), a function \(g(b) = o(\log b)\), and assume that there is an algorithm \(A\) for \((a:b)\)-coloring that runs in time \(f(b) \cdot 2^{g(b)} \cdot n\) for a given \(n\)-vertex graph whenever \(a = \Theta(b^2 \log b)\). Without loss of generality, we can replace \(f(b)\) by any nondecreasing function \(f'(n)\) such that \(f'(n) \geq f(n)\) and \(f'(n) > n\). Intuitively, we now define an unbounded function \(b(N)\), which should be at least 2, at most the inverse of \(f\), and small enough so that \(2^{O(b \log b)} \leq \frac{N}{\log b}\). The following function is \(\omega(1)\); a standard argument shows how to compute it in \(\text{poly}(N)\) time (see Lemmas 3.2 and 3.4 in [4]).

\[
b(N) = \min \{ \max \{ b : f(b) \leq N \}, \max \{ b : b \log b \leq \log N / \log \log N \} \} + 2.
\]

Consider an instance of \((3,4)\)-SAT with \(N\) variables. Let \(b = b(N)\). By Theorem 3.1 in \(\text{poly}(N)\) time, we get an equivalent instance \((G, \beta, L)\) of \text{NONUNIFORM LIST \((a:b)\)-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 3.5 in \(\text{poly}(N)\) time, we get an equivalent instance \((G', \mathcal{L}')\) of \text{LIST \((a:b)\)-COLORING}. Then, by Lemma 4.2, in \(\text{time}\ \text{poly}(N)\) we get an equivalent instance \(G'\) of \((a + 8b):(2b)\)-COLORING such that \(|V(G')| = |V(G)| + (\frac{a + 8b}{2b})\). Observe that since \(a = \Theta(b^2 \log b)\) and \(b \log b \leq \log N / \log \log N\),

\[
(a + 8b) \leq 2b \log b = 2^{O(b \log b)} = 2^{O(\log N / \log \log N)} = N^\omega(1) = o(N / \log \log N).
\]

Hence, \(|V(G')| = \mathcal{O}(\frac{N}{\log b})\). Finally, we solve the instance \(G'\) using algorithm \(A\). Since \(b(N) = \omega(1)\), we have that \(g(b(N)) = o(\log(b(N)))\). Therefore, \(A\) runs in time

\[
f(b) \cdot 2^{o(\log b)} \cdot |V(G')| \leq N \cdot 2^{o(\log b)} \cdot 2^{O(\log N / \log \log N)} = 2^{O(N)}
\]

solving the instance \(\phi\) of \((3,4)\)-SAT in time \(2^{o(N)}\). By Corollary 2.3, this contradicts the ETH.

Corollary 4.3. If there is an algorithm for \text{GRAPH HOMOMORPHISM} that runs in time \(f(h) \cdot 2^{o(\log \log h)} \cdot n\), for some computable function \(f(h)\), then the ETH fails. This holds even if the algorithm is required to work only in instances where \(H\) is a Kneser graph \(KG_{a,b}\) with \(a = \Theta(b^2 \log b)\).

Proof. Fix a computable function \(f(h)\) and assume that there is an algorithm \(A\) for \text{GRAPH HOMOMORPHISM} that runs in time \(f(h) \cdot 2^{o(\log \log h)} \cdot n\) for a given \(n\)-vertex graph whenever \(H\) is a Kneser graph \(KG_{a,b}\) with \(a = \Theta(b^2 \log b)\). Consider an instance of \((a:b)\)-COLORING with \(n\) vertices and \(a = \Theta(b^2 \log b)\). This is an instance of \text{GRAPH HOMOMORPHISM} with \(h = (\frac{a}{b}) \leq a^b = 2^{O(b \log b)}\).

Hence, \(A\) solves it in

\[
f(h) \cdot 2^{o(\log h)} \cdot n = f(2^{O(b \log b)} \cdot 2^{o(\log (b \log b))} \cdot n) \leq f'(b) \cdot 2^{o(\log b)} \cdot n
\]

for some computable function \(f'(b) \geq f(2^{O(b \log b)})\), which contradicts Theorem 1.1.

5 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 \(\mathbb{F}\). Such a circuit may contain input, constant, addition, negation, multiplication, and inversion gates. One gate is designated to be the output gate, which 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 of 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 parameter \(r \leq k\). Abasi et al. [1] gave a very fast randomized algorithm for \((r,k)\)-MONOMIAL TESTING.
Theorem 5.1 (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 $g \in GF(p)[x_1, \ldots, x_n]$ be a homogenous polynomial of degree $k$, computable by a circuit $C$. Then, there is a randomized algorithm running in time $O(r^{2k/r}|C|rn^{O(1)})$ that

- 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 noncancelling, i.e., it contains only input, addition, and multiplication gates. Many concrete problems such as $r$-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 will see in a moment, this is also the case for LIST $(a:b)$-COLORING.

Let $(G = (V, E), L)$ be an instance of the LIST $(a:b)$-COLORING problem. We denote $n = |V|$. Let $C_a(G, L)$ denote the set of all functions $c : V \rightarrow 2^{|a|}$ such that, for every edge $uv \in E$, the sets $c(u)$ and $c(v)$ are disjoint and for every vertex $v$, we have that $c(v) \subseteq L(v)$. Consider the following polynomial in $n(a + 1)$ variables $(x_v)_{v \in V}$ and $(y_{v,j})_{v \in V, j \in [a]}$ over $GF(2)$:

$$p_G = \sum_{c \in C_a(G, L)} \prod_{v \in V} x_v^{c(v)} \prod_{j \in c(v)} y_{v,j}.$$  

(1)

Note that every summand in Equation (1) has a different set of variables; therefore, it corresponds to a monomial (with coefficient 1). Then, the following proposition is immediate.

Proposition 5.2. There is a list $(a:b)$-coloring of graph $G$ if and only if $p_G$ contains a $b$-monomial.

Now, we show that $p_G$ can be evaluated relatively fast.

Lemma 5.3. The polynomial $p_G$ can be evaluated using a circuit of size $2^n \text{poly}(a, n)$, which can be constructed in time $2^n \text{poly}(a, n)$.

Proof. Let $I$ be the family of all independent sets of $G$. Consider the following polynomial:

$$q_G = \prod_{j=1}^a \sum_{I \in I} \prod_{v \in I} x_v y_{v,j}.$$  

(2)

Observe that $p_G$ is obtained from $q_G$ by removing all monomials of degree different than $2bn$. Equation (2) shows that $q_G$ can be evaluated by a circuit $C_q$ of size $|I| \text{poly}(a, n) \leq 2^n \text{poly}(a, n)$, which can be constructed in time $2^n \text{poly}(a, n)$. We obtain from $C_q$ a circuit $C_p$ for $p_G$ by splitting gates according to degrees, in a bottom-up fashion, as follows.

Every input gate $u$ of $C_q$ is replaced with a gate $u_1$ in $C_p$. Every addition gate $u$ with inputs $x$ and $y$ in $C_q$ is replaced in $C_p$ by $2an$ addition gates $u_1, \ldots, u_{2an}$, where $u_1$ has inputs $x_i$ and $y_i$ (whenever $x_i$ and $y_i$ exist). Every multiplication gate $u$ with inputs $x$ and $y$ in $C_q$ is replaced in $C_p$ by $2an$ addition gates $u_1, \ldots, u_{2an}$. Moreover, for every pair of integers $1 \leq r, s \leq 2an$, we create a multiplication gate $ur,s$ with inputs $x_r$ and $y_s$ (whenever they exist) and make it an input of the addition gate $u_{r+s}$. It is easy to see that, for every gate $u$ of $C_q$, for every $i$, the gate $u_i$ of $C_p$ evaluates the same polynomial as $u$ but restricted to monomials in which the total degree is equal to $i$. When $o$ is the output gate of $C_q$, then $o_{2bn}$ is the output gate of $C_p$. Clearly, $|C_p| \leq (2an + 1)^2|C_q|$, and $C_p$ can be constructed from $C_q$ in time $2^n \text{poly}(a, n)$.

Since $p_G$ is a homogenous polynomial of degree $k = 2bn$, by putting $r = b$ we can combine Proposition 5.2, Theorem 5.1, and Lemma 5.3 to get yet another polynomial-space algorithm for
List \((a:b)\)-coloring, running in time \(b^{O(n)} \cdot \text{poly}(n)\). Similarly, if the runtime in Theorem 5.1 was improved from \(2^{o(\log r) \cdot k} \cdot |C| \cdot \text{poly}(r, n)\), then we would get an algorithm for List \((a:b)\)-coloring in time \(2^{o(\log b) \cdot n} \cdot \text{poly}(n)\), which contradicts the ETH by Theorem 1.3. However, a careful examination shows that this chain of reductions would yield instances of \((r, k)\)-Monomial Testing with \(r = O(\sqrt{k}/ \log k)\) only. 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 \(\sum_{i=1}^{n} a^{(j)}_i < 10\) for every \(j = 1, \ldots, n\). The goal is to verify whether there is a sequence of indices \(1 \leq i_1 < \cdots < i_k \leq n\) such that \(\sum_{q=1}^{k} a_{i_q} = s\). (Note that by the small sum assumption, this is equivalent to the statement that \(\sum_{q=1}^{k} a^{(j)}_{i_q} = s^{(j)}\) for every \(j = 1, \ldots, n\).) The standard NP-hardness reduction from 3-SAT to Subset Sum, in fact, outputs an instance of Carry-Less Subset Sum of linear size, yielding the following.

**Lemma 5.4.** Unless the ETH fails, there is no algorithm that solves Carry-Less Subset Sum with \(n\) numbers in time \(2^{o(n)}\).

**Proof.** Let \(\varphi\) be an instance of 3-SAT with \(N\) variables and \(M\) clauses. By a standard NP-hardness reduction for Subset Sum (see, e.g., the textbook of Cormen et al. [8]) in polynomial time, one can build an equivalent instance of Carry-Less Subset Sum with \(O(N + M)\) numbers, each having \(O(N + M)\) decimal digits and with the sum of the \(j\)th digit in all of the numbers not exceeding 7. In case the number of numbers is different from the length of their decimal representations, we can make them equal by padding the instance by zero numbers or with zeroes in the decimal representations. Thus, by Theorem 2.1, a \(2^{o(n)}\) algorithm for Carry-Less Subset Sum would contradict the ETH.

We proceed to reducing the Carry-Less Subset Sum to \((r, k)\)-Monomial Testing. Let us choose a parameter \(t \in \{1, \ldots, n\}\). We assume without loss of generality that \(n \mod t = 0\); otherwise, we add \(t - (n \mod t)\) 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.,

\[x^{(j)} = (x^{(j-1)}q) \cdots x^{(j-1)}q)\]  

Let \(r = 10^q - 1\). Define the following polynomial over \(\text{GF}(2)\):

\[qs = \prod_{i=1}^{n} \left( y_i + z_i \cdot \prod_{j=1}^{t} x_j^{a_i^{(j)}} \cdot \prod_{j=1}^{t} x_j^{y_i^{(j-1)}} \right).\]

**Proposition 5.5.** \((s, a_1, \ldots, a_n)\) is a YES instance of Carry-Less Subset Sum if and only if \(qs\) contains the monomial \(\prod_{j=1}^{t} x_j^{y_i} \prod_{i \in S} y_i \prod_{i \in S} z_i\) for some \(S \subseteq \{1, \ldots, n\}\).

**Proof.** Consider the following polynomial over \(\text{GF}(2)\):

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

The summands in the expression above have unique sets of \(y_i\) variables; thus, each corresponds to a monomial (of coefficient 1). It is clear that these monomials, where for every \(j\) the degree of \(x_j\) is exactly \(r\), are in one-to-one correspondence with solutions of the instance \((s, a_1, \ldots, a_n)\). The claim follows by observing that polynomials \(r_s\) and \(q_s\) coincide.
Let $p_S$ denote the polynomial obtained from $q_S$ by filtering out all the monomials of degree different than $k = tr + n$.

**Proposition 5.6.** $(s, a_1, \ldots, a_n)$ is a YES instance of CARRY-LESS SUBSET SUM if and only if $p_S$ contains an $r$-monomial.

**Proof.** If $(s, a_1, \ldots, a_n)$ is a YES instance, then by Proposition 5.5, polynomial $q_S$ contains the monomial $\prod_{j=1}^{t} x_j \prod_{i \in S} y_i \prod_{i \in S} z_i$, which is an $r$-monomial. This monomial has degree $tr + n$; thus, it is contained in $p_S$ as well.

Conversely, assume that $p_S$ contains an $r$-monomial $m$. Every monomial of $q_S$ (hence, also of $p_S$) contains exactly one of the variables $y_i$ and $z_i$, with degree 1, for every $i = 1, \ldots, n$. It means that the total degree of $x_j$-type variables in $m$ is $tr$. Hence, since $m$ is an $r$-monomial, each of the $x_j$'s has degree exactly $r$. In other words, $m$ is of the form $\prod_{j=1}^{t} x_j \prod_{i \in S} y_i \prod_{i \in S} z_i$ for some $S \subseteq \{1, \ldots, n\}$. Then, $(s, a_1, \ldots, a_n)$ is a YES instance of CARRY-LESS SUBSET SUM by Proposition 5.5. □

**Proposition 5.7.** $p_S$ can be evaluated by a circuit of size $O(nt^2r + n^2t)$, which can be constructed in time polynomial in $n + t + r$.

**Proof.** Polynomial $q_S$ can be evaluated by a circuit of size $O(nt)$. The circuit for $p_S$ is built using the construction from Lemma 5.3. Thus, its size is $O(nt(tr + n)) = O(nt^2r + n^2t)$. □

We are ready to give our main lower bound for $(r, k)$-MONOMIAL TESTING. We state it in the most general form, which is also quite technical, unfortunately. Next, we derive an exemplary corollary that gives a lower bound for $r$ expressed as a function of $k$.

**Theorem 5.8.** If there is an algorithm solving $(r, k)$-MONOMIAL TESTING in time $2^{O(k \log r/r)}|C|$ $O(1)$, then the ETH fails. The statement remains true even if the algorithm works only for instances in which $r = 2^{\Theta(n/t(n))}$ and $k = t(n)2^{\Theta(n/t(n))}$ for an arbitrarily chosen function $t: \mathbb{N} \rightarrow \mathbb{N}$ computable in $2^{O(n)}$ time, such that $t(n) = \omega(1)$ and $t(n) \leq n$ for every $n$.

**Proof.** By Lemma 5.4, it suffices to give an algorithm for CARRY-LESS SUBSET SUM that works in time $2^{O(n)}$, where $n$ is the number of input numbers. Let $t = t(n)$ and $q = n/t$, $r = 10^q - 1$, $k = tr + n$ as before. Note that $r = 2^{\Theta(n/t(n))}$. Also, since $10^{n/t(n)} = \Omega(n/t(n))$, $k = \Theta(t(n)10^{n/t(n)} + n) = \Theta(t(n)10^{n/t(n)}) = t(n)2^{\Theta(n/t(n))}$.

By Proposition 5.6, solving CARRY-LESS SUBSET SUM is equivalent to detecting an $r$-monomial in $p_S$, which is a homogenous polynomial of degree $k = tr + n$. Let $C$ be the circuit for $p_S$; by Proposition 5.7, we have that $|C| = O(nt^2r + n^2t)$. If this can be done in time $2^{O(k \log r/r)}|C|$ $O(1)$, we get an algorithm for CARRY-LESS SUBSET SUM running in time

$$2^{O(k \log r/r)}|C|$O(1) = 2^{O((tr+n)q/r)}(ntr)$O(1) = 2^{O(n+q/10^q)}(ntr)$O(1) = 2^{O(n)}(ntr)$O(1).$$

Recall that $t \leq n$ and $r = 10^{n/t} - 1 = 2^{O(n)}$ since $t = t(n) = \omega(1)$. Hence, $(ntr)$ $O(1) = 2^{O(n)}poly(n)$. The claim follows. □

**Theorem 1.4.** Let $\sigma \in [0, 1)$. Then, unless the ETH fails, there is no algorithm for $(r, k)$-MONOMIAL TESTING that solves instances with $r = \Theta(k^\sigma)$ in time $2^{O(k^{1-1/(2\sigma)})} . |C|$ $O(1)$.

**Proof.** We prove that an algorithm for $(r, k)$-MONOMIAL TESTING with properties as in the statement can be used to derive an algorithm for the same problem with properties as in the statement of Theorem 5.8, which implies that the ETH fails. Take $t$ to be a positive integer not larger than $n$ such that

$$\frac{1}{2} \leq \frac{10^{n/t} - 1}{(t \cdot (10^{n/t} - 1) + n)\sigma} \leq 2;$$



ACM Transactions on Computation Theory, Vol. 11, No. 3, Article 13. Publication date: March 2019.
it can be easily verified that since $\sigma < 1$, for large enough $n$, such an integer $t \leq n$ always exists. Moreover, we have that $t = t(n) \in \omega(1)$ and $t(n)$ can be computed in polynomial time by brute force. Hence, $t(n)$ satisfies the properties stated in Theorem 5.8.

Let $t = t(n)$ and $q = n/t$. Define $r = 10^q - 1$ and $k = tr + n$; then, Equation (3) is equivalent to

$$1/2 \leq r/k^\sigma \leq 2.$$ 

Hence, $r = \Theta(k^\sigma)$. Consequently, the assumed algorithm solves $(r, k)$-MONOMIAL TESTING in time $2^{o(k \log r/r)}|C|^{O(1)}$. However, in the proof of Theorem 5.8, we have shown that the existence of an algorithm that achieves such a runtime for this particular choice of parameters implies that the ETH fails. □

Note that Theorem 1.4, in particular, implies that $(r, k)$-MONOMIAL TESTING does not admit an algorithm that achieves runtime $2^{o((\log r/r) \cdot k)} \cdot |C|^{O(1)}$ for any given $r$.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Hasan Abasi, Nader H. Bshouty, Ariel Gabizon, and Elad Haramaty. 2014. On $r$-simple $k$-path. In Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS’15), Giuseppe F. Italiano, Giovanni Fighizzini, and Donald T. Sannella (Eds.), Vol. 8635. Springer, Berlin, 1–12.

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

[3] Nader H. Bshouty. 2009. Optimal algorithms for the coin weighing problem with a spring scale. In Proceedings of the 22nd Conference on Learning Theory. McGill University, 1–10. https://www.cs.mcgill.ca/~colt2009/proceedings.html.

[4] Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. 1995. On the structure of parameterized problems in NP. Inf. Comput. 123, 1 (1995), 38–49.

[5] Marie G. Christ, Lene M. Favrholdt, and Kim S. Larsen. 2016. Online multi-coloring with advice. In Proceedings of the 14th International Workshop on Approximation and Online Algorithms (WAOA’16), Klaus Jansen and Monaldo Mastrolilli (Eds.), Vol. 8952. Springer, Berlin, 83–94.

[6] Vasek Chvátal. 1983. Mastermind. Combinatorica 3, 3 (1983), 325–329.

[7] V. Chvátal, M. R. Garey, and D. S. Johnson. 1978. Two results concerning multicoloring. In Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’16), Robert Krauthgamer (Ed.), Vol. 2. Elsevier, 151–154.

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

[9] Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub Pachocki, and Arkadiusz Socała. 2017. Tight lower bounds on graph embedding problems. J. ACM 64, 3 (2017), 18:1–18:22. DOI: https://doi.org/10.1145/3051094

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

[11] Reinhard Diestel. 2010. Graph Theory. Springer-Verlag Heidelberg.

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

[13] Fedor V. Fomin, Pinar Heggernes, and Dieter Kratsch. 2007. Exact algorithms for graph homomorphisms. Theory Comput. Syst. 41, 2 (2007), 381–393.

[14] Ariel Gabizon, Daniel Lokshtanov, and Michal Pilipczuk. 2015. Fast algorithms for parameterized problems with relaxed disjointness constraints. In Proceedings of the 23rd Annual European Symposium on Algorithms (ESA’15), Nikhil Bansal and Irene Finocchi (Eds.), Vol. 9294. Springer, Berlin, 545–556.

[15] Chris Godsil and Gordon F. Royle. 2001. Algebraic Graph Theory. Springer.

[16] Vladimir Grebinski and Gregory Kucherov. 2000. Optimal reconstruction of graphs under the additive model. Algorithmica 28, 1 (2000), 104–124.
[17] Martin Grötschel, László Lovász, and Alexander Schrijver. 1981. The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1, 2 (1981), 169–197. Corrigendum available at: http://dx.doi.org/10.1007/BF02579139.

[18] Magnús M. Halldórsson and Guy Kortsarz. 2004. Multicoloring: Problems and techniques. In Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science (MFCS’13), Krishnendu Chatterjee and Jirí Sgall (Eds.), Vol. 3153. Springer, Berlin, 25–41.

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

[20] Frédéric Havet. 2001. Channel assignment and multicolouring of the induced subgraphs of the triangular lattice. Discrete Math. 233, 1–3 (2001), 219–231.

[21] Pavol Hell and Jaroslav Nešetřil. 1990. On the complexity of H-coloring. J. Comb. Theory, Ser. B 48, 1 (1990), 92–110.

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

[23] Russell Impagliazzo and Ramamohan Paturi. 2001. On the complexity of k-SAT. J. Comput. Syst. Sci. 62, 2 (2001), 367–375.

[24] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which problems have strongly exponential complexity? J. Comput. Syst. Sci. 63, 4 (2001), 512–530.

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

[26] Fabian Kuhn. 2009. Local multicoloring algorithms: Computing a nearly-optimal TDMA schedule in constant time. In Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS’09), Susanne Albers and Jean-Yves Marion (Eds.), Vol. 3. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, Germany, 613–624.

[27] Wensong Lin. 2008. Multicoloring and Mycielski construction. Discrete Math. 308, 16 (2008), 3565–3573.

[28] Bernt Lindström. 1965. On a combinatorial problem in number theory. Canad. Math. Bull 8, 4 (1965), 477–490.

[29] László Lovász. 1978. Kneser’s conjecture, chromatic number, and homotopy. J. Comb. Theory, Ser. A 25, 3 (1978), 319–324.

[30] Dániel Marx. 2002. The complexity of tree multicolorings. In Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS’02), Krzysztof Diks and Wojciech Rytter (Eds.), Vol. 2420. Springer, 532–542.

[31] Colin McDiarmid and Bruce A. Reed. 2000. Channel assignment and weighted coloring. Networks 36, 2 (2000), 114–117.

[32] J. W. Moon and L. Moser. 1965. On cliques in graphs. Israel J. Math. 3, 1 (1965), 23–28.

[33] Jesper Nederlof. 2008. Inclusion exclusion for hard problems. Master’s thesis. Department of Information and Computer Science, Utrecht University. Retrieved February 26, 2019 from http://www.win.tue.nl/jnederlo/MScThesis.pdf.

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

[35] Craig A. Tovey. 1984. A simplified NP-complete satisfiability problem. Discrete Appl. Math. 8, 1 (1984), 85–89.

[36] Magnus Wahlström. 2011. New plain-exponential time classes for graph homomorphism. Theory Comput. Syst. 49, 2 (2011), 273–282.

Received November 2017; revised October 2018; accepted December 2018